Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Research Paper

AI for AI-based intrusion detection as a service: Reinforcement learning to configure models, tasks, and capacities

Ying-Dar Lin [a], Hao-Xuan Huang [a], Didik Sudyana [a,*], Yuan-Cheng Lai [b]

[a] Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, 300, Taiwan
[b] Department of Information Management, National Taiwan University of Science and Technology, Taipei, 106, Taiwan

ARTICLE INFO

Keywords:
ML-based IDaS
Auto-IDaS
Dynamic model selection
Capacity allocation optimization
Auto-configuration

ABSTRACT

Intrusion Detection Systems (IDS) increasingly leverage machine learning (ML) to enhance the detection of zero-day attacks. As operational complexities increase, enterprises are turning to Intrusion Detection as a Service (IDaS), requiring advanced solutions for efficient ML model selection and resource allocation. Existing research often focuses primarily on accuracy and computational efficiency, leaving a gap in solutions that can dynamically adapt. This study introduces a novel integrated solution, Auto-IDaS, which employs advanced Reinforcement Learning (RL) techniques for real-time, adaptive management of IDS. Auto-IDaS uses the Deep Q-Network (DQN) algorithm for dynamic ML model selection, automatically adjusting configurations of IDaS in response to fluctuating network traffic conditions. Simultaneously, it utilizes the Twin Delayed Deep Deterministic (TD3) algorithm for optimizing capacity allocation, aiming to minimize computational costs while maintaining service quality. This dual approach is innovative in its use of RL to address both selection and allocation challenges within IDaS frameworks. The effectiveness of TD3 is compared against Simulated Annealing (SA), a traditional optimization technique. The results demonstrate that utilizing DQN to dynamically select the model significantly improves the reward by 0.29% to 27.04%, effectively balancing detection performance (F1 score), detection time, and computation cost. Regarding capacity allocation, TD3 accelerates decision times approximately 5×10^6 times faster than SA while retaining decision quality within a 10% range comparable to SA's performance.

1. Introduction

In recent years, due to the fast-evolving nature of network threats, IDS have become a vital component of cybersecurity, with a current emphasis on employing ML to improve detection capabilities. ML-based IDS tasks range from preprocessing to traffic detection or classification. Preprocessing involves extracting features and discarding irrelevant data from the raw input. Subsequently, ML-based IDS employs either binary detection, which classifies network traffic as benign or malicious (Nassif et al., 2021), or multi-class classification, which identifies specific types of malicious traffic (Masdari and Khezri, 2020). Additionally, a two-stage machine learning model combines these methods, enhancing the detection rate for various attacks (Verkerken et al., 2023).

However, developing an ML-based IDS system requires knowledge of AI and cybersecurity, resulting in high development and management costs. Therefore, the concept of Intrusion Detection as a Service (IDaS) has emerged, where subscribers can offload those development and management processes to service providers (Lai et al., 2021). This concept reduces the operational and development costs, making the service more accessible and affordable for subscribers.

From a service provider's perspective, there are key considerations in choosing the right IDS models for IDaS. The choices range from a straightforward single-stage model using either binary detection or multi-class classification, to a more complex two-stage model that combines both methods. Binary detection offers simplicity, lower cost, and faster detection time but does not provide detailed threat insights, which can be critical for effective mitigation (Omar et al., 2013; Nandanwar and Katarya, 2024). Multi-class classification, while offering deeper insights into attack types, it may come with increased costs, longer detection times, and the risk of failing to detect zero-day attacks (Holm, 2014; Xu et al., 2023). A two-stage model integrates the strengths of both, though at a higher computational cost. When selecting the appropriate IDS models, it is crucial to evaluate the accuracy and the F1 score for detecting various attacks, detection speed,

* Corresponding author.
E-mail addresses: ydlin@cs.nctu.edu.tw (Y.-D. Lin), hhs.alex.cs10@nycu.edu.tw (H.-X. Huang), dsudyana@cs.nctu.edu.tw (D. Sudyana), laiyc@cs.ntust.edu.tw (Y.-C. Lai).

https://doi.org/10.1016/j.jnca.2024.103936
Received 18 February 2024; Received in revised form 6 May 2024; Accepted 16 May 2024
Available online 22 June 2024

and computational expenses. This study seeks to balance these factors to enhance IDaS effectiveness.

The second aspect crucial for service providers offering IDaS is the strategic allocation of resources across the service architecture, which often involves multi-tier systems including cloud, edge, and fog layers (Bierzynski et al., 2017). In this setup, tasks like preprocessing may occur in the fog, binary classification at the edge, and multi-class classification in the cloud. Each tier then requires specific computational capacity allocations based on cost considerations, as computing resource costs vary by tier. Efficient capacity allocation is vital, as it impacts both the performance and the cost-efficiency of the service. Service providers aim to configure these capacities to minimize costs while adhering to delay constraints, thereby optimizing the balance between operational efficiency and service quality in IDaS.

The increasing complexity and dynamics of networks create challenges in efficiently selecting IDS models and allocating capacities manually, underscoring the need for automation. Previous research in service management has primarily concentrated on three areas: resource management (Diro and Chilamkurti, 2018; Sundararaj, 2019; Zeng et al., 2019), which allocates computing resources, task management (De Souza et al., 2020; Almiani et al., 2020; Qu et al., 2021; Zhu et al., 2022), focusing on task assignment and effectiveness, and their joint optimization (Lai et al., 2021; Bovenzi et al., 2020; Idhammad et al., 2018; Zhang et al., 2017). While these studies provide valuable insights, they often treat resource and task management separately without considering their interdependencies and typically focus on detection times, neglecting other critical factors. Furthermore, they do not adequately address the need for an adaptive service management model that can dynamically configure IDS models, tasks, and resources in response to evolving cybersecurity threats, and traditional optimization methods have proven too slow for these complex optimizations.

This study introduces an integrated solution named Auto-IDaS with RL, which addresses the challenges of auto-model selection and capacity allocation. The model selection problem in IDaS is solved with a dynamic model selection solution to adaptively choose the most suitable IDS models in response to fluctuating traffic conditions. RL plays a pivotal role by comprehensively understanding the system's behavior and making informed decisions in real-time, thus enabling the adaptive selection of IDS models that balance crucial metrics such as the F1 score of detecting variant attacks, detection time, and computation costs. This solution enables service providers to have a flexible IDS model that can adjust to current network traffic, ensuring efficiency and effectiveness.

Moreover, in addressing the challenge of optimizing capacity allocation within the IDaS system, we leverage RL in Auto-IDaS with the goal of minimizing computation costs. This method contrasts with traditional optimization techniques by focusing on reducing the long convergence times typically associated with them. Through RL, the system is able to dynamically adjust to changes, learn from network behavior, and anticipate future demands, leading to a more expedient and effective decision-making process. Upon completion of the optimization, we offer service providers strategic recommendations for managing IDaS. These insights provide valuable direction for service providers to enhance their IDaS offerings for customers.

To deepen our understanding, we conducted comprehensive evaluations across several dimensions. We examined the IDS model selection, comparing single- and two-stage IDS models, to investigate how different configurations perform in handling malicious traffic. We also explored model selection, assessing both dynamic and static approaches to determine the performance of a dynamic model in selecting appropriate models in response to changing traffic patterns. Additionally, we considered various architecture alternatives, including 1-tier, 2-tier, and 3-tier networks, to evaluate how different network architectures manage IDaS traffic. Finally, we compared traditional optimization methods with RL to investigate the relative performance of these optimization techniques.

In relation to related work, the contributions of this paper are three-fold:

- To the best of our knowledge, this is the first study that identifies the dual challenge of adaptive IDS model selection and efficient capacity allocation within multi-tier IDaS architectures, which are crucial for the effectiveness and cost-efficiency of cybersecurity services.
- We introduce an integrated solution named Auto-IDaS, which utilizes RL to dynamically select IDS models and optimize capacity allocation. This addresses the requirement for adaptability in response to evolving network conditions and cost constraints. The technical contributions of our Auto-IDaS solution are as follows:
  - **Novel Integration of RL Algorithms:** We innovatively combine DQN and TD3 within a unified framework tailored for IDaS. This integrated approach is designed to handle the dynamic and complex requirements of real-time model selection and capacity allocation, which existing research has not explored within the IDS field.
  - **Adaptation and Fine-Tuning for IDaS:** The adaptation of DQN and TD3 for the specific challenges of intrusion detection represents a significant advancement. We have fine-tuned these algorithms to the peculiarities of network traffic patterns and attack behaviors, requiring a deep understanding of both cybersecurity and reinforcement learning.
  - **Real-World Applicability:** Auto-IDaS is among the first frameworks to apply these RL algorithms to the practical domain of IDaS, showing not only that they work in theory but also providing actionable insights and tangible improvements in a critical area of cybersecurity.
- We conduct a thorough evaluation of our RL-based approach involving various IDS models and IDS architectural configurations. This demonstrates the advantages of our approach over traditional methods. The robustness of Auto-IDaS is validated under various scenarios that mimic real-world conditions, underscoring the applicability and effectiveness of our model.

This work is organized into six sections. In Section 2, the background and related work are presented. Section 3 presents the problem formulation for IDS model selection and IDaS task assignments and capacities. In Section 4, the proposed solution approaches are discussed in detail. Section 5 presents the results of the experiments conducted to evaluate the performance of the proposed approach. Finally, in Section 6, the conclusions of the study are summarized, and future research directions are outlined.

2. Background and related work

This section begins with an overview of the technical background of this work, including multi-tier architecture, task assignment for IDaS, and the algorithms we implement for managing IDaS. At the end of this section, we compare the differences with other studies in terms of ML-based distributed IDS, task assignments, and capacities optimization.

2.1. Multi-tier architecture

Distributed architecture can provide a viable solution to the limitations of centralized systems by enabling parallel computing and proximity to users. This can be achieved by leveraging "multi-tier" systems consisting of multiple nodes, where tasks can be distributed among the components. With the advent of cloud, edge, and fog computing, these technologies have become promising paradigms that can be integrated into multi-tier architecture systems. Cloud computing offers many advantages, including flexibility, scalability, and cost-effectiveness, but it is far from the end-users, which can lead to slow response times. Edge and fog computing is designed to solve the issues by bringing computing resources closer to the user. However, these technologies

Table 1
Task assignments with single- and two-stage IDS model.

Stage type	Number of tiers	Architecture	Fog	Edge	Cloud	ID	Abbreviation
Single-stage IDS model	1	Fog	p, b(m)			1	pb(m)/–/–
		Edge		p, b(m)		2	–/pb(m)/–
		Cloud			p, b(m)	3	–/–/pb(m)
	2	Fog–Edge	p	b(m)		4	p/b(m)/–
		Fog–Cloud	p		b(m)	5	p/–/b(m)
		Edge–Cloud		p	b(m)	6	–/p/b(m)
Two-stage IDS model	1	Fog	p, b, m			7	pbm/–/–
		Edge		p, b, m		8	–/pbm/–
		Cloud			p, b, m	9	–/–/pbm
	2	Fog–Edge	p, b	m		10	pb/m/–
		Fog–Cloud	p, b		m	11	pb/–/m
		Edge–Cloud		p, b	m	12	–/pb/m
		Fog–Edge	p	b, m		13	p/bm/–
		Fog–Cloud	p		b, m	14	p/–/bm
		Edge–Cloud		p	b, m	15	–/p/bm
	3	Fog–Edge–Cloud	p	b	m	16	p/b/m

have limited capacity, making them ideal for small-scale applications. Combining cloud, edge, and fog to build a multi-tier architecture can effectively manage a diverse range of services with varying characteristics, all while ensuring both availability and reliability (Bierzynski et al., 2017).

