International Journal of Computers and Applications

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/tjca20

Evaluating the cost of classifier discrimination choices for IoT sensor attack detection

Mathew Nicho, Brian Cusack, Shini Girija & Nalin Arachchilage

To cite this article: Mathew Nicho, Brian Cusack, Shini Girija & Nalin Arachchilage (10 Sep 2024): Evaluating the cost of classifier discrimination choices for IoT sensor attack detection, International Journal of Computers and Applications, DOI: 10.1080/1206212X.2024.2401069

To link to this article: https://doi.org/10.1080/1206212X.2024.2401069

Published online: 10 Sep 2024.

INTERNATIONAL JOURNAL OF COMPUTERS AND APPLICATIONS
https://doi.org/10.1080/1206212X.2024.2401069

Taylor & Francis
Taylor & Francis Group

Evaluating the cost of classifier discrimination choices for IoT sensor attack detection

Mathew Nicho[a], Brian Cusack[b], Shini Girija[c] and Nalin Arachchilage[d]

[a]Research and Innovation Center, Rabdan Academy, Abu Dhabi, UAE; [b]Cloud Security Research Center, AUT University, Auckland, New Zealand; [c]College of Technological Innovation, Zayed University, Dubai, UAE; [d]School of Computing Technologies, RMIT University, Melbourne, Australia

ABSTRACT

The intrusion detection of IoT devices through the classification of malicious traffic packets have become more complex and resource intensive as algorithm design and the scope of the problems have changed. In this research, we compare the cost of a traditional supervised pattern recognition algorithm (k-Nearest Neighbor (KNN)), with the cost of a current deep learning (DL) unsupervised algorithm (Convolutional Neural Network (CNN)) in their simplest forms. The classifier costs are calculated based on the attributes of design, computation, scope, training, use, and retirement. We find that the DL algorithm is applicable to a wider range of problem-solving tasks, but it costs more to implement and operate than a traditional classifier. This research proposes an economic classifier model for deploying suitable AI-based intrusion detection classifiers in IoT environments. The model was empirically validated on the IoT-23 dataset using KNN and CNN. This study closes a gap in prior research that mostly concentrated on technical elements by incorporating economic factors into the evaluation of AI algorithms for IoT intrusion detection. This research thus evaluated the economic implications of deploying AI-based intrusion detection systems in IoT environments, considering performance metrics, implementation costs, and the cost of classifier discrimination choices. Researchers and practitioners should focus on the cost–benefit trade-offs of any artificial intelligence application for intrusion detection, recommending an economic evaluation and task fit assessment before adopting automated solutions or classifiers for IoT intrusion detection, particularly in large-scale industrial settings that involve active attacks.

ARTICLE HISTORY
Received 27 May 2024
Accepted 31 August 2024

KEYWORDS
Internet of things; classifier costs; cyber security; cost evaluation; pattern recognition; threat detection

1. Introduction

The adoption and application of IoT devices is projected to double from 15.9 billion in 2023–21 billion in 2030, while during the same period, attacks increased by 41% in 2023 alone [1,2]. IoT devices have been classified as low-end, medium-end, and high-end devices. Low-end and medium-end devices, which use low processing power, often have weaker security layers [3]. With existing IoT literature mainly focusing on application domains, services, cloud and business solutions, IoT technologies, and security [3], the proposed research focuses on implementation cost, total cost, and algorithm performance of classifiers in intrusion detection to optimize detection processing. This assumes significance since critical industry verticals namely electricity, gas, steam & A/C, water supply & waste management, retail & wholesale, transportation & storage, and government have currently more than 100 million connected IoT devices [4].

As a computing system, the IoT possesses inherent potential information security vulnerabilities and associated negative risk impacts [5]. IoT architectures are susceptible to attacks since they encompass peripheral sensors, connected devices, and an IoT platform consisting of gateways, servers, applications, and other internetwork connectivity elements, including satellite systems. Consequently, vulnerabilities can arise during network transmission, authentication and key exchange, and system command-and-control processes. IoT networks utilize sensor data for processing, decision-making, and initiating actions [6]. Sensors, despite their small size and large data-transmission capabilities, represent the most vulnerable components. Although lightweight encryption algorithms can offer protection, the limited computational resources of sensors restrict the extent of this protection [7,8]. Therefore, automated solutions for detecting malicious IoT traffic have been explored by various researchers [9–12]. Each solution presents a trade-off between costs and performance benefits, thereby posing the ongoing challenge of determining the most effective solution for specific use cases. Effective classification algorithms are necessary because IoT devices have limited computing capability [13,14]. The Cost of Classifier Discrimination Choices is important because it establishes the computational resources required, thereby affecting the choice of algorithms to balance resource efficiency and accuracy for IoT applications.

The problem of accurately categorizing observed IoT network traffic packets as either beneficial or malicious necessitates the development of a classification mechanism capable of effectively distinguishing class membership while maintaining acceptable costs [15]. Consequently, researchers face the challenge of devising cost-efficient algorithms that operate within acceptable risk parameters. Four potential outcomes can result from applying a classifier to a dataset: correct identification of a packet, incorrect identification of a packet, false classification of a beneficial packet, or false classification of a malicious packet [16]. By employing standardized metrics (Section 3.2), the performance of a classifier can be assessed, and its predicted utility determined. There are various aspects of the cost of implementing an algorithm, including performance as well as other dimensions of constraints and impacts. In this study, we evaluate and test each algorithm based on such cost attributes as design, computation, scope, training, usage, and retirement (Table 1). Furthermore, additional costs – such as performance failure, access, and assistance – that contribute to algorithm utilization are addressed in the discussion in Section 5.

CONTACT Shini Girija shini.girija@zu.ac.ae College of Technological Innovation, Zayed University, Dubai, UAE

Table 1. Cost attributes of the tested algorithms.

Cost	KNN	CNN
Design	Simple	Complex
Computation	Scales on dataset size and k-value	Scales by layers and design complexity
Scope	Supervised	Unsupervised
Training	None or little	Many runs
Use	Requires management	No management once trained
Retirement	Turn off and erase data	Turn off and erase layer contents

The two algorithms selected for this study represent extreme cases, with one focusing on nearest neighbor similarities and the other involving multiple layers of relationships. These choices are intentional, for one algorithm exhibits well-known limitations while the other is in the developmental stage and the subject of ongoing research projects. A convolutional neural network (CNN), a deep learning (DL) network architecture that learns directly from data, is often employed in contemporary application research [14]. CNNs are adept at identifying patterns in images to discern objects, classes, and categories and have proved effective in classifying audio, time-series, and signal data [17].

The K-nearest neighbor (KNN) algorithm, a traditional statistical metric approach to classification, performs optimally when data have been normalized and labeled [18]. As a supervised algorithm, much of the feature selection and data normalization can be delegated to supervisor actions, enabling rapid implementation and efficient discrimination. However, the drawbacks of the algorithm include efficiency limitations on large unclassified datasets and difficulties in determining the optimal k constant [19,20]. The sections of this study are organized into a review of the theoretical context and research methods. We report and discuss the experimental results to elucidate the learning process for making cost-based decisions among algorithms.