2.2. Task assignment for IDaS

IDaS uses a multi-tier architecture and combines IDS tasks, including pre-processing, binary detection, and multi-class classification, to detect or classify malicious traffic in a computer network.

These tasks can be decomposed and mapped to multi-tier architecture, resulting in a total of 16 possible task assignments, as shown in Table 1. In this work, we use IDs and abbreviations to symbolize the tasks and placements (xxx/xxx/xxx). The left, middle, and right represent fog, edge, and cloud, respectively. The symbols p, b, and m represent pre-processing, binary detection, and multi-class classification, respectively. If no task is placed in this part, it will be represented by '-' (Lai et al., 2021). In a single-stage IDS model, b(m) means it may be either binary or multi-class. For example, task assignment 16 adopts a three-tier architecture with a two-stage ML model, abbreviated as p/b/m, indicating pre-processing, binary, and multi-class, placed in fog, edge, and cloud, respectively.

2.3. Optimization algorithms

This subsection introduces the two algorithms used to address our problems. Firstly, we employ RL to dynamically select IDS models and optimize resource costs. Additionally, we utilize the SA algorithm to optimize capacity allocation for performance comparison with RL. Our selection of SA was informed by its well-established reputation and proven track record in a wide range of optimization tasks, particularly in system architecture optimizations (Kar et al., 2023; Jin et al., 2022; Mahjoubi et al., 2022). SA is recognized for its ability to achieve a near-global optimum solution, a critical attribute when dealing with complex and high-dimensional optimization problems like those encountered in IDaS.

SA's methodology, which involves an exhaustive search process that evaluates the current solution against potential alternatives generated through a random function, is integral to its capability to escape local optima and explore a broader solution space. This attribute is especially beneficial in IDS settings, where the landscape of potential solutions can be rugged and highly variable.

There are different machine learning types, including supervised, unsupervised, and RL. Supervised learning requires labeling all traffic data, while unsupervised learning is used for current prediction without labeling data. RL does not require labeled data and is used for future prediction (Sutton and Barto, 2018). In this work, we need to learn from the network, which is difficult to label the data and obsolete to predict the current state. Thus, we chose RL with the aim of finding a balance between exploration and exploitation to achieve optimal performance.

RL involves at least one agent learning to make decisions by interacting with the environment. The environment is represented by its current state, and the agent takes actions to move from one state to the next. There are two types of actions: discrete actions and continuous actions. The main difference between them is the types of actions available, where a discrete action space is a finite predefined set, while a continuous action space involves continuous variables.

There are various algorithms for RL, including model-free and model-based, where the former represents algorithms that learn directly from interactions with the environment without a model of its dynamics, focusing on learning the value of actions or policies through trial and error; and the latter represents algorithms that involve creating and utilizing a model of the environment's dynamics to make decisions, allowing for planning and potentially more efficient learning, but requiring an accurate model for effective application. A model of the dynamics in the context of RL refers to a representation or approximation of how the environment behaves and changes in response to the actions taken by the learning agent.

In this work, we selected model-free algorithms for our IDaS framework due to the network environment's lack of reliable prior knowledge of transition probabilities and reward functions between states. Between the three categories of model-free RL — policy-based, value-based, and actor–critic — the value-based Deep Q-Network (DQN) and the actor–critic Twin Delayed Deep Deterministic Policy Gradient (TD3) were chosen for their theoretical soundness and proven practical efficacy.

DQN, a value-based algorithm, integrates Q-learning with deep neural networks to manage environments with high-dimensional and discrete action spaces effectively. Its ability to optimize discrete decision-making processes has been demonstrated in domains such as video gaming and robotic control, where it achieves performance comparable to human levels. This makes DQN particularly suited for IDS model selection, where the action space is discrete and Q-values directly inform the optimal weighting between different model combinations.

On the other hand, TD3, an enhancement of the actor–critic method, employs techniques such as double Q-learning and policy updates to prevent value overestimations and enhance policy stability, respectively. Its additional use of target policy smoothing regularization broadens exploration spaces, leading to faster convergence times compared to other actor–critic algorithms (Fujimoto et al., 2018). TD3's application in fields requiring continuous action decision-making, such as financial portfolio optimization and dynamic robotics movement, underscores its adaptability and effectiveness.

Table 2
Summary of the research on distributed IDS and service management.

Paper	Category	Service		IDS tasks			Architecture			Optimization part	Method	Purpose
		IDS	Others	P	B	M	F	E	C			
Cil et al. (2021)	Model management	O	X	X	O	X	X	X	X	DL model	Feed forward DNN	Accuracy
Nandanwar and Katarya (2024)	Model management	O	X	X	X	O	O	X	X	DL model	CNN-GRU	Accuracy
Diro and Chilamkurti (2018)	Model management	O	X	X	O	O	O	X	O	DL model	Federated LSTM	Accuracy
Syed et al. (2023)	Model management	O	X	O	O	X	O	X	O	DL model	RNN	Accuracy
Selamnia et al. (2022)	Model management	O	X	X	X	O	X	O	O	DL model	FL with DNN	Accuracy
Bovenzi et al. (2020)	Model management	O	X	X	O	O	X	X	O	DL model	Multi-modal DAE	Accuracy
Labiod et al. (2022)	Model management	O	X	X	O	O	O	X	X	DL model	vAE with MLP	Accuracy
Antonio et al. (2020)	Model management	O	X	X	O	O	O	X	X	DL model	DNN-KNN	Accuracy
Almiani et al. (2020)	Model management	O	X	X	O	O	O	X	X	DL model	RNN	Accuracy
Sundararaj (2019)	Resource management	X	O	X	X	X	X	X	O	Offloading	Ant-Bee	Minimizeexecution time
Zeng et al. (2019)	Resource management	X	O	X	X	X	X	O	X	Service migration	Deep RL	Minimizeoperational cost
Qu et al. (2021)	Resource management	X	O	X	X	X	X	O	O	Task offloading	Deep RL	Minimize delay
Mahjoubi et al. (2022)	Resource management	X	O	X	X	X	X	X	O	Task scheduling	SA	Minimize delay
Zhang et al. (2017)	Resource management	X	O	X	X	X	X	X	O	Resource configuration	Deep RL	Minimize cost
Lai et al. (2021)	Resource management	O	X	O	O	O	O	O	O	Capacity allocation	SA	Minimize delay
Lai et al. (2023)	Resource management	O	X	O	O	O	O	O	O	Cost allocation	SA	Minimize delay
Auto-IDaS (Ours)	Model & Resource management	O	X	O	O	O	O	O	O	Model selection & resource allocation	RL	Minimize cost

P/B/M: Preprocessing/Binary/Multi-class.
F/E/C: Fog/Edge/Cloud.
O/X: Applied/Not Applied.

In the context of IDaS, the selected algorithms — DQN and TD3 — facilitate an adaptive and efficient configuration of service models in response to dynamic network conditions. Their capability to continuously learn and make optimized decisions from a changing environment aligns perfectly with the needs of IDaS, which involves managing varying traffic patterns and evolving threat landscapes. This alignment not only demonstrates the suitability of these algorithms for our application but also ensures that the service models are both robust and responsive to real-world challenges.

Furthermore, the SA algorithm is a global optimization technique inspired by the annealing process in metallurgy. The motivation behind the SA algorithm is to avoid getting trapped in local optima and find the global optimum solution (Bertsimas and Tsitsiklis, 1993). In the SA algorithm, a random initial solution is selected and gradually improved by allowing for uphill moves (i.e., accepting worse solutions with some probability) to escape from local optima. As the algorithm proceeds, the temperature gradually decreases, and correspondingly, the probability of accepting a worse solution also decreases. This cooling process allows the algorithm to explore the solution space thoroughly. While simulated annealing is designed to converge towards the global optimum over time, it does not guarantee that the absolute global optimum will always be found, but rather, it often arrives at a solution that is close to the optimum.

2.4. Related works

The research of service management on distributed IDS systems has been extensively studied. In the landscape of this research, the domain can be broadly classified into two categories: model management and resource management. Table 2 summarizes the related papers to this study, elaborating on the respective service, the utilized IDS tasks, the architectures they employ, the focus of their optimization part, the method they use, and their purpose. In Table 2, the symbol 'O' indicates that a feature or characteristic is included or applicable ('applied/yes'), while 'X' signifies that it is excluded or not applicable ('not applied/no').

In the domain of model management, the primary focus is on IDS services. Various works have attempted to enhance system accuracy by optimizing DL models in this domain. For example, Cil et al. (2021) employed a Feed Forward DNN to improve the accuracy of IDS models against DDoS attack, Nandanwar and Katarya (2024) employed CNN-GRU to enhance the accuracy of IDS models for IIoT. Diro and Chilamkurti (2018) explored the utilization of both binary and multi-class tasks, implementing their federated Long Short-Term Memory (LSTM) model within a fog computing architecture. Syed et al. (2023) employed a Recurrent Neural Network (RNN). Similarly, Selamnia et al. (2022) integrated a Deep Learning model with a federated approach, and Bovenzi et al. (2020) adopted a multi-modal Deep Autoencoder (DAE). Papers Labiod et al. (2022) and Antonio et al. (2020) advanced this approach by incorporating Variational Autoencoders (VAE) with Multi-Layer Perceptrons (MLP) and a hybrid Deep Neural Network and K-Nearest Neighbors (DNN-KNN) method, respectively. However, these works often do not delve into the deployment details of the architecture or how well the proposed models handle varying traffic in different architectural settings.

On the other hand, the resource management category has a different focus, dealing with the use of architecture, optimization parts, and the purpose of the management systems. For example, Sundararaj (2019) utilized offloading with an 'Ant-Bee' algorithm to minimize execution time, while Zeng et al. (2019) discussed service migration using Deep RL to reduce operational costs, and Mahjoubi et al. (2022) focused on task scheduling using SA. Qu et al. (2021) highlighted task offloading, and Zhang et al. (2017) concentrated on resource configuration, both employing Deep RL to minimize delay and cost, respectively. Papers Lai et al. (2021) and Lai et al. (2023) are more closely related to our work, emphasizing capacity allocation with the objective of minimizing computation delay. These studies mainly utilized SA, which has a longer convergence time than RL.

Our work stands apart from these previous studies by not only focusing on capacity allocation but also on the autonomous selection of the appropriate model. We recognize that the combination of model and resource management is critical for efficient IDaS service deployment for service providers. Our novel approach combines model selection with capacity allocation, leveraging RL for faster convergence and better adaptability to dynamic conditions, thereby addressing the limitations of prior work in both categories.

3. System architecture and problem formulation

This section provides an explanation of system architecture, problem formulation, and a delay model. Incorporating the delay model with queueing theory is essential for simulating real-time processing demands in IDaS. It anticipates bottlenecks, optimizes capacity allocation and ensures efficient system response to threats. This framework mitigates delay impacts, enabling resilient and efficient IDaS systems design.

The variables and notations that were utilized are outlined in Table 3. For problem formulation, we divided it into two parts: IDS model selection and optimizing capacity allocation.

3.1. Notation table

Table 3 shows the used notation in this work, which is classified into five categories: topology, IDS model, network, service capacity, and performance. Topology encompasses the IDaS architecture, which refers to the layout of the system and the nature of the tasks to be performed. IDS models represent ML-based IDS models. The network shows the behavior in IDaS pertains to the interaction between the various components of the system. Service capacity includes the amount of data and processing that is sufficient to handle the task. Finally, performance is defined as the ability to provide efficient solutions to problems.

3.2. IDS model selection problem

In this study, we deploy binary detection and multi-class classification, with different services having different attack detection capabilities and response performance. The choice between a single or two-stage IDS model may vary depending on different factors.