IoT ecosystems have essential and constantly changing features that may introduce new vulnerabilities and risks [21]. Attacks that target various levels of IoT architecture are frequently the result of these vulnerabilities. IoT device attacks can be classified as either passive or active depending on the attacker's actions [22]. Passive attackers can be difficult to identify since they pose as legitimate users to listen in on communication channels and steal data. In active attacks, which are readily identifiable, a hacker prevents the network from operating normally. Since active attacks are complicated and resource-intensive, they demand significantly more processing power than passive attacks since they manipulate data or inject data into communication streams [23]. For instance, man-in-the-middle (MitM) attacks necessitate real-time data interception, decryption, modification, and re-encryption by attackers, all of which involve considerable computing power [24]. In a similar vein, distributed denial-of-service (DDoS) attacks involve the coordination of multiple infected devices to overwhelm a target system with traffic, likewise requiring significant computational power [25]. This study focuses on identifying the ways in which active attacks use processing power, thereby highlighting the vital need for strong security measures and sophisticated processing power to recognize, mitigate, and respond to the increasingly complex and resource-intensive threats in large scale IoT ecosystems.

Although techniques based on machine learning (ML) and artificial intelligence (AI) have excellent detection accuracy, previous research has revealed the drawback that their assessments do not consider the social and, in particular, economic consequences of advances in the field of AI [26–29]. Therefore, the novelty of this research lies in the inclusion of economic factors in the assessment of the use of AI algorithms in IoT intrusion detection [27–29]. The economic factors, performance measures, advantages, and disadvantages of each approach are considered when evaluating the cost of the classifier discrimination choices for detecting IoT sensor attacks [27]. Researchers have recommended assessing the economic value of AI algorithms regarding their performance and implementation costs when adopting intrusion-detection systems in IoT environments to ensure that they are both effective and affordable for enterprises [30,31]. Therefore, in this paper, we address the often-overlooked economic consequences of algorithm selection [27,28], for the previous research has tended to focus on the technical elements of intrusion detection systems [27,32–34].

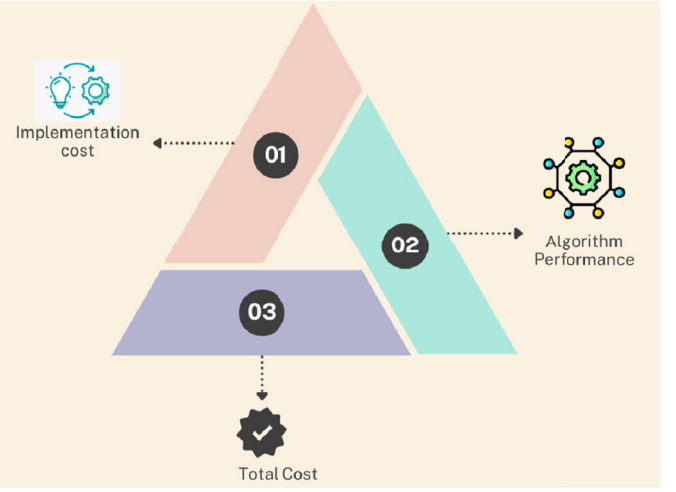

Figure 1. Dimensions of cost.

The research presented here sheds light on the intricate relationship between the performance of algorithms, the implementation costs, and the possible consequences of undetected intrusions on IoT systems by utilizing a total cost of ownership (TCO) (Figure 1). The trade-offs involved in algorithm selection are clarified by contrasting supervised and unsupervised algorithms and examining thoroughly the design factors and usability costs. Additionally, our examination of the cost of failure in terms of false negatives emphasizes the importance of reliable intrusion detection systems for the security of IoT networks. This study thus provides an innovative perspective that can guide decision-making for enterprises looking to implement intrusion detection systems in IoT environments that are both effective and affordable by highlighting the economic value of AI algorithms in relation to their performance and implementation costs. Accordingly, this study addresses the research question:

> How does the DL unsupervised algorithm for IoT intrusion detection compare in terms of cost and performance with a typical supervised pattern recognition algorithm, and how do the trade-offs affect the utility and application of these algorithms in real-world situations?

1.1. Research contributions

The primary objective of this research is to compare and analyse systematically the costs and performance trade-offs of a traditional supervised pattern recognition algorithm (KNN) and a DL unsupervised algorithm (CNN) when these algorithms are used to detect IoT intrusion. This analysis focuses on the cost attributes of design, computation, scope, training, usage, and retirement. Utilizing a dataset of malicious and benign traffic from IoT devices, we offer insights into the practical implications of employing each algorithm. Future work on this topic will involve replicating our research using datasets from IoT ecosystem and lightweight classifiers. In this way, researchers can investigate the applicability of the findings presented here to

other datasets and evaluate the reliability of various classification algorithms in detecting IoT attacks.

The study thus makes the following contributions to the literature.

(1) We propose an economic classifier model for deploying suitable AI-based intrusion detection classifiers in IoT environments, empirically validated using KNN and CNN on the IoT-23 dataset. This model can be applied to any type of IoT sensor, for evaluating and deploying the most cost-effective classifiers across diverse IoT environments, particularly in industrial settings where cost-efficiency is crucial due to the large number of sensors deployed.
(2) We present a comprehensive comparison of the costs involved in implementing and maintaining a traditional supervised pattern recognition algorithm (KNN) with the costs of a DL unsupervised algorithm (CNN) for the purpose of IoT intrusion detection. The lightweight structure of these algorithms makes them appropriate for IoT applications with limited resources and provides unique methods for intrusion detection. The cost attributes considered include design, computation, scope, training, usage, and retirement.
(3) We utilize a IoT-23 dataset of malicious and benign traffic collected from IoT devices to evaluate the performance and computational costs of the KNN and CNN classifiers. This practical assessment clarifies the trade-offs involved in employing each algorithm. The selection of the IoT-23 dataset for this study was based on its extensive collection of benign and malicious traffic from real-world IoT devices as well as its widespread use in the research community.
(4) We emphasize the importance of economic evaluation and assessing task fit before adopting automated solutions for IoT intrusion detection to ensure that the selected method (methods) is (are) cost-effective and appropriate for the identified case.
(5) By analysing the cost attributes of the algorithms, we facilitate informed decision-making in organizations that are considering the implementation of automated solutions for detecting intrusions in IoT sensors and associated systems.
(6) The findings presented here serve to alert stakeholders in the cybersecurity and IoT industries to the trade-off in terms of costs and benefits when using AI applications to detect intrusion and, thus, can inform future research and development efforts in this field.

1.2. Structure of paper

In what follows, we present in Section 2 an outline of the theoretical background for the study. In Section 3, we describe the methods and metrics used in the research. In Section 4, we present the results of our experiments. In Section 5, we discuss our results in detail, and we summarize our conclusions in Section 6.

2. Theoretical background

IoT intrusion detection research is characterized by the need to classify large volumes of data and the drive to develop automated solutions. As a result, the standard statistical techniques for processing big data and supervised learning approaches often fail to provide effective solution. Novel capabilities in ML are necessary to ensure problem adaptation and efficiency in autonomous decision-making. The recent research on DL has included a proposal to employ neural network technologies as a solution for automating IoT intrusion detection and addressing the associated challenges [12,35]. In the following discussion, we present a background literature review on IoT security attacks, classification algorithms, and the cost considerations relating to algorithm ownership.

2.1. IoT security attacks

Cybercriminals exploit vulnerabilities in IoT devices to gain remote access and launch attacks to acquire valuable information. In [10], the authors distinguished nine broad categories of IoT vulnerabilities: lack of physical security, constrained energy capacity, weak encryption, vulnerable authentication, access control violations, obsolescent software, open ports, programming errors, and inadequate auditing. In [36], the authors classified IoT vulnerabilities based on three architectural features: internal systems, interfaces, and shared architecture. Internal system vulnerabilities are targeted by ransomware and phishing attacks while interfaces are prone to cross-site scripting attacks and malicious remote access. Unauthorized access to user information and malicious user account generation compromise both interfaces and internal systems [23,37].

IoT security vulnerabilities can also be categorized according to their components as platform software/firmware vulnerabilities, network vulnerabilities, gateway vulnerabilities, and people, policy, and procedure vulnerabilities [38]. Platform software and firmware vulnerabilities arise from flaws introduced during the design, development, and deployment of IoT software or firmware that allow cybercriminals to infiltrate the IoT network and make unauthorized changes [39]. Network vulnerabilities result from policy and procedure violations, and gateway vulnerabilities result from malicious modifications to any gateway connected to the IoT. People, policy, and procedure vulnerabilities are caused by unskilled or inadequately trained users [40].

IoT device attacks can be categorized based on the IoT architecture layers of perception, network, and application [41]. The perception layer encompasses the actuators and sensors responsible for generating and transmitting data to the network layer. Primary attacks targeting this layer include physical attacks aimed at disrupting physical devices and impersonation attacks involving fake identities or connection points [42]. The network layer entails the interconnection of IoT devices and the transfer of data from the perception layer to the application layer. Prominent attacks on the application layer include MitM and routing attacks, such as sinkhole attacks and selective forward attacks. The application layer hosts services for IoT applications and is also susceptible to attacks related to software changes, which can introduce new vulnerabilities that can allow for the injection of malicious code or leakage of data [36].

IoT attacks can also be classified as either intrinsic or extrinsic. Intrinsic attacks originate internally and can be mitigated through policy enforcement and intrusion detection systems. Extrinsic attacks target a system's firewall externally, thus posing a challenge to the security of the entire IoT system and may involve malicious nodes or compromised sensors [43].

IoT attacks can also be categorized based on whether the attacker's activity is passive or active [44]. Passive attackers masquerade as legitimate users and eavesdrop on communication channels to acquire data, so they are difficult to detect. Various types of passive attacks are outlined in Table 2, in which IoT entities are denoted numerically as 1 – IoT device, 2 – IoT device developer, 3 – IoT platform, 4 – IoT system, 5 – IoT solution, and 6 – IoT ecosystem.

Table 3 summarizes the vulnerabilities of IoT networks to active attacks across the categories of IoT entities.

2.2. Classification algorithms

Classification algorithms assign observations to classes or categories. The simplest classifiers distinguish between two categories or a

Table 2. Passive IoT attacks.

Attacks	IoT Entity	Description
Eavesdropping	1,2,4	Gathering of information by tapping network communication lines [35].
Node Destruction	1,3,4,6	Attempts to damage nodes through an electric overflow or physical force [42].
Node Malfunctioning	1,2, 3,4	Improper functioning of nodes resulting from DDoS attacks or overwhelmed sensors [39].
Node Outage	1,3,4,6	Disruption of the functionality of sensor nodes [44].
Traffic analysis	2,4,6	Gaining network topology information through traffic pattern analysis. Typically, DDoS attacks are used to access this information [43].

Table 3. Active IoT attacks.

Attacks	IoT Entity	Description
Jamming	1,4	An intercepted device can jam a signal by transmitting at a similar frequency. Jamming can be performed at random intervals or continuously to hamper the smooth flow of data [36].
Tampering	1,4	Tampering with physical devices can help attackers generate fake queries to confuse legitimate users with false data [41].
Collision	1,2, 3,4	A transmission is sent on the same channel on which a legitimate user is finishing a transmission. The transmissions collide, and the receiver asks that the data be transmitted again [43].
Denial of sleep	1,2,4,6	The energy of battery-powered devices can be drained by collision attacks or recurring handshakes [45].
Spoofing	1,2,4	An intercepted node spoofs the physical address of a victim node and then generates fake identities for it and uses them elsewhere in the network [46].
Flooding attacks	3,4,6	Intruders broadcast a message throughout the network and convince other nodes that is in the vicinity [44].
Wormhole attacks	1,2, 4	A sinkhole attracts all traffic to itself and does not drop any packets. Sinkholes achieved through the cooperation of two nodes that use a different fast channel than the network itself are termed wormholes [46].
Node Replication	1,2,3,4	An adversary places duplicates of victim nodes in many places on a network to generate inconsistencies [47].
Sybil attacks	1,2, 3,4, 6	Compromised nodes present multiple identities to the other nodes in a network, thereby contradicting the routing paths and reducing the effectiveness of fault tolerance schemes [44].
DDoS attacks	1,3,4	Attempts to shut down the targeted node partially or completely by flooding it with traffic, thereby interrupting regular traffic to the victim network [9].
Phishing attacks	1,2,4	An adversary impersonates a legitimate user to gain sensitive information such as bank account details or credit card information. Usually, these attacks are launched with an email that, when opened, causes the leakage of information [47].

binary while the more complex assign observations to multiple classes [48]. The decision to assign an observation is made based on predetermined criteria and rules, and performance metrics serve to assess the algorithm's success. Researchers choose algorithms based on their previous successful performance in similar contexts and on similar problems to ensure that they are suited to their research problems [49]. In the following discussion, we review six algorithms that reflect the evolution of the statistical methods used for classification, from linear modeling to learning and CNNs.

Regression analysis is the fundamental basis for examining variable relationships and constructing predictive models, which involve the supervised classification of dependent variables in relation to independent variables. Typically, a regression is not perceived as a classification algorithm; rather, it underpins the mathematical concepts essential for the processes of discrimination, grouping, and classification [50]. The certainty of a relationship is often insufficient to include or exclude a variable definitively from a particular class. Nevertheless, the application of clustering and grouping methodologies to regression data enables the establishment of a model for differentiating among distinct classes of information. Logistic regression is a notable example of a technique used to predict a categorical dependent variable based on a set of independent factors.