The arrival traffic rate, probability of malicious flows, the set of binary models, and multi-class classification models are inputs crucial for achieving the goal of maximizing the reward for the optimal IDS model selection. This selection could involve a single-stage IDS model utilizing a binary or multi-class model or a two-stage IDS model combining both types. The objective is to identify the optimal IDS model selection capable of effectively handling diverse network conditions. The optimal IDS model selection reward is determined by three key factors: detection performance measured by the F1 score, detection time, and computation cost. A higher F1 score, indicating improved detection accuracy, contributes positively to the reward, as do shorter detection times and lower computational costs.

The detailed formula for this reward calculation is explained in Section 4.1. This formula is carefully designed to strike an optimal balance between effective threat detection, fast detection response, and efficient resource usage, ensuring that the chosen IDS model is not only accurate but also capable of dynamic adaptation to changing traffic conditions. The formal definition of the problem statement is provided as follows

- Input:
  - Arrival traffic rate (λ)
  - Probability of malicious flow (p^A)
  - Set of binary models ($\mathcal{V}^B$)
  - Set of multi-class models ($\mathcal{V}^G$)
- Output:
  - The optimal IDS model selection f^{Q^*}
- Objective:
  - Maximize the reward of the IDS model selection ($P \times p^A - (U + S^Q) \times \lambda$)
- Constraint:
  - F1 score of the combination of IDS model (P) $\geq$ threshold (Δ^P)
  - Detection time (U) $\leq$ threshold (Δ^U)
  - Computation cost for IDS model (S^Q) $\leq$ threshold (Δ^{S^Q})

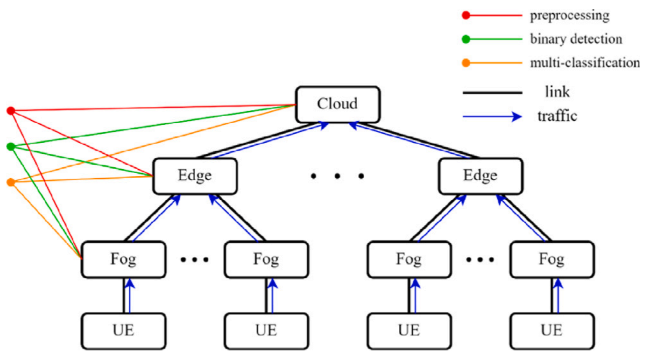

Fig. 1. IDaS architecture.

3.3. Optimization for task assignments and capacities problem

System architectures

Fig. 1 illustrates the IDaS architecture, which consists of a multi-tier architecture comprising one cloud, M edge nodes, and N fog nodes located beneath the edge nodes. We then need to align the optimal IDS model selection, denoted as f^{Q^*}, which are the outcomes of the model selection process, with the pre-processing task to the network architecture. This may involve employing either a single-stage or a multi-stage model.

Both single-stage and two-stage IDS models involve preprocessing tasks. This preprocessing task will be executed at the lowest-tier node, which could be located at the fog, edge, or cloud levels. Upon the arrival of traffic flow from user equipment (UE) at the lowest-tier node, with a length of $1/\mu_1^L$, it undergoes the preprocessing stage with a workload of $1/\mu_1^W$. This stage involves extracting features and eliminating irrelevant parts from raw data for machine learning detection, resulting in a reduced flow length.

Subsequently, in the case of a single-stage IDS model utilizing a binary model, the data with a flow length of $1/\mu_2^L$ passes through binary detection with a workload of $1/\mu_2^W$. If a multi-class model is used, the data with a flow length of $1/\mu_3^L$ undergoes multi-class classification with a workload of $1/\mu_1^W$. Additionally, in a two-stage IDS model scenario, the data is first processed through binary detection with a workload of $1/\mu_2^W$. Subsequently, only malicious traffic with a probability of p^A and flow length of $1/\mu_3^L$ is forwarded to multi-class classification to classify specific attack classes, with a workload of $1/\mu_3^W$.

Problem statement

The output of the previous problem, which is a combination of IDS models, along with the parameters of one cloud, edge nodes, fog nodes, arrival traffic rate, flow lengths, and workloads, the probability of

Table 3
Notations.

Category	Notation	Meaning	Attr
Topology	M	Number of edges	Input
	N	Number of fogs for each edge	Input
	i	IDS task and models (1: pre-processing, 2: binary model, 3: multi-class model)	Input
IDS model	Y	Number of binary models	Input
	K	Number of multi-class models	Input
	f_y^B	Binary detection model $y = \{1, 2, \dots, Y\}$	Input
	f_k^G	Multi-class classification model $k = \{1, 2, \dots, K\}$	Input
	$\mathcal{V}^B$	Set of binary models $\mathcal{V}^B = \{f_1^B, f_2^B, \dots, f_Y^B\}$	Input
	$\mathcal{V}^G$	Set of multi-class models $\mathcal{V}^G = \{f_1^G, f_2^G, \dots, f_K^G\}$	Input
	f^Q	The selected algorithm action	Output
	f^{Q^*}	The optimal IDS model selection	Output
Network	λ	Arrival traffic rate	Input
	$1/\mu_i^L$	Flow length	Input
	$1/\mu_i^W$	Flow workload	Input
	C^{UF}	Link bandwidth from UE to fog	Input
	C^{FE}	Link bandwidth from fog to edge	Input
	C^{EC}	Link bandwidth from edge to cloud	Input
	p^A	Probability of malicious flows	Input
	Z^{UF}	Propagation delay between UE and fog	Input
	Z^{FE}	Propagation delay between fog and edge	Input
	Z^{EC}	Propagation delay between edge and cloud	Input
Service capacity	C^C	Computation capacity of cloud	Output
	C^E	Computation capacity of edge	Output
	C^F	Computation capacity of fog	Output
	S^C	Cost per unit of capacity in cloud	Input
	S^E	Cost per unit of capacity in edge	Input
	S^F	Cost per unit of capacity in fog	Input
Performance	P	F1-score of IDS model	Output
	U	Detection time of IDS model	Output
	S^Q	Computation cost of IDS model	Output
	S^T	Total cost of IDaS architecture	Output
	D	Total delay of IDaS	Output
	Δ^X	Threshold. $X = \{P, D, U, S^Q\}$	Input

malicious flow, propagation delay, link bandwidths between each node, and the cost per unit capacity for cloud, edge, and fog, serve as inputs for determining the optimal task assignment. This objective is subject to the constraint that the total delay must not exceed the threshold Δ^D. To accomplish this, it is necessary to appropriately allocate capacity to each tier based on the computation cost. The following is the formal definition of the problem

- **Input:**
  - Optimal IDS model selection (f^{Q^*})
  - M edge nodes
  - N fog nodes per edge node
  - Arrival traffic rate (λ)
  - Flow length ($1/\mu_i^L$)
  - Flow workload ($1/\mu_i^W$)
  - Probability of malicious flows (p^A)
  - Link bandwidth (C^{UF}, C^{FE}, C^{EC})
  - Propagation delay between tiers (Z^{UF}, Z^{FE}, Z^{EC})
  - Cost per unit of capacity (S^F, S^E, S^C)
- **Output:**
  - The allocation of capacity to each tier (C^F, C^E, C^C) and corresponding task assignment
- **Objective:**
  - Minimize the total cost: $S^T = C^F S^F + C^E S^E + C^C S^C$
- **Constraint:**
  - The total delay (D) $\leq$ threshold (Δ^D)

Delay model

The delay model utilized in this study employs queueing theory to simulate and examine data transmission and processing delays. This approach adeptly captures the dynamics of network traffic and service mechanisms within IDaS systems. The representation is tailored to handle scenarios characterized by arrivals following a Poisson process and exponentially distributed service times, ensuring a realistic and effective simulation.

This model is instrumental in understanding and predicting the system's behavior across various load conditions, offering insights into average wait times and queue lengths. Such information proves crucial for designing and optimizing network architectures, especially in ensuring the efficient transmission and processing of data while minimizing delay in IDaS operations. In this subsection, we provide a detailed exploration of the employed delay model and its calculation process.

In order to calculate the total delay in each network component, including transmission, computation, and propagation delays, we have to know the task size, the arrival rate, the service capacity, and the different architectures used for each assignment.

Tables 4 and 5 below illustrate the task sizes and arrival rates for each task assignment. The variable j denotes the index of the task assignment, which is used to differentiate between different task assignments. Specifically, the variable "j" corresponds directly to the IDs listed in Table 1. For instance, $j = 1$ corresponds to task assignment 1 as defined within that table.

Regarding the variable r, it is utilized to differentiate between transmission and computation entities for delay calculation. The indices

Table 4
Task size for task assignment, V[j, r].

V[j, r]	Transmission			Computation (workload)		
	r = 1	r = 2	r = 3	r = 4	r = 5	r = 6
V[1, r]	$1/\mu_1^L$	ϕ	ϕ	$1/\mu_1^W + 1/\mu_2^W(1/\mu_3^W)$	ϕ	ϕ
V[2, r]	$1/\mu_1^L$	$1/\mu_1^L$	ϕ	ϕ	$1/\mu_1^W + 1/\mu_2^W(1/\mu_3^W)$	ϕ
V[3, r]	$1/\mu_1^L$	$1/\mu_1^L$	$1/\mu_1^L$	ϕ	ϕ	$1/\mu_1^W + 1/\mu_2^W(1/\mu_3^W)$
V[4, r]	$1/\mu_1^L$	$1/\mu_2^L(1/\mu_3^L)$	ϕ	$1/\mu_1^W$	$1/\mu_2^W(1/\mu_3^W)$	ϕ
V[5, r]	$1/\mu_1^L$	$1/\mu_2^L(1/\mu_3^L)$	$1/\mu_2^L(1/\mu_3^L)$	$1/\mu_1^W$	ϕ	$1/\mu_2^W(1/\mu_3^W)$
V[6, r]	$1/\mu_1^L$	$1/\mu_1^L$	$1/\mu_2^L(1/\mu_3^L)$	ϕ	$1/\mu_1^W$	$1/\mu_2^W(1/\mu_3^W)$
V[7, r]	$1/\mu_1^L$	ϕ	ϕ	$1/\mu_1^W + 1/\mu_2^W + 1/\mu_3^W * p^A$	ϕ	ϕ
V[8, r]	$1/\mu_1^L$	$1/\mu_1^L$	ϕ	ϕ	$1/\mu_1^W + 1/\mu_2^W + 1/\mu_3^W * p^A$	ϕ
V[9, r]	$1/\mu_1^L$	$1/\mu_1^L$	$1/\mu_1^L$	ϕ	ϕ	$1/\mu_1^W + 1/\mu_2^W + 1/\mu_3^W * p^A$
V[10, r]	$1/\mu_1^L$	$1/\mu_3^L$	ϕ	$1/\mu_1^W + 1/\mu_2^W$	$1/\mu_3^W * p^A$	ϕ
V[11, r]	$1/\mu_1^L$	$1/\mu_3^L$	$1/\mu_3^L$	$1/\mu_1^W + 1/\mu_2^W$	ϕ	$1/\mu_3^W * p^A$
V[12, r]	$1/\mu_1^L$	$1/\mu_1^L$	$1/\mu_3^L$	ϕ	$1/\mu_1^W + 1/\mu_2^W$	$1/\mu_3^W * p^A$
V[13, r]	$1/\mu_1^L$	$1/\mu_2^L$	ϕ	$1/\mu_1^W$	$1/\mu_2^W + 1/\mu_3^W * p^A$	ϕ
V[14, r]	$1/\mu_1^L$	$1/\mu_2^L$	$1/\mu_2^L$	$1/\mu_1^W$	ϕ	$1/\mu_2^W + 1/\mu_3^W * p^A$
V[15, r]	$1/\mu_1^L$	$1/\mu_1^L$	$1/\mu_2^L$	ϕ	$1/\mu_1^W$	$1/\mu_2^W + 1/\mu_3^W * p^A$
V[16, r]	$1/\mu_1^L$	$1/\mu_2^L$	$1/\mu_3^L$	$1/\mu_1^W$	$1/\mu_2^W$	$1/\mu_3^W * p^A$

Table 5
Arrival rate for task assignment, H[j, r].