Decision trees are widely used to classify and group patterns in data. A decision tree consists of a hierarchical arrangement of questions shaped like a tree, with each question representing a condition and generating a split that leads to other questions [48]. The concept of a tree is extended with nodes and leaves that represent conditions and predictions, respectively. In this manner, an investigation progresses through the tree structure, which conditions a problem for solving and creates predictions as answers. The decision tree classification structure is commonly used in ML problem-solving. The key challenge is to develop a series of relevant questions that sequentially lead to a required outcome. The decision trees from one or multiple learning models can be combined to obtain a generalized outcome through an average or majority vote.

The naive Bayes classifier is a probabilistic classifier that estimates the probability of an observation belonging to a particular category or group [51]. This classifier uses chance or probability scales to make predictions quickly. The algorithm stores the features of an object independently so that, when a feature or combination of features is detected, it predicts the object's category. It is based on the Bayes conditional probability theorem and provides error rates for each prediction. This supervised learning algorithm thus predicts the probability of a hypothesis being true without prior knowledge. To make a classification decision, any dataset can be transformed into a frequency table and a likelihood table, and the posterior probability is calculated. This classifier can be applied to both binary and multi-class classifications. One of its limitations is that it assumes all features to be independent or unrelated, so it may result in a failure to learn the relationships between features.

The KNN method references all existing examples in a dataset (n in size) and categorizes new examples using a similarity metric (e.g. distance functions). Since the early 1970s, KNN has been used as a non-parametric approach in statistical estimates and pattern recognition [48]. The method depends on the similarity comparison mechanism to drive decision-making and only stores instances of the training data. Classification is determined by a simple majority vote of each k closest neighbor, with new examples allocated to the class with the most members among such neighbors as determined by a distance function [20]. When k equals 1, the new example is allocated to the nearest neighbor's class whereas, if k is greater than 1, the computational load increases linearly until k equals n (the total entries in the dataset). The challenge in using KNN is to select or estimate the optimal k that balances the computational cost against the accuracy of the predictions.

Artificial neural networks (ANNs) are supervised learning networks used for classification problems [52]. Clustering, on the other hand, is an unsupervised ML approach for categorizing data points based on their traits and characteristics. Clustering is a statistical technique while ANNs consist of various patterning methodologies [53]. ANNs are designed to resemble the structure of the brain, with

each artificial neuron connecting to many others and millions of neurons working together to form a complex cognitive structure. The structure of neural networks is multi-layered, with the neurons in one layer transferring data to multiple neurons in the next layer and so on. The data eventually reach the output layer, where the network decides how to handle a problem, categorize an object, and so on [54]. The study of neural networks is characterized as DL because its multi-layer structure is supervised and learns acceptable behaviors progressively. A CNN retains the ANN structure of the input, hidden, and output layers but is a class of ANN that reduces the complexity of the neural connections between layers [15]. The input is a matrix image representation which can be represented as (number of inputs) × (input height) × (input width) × (input channels). After passing through a convolutional layer, an image is transformed into a feature map or activation map, and the number of required neuron connections to the next layer decreases. This feature map serves as the input matrix for the subsequent layers, such as the pooling, fully connected, and normalization layers [16]. CNNs solve the issues of overfitting and computational complexity through the reuse of filters and the use of dilatable receptive input fields between the hidden layers.

Classification algorithms play a crucial role in assigning observations to classes or categories, with their suitability depending on the research problem at hand. Here, we explore six prominent classification algorithms: regression analysis, decision trees, naive Bayes classifiers, KNNs, and ANNs, specifically, CNNs. Since each ML algorithm is unique, with its own strengths and weaknesses, selecting the most appropriate entails carefully considering the specific context and requirements.

Because IoT environments have inherent resource limits, lightweight classification methods are essential for the detection of IoT intrusions. We selected CNN (an unsupervised deep learning technique) and KNN (a conventional supervised pattern recognition algorithm) because they function well in resource-constrained environments. As IoT devices proliferate, lightweight algorithms are increasingly necessary to enable effective data processing and accurate intrusion detection capabilities. The intention of the research described here, then, is to resolve issues specific to IoT environments and contribute to the development of efficient intrusion detection systems for IoT devices by utilizing lightweight techniques such as CNN and KNN.

2.3. Computational costs

The total cost of ownership (TCO) is a comprehensive estimation of the expenses related to the acquisition, deployment, utilization, and retirement of a product or piece of equipment [55]. The TCO is demonstrated by the economic classifier model. This concept is also applicable to software applications and algorithms implemented for work purposes. The TCO thus includes the acquisition, operation, and retirement costs of any asset, with the value of software and algorithms evaluated and reported with respect to the functional attributes of the services that they provide or facilitate. Table 4 illustrates the TCO for an algorithm. In addition to computational costs, the dimensions of an algorithm's TCO include design, scope of use, challenges for use, and retirement opportunities [56]. The design influences the ease of use, functionality, and problem-solving scope.

The model illustrated in Table 4 outlines the phases, attributes, and metrics for evaluating the total cost of ownership (TCO) for algorithms used in IoT intrusion detection through simulation. It provides a useful model that can be replicated by industries to select the optimal classifier for appropriate IoT types as it breaks down the costs related to the acquisition, operation, and retirement phases through considering the attributes and metrics. Organizations can thus make better decisions by using this method to assess and contrast the actual costs of implementing various algorithms in their setting.

Table 4. Economic classifier model for owning an algorithm.

Phase	Attribute	Metric
Acquire	Procure, Availability, Fit assessment	Finance, Time, Method execution
Operate	Implement, Test, Train, Run	Time, Expert knowledge, Computing resources, Success rates
Retire	Data erasure, Decommission, Reuse market potential assessment	Time, Expert knowledge

The KNN and CNN algorithms chosen for this research have distinct designs but are applicable to network traffic classification. KNN features a simple geometric design that enables its application across various discrimination tasks whereas CNN possesses a complex design, so users must calculate the number of layers and nodes to achieve the desired outcome [57]. Additionally, CNN requires 3D-format visual input data for computation. These factors contribute to or reduce the costs of use. CNN entails training costs unless a pretrained CNN is acquired and customized for the required task [58]. Consequently, we select as attributes the design, computation, scope, training, use, and retirement of the algorithms to provide adequate scope for comparing the costs between them.

CNN and KNN were chosen for this study because of their complementing advantages in handling the challenges associated with IoT intrusion detection. CNNs excel at detecting intricate patterns in unstructured data, which is crucial for identifying small anomalies within the large volumes of diverse network traffic commonly found in IoT ecosystems [59]. They are useful for recognizing forms of attacks that have never been seen before because of their capacity to function in unsupervised learning environments [60]. However, KNN is an effective benchmark for comparison due to its ease of use, low computational cost, and interpretability, particularly in resource-constrained IoT devices where real-time detection is crucial [61]. The study focuses on examining the trade-offs between these two algorithms, which represent different extremes of the complexity spectrum, in terms of cost and performance. Other classifiers, like ensemble approaches or support vector machines, were taken into consideration but eventually discarded because of their increased computational overhead, propensity for overfitting, and added complexity – factors that might not be well suited to the dynamic and limited environments of IoT systems [19]. Thus, CNN and KNN were selected to offer a thorough, unbiased analysis of how algorithmic complexity affects both detection accuracy and operating costs, which is exactly in line with the study's goals of creating an intrusion detection system that is both cost-effective and widely applicable.

2.4. State of the art

Table 5 presents a comparison of the proposed approach with the existing IDS approaches. The economic factors, performance measures, advantages, and disadvantages of each approach are considered when evaluating it. Although ML – and AI-based techniques show excellent detection accuracy, their assessments lack economic consequences. DL-based algorithms are quite accurate, but they ignore design considerations and usability costs. Our approach, by contrast, provides a thorough evaluation of the performance, design elements, and usability costs since it takes economic considerations into account and uses a TCO paradigm. This table is useful for understanding the various IDS approaches and the trade-offs

Table 5. Comparison with related research.

IDS Method	Economical	Performance	Strengths	Limitations
ML-based algorithms [29]	No	Precision, recall, F1 score, AUC	Good detection accuracy (95%)	Economic implications not considered
DL-based algorithms [30]	No	Precision, recall, accuracy	High detection accuracy (99.75%)	Usability cost and design factors not considered
AI-based detection [31]	No	Precision, recall, accuracy, time	High detection accuracy (99.45%); execution time also considered	Economic implications not considered
Our Method	Yes	Precision, recall, accuracy, F1 score, TOC Model	Takes into account the performance, design factors, and usability costs of algorithms in addition to comparing costs (99.0%, 99.9%)	Conducted using a limited number of PCAP files

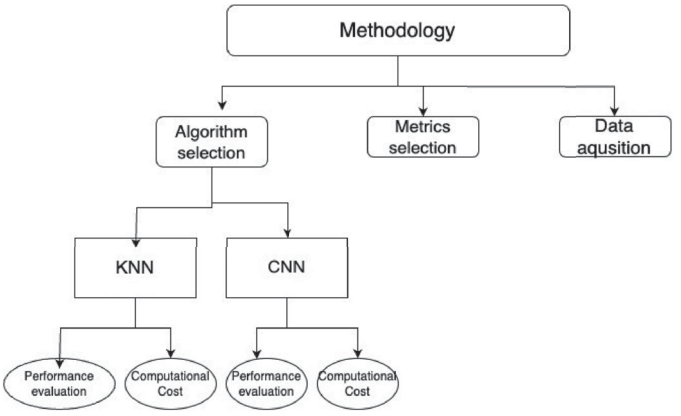

Figure 2. Diagram of our methodology.

associated with each in terms of usability, performance, and economic sustainability.

Our study thus fills a major gap in the literature by incorporating economic factors into the assessment of AI algorithms for IoT intrusion detection. Our research objective is to offer a comprehensive understanding of the economic implications of the choice, execution, and maintenance of algorithms by utilizing a TCO model relevant to large scale industrial settings that deal with active attacks on IoT. Furthermore, we explore the performance, design factors, and usability costs of various algorithms in addition to comparing the costs of the algorithms.

3. Methods and cost metrics

Previous research on intrusion detection in IoT devices has focused on ML/DL models for distinguishing benign and malicious packets and classifying attacks [7,62,63]. There have been reports of the performance of various algorithms and classifier designs to demonstrate the potential advantages and problem-solving capabilities of DL. Methods for intrusion detection are chosen, designed, and employed to address the problem context and the research objectives. Many CNN methods are developmental, being selected to test performance-based hypotheses, expand knowledge, and design capabilities for autonomous systems [53,64]. The metrics used are chosen from confusion matrix relationships and statistical methods [50]. The following discussion describes the choices made to facilitate this research, as shown in Figure 2. We define the two algorithms selected for testing, elaborate on the choice of the four relevant metrics, and specify the IoT datasets.

3.1. Algorithms

An algorithm provides a sequence of instructions to achieve a predicted outcome. For our comparison of the KNN and CNN algorithms, we specify them here along with the steps for computation and review the decisions that researchers make regarding the design and scope of each and identify the cost implications.

3.1.1. KNN

The KNN algorithm, when used for supervised classification, takes an observation input, computes the relationship to current classifications, and outputs a classification class for the observation [20]. The algorithm learns to classify through the supervisory labeling of features and comparison of an observation for similarity to the features in the stored dataset. It performs best when the dataset has consistencies that can be matched to categories. When the data are inconsistent or diverse, greater supervisory intervention is required, and higher error rates occur. However, many of the attacks on the IoT described in section 2 above have consistent features and can be pre-labeled by the supervisor for automated KNN classifications. The accuracy of KNN is sensitive to the data distribution, so a dominant class can skew the determination of a new observation. The accuracy with which the algorithm appraises a new observation depends on the abstraction or weighting of the data from a dataset. Hence, the dataset requires data abstraction or weightings to assure a fair appraisal of any new observation. The training phase of the algorithm is minimal when the distribution of the data is consistent, and the supervisor accurately identifies feature vectors and class labels. Since these preconditions may not be satisfied, the performance metrics require monitoring during the computation process.

The simplest application of KNN to classification problems involves selecting a binary output class. For the classification of IoT malware, this selection determines whether the outcome is malicious [65]. The KNN algorithm can be scaled to multiple output classes, but we focus on the simplest form of it that is relevant to the IoT intrusion detection problem. The additional costs of the added features can be estimated accurately from the simple binary case. The training data are also simplified, consisting of labeled IoT PCAP files that are divided equally between the two classes. Based on these training data, the objective is to predict the classification of new data from the test dataset. In the simplest computation format, the nearest k neighbor(s) of the new input is (are) computed, and the new data are classified by vote as either malicious or not. The KNN classifier requires supervisory input to function but has a simple design and minimal computational steps for a small n.