H[j, r]	Transmission			Computation		
	r = 1	r = 2	r = 3	r = 4	r = 5	r = 6
H[1, r]	λ	ϕ	ϕ	λ	ϕ	ϕ
H[2, r]	λ	λN	ϕ	ϕ	λN	ϕ
H[3, r]	λ	λN	λNM	ϕ	ϕ	λNM
H[4, r]	λ	λN	ϕ	λ	λn	ϕ
H[5, r]	λ	λN	λNM	λ	ϕ	λNM
H[6, r]	λ	λN	λNM	ϕ	λN	λNM
H[7, r]	λ	ϕ	ϕ	λ	ϕ	ϕ
H[8, r]	λ	λN	ϕ	ϕ	λN	ϕ
H[9, r]	λ	λN	λNM	ϕ	ϕ	λNM
H[10, r]	λ	$\lambda N * p^A$	ϕ	λ	λN	ϕ
H[11, r]	λ	$\lambda N * p^A$	$\lambda NM * p^A$	λ	ϕ	λNM
H[12, r]	λ	λN	$\lambda NM * p^A$	ϕ	λN	λNM
H[13, r]	λ	λN	ϕ	λ	λN	ϕ
H[14, r]	λ	λN	λNM	λ	ϕ	λNM
H[15, r]	λ	λN	λNM	ϕ	λN	λNM
H[16, r]	λ	λN	$\lambda NM * p^A$	λ	λN	λNM

Table 6
Capacity in the rth resource, Y[r].

r = 1	r = 2	r = 3	r = 4	r = 5	r = 6
C^{UF}	C^{FE}	C^{EC}	C^F	C^E	C^C

$r = 1, r = 2$, and $r = 3$ denote the communication resources, representing the links connecting User Equipment (UE) to fog nodes ($r = 1$), fog nodes to edge nodes ($r = 2$), and edge nodes to the cloud ($r = 3$), respectively. Conversely, $r = 4, r = 5$, and $r = 6$ represent the computational resources within the fog nodes ($r = 4$), edge nodes ($r = 5$), and the cloud ($r = 6$), respectively.

In Table 4, terms within parentheses are used to denote alternative parameters for multiclass task processing, analogous to the notations presented in Table 1. These parentheses present options for the system to choose from based on the specific task scenario.

With the task sizes, arrival rates, and service capacity for each task assignment above, the total delay for each task assignment can be expressed as

$$D_j = \Sigma_{r=1}^{6} b(j,r) \tag{1}$$

where

$$b(j,r) = \begin{cases} \frac{1}{\frac{Y[r]}{V[j,r]} - H[j,r]} & \text{,if V[j,r]} \neq \phi \\ 0 & \text{,if V[j,r]} = \phi. \end{cases} \tag{2}$$

In the process of calculating the delay for communication links and computation nodes, it needs the values provided in Tables 4–6. Additionally, it is crucial to consider transmission and propagation delays when evaluating communication delays. Therefore, we incorporate the propagation delay as an additional factor, which can be expressed as

$$Z^{UF} \times f(V[j,1]) + Z^{FE} \times f(V[j,2]) + Z^{EC} \times f(V[j,3]) \tag{3}$$

where

$$\begin{aligned} f(x) &= 1 \quad \text{, if } x \neq \phi \\ f(x) &= 0 \quad \text{, if } x = \phi. \end{aligned} \tag{4}$$

Eq. (1) calculates the total delay, integrating various types of delays experienced within the network. It specifically aggregates communication and computation delays encountered across different network entities. The purpose is to provide a comprehensive measure of delay, reflecting the cumulative impact of all processing and transmission activities within the IDS system.

Eq. (2) then applies the queuing theory to compute the delay for processing a flow at each network entity. It is derived by dividing the capacity by the workload and then subtracting the arrival rate. This approach helps in understanding how efficiently each entity processes its portion of network traffic, which is crucial for optimizing the overall system performance.

Lastly, Eq. (3) is used to determine the propagation delay between nodes. It focuses on the time taken for data to travel from one point to another within the network, which is critical for evaluating the network's response time and efficiency.

Consider Task Assignment 1 as the example for delay calculation, where traffic processing occurs exclusively at the fog layer. The traffic is transmitted from UE to a fog link with a flow length of $1/\mu_1^L$, an arrival rate of λ in $H[1,1]$, and a capacity of C^{UF} in $Y[1]$. The transmission delay for this link can be calculated by inputting all variables into Eq. (2).

Upon reaching the fog node, the traffic undergoes processing with a flow workload of $V[1,4]$, an arrival rate of $H[1,4]$, and the fog node's processing capacity of $Y[4]$. The computation delay is determined by applying these values to Eq. (2). The total delay associated with task assignment 1 is obtained by summing the computed transmission delay from UE to the fog node and the computation delay at the fog node, as determined by Eq. (1). Additionally, the propagation delay is included as calculated from Eq. (3).

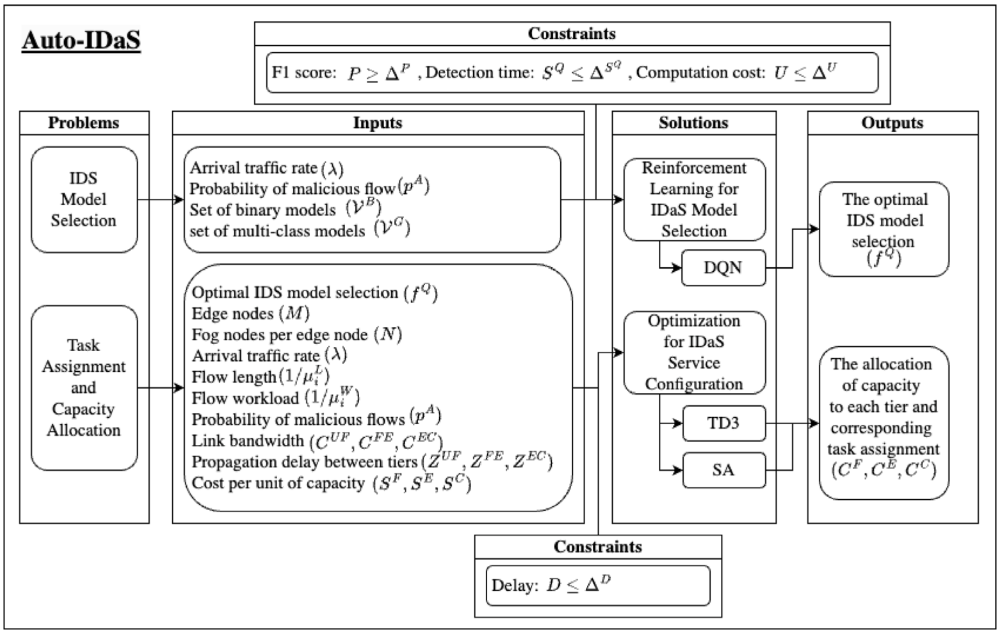
Auto-IDaS
Constraints
F1 score: $P \geq \Delta^P$, Detection time: $S^Q \leq \Delta^{S^Q}$, Computation cost: $U \leq \Delta^U$
Problems
IDS Model Selection
Task Assignment and Capacity Allocation
Inputs
Arrival traffic rate (λ)
Probability of malicious flow (p^A)
Set of binary models ($\mathcal{V}^B$)
set of multi-class models ($\mathcal{V}^G$)
Optimal IDS model selection (f^Q)
Edge nodes (M)
Fog nodes per edge node (N)
Arrival traffic rate (λ)
Flow length ($1/\mu_i^L$)
Flow workload ($1/\mu_i^W$)
Probability of malicious flows (p^A)
Link bandwidth (C^{UF}, C^{FE}, C^{EC})
Propagation delay between tiers (Z^{UF}, Z^{FE}, Z^{EC})
Cost per unit of capacity (S^F, S^E, S^C)
Solutions
Reinforcement Learning for IDaS Model Selection
DQN
Optimization for IDaS Service Configuration
TD3
SA
Outputs
The optimal IDS model selection (f^Q)
The allocation of capacity to each tier and corresponding task assignment (C^F, C^E, C^C)
Constraints
Delay: $D \leq \Delta^D$

Fig. 2. The overview of Auto-IDaS.

4. Auto-IDaS: AI for AI-based IDaS solution

The solutions in Fig. 2 outline how Auto-IDaS solves the problems. This study has two main problems: the IDS model and service deployment, which includes task assignment and capacity allocation. In order to solve those problems, we utilize DQN and TD3 algorithms.

The process pipeline of Auto-IDaS framework is initiated with the capture of network traffic, which is served as the input data for subsequent steps. Following this, the best-suited IDS model for the current traffic conditions is evaluated and selected by DQN algorithm, aiming to maximize detection effectiveness while considering system constraints. Subsequently, the necessary processes across the network architecture are allocated using the TD3 algorithm, taking into account various factors such as computational resources and latency requirements to ensure optimal distribution and execution of tasks.

Furthermore, several enhancements have been implemented to tailor these algorithms to the specific challenges of our IDaS framework:

- **Customization of Reward Function:** We have designed reward functions for both DQN and TD3 that are tailored to the IDS context, taking into account not just accuracy, but also cost efficiency and detection speed, which are critical in IDaS.
- **Action Space Optimization:** The action spaces for both algorithms have been carefully constructed to address the granular nuances of IDS tasks, allowing for more precise model selections and resource allocations.
- **State Representation Enhancements:** The state representations used by the algorithms have been enriched to incorporate a more comprehensive view of the system state, enabling more informed and strategic decision-making by the agents.
- **Hyperparameter Tuning:** Through extensive experimentation, we have fine-tuned the hyperparameters of both algorithms, ensuring that they are optimally configured for the IDS environment.
- **Integration with IDS:** Both DQN and TD3 have been integrated into the IDS pipeline to demonstrate a novel approach to model selection and task assignment, one that is more responsive to dynamic network conditions and capable of real-time adaptation.
- **Benchmarking Against Traditional Models:** We have included a comparative analysis that benchmarks our improved DQN and TD3 models against their traditional counterparts. This comparison highlights the performance gains achieved through our enhancements.
- **Contribution to Processing:** We clarify the contributions made during the processing phase by articulating the improvements in model selection accuracy and system efficiency. We also demonstrate how these enhancements lead to a more effective and responsive IDS.

Section 4.1 details the IDS model selection using the DQN algorithm. Following this, Section 4.2 elaborates on the capacity allocation solution utilizing the TD3 algorithm and compares its performance with SA.

4.1. *DQN for IDS model selection*

A Deep Q-Network (DQN) is described in Algorithm 1, which finds the optimal IDS model selection in different situations. RL is chosen because it can dynamically predict future information and achieve approximate optimal performance based on exploration and exploitation. The use of DQN can solve the problem of the state space (inputs) being continuous, but the action space (outputs) being discrete.

In our DQN-based approach for optimal IDS model selection, it is essential to define the environment (state), the agent (action), and the reward, in addition to setting up an action-value neural network $\mathcal{N}$ and a replay buffer B_{DQN}.

The problem addressed in Algorithm 1 is episodic in nature. A terminal state in the context of this algorithm is reached under two conditions: firstly, the end of an individual episode occurs when the maximum number of steps, denoted by Q, within that episode is attained, as indicated by the termination of the inner loop `For`{$q = 1$ `to` Q}. Secondly, the global terminal state of the entire algorithm is achieved when the episode count equals the total number of episodes E, as denoted by the completion of the outer loop `For{episode = 1 to` E}. Thus, the algorithm runs for E episodes, with each episode comprised of Q steps. The terminal state of an individual episode is marked by reaching the Qth step, and the global terminal state of the algorithm is signified by the fulfillment of all E episodes.

First, the state of our environment comprises the arrival traffic λ and the malicious rate p^A. The choice of λ represents the volume of network traffic, impacting the performance demands on the IDS. The malicious

Algorithm 1 DQN for IDS Model Selection

```
1: Input: Arrival traffic rate $\lambda$, probability of malicious flow $p^A$, set of binary
   models $\mathcal{V}^B$, set of multi-class models $\mathcal{V}^G$;
2: Output: The optimal IDS model selection $f^{Q^*}$;
3: Initialize replay buffer $B_{DQN}$;
4: Initialize action-value network $\mathcal{N}$ with random weights $\theta$;
5: for episode = 1 to $E$ do
6:     Initialize sequence $s_1 = \{h_1\}$ and pre-processed sequence $\phi_1 = \phi(s_1)$
7:     Initialize the best known action-value $f^{Q^*}$ if available
8:     for $q = 1$ to $Q$ do
9:         With probability $\epsilon$ select a random action $f^Q$
10:        otherwise select $f^Q = \arg\max_a \mathcal{N}(\phi(s_q), a; \theta)$
11:        Execute action $f^Q$ in emulator and observe reward $r_q$ and next state
   $h_{q+1}$
12:        if reward $r_q$ for $f^Q$ is higher than for $f^{Q^*}$ then
13:            Update the best known action-value: $f^{Q^*} = f^Q$
14:        end if
15:        Set $s_{q+1} = s_t, a_t, h_{q+1}$ and pre-process $\phi_{q+1} = \phi(s_{q+1})$
16:        Store transition $(\phi_q, a_q, r_q, \phi_{q+1})$ in $B_{DQN}$
17:        Sample random mini-batch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $B_{DQN}$
18:        Set $y_j = \begin{cases} r_j, \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \mathcal{N}(\phi_{j+1}, a'; \theta), \text{otherwise} \end{cases}$
19:        Perform a gradient descent step on $(y_j - \mathcal{N}(\phi_j, a_j; \theta))^2$ with network
   parameters $\theta$
20:    end for
21:    return the best known action-value function $f^{Q*}$
22: end for
23: return the trained action-value network $\mathcal{N}$ and the best known
   action-value function $f^{Q^*}$
```

rate p^A measures the proportion of potentially harmful traffic, which is vital for assessing the severity of threats the IDS must address. These factors provide a comprehensive view of the network environment's current state.

Second, the action is denoted by the chosen IDS model selection, represented as f^{Q^*}. This aligns the learning process with the practical objective of identifying the optimal IDS model selection for the given network state.

Third, the reward structure is based on the performance of the chosen IDS model combination, measured by metrics P, U, and S^Q, and defined as $P \times p^A - (U + S^Q) \times \lambda$. This reward structure balances effective detection performance, fast detection time, and efficient resource use under varying traffic and threat levels.

The process of Algorithm 1 commences with the initialization of a replay buffer B_{DQN} for storing experiences and an action-value network $\mathcal{N}$ with randomly assigned weights θ. Across multiple episodes, each representing a unique sequence of actions and outcomes, the state of the environment is established using arrival traffic λ that is randomly taken from the CICIDS-2017 dataset and malicious rate p^A and subsequently pre-processed.

Within each episode step, the algorithm either selects a random action f^Q with a probability ϵ to explore, or it exploits known strategies by choosing the action that maximizes the value predicted by the network $\mathcal{N}$. Post-action execution observes the resultant reward and the ensuing state, updates the sequence, and stores this transition in B_{DQN}. Furthermore, if the current reward r_q for current f^Q is higher than the previous one, then it updates the best-known action value of $f^{Q^*} = f^Q$. A significant aspect of the algorithm is the periodic sampling of mini-batches of transitions from the replay buffer, which are then used to update the network weights θ through gradient descent. This updating process aims to minimize the discrepancy between the network's predicted values and the computed target values based on observed rewards and the discounted value of future actions.

This iterative procedure enables the DQN to progressively refine its approach for deciding the optimal IDS model selection f^{Q^*}, adapting to varying network conditions as indicated by λ and p^A. The reward function, integrating metrics such as detection performance (P), detection time (U), and computation cost (S^Q), steers the DQN towards a balance of F1 score, efficiency, and resource management in diverse scenarios.

4.2. Optimization for task assignments and capacities

This section explains our proposed solution, which employs the TD3 policy gradient algorithm to optimize task assignments and capacities. Additionally, we utilize the SA algorithm as a basis for performance comparison with TD3.

RL for optimizing task assignments and capacities

TD3 algorithm is described in Algorithm 2 for optimizing task assignments and capacities. This optimization aims to minimize the total cost while identifying optimal tasks and capacities. This method can solve complex combinatorial optimization problems with a shorter decision time.

Algorithm 2 TD3 for Optimizing Task Assignments and Capacities

```
1: Input: $f^{Q^*}$, $M$, $N$, $\lambda$, $1/\mu_i^L$, $1/\mu_i^W$, $p^A$, $C^{UF}$,$C^{FE}$,$C^{EC}$, $Z^{UF}$,$Z^{FE}$,$Z^{EC}$,
   $S^F$, $S^E$, $S^C$;
2: Output: $C^F$, $C^E$, $C^C$;
3: Initialize replay buffer $B_{TD3}$
4: Initialize critic networks $\mathcal{R}_{\theta_1}, \mathcal{R}_{\theta_2}$, actor-network $\pi_\phi$ with random
   parameters $\theta_1, \theta_2, \pi_\phi$;
5: Initialize target networks $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$
6: for $t = 1$ to $\mathcal{T}$ do
7:     State $s = [S^T, D]$
8:     Select action $a = [C^C, C^E, C^F]$ with exploration noise $\pi_\phi s + \epsilon$
9:     Calculate total cost $S^T$ and total delay $D$
10:    Calculate reward $r = -S^T \times c_1 - (D - \Delta^D) \times c_2$
11:    Observe new state $s'$
12:    Store transition tuple $(s, a, r, s')$ in $B_{TD3}$
13:    Sample random mini batch of $\mathcal{N}$ transitions $(s, a, r, s')$ from $B_{TD3}$
14:    Calculate $\tilde{a}, y$ and update $\theta$
15:    if $t$ mod $d$ then
16:        Update $\phi$ by deterministic policy gradient:
   $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_\theta(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
17:        Update target networks
   $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$
   $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
18:    end if
19: end for
20: return $C^F$, $C^E$, $C^C$
```

Using this approach, we must define the environment (state), agent (action), and reward. First, the environment is the total cost S^T, and the total delay D. Second, the action is represented by corresponding capacity allocations. Third, reward is defined as $r = -S^T \times c_1 - (D - \Delta^D) \times c_2$, where c_1 and c_2 are used to balance the weights. To address the overestimation problem, the TD3 algorithm incorporates two critic networks, Q_{θ_1} and Q_{θ_2}. In addition, it includes an actor-network π_θ for policy gradient optimization. Each network has a corresponding target network. Lastly, the presence of a replay buffer B_{TD3} generates a stable network of actors.

The process of Algorithm 2 commences with initializing a replay buffer, the critic, and actor networks with random parameters to facilitate unbiased learning from scratch. Corresponding target networks are also established to provide stable target values during the training updates. As the algorithm iterates through a predetermined number of episodes, it captures the state of the environment, denoted by S, S^T, D. Then, the algorithm selects an action regarding capacity for each tier. This action is augmented with exploration noise to ensure a comprehensive search across the action space, thus preventing premature convergence to suboptimal policies.

The algorithm then calculates the total cost associated with the selected action and observes the immediate reward, which reflects the efficacy of the action relative to the system's objectives. Following this, the algorithm updates its state representation and records the transition in the replay buffer, ensuring that these experiences can be revisited to refine the decision-making process. In a key step, the algorithm employs a mini-batch approach to sample past transitions, which helps break the temporal correlations and promotes more robust learning.

One of the distinguishing features of TD3, which is incorporated in this algorithm, is the delayed policy update mechanism. By updating the policy only at specific intervals, the algorithm stabilizes the learning process by preventing the actor-network from frequently chasing moving targets. The deterministic policy gradient then updates the actor-network, guiding it toward actions expected to yield higher rewards. Complementing this, the target networks are softly updated, which blends the old and new parameters to ensure that the targets change slowly over time, thereby contributing to the overall stability of the learning process. This iterative cycle of action selection, evaluation, and refinement continues until the algorithm converges to a policy that effectively balances task performance with computational efficiency.

SA for optimizing task assignments and capacities

The SA algorithm, detailed in Algorithm 3, is applied to optimize task assignments and capacities (resources for tasks) and is a benchmark against the TD3 algorithm. This algorithm excels in identifying globally optimal capacity allocations at minimal costs. Its ability to perform stochastic variations of the current solution helps it avoid getting trapped in local minima, thereby making it a highly effective tool for addressing complex combinatorial optimization problems.

Initially, we randomly select C^C, C^E, and C^F between minimum and maximum constraints. The minimum constraint is set to avoid delay values less than zero since a negative delay means the traffic is never processed. Typically, the maximum constraint is determined based on the actual maximum capacity of the node; however, in this study, it is assumed to be 30 times the minimum constraint. Next, calculate the total cost and average delay. The new solution replaces the current solution if it has a lower total cost and meets the delay threshold. Otherwise, it may be accepted based on a $1/exp(\Delta S'/t)$ probability.

Algorithm 3 SA for Optimizing Task Assignments and Capacities

```
1: Input: $f^{Q^*}$, $M$, $N$, $\lambda$, $1/\mu_i^L$, $1/\mu_i^W$, $p^A$, $C^{UF}, C^{FE}, C^{EC}$, $Z^{UF}, Z^{FE}, Z^{EC}$, $S^F, S^E, S^C$;
2: Output: $C^F, C^E, C^C$;
3: Initialize $C^C, C^E, C^F, S^T, t_{ini}, \alpha$;
4: $t = t_{ini}$
5: while $t > t_{trm}$ do
6:     $C^{C'} = \text{rand}[C^C_{min}, C^C_{max}]$
7:     $C^{E'} = \text{rand}[C^E_{min}, C^E_{max}]$
8:     $C^{F'} = \text{rand}[C^F_{min}, C^F_{max}]$
9:     Calculate total cost $S' = C^{F'} S^F + C^{E'} S^E + C^{C'} S^C$
10:    Calculate average delay $D'$
11:    $\Delta S' = S' - S^T$
12:    if $S' \leq S$ and $D' \leq \Delta^D$ then
13:        Accept new $C^C = C^{C'}, C^F = C^{F'}, C^E = C^{E'}, S^T = S'$
14:    else if random() $< \frac{1}{\exp(\Delta S'/t)}$ then
15:        Accept new $C^C = C^{C'}, C^F = C^{F'}, C^E = C^{E'}, S^T = S'$
16:    end if
17:    Decrease the temperature: $t = \alpha \times t$
18: end while
19: return $C^F, C^E, C^C$
```

In this study, the control parameters t (temperature) and the cooling parameter (α) were used to control the stochasticity of the search process. At high temperatures, poor solutions are accepted with a probability of $1/exp(\Delta S'/t)$ to prevent local optimization. Because exploration takes precedence in the heat phase. Instead, the focus under low-temperature shifts to only acceptable and reasonable solutions. The search terminates when the temperature reaches the t_{trm}.

5. Results and analysis

This section analyzes various detection modes of ML-based IDS and capacity optimization of IDaS task assignments. The section begins by explaining the various detection modes and the baseline configurations, detailing the parameters used. It then analyzes the IDS model selection by investigating tenant traffic learning to determine the performance of a dynamic model in selecting the appropriate models for different traffic patterns. Subsequent subsections explore the architecture alternative of IDaS and compare the performance between traditional optimization methods and RL techniques, focusing on decision time and quality.