The KNN supervisor must choose the k value and the measurement model [20]. The k value and the n value settings determine the scope of the costs for the algorithm. Many guides suggest that k should be the square root of n ($\sqrt{n}$), but, in practice, each dataset requires test runs with various k values to identify the optimal performance. The k value should be an odd number to break any ties in vote counting for $k > 1$. The k value sets the number of computations to be referenced in order from highest to k for the classification vote. Low and high values of k impact the accuracy and cost, and the supervisor must optimize k for each dataset. For this research, we chose a simplified configuration with the most frequent vote and

Algorithm 1. KNN Intrusion detection

1: Start
2: Inspect dataset
3: Determine best data abstraction
4: Test for optimal k value
5: Run
6: Rank metric results
7: Use k to select comparison set members
8. Decide by vote classification
8: Stop

Algorithm 2. Simplified CNN intrusion detection

1: Convert PCAP files to grey scale matrixes
2: Input grey scale matrix into convolution function
3: Apply ReLU to reduce output size and repeat step 2 once
4: Send reduced matrix to fully connected layer
5: Compute Softmax output and report classification
6: Compute metrics
7: Repeat steps 2–6 until training complete
8: Run test
9: Compute metrics and report

experimented with values of k around $\sqrt{n}$ for the trade-off between costs and accuracy for the IoT dataset.

The KNN algorithm is.

3.1.2. CNN

A CNN possesses DL and autonomous capabilities. This multi-layered network can extract complex automated features from IoT malicious codes [15]. Implementing the model requires structural determination, the conversion of data to images, and training until an acceptable error rate is achieved. Once trained, the CNN operates autonomously and can be utilized for intrusion detection [17]. Its value lies in the lack of supervision required after training and the ability to process large volumes of data. However, the implementation and training of the algorithm can be costly. The structural design determines the constraints, and the process steps contribute to the substantial computational costs. The PCAP files and labels need to be pre-processed to reshape the features into grayscale, as described in section 2.2; the input is a 2D x 1 array. The 2D effect is obtained by setting one dimension to the value of 1, and zero padding is inserted to create the required input. In an ANN, each neuron is directly connected, so each image input has 16 × 16 × 1 pixels based on grayscale weightings from 0 to 255 plus 1 for the bias, which equals 257 connections per neuron. However, the CNN is more cost-effective, and each convolution layer forward feeds 3 × 3 × 1 weightings plus 1 bias, resulting in significant computational cost savings. Each image data array is fed into the convolution function to reduce the amount of data sent to the neurons in the fully connected layers. Normalization and pooling decrease the data size, and the activation layer ReLU function and softmax output function are cost-effective design choices. We simplified the research CNN design to include the input array, a convolution function, a pooling layer with ReLU output activation function (x2 for a DL aspect), the fully connected layer, the softmax output function set to the malicious or non-malicious classification (0 or 1, respectively), and the performance metric computation.

Thus, we minimize the cost of the CNN algorithm by streamlining the design and selecting cost-effective output functions. In doing so, we establish a minimal cost baseline for malware detection that can then serve as the foundation for estimates of the cost of incorporating additional computational options into the algorithm. The following is a detailed overview of the general steps.

For our comparison of KNN and CNN, we examine their respective designs and the steps for computation, discuss the decisions made by researchers regarding the design and scope of each algorithm with an emphasis on the cost implications, and outline the four relevant metrics and the IoT datasets utilized. Because the KNN algorithm, with its simple design, requires supervisory input but offers minimal computational steps for a small n, it is cost-effective for certain applications. On the contrary, the CNN algorithm, with its DL capabilities and autonomous functioning, provides a more powerful solution for intrusion detection but has higher implementation and training costs. By comparing these algorithms, we can better understand the trade-offs among cost, accuracy, and ease of use in the context of detecting intrusions in the IoT, thereby, ultimately, facilitating the selection of appropriate methods and metrics for future research in this area.

3.2. Metrics and computation

The following metric relationships are based on the classification confusion outcomes matrix in Equations 1–4 below [20,63]. The accuracy measure computes the number of correct classifications in the total number of classifications (1). (TP = true positive correct malicious classification; TN = true negative correct benign classification; FP = false positive = incorrect malicious classification; FN = incorrect benign classification.)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

The precision of the algorithm is computed as the ratio of correct malicious classifications and the sum of correct malicious classifications and incorrect malicious classifications.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

The recall or true positive rate is computed as the ratio of correct malicious classifications and the sum of the correct malicious classifications and the incorrect benign classifications (3).

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

The F1 score is the harmonic mean for precision and recall; it is computed using Equation (4). The purpose of the F1 score is to acknowledge the extreme values in a dataset and, hence, moderate excessive theoretical normalization (P = precision; R = recall).

$$F1 = 2x\frac{PxR}{P + R} \tag{4}$$

3.3. Dataset used

The widely used Aposemat IoT-23 dataset is among the numerous data sources available to researchers online [66]. This dataset comprises both the malicious and the benign IoT network traffic captured in PCAP files for analysis. It contains 23 scenarios of network traffic ranging in size from 2.1 MB to 21 GB. Of these scenarios, 20 are malicious, and 3 are benign. Two scenarios were chosen for training and testing the algorithms based on the closest match in the packet numbers between a benign and a malicious scenario. One scenario contains 21,000 packets and the other 23,000 packets. In [67], the authors suggested that 75% of the data should be allocated for training and 25% for testing. These scenarios enable the random selection of 10,000 packets from each scenario and provide an

additional 2,500 randomly selected packets from both scenarios for testing. The experimental design involves training each algorithm on 10,000 benign and 10,000 malicious labeled PCAP packets and then testing on 2,500 randomly chosen packets. The size and scope of the KNN dataset are controlled by the experimental settings to establish the k-value and maintain manageable computational costs.

Although there were 23 scenarios, several of them shared similar traits. We selected two scenarios that captured much of the volatility in the dataset using feature analysis and dimensionality-reduction approaches. We were able to streamline the training procedure while maintaining the key components required for precise model performance by using only these two situations.

3.4. Experimental setup

The platforms that we used had hardware requirements that included high-performance servers with multi-core Intel Xeon E5 series processors and 64 GB of RAM. To improve these servers further, we used NVIDIA GeForce RTX GPUs to accelerate the training and interference procedures of our CNN models. Because of its reliability and compatibility with DL frameworks, we used the Ubuntu Linux operating system (version 20.04) for our software environment. To compute the neural network operations efficiently, we made use of the GPU-accelerated implementation of the TensorFlow library (version 2.5). Furthermore, the major programming language for the model-building and experimentation was Python (version 3.8). We used Jupyter Notebook as an interactive computing environment to facilitate the experimentation and analysis. This approach allowed for the seamless integration of code, documentation, and visualizations.

We used a CNN architecture with numerous convolutional layers, max pooling layers, fully connected layers, and softmax activation for classification in our investigation. To down-sample the feature maps, we specifically used a deep CNN architecture with six convolutional layers – each with a distinct number of filters and kernel size – sprinkled with max-pooling layers. We added nonlinearity to the network by using rectified linear unit (ReLU) activation functions. We also added batch normalization layers to increase training stability and speed up convergence. To avoid overfitting, we used dropout layers for regularization following the fully connected layers. The CNN was trained with a categorical cross-entropy loss function and a learning rate of 0.001 using the Adam optimization algorithm. During training, data augmentation methods such as random rotation, horizontal flipping, and scaling served to improve the generalization performance further. A high-performance computing platform using NVIDIA GPUs to speed up calculation was employed for the training procedure. Additionally, through testing and experimentation, we carefully adjusted certain hyperparameters to maximize performance, including batch size, the number of epochs, and weight initialization approaches.