5.1. ML models for IDS

Each detection mode in our study leverages a selection of IDS models chosen for their effectiveness in intrusion detection scenarios. For binary detection tasks, we have opted for One-Class Support Vector Machines (OCSVM) and Autoencoders (AE) due to their capability to capture anomalies and deviations from normal network behavior. Similarly, for multi-class classification, we employ Neural Networks (NN) and Random Forests (RF) for their ability to handle complex patterns and diverse attack types.

These ML models were selected based on Verkerken et al. (2023), which provides an in-depth analysis of various models within a multi-tiered, hierarchical intrusion detection context. Verkerken et al. (2023) evaluated each model's unique capabilities to address specific intrusion threats, leveraging their individual strengths to bolster overall detection performance. Our decision to use these models was underpinned by their proven performance in layered defense architectures similar to those employed in our study. This relevance to our work's objective — to explore and refine IDS capabilities within a hierarchical framework — made them particularly pertinent for inclusion in our analysis.

We then proceed to train these models using the CICIDS-2017 dataset. Our selection of this dataset is based on several considerations. Firstly, CICIDS-2017 is widely recognized in the research community as a benchmark dataset for evaluating IDS solutions, offering a diverse range of network traffic scenarios and attack types. Secondly, the dataset contains a large volume of labeled network traffic data, enabling comprehensive training and evaluation of our proposed method. Lastly, the dataset's design mirrors real-world complexities, including the latest attack vectors and benign activities, making it an ideal choice for testing the effectiveness of IDS solutions in detecting and differentiating between malicious and normal traffic. To facilitate comprehensive training and evaluation of our proposed IDS framework, we partitioned the CICIDS-2017 dataset into three segments: training, validation, and testing.

It is noteworthy that, based on findings from Verkerken et al. (2023), we opted not to include neural networks for the two-stage analysis. This decision was informed by evidence suggesting that neural networks may not consistently outperform random forests in this specific context, further supporting our experimental design choices.

5.2. Parameter settings

Table 7 lists the parameters utilized in IDS model selection and optimization for task assignments and capacities. The parameters within our Auto-IDaS framework have been set manually. This approach was chosen to ensure that our experimental setup reflects both the established research and the operational realities of contemporary network systems. Some of the parameters were derived from existing literature to align our work with recognized benchmarks and ensure comparability with prior studies. We believe that this combination of literature-derived and empirically-measured parameters enables our study to present a realistic and applicable reflection of IDaS system operations. It also facilitates the replication of our results and contributes to our research's overall integrity and utility.

Table 7
Baseline configuration for experiment.

Notation	Configuration	Unit
M	400	Nodes
N	20	Nodes
λ	100	Flows/s
p^A	0.2	
$1/\mu_1^L$	124.99	kb/flow
$1/\mu_2^L$	6.62	kb/flow
$1/\mu_3^L$	6.62	kb/flow
$1/\mu_1^W$	1 137 406	Instruction/flow
$1/\mu_2^W$ (AE)	310 423	Instruction/flow
$1/\mu_3^W$ (RF)	142 032	Instruction/flow
Z^{UF}	3.335×10^{-6}	Seconds. Distance: 1 km
Z^{FE}	3.335×10^{-5}	Seconds. Distance: 10 km
Z^{EC}	3.335×10^{-3}	Seconds. Distance: 1000 km
C^{UF}	1	Gbps
C^{FE}	100	Gbps
C^{EC}	100	Gbps
S^F	20	Money units/instruction
S^E	15	Money units/instruction
S^C	10	Money units/instruction
Δ^D	0.5 and 0.02	s
Δ^U	30	s
Δ^{S^Q}	20	s (GPU time)
Δ^{S^Q}	20	s (GPU time)
Δ^P	0.75	

During the IDS model selection experiment, we conducted 200 iterations of simulations with varying traffic and malicious rates. In each iteration, we randomly sampled traffic from the CICIDS-2017 dataset. We then compared the average performance of six fixed IDS models against the performance of six IDS models dynamically selected by DQN.

For task assignments and capacities optimization, our setup comprised a cloud node and 400 edge nodes, with each edge node covering 20 fog nodes to ensure comprehensive coverage across the large area, and the link bandwidth based on a 5G network (Lai et al., 2021). We determined the distances between tiers from Thai et al. (2019) and used the flow length from Megyesi and Molnár (2013) for pre-processing. The probability of malicious flow (p^A) is manually defined. We set this probability to 0.2 by default, aligning it with a realistic proportion of malicious traffic in typical network environments. This value balances low-threat activity and heavy attack scenarios, offering a practical middle ground for evaluations. By selecting this default value, we aim to approximate real-world IDaS settings, ensuring the practicality and applicability of our results.

Furthermore, to measure the performance of the binary (autoencoder) and multi-class (random forest) models, we monitored instruction numbers using PERF on Linux and conducted tests on an Intel I5-12400F 2.5 GHz CPU. Then we examined the computation costs, assigning higher costs to the lower tiers of computing power. This accounts for the increased costs in the lowest tier, including real estate, cooling, and maintenance. In contrast, using large hyperscale data centers, costs can be reduced through economies of scale (Lai et al., 2021). Finally, we set the delay threshold between 0.5 and 0.02 to compare the difference in the cost of task assignment.

5.3. IDS model selection: Single- vs. Two-stages

In this study, we developed six ML-based IDS models. In the single-stage IDS framework, we implemented AE and OCSVM for binary detection, along with RF and NN for multi-class classification. Additionally, we constructed two-stage IDS models, namely AE+RF and OCSVM+RF. The evaluation of IDS models is influenced by three critical factors: detection performance (F1 score), detection time, and computation cost. Our experiments also revealed that RF is more suitable as the second-stage model than NN, exhibiting superior performance in those three factors. This result is elaborated on in the following subsection, providing a comprehensive understanding of the performance of different IDS models.

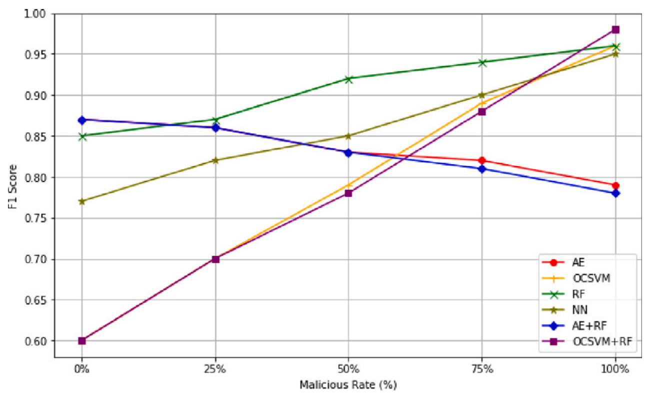

Fig. 3. The detection performance of IDS model.

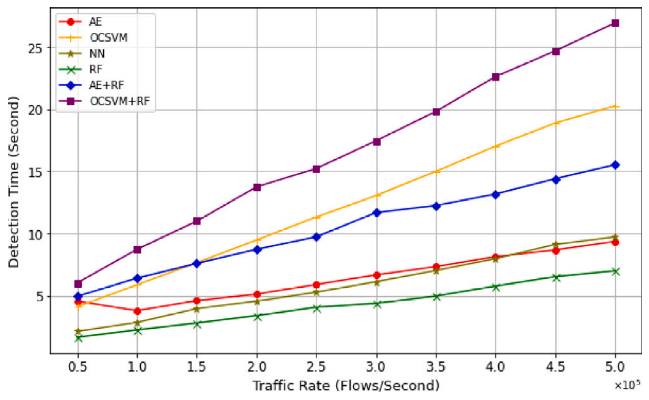

Fig. 4. Detection time for various IDS models.

Fig. 3 shows the performance of the detection performance from all IDS models. We tested the performance by adjusting the malicious rate from 0 to 100%. Observing the results in Fig. 3, in the binary detection, we noted that AE achieves optimal performance in benign traffic scenarios but experiences a substantial decline in F1 score as the volume of malicious traffic increases from 0.87 to 0.79. Conversely, OCSVM exhibits the opposite behavior, with its F1 score improving as the amount of malicious traffic grows. These findings highlight AE's proficiency in handling normal traffic patterns and OCSVM's ability to adapt to more complex malicious traffic situations.

Furthermore, RF outperforms NN in multi-class classification. Based on this result, we have selected RF as the second stage in two-stage IDS models. The results indicate that employing two-stage IDS models demonstrates superior capabilities compared to using a single NN or RF model alone. The combinations of OCSVM+RF effectively leverage the strengths of both anomaly-based and signature-based detection methods, resulting in enhanced performance.

Analyzing Figs. 4 and 5, we investigated the resource consumption and detection times of OCSVM and RF as the traffic rate increases. OCSVM exhibits a significant increase in resource consumption and detection times due to its algorithmic complexity. In contrast, RF demonstrates relatively lower resource consumption and faster detection times. These findings highlight the advantages of RF, which are attributed to its parallelized nature and efficient tree-based algorithm.

Our experiments found that IDS models perform differently in various scenarios. No single-stage IDS model can handle all traffic and

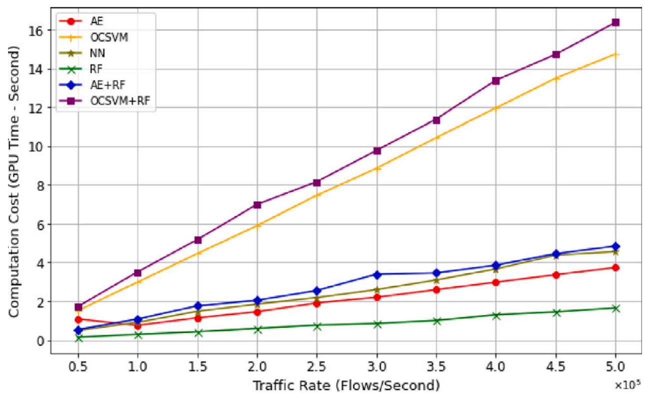

Fig. 5. Computation cost (GPU time) for various IDS models.

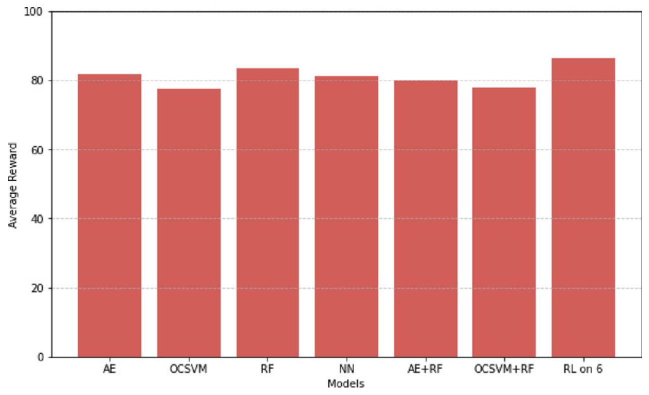

Fig. 6. Average reward of IDS models in handling fluctuating traffic.

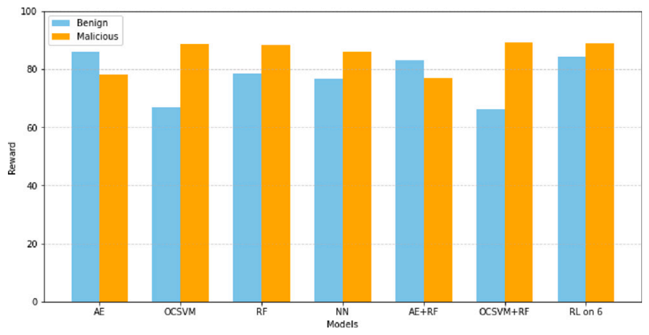

Fig. 7. Reward of IDS models in handling benign and malicious traffic.

malicious rates effectively. Employing a two-stage IDS model can enhance the detection performance but result in a longer detection time and higher computation cost, which is inappropriate when a massive amount of incoming traffic needs to be handled. Therefore, the selection of suitable IDS combinations is crucial. We utilize DQN to choose IDS combinations dynamically based on our observations to address this dynamic requirement.