4. Results

Table 1 presents the scope of the algorithm cost attributes and categorizes the properties of the KNN and CNN algorithms. Table 4 presents definitions of the general parameters for measuring the total cost of ownership for an algorithm based on a simplified software ownership lifecycle. The testing conducted on each classifier represents the minimum number of runs needed to identify the classification performance. The aim of the experimental design is to determine the minimal cost and performance trade-off, which serves as a baseline for extrapolating the value-added costs and benefits when other algorithm options are considered.

The first decision to minimize costs involves selecting the binary classification for malicious and non-malicious IoT traffic, which the dataset labels provide. This choice supports a minimalistic approach and leaves multiple classification tasks for quantification by extrapolation. It also lays the foundation for setting expectations for more complex experimental designs. Importantly, this approach establishes a benchmarking process for measuring the total cost of ownership of AI applications and methods for object-oriented measurement.

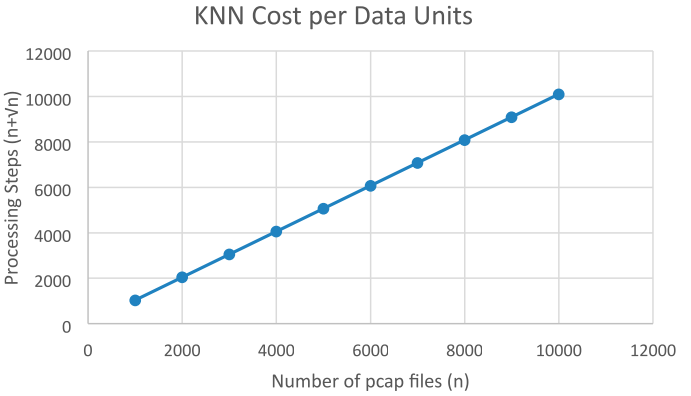

Figure 3. KNN cost per data unit.

The second set of decisions entails simplifying the KNN and CNN algorithms to the bare minimum of steps and performance features to ensure operation and realistic outcomes. As a result, the KNN algorithm utilizes Euclidean distances, powers of k iterations up to $\sqrt{n}$, and vote decision-making. The CNN algorithm features only two layers after the feature convolution selection and before the connected output layer. This configuration represents the minimum required for functionality, necessitating at least 10,000 training runs to achieve meaningful outcomes.

In what follows, we address the research question and the implications for further research concerning the potential cost–benefit and performance trade-offs that may be quantified based on the benchmarks achieved. The results for each algorithm are discussed in the following subsections.

4.1. KNN results

The KNN algorithm is often touted for its quick setup and ease of use. However, we found that configuring the 2D data format and selecting the best-performing k value proved to be time-consuming and computationally expensive. Consequently, we reduced the number of trained data points to 50 normal PCAP files and 50 malicious PCAP files. These files were randomly and equally selected from the Malware-capture-8-1 dataset and the Honeypot-capture-4-1 dataset. For the testing, we adhered closely to the proportions specified in the literature, selecting 25 data points randomly from each dataset.

The 2D rendering of the data points was achieved by randomly allocating coordinates to each trained dataset in the binary segregated regions. The test points were then randomly allocated coordinates in a region equidistant from the trained regions and randomly entered for computation. The number of data points was significantly reduced from the planned 10,000 because it proved possible to address the research question using the smaller dataset and apply the proposed extrapolation method to predict the results from larger datasets. These results are sufficiently stable for extrapolation and prediction of the costs and benefits of larger datasets. Table 6 reports the total cost of KNN algorithm ownership from Table 4, and Figure 3 illustrates the trade-off curves of k value, performance, and costs.

The test runs of the KNN algorithm for various k values yielded 100% accuracy on the labeled datasets, with only a minor cost

Table 6. KNN Total cost of ownership results.

Phase	Attribute	Observation	Metric	Measure
Acquire	Procure	Easy to find	Finance	Free
	Availability	Easy to acquire	Time	15-20 minutes
	Fit	Customization required	Method	K step count 2D data format
Operate	Implement	Fit format	Time	30-40 minutes
	Train	Structure clusters	Expert help	Setting data format and k
	Test	Easy once set up	Resources	Higher for larger n and k
	Run	Quick for small n and k	Success rate	High for larger k
Retire	Data erasure	Delete all datasets	Time	5 minutes
	Decommission	Remove data, save code	Knowledge	Tactic functionality
	Reuse	Code and 2D data method	Market Value	Little. Too common / free

Table 7. Metrics for test accuracy of DT algorithm.

Accuracy	Execution time	CPU	Memory
99.99%	33.38	99% (24 threads)	2.84 GB

Table 8. DT total cost of ownership results.

Phase	Attribute	Observations	Metric	Measure
Acquire	Procure	Easy to get	Financial implications	Free
	Availability	Available through multiple libraries	Libraries	4
	Fit	Customization available	Method	Criterion leaf
Operate	Implement	Time-consuming	Time	33.38 sec
	Train	Tree algorithm	Expert help/ resources	Code customization memory: 2.84 GB CPU usage: 99%
	Test	Easy once set up	Resources	High
	Run	Quick for data < 1,000,000	Success rate	Always successful
Retire	Data erasure	Delete all datasets	Time	5 minutes
	Decommission	Remove data and save code	Knowledge	Tactic functionality
	Reuse	Code and dataset	Market Value	Little

increase for each k unit resulting from the steps for selecting the least distance. However, the relationship between k and n is significant. The selected n was 125 (100 + 25). As n increases, the cost increases by n + k, where ideally $k = \sqrt{n}$.

Table 7 presents the performance metrics for the DT algorithm. The 99.99% stated accuracy shows the precision of the DT algorithm in classifying data. Its execution duration of roughly 33.38 s indicates its effectiveness in processing data. With 24 threads and a CPU utilization of 99%, these metrics indicate substantial parallelization to maximize computing resources. Furthermore, the 2.84 GB memory use is the memory footprint needed to run the method. Table 8 reports the total cost of DT algorithm ownership.

This relationship renders large datasets increasingly expensive to process in KNN. Computing the distance of any new PCAP file once and then calculating the votes for the various k values from a static rank table reduced the costs when optimizing k. Figure 3 illustrates the experimental relationship and extrapolation of costs.

4.2. CNN results

The CNN algorithm required design and training prior to use. We intentionally simplified the CNN design to a minimal specification to ensure that the trade-off constraints were satisfied, stable results were

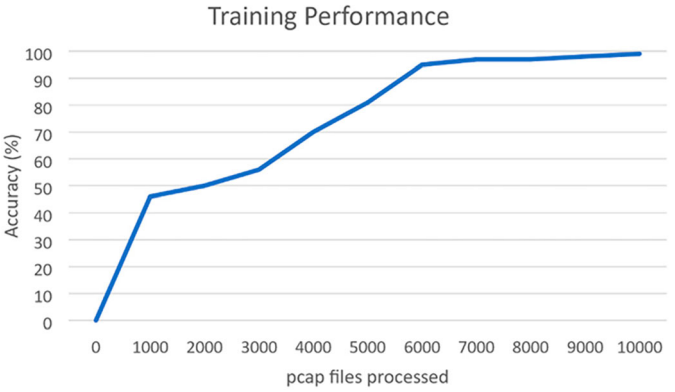

Figure 4. CNN training accuracy rates per PCAP file.

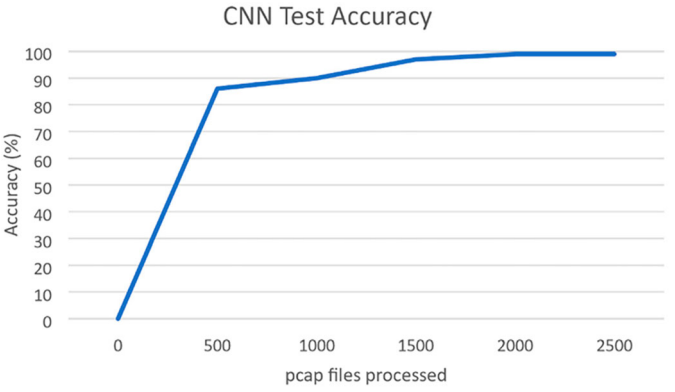

Figure 5. CNN test accuracy rates per PCAP file.

obtained, and the potential for further design additions was revealed for costing. The 3D data format setup was also simplified by setting the third dimension to 1 and adopting grayscale values with decimal representations in [0,225] to fit the image format. The resulting array texture is a unique grayscale image representing each PCAP file. Consequently, the minimal CNN has a normalized matrix input to the convolution function that identifies the characteristic features from each matrix.

The two pooling layers reduce the computational costs, and the activation ReLU function output is used for cost efficiency, followed by the second pooling layer input to a directly connected layer. The final output is obtained using the SoftMax function for classification, and the result is collected for metric analysis. The planned 10,000 PCAP files were used as outlined in Section 3.3. These results are sufficiently stable to predict the costs and benefits of CNN with various designs and larger datasets. Table 9 reports the total cost of CNN algorithm ownership from Table 4, and Figures 4 and 5 display the performance of CNN during training and testing, respectively. Table 10 quantifies the results.

5. Discussion

Adopters of AI applications for IoT intrusion detection must address economic questions and evaluate the costs of the choices made prior to use. This experiment was deliberately structured to compare substantially different algorithms in order to calculate the cost differences in their utilization for IoT intrusion detection. The literature review established the vulnerability of IoT systems to a wide range of passive and active attacks and confirmed the volume of data generated by sensors for information processing. Consequently, current

Table 9. CNN total cost of ownership results.

Phase	Attribute	Observation	Metric	Measure
Acquire	Procure	Confusing options	Finance	Free
	Availability	Must select relevant CNN	Time	2-3 hours
	Fit	Customization required	Method	Design choices 3D data format
Operate	Implement	Time-consuming	Time	6-8 hours
	Train	Check blocks work and metrics	Expert help	Code audit data formatting
	Test	Easy once set up	Resources	Higher for larger n and more layers
	Run	Requires up to 10,000 training runs	Success rate	Low with $< 5{,}000$ runs; expensive
Retire	Data erasure	Delete each layer	Time	30 minutes
	Decommission	Remove layers and data	Knowledge	Tactical functionality
	Reuse	Whole trained CNN	Market Value	Reuse and sale possible

Table 10. Metrics for test accuracy for unsupervised algorithm.

Accuracy	Precision	Recall	F1 Score
99.0%	99.0%	99.0%	99.3

research proposals to automate malicious intrusion detection in an IoT network are relevant and necessary. The performance of an AI algorithm is economically valuable when the benefits outweigh the costs. In this research, we employed the TCO as a for assessing AI algorithm benefits using the economic classifier model (Table 4).

The comparison of the cost of a supervised algorithm with that of an unsupervised algorithm introduced the cost of a supervisor and the benefits of autonomous detection systems. For a small number of PCAP files, the supervised algorithm demonstrated superior setup, performance, and accuracy benefits. However, there are significant challenges in IoT intrusion detection associated with the large datasets of the continuous data feeds. A supervisor can only manage a portion of the requirement, and the computational costs escalate as the number of PCAP files increases. Extrapolating the current findings, we speculate that there will be a crossover point at which the unsupervised algorithm offers computational cost advantages. Once trained, the unsupervised algorithm provided an accuracy level of 99.0% on the test data and had an F1 score of 99.3 (Table 10). A perfect result would be 100% accuracy and an F1 score of 1 These findings suggest that an unsupervised algorithm can approach a near perfect detection rates, but an expectation is set for a margin of error. Allowing for a few points of potential failure and the user's appetite for risk, the unsupervised algorithm (CNN) offers significant economic advantages for IoT intrusion detection in large datasets.

Design is a critical factor in the selection of a cost-effective and high-performing AI application for IoT intrusion detection. The supervised algorithm assigns to the supervisor a range of data-abstraction judgments, data-conditioning choices, and add-on options for statistical data treatments. All these design considerations have cost and benefit trade-offs for task performance. The unsupervised algorithm has significant training costs as well as design costs depending on the number of layers chosen, the data management techniques, and the operational scope. Many ANN designs are available, and we chose a CNN for cost efficiencies and reduced neuron node loadings.

However, a CNN can become more costly when targeted and refined for specific detection tasks that require more than two pooling layers. In large datasets from IoT sensor inputs, many more layers are feasible for better responsiveness, accuracy, and multi-class classification. The scope for IoT intrusion detection scales up the potential number of elements in a design and the performance expectations for detection to a full range of attacks. To meet these requirements, the CNN design may become too complex and costly to use.

The first research question addresses how a deep learning (DL) unsupervised algorithm for IoT intrusion detection compares in terms of cost and performance with a typical supervised pattern recognition algorithm. The use of an algorithm for IoT intrusion detection entails usability costs and implementation costs that must be factored in before the expected benefits of deliverables are calculated. The efficiency and effectiveness of the algorithm are the expected deliverables, but the difficulty or simplicity of implementation and use are drag factors that create tangible costs. Both tested AI algorithms required data structuration prior to use, representing a hidden and ongoing cost for computation, memory use, and response times. In some instances, a CNN pretrained on similar classification problems may be purchased to avoid the training costs. However, a pre-trained CNN still requires input data structuration, system implementation, customization, and monitoring before release to autonomy. Hence, hidden costs reduce the value of the expected benefits of using a CNN, but the extrapolation from our experiments suggests