5.4. Model selection (dynamic vs. Static)

The performance of IDS models varies across different scenarios, making it challenging to find a single model that can handle diverse traffic rates and levels of malicious activity. Recognizing the dynamic nature of network environments, we address this challenge by employing DQN to dynamically select IDS models that are well-suited for specific traffic conditions. This allows us to adapt our approach to the changing nature of network traffic and optimize the performance of the IDS system.

Fig. 6 illustrates the average reward for each model combination in managing fluctuating benign and malicious traffic. Notably, employing DQN to dynamically select models for handling traffic proves more effective than other fixed IDS models, consistently maintaining a reward above 80%.

Furthermore, a detailed examination of each model's reward when facing exclusively benign or malicious traffic in fluctuating environments reveals valuable insights. As illustrated in Fig. 7, the rewards for models tackling only benign or malicious traffic are examined separately. This analysis highlights that no static model consistently excels in handling the dynamic nature of such traffic. For example, AE shows commendable performance in benign traffic scenarios, achieving a high reward, but its efficacy significantly diminishes in the presence of malicious traffic. On the other hand, AE+RF presents a more balanced performance, resulting in rewards of 76.81 and 79.97 in benign and malicious traffic scenarios, respectively. However, this amalgamation incurs a substantial computation cost, adversely impacting the overall reward. In contrast, employing a DQN to dynamically select the most appropriate model in response to fluctuating traffic conditions emerges as a more efficient strategy.

Moreover, we assess the percentage differences in performance among various IDS models. To quantify these disparities when employing DQN for dynamic model selection compared to static models in handling both benign and malicious traffic, we apply the following formula:

$$\left(\frac{\text{RL on 6 (DQN)} - \text{Value of Other Model}}{\text{Value of Other Model}}\right) \times 100\%$$

In handling the benign traffic, utilizing DQN to dynamically select models surpasses the AE model by 1.34%, OCSVM by 26.09%, RF by 7.45%, NN by 9.97%, and OCSVM+RF by 27.04%. In malicious traffic, it outperforms AE by 13.79%, OCSVM by 0.29%, RF by 0.42%, NN by 3.29%, and AE+RF by 15.51%. Summarizing these findings, utilizing DQN to select models dynamically outperforms the other models with a range of effectiveness improvement from 0.29% to 27.04% across different scenarios, highlighting its varied but generally superior performance in detecting benign and malicious activities. This suggests the approach effectively balances the tradeoff between F1 score, detection time, and computation cost.

Table 8 shows the decision of DQN made under different traffic conditions. As the traffic increases, the impact on detection time is considered by DQN, which then selects a single model with a higher reward. Moreover, in the presence of malicious traffic, DQN tends to choose an IDS model with better detection capabilities. The ability of DQN to dynamically select IDS model architectures aligns with our expectations and offers valuable insights. It empowers the IDS system to adapt and react to evolving network conditions, ensuring optimal performance across various traffic scenarios. By leveraging RL techniques, such as DQN, we can harness the power of intelligent decision-making to enhance the effectiveness and efficiency of IDS systems in real-world environments.

5.5. Optimization for task assignment and capacities

Architecture alternatives: 1- vs. 2- vs. 3-tier

Different task assignments can result in varying optimal capacity allocations and total costs under the same data arrival rate. We utilized SA to determine the optimal capacity allocations while keeping the data arrival rate and malicious rate fixed at 100 flows/s and 0.2, respectively. By comparing the total costs, we evaluate the performance of different task assignments in terms of cost efficiency.

Figs. 8 and 9 compare the total cost after allocating capacity to each tier between task assignments with tightened delay thresholds (0.02 s)

Table 8
DQN decision under different conditions.

Traffic rate	100 000	100 000	100 000	100 000	500 000	500 000	500 000	500 000
Malicious rate	0	0.2	0.4	1	0	0.2	0.4	1
Decision	AE	AE+RF	OCSVM+RF	OCSVM+RF	AE	RF	OCSVM	OCSVM

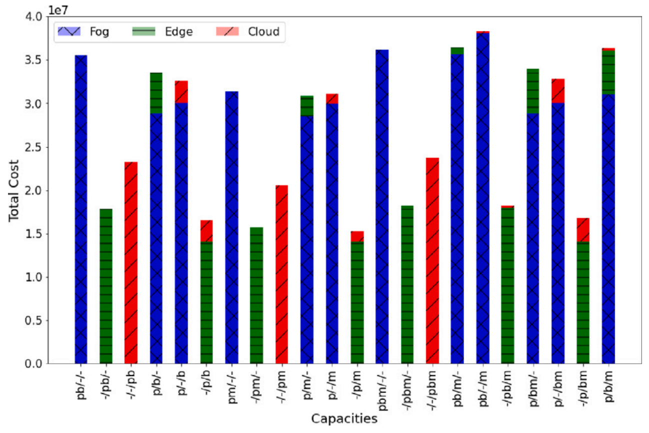

Fig. 8. Total cost of task assignment with delay threshold = 0.02 s.

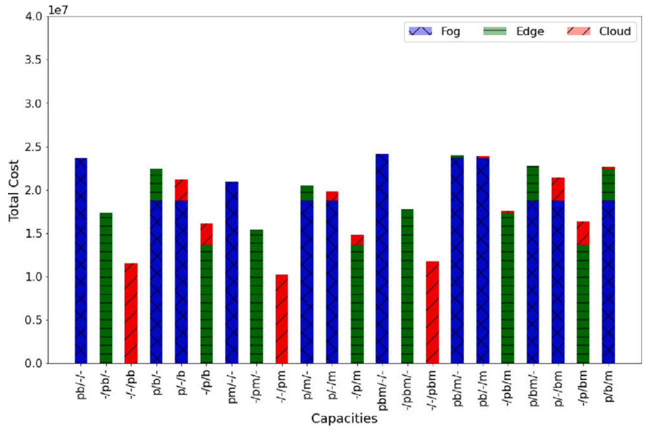

Fig. 9. Total cost of task assignment with delay threshold = 0.5 s.

and loose delay thresholds (0.5 s). In scenarios where a single-tier architecture is employed and the delay threshold is set tightly, utilizing the edge results in the lowest total cost (task assignment: –/pb/– and –/pm/–). This is because the edge provides the most suitable location and computation capacity compared to the fog and cloud. The fog, being the closest computation resource to the tenant, incurs a higher cost due to its highly distributed computation capacity. On the other hand, the cloud has lower computation costs but is unsuitable due to higher propagation delays.

However, as seen in Fig. 8, collaborating the edge with the cloud by splitting the IDS tasks and forming a two-tier architecture yields a better result, requiring a lower computation cost than utilizing only the edge. Balancing the workload to the higher tier provides more scalability, thereby reducing costs. Furthermore, employing more tiers in a three-tier architecture scenario does not offer advantages, as it increases fog computing costs and extends transmission time between tiers. As a result, the optimization algorithm incurs additional costs to meet the delay threshold.

Furthermore, in scenarios where the delay threshold is loose, as shown in Fig. 9, utilizing the cloud results in the lowest computation cost. This is attributed to the system's ability to tolerate the effects of propagation delay. Consequently, processing the entire workload in the cloud is the optimal strategy for reducing computation costs.

These results underscore the significance of selecting the appropriate architecture based on the characteristics of arrival traffic and the specified delay threshold. Service providers must continually adjust their configurations to minimize computation costs, emphasizing the need for adaptability in response to evolving network conditions and cost constraints. However, traditional optimization methods pose a challenge for service providers, as they necessitate longer optimization times whenever adjustments are needed.

While other traditional optimization algorithms could potentially be used, the performance differential between SA and other conventional methods may not be substantial. SA's robustness and depth of exploration provide unique benefits that are valuable in the comprehensive evaluation of IDS architectures.

In the following section, we delve into a performance comparison between utilizing traditional optimization (SA) and RL for optimizing capacity allocation to reduce computation costs, explaining why RL serves as a solution to address this challenge.

Traditional optimization vs. Reinforcement learning

In the last section, we utilized SA to optimize capacities for various task assignments, aiming to minimize costs under a fixed arrival rate. However, real-world scenarios often involve fluctuating arrival rates, necessitating different capacities to achieve cost efficiency. While simulated annealing can provide the global optimum, its slow convergence limits its applicability. We introduced RL in the control plane to address this, enabling us to make faster decisions and obtain satisfactory capacities. This approach allows for adaptive capacity adjustments, accommodating changing arrival rates and reducing decision times compared to traditional optimization algorithms.

The experimental results demonstrate that RL incurs approximately 10% higher costs compared to simulated annealing in terms of performance quality, as shown in Fig. 10. In terms of decision time, RL requires a mere 4.2×10^{-5} s, as shown in Fig. 11, whereas simulated annealing takes 2.3×10^{2} s, indicating a remarkable difference of 5×10^{6} times.

This significant discrepancy in decision time can be attributed to the operational mechanisms of RL and SA. SA relies on continuous recalculations of optimization parameters upon receiving updated arrival rates, leading to prolonged decision times. In contrast, RL, particularly the TD3 algorithm utilized in our study, leverages a pre-trained actor model to determine capacity allocation swiftly and efficiently. TD3 streamlines the decision-making process using this pre-trained model, resulting in remarkably faster optimization.

This distinct advantage of TD3 over SA underscores the effectiveness of RL approaches, particularly in scenarios where rapid and reliable optimization solutions are paramount. While SA remains a viable traditional optimization method, its inherent computational overhead limits its applicability in time-sensitive contexts, highlighting the significance of adopting the RL technique for optimizing tasks and capacities.

6. Conclusions and future works

Our study proposed an innovative solution named Auto-IDaS, which utilizes RL to optimize the IDaS system. This approach involves applying RL to select the most suitable IDS models intelligently, ensuring a balance between detection performance (F1 score), detection time, and computation costs. This methodology empowers IDaS to dynamically adapt to the constantly shifting IDS model based on network

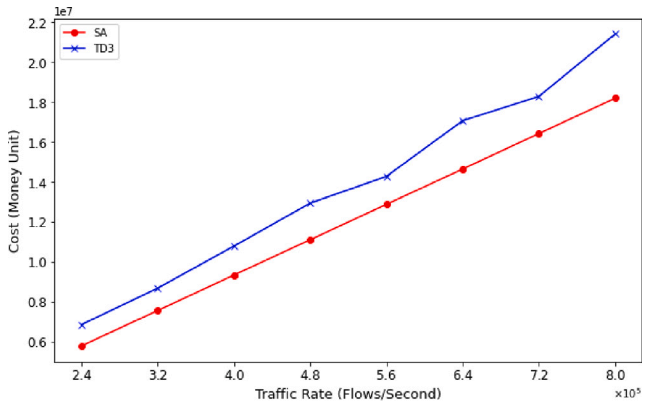

Fig. 10. Total cost with –/p/bm (SA vs. TD3).

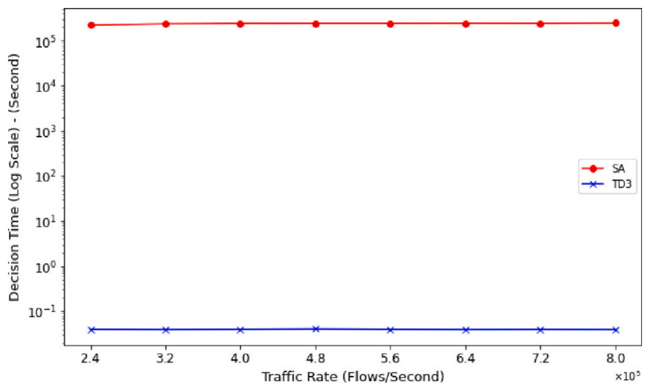

Fig. 11. Decision time with –/p/bm (SA vs. TD3).

traffic conditions, significantly enhancing the service's efficiency and effectiveness.

In addition, the paper addresses the challenge of optimizing cost allocation within the IDaS system. TD3, as the RL algorithm, is employed to minimize computation costs, diverging from traditional optimization techniques that often face extended convergence times. The system can make faster and more effective decisions by leveraging RL's dynamic adaptability to network changes and its ability to learn from ongoing behavior.

Our evaluations revealed that different IDS models, such as the Autoencoder and OCSVM, have unique strengths, with the former excelling in benign traffic scenarios and the latter in complex malicious situations.

A significant breakthrough was achieved by employing RL's DQN algorithm for dynamic model selection, which surpassed static models by achieving higher rewards ranging from 0.29% to 27.04%. This approach effectively harmonized crucial factors such as F1 score, detection time, and computation cost. In terms of capacity allocation, our comparison of SA and RL, specifically using the TD3 algorithm, showed that TD3 significantly accelerated decision times by about 5×10^6 times while maintaining decision quality close to SA's performance, within a 10% range.

Drawing from these findings, we recommend that IDaS providers focus on optimizing model selection and adapting to tenant traffic patterns. This is critical for achieving an effective balance of F1 score, detection time, and cost. Particularly, the adoption of TD3 for capacity allocation is advised due to its rapid decision-making capability and satisfactory decision quality, making it highly suitable for dynamic network environments.

Future research could focus on broadening the application of AI within IDaS, especially through the advancement of ML and RL algorithms. This could include exploring hybrid models that amalgamate various ML techniques to enhance detection precision and efficiency. There is also an opportunity to enrich the RL models used in Auto-IDaS by integrating them with cutting-edge RL algorithms, potentially refining the optimization of IDS model selection and resource allocation.

Additionally, the scalability and flexibility of the proposed RL-based approaches warrant further investigation, particularly in expansive and fluctuating network contexts. Addressing interpretability and explainability in ML and RL-driven systems represents a crucial research frontier, promoting transparency and building trust in automated decision-making within cybersecurity.

Moreover, evaluating the performance of the Auto-IDaS framework using live data streams is essential to confirm its efficacy in real-world settings. Finally, considering the rapid evolution of next-generation machine learning — such as integrating IDS with federated learning — there is a pressing need to analyze how Auto-IDaS manages increased complexities in more sophisticated environments.

CRediT authorship contribution statement

Ying-Dar Lin: Supervision, Resources, Funding acquisition, Conceptualization. **Hao-Xuan Huang:** Writing – original draft, Methodology, Investigation, Formal analysis. **Didik Sudyana:** Writing – review & editing, Visualization, Validation, Methodology. **Yuan-Cheng Lai:** Validation, Supervision, Resources, Conceptualization.

Declaration of competing interest

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

Data availability

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

References

Almiani, M., AbuGhazleh, A., Al-Rahayfeh, A., Atiewi, S., Razaque, A., 2020. Deep recurrent neural network for IoT intrusion detection system. Simul. Model. Pract. Theory 101, 102031.

Antonio, C., Souza, D., Becker, C., Bobsin, R., Bosco, J., Sobral, M., Vieira, S., 2020. Hybrid approach to intrusion detection in fog-based IoT environments. Comput. Netw. 180, 107417.

Bertsimas, D., Tsitsiklis, J., 1993. Simulated annealing. Stat. Sci. 8 (1), 10–15.

Bierzynski, K., Escobar, A., Eberl, M., 2017. Cloud, fog and edge: Cooperation for the future? In: 2017 Second International Conference on Fog and Mobile Edge Computing. FMEC, IEEE, pp. 62–67.

Bovenzi, G., Aceto, G., Ciuonzo, D., Persico, V., Pescapé, A., 2020. A hierarchical hybrid intrusion detection approach in IoT scenarios. In: GLOBECOM 2020-2020 IEEE Global Communications Conference. IEEE, pp. 1–7.

Cil, A.E., Yildiz, K., Buldu, A., 2021. Detection of DDoS attacks with feed forward based deep neural network model. Expert Syst. Appl. 169, 114520, [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2020.114520.

De Souza, C.A., Westphall, C.B., Machado, R.B., Sobral, J.B.M., dos Santos Vieira, G., 2020. Hybrid approach to intrusion detection in fog-based IoT environments. Comput. Netw. 180, 107417.

Diro, A., Chilamkurti, N., 2018. Leveraging LSTM networks for attack detection in fog-to-things communications. IEEE Commun. Mag. 56 (9), 124–130.

Fujimoto, S., Hoof, H., Meger, D., 2018. Addressing function approximation error in actor-critic methods. In: International Conference on Machine Learning. PMLR, pp. 1587–1596.

Holm, H., 2014. Signature based intrusion detection for zero-day attacks:(not) a closed chapter? In: 2014 47th Hawaii International Conference on System Sciences. IEEE, pp. 4895–4904.

Idhammad, M., Afdel, K., Belouch, M., 2018. Distributed intrusion detection system for cloud environments based on data mining techniques. Procedia Comput. Sci. 127, 35–41.

Jin, H., Gregory, M.A., Li, S., 2022. A review of intelligent computation offloading in multiaccess edge computing. IEEE Access 10, 71481–71495.

Kar, B., Yahya, W., Lin, Y.-D., Ali, A., 2023. Offloading using traditional optimization and machine learning in federated cloud–edge–fog systems: A survey. IEEE Commun. Surv. Tutor. 25 (2), 1199–1226.

Labiod, Y., Amara Korba, A., Ghoualmi, N., 2022. Fog computing-based intrusion detection architecture to protect IoT networks. Wirel. Pers. Commun..
Lai, Y.-C., Sudyana, D., Lin, Y.-D., Verkerken, M., D'hooge, L., Wauters, T., Volckaert, B., De Turck, F., 2021. Machine learning based intrusion detection as a service: task assignment and capacity allocation in a multi-tier architecture. In: Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion. pp. 1–6.
Lai, Y.-C., Sudyana, D., Lin, Y.-D., Verkerken, M., D'hooge, L., Wauters, T., Volckaert, B., De Turck, F., 2023. Task assignment and capacity allocation for ML-based intrusion detection as a service in a multi-tier architecture. IEEE Trans. Netw. Serv. Manag. 20 (1), 672–683.
Mahjoubi, A., Grinnemo, K.-J., Taheri, J., 2022. An efficient simulated annealing-based task scheduling technique for task offloading in a mobile edge architecture. In: 2022 IEEE 11th International Conference on Cloud Networking. CloudNet, pp. 159–167.
Masdari, M., Khezri, H., 2020. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. Appl. Soft Comput. 92, 106301.
Megyesi, P., Molnár, S., 2013. Analysis of elephant users in broadband network traffic. In: Advances in Communication Networking: 19th EUNICE/IFIP WG 6.6 International Workshop, Chemnitz, Germany, August 28-30, 2013. Proceedings 19. Springer, pp. 37–45.
Nandanwar, H., Katarya, R., 2024. Deep learning enabled intrusion detection system for industrial IOT environment. Expert Syst. Appl. 249, 123808, [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2024.123808.
Nassif, A.B., Talib, M.A., Nasir, Q., Dakalbab, F.M., 2021. Machine learning for anomaly detection: A systematic review. IEEE Access 9, 78658–78700.
Omar, S., Ngadi, A., Jebur, H.H., 2013. Machine learning techniques for anomaly detection: an overview. Int. J. Comput. Appl. 79 (2).
Qu, G., Wu, H., Li, R., Jiao, P., 2021. DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing. IEEE Trans. Netw. Serv. Manag. 18 (3), 3448–3459.
Selamnia, A., Brik, B., Senouci, S.M., Boualouache, A., Hossain, S., 2022. Edge computing-enabled intrusion detection for C-V2X networks using federated learning. In: GLOBECOM 2022 - 2022 IEEE Global Communications Conference. pp. 2080–2085.
Sundararaj, V., 2019. Optimal task assignment in mobile cloud computing by queue based ant-bee algorithm. Wirel. Pers. Commun. 104, 173–197.
Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: an Introduction. MIT Press.
Syed, N.F., Ge, M., Baig, Z., 2023. Fog-cloud based intrusion detection system using recurrent neural networks and feature selection for IoT networks. Comput. Netw. 225, 109662.
Thai, M.-T., Lin, Y.-D., Lai, Y.-C., Chien, H.-T., 2019. Workload and capacity optimization for cloud-edge computing systems with vertical and horizontal offloading. IEEE Trans. Netw. Serv. Manag. 17 (1), 227–238.
Verkerken, M., D'hooge, L., Sudyana, D., Lin, Y.-D., Wauters, T., Volckaert, B., De Turck, F., 2023. A novel multi-stage approach for hierarchical intrusion detection. IEEE Trans. Netw. Serv. Manag..
Xu, H., Sun, Z., Cao, Y., Bilal, H., 2023. A data-driven approach for intrusion and anomaly detection using automated machine learning for the internet of things. Soft Comput. 27 (19), 14469–14481, [Online]. Available: http://dx.doi.org/10.1007/s00500-023-09037-4.
Zeng, D., Gu, L., Pan, S., Cai, J., Guo, S., 2019. Resource management at the network edge: A deep reinforcement learning approach. IEEE Netw. 33 (3), 26–33.
Zhang, Y., Yao, J., Guan, H., 2017. Intelligent cloud resource management with deep reinforcement learning. IEEE Cloud Comput. 4 (6), 60–69.
Zhu, S., Ota, K., Dong, M., 2022. Energy-efficient artificial intelligence of things with intelligent edge. IEEE Internet Things J. 9 (10), 7525–7532.

Ying-Dar Lin is a Chair Professor of computer science at National Yang Ming Chiao Tung University (NYCU), Taiwan. He received his Ph.D. in computer science from the University of California at Los Angeles (UCLA) in 1993. He was a visiting scholar at Cisco Systems in San Jose during 2007–2008, CEO at Telecom Technology Center, Taiwan, during 2010–2011, and Vice President of National Applied Research Labs (NARLabs), Taiwan, during 2017–2018. He cofounded L7 Networks Inc. in 2002 and O'Prueba Inc. in 2018. His research interests include cybersecurity, wireless communications, network softwarization, and machine learning for communications. He is an IEEE Fellow (class of 2013). He has served or is serving on the editorial boards of several IEEE journals and magazines, including Editor-in-Chief of IEEE Communications Surveys and Tutorials (COMST, 2017–2020).

Hao-Xuan Huang received his M.S. degree in the program of cybersecurity from National Yang Ming Chiao Tung University (NYCU), Hsinchu, Taiwan, in 2023. He was an associate researcher at High Speed Network Lab, NYCU, in 2021–2023. His research interests include cybersecurity and the application of machine learning in security contexts.

Didik Sudyana received his Ph.D. degree in the Department of Electrical Engineering and Computer Science (EECS) from National Yang Ming Chiao Tung University (NYCU) in 2024. He is currently a research assistant with the High-Speed Network Lab at NYCU. His research interests include cybersecurity, machine learning, and network design and optimization.

Yuan-Cheng Lai received his Ph.D. degree in the Department of Computer and Information Science from National Chiao Tung University in 1997. He joined the faculty of the Department of Information Management at National Taiwan University of Science and Technology in August 2001 and has been a distinguished professor since June 2012. His research interests include performance analysis, software-defined networking, wireless networks, and IoT security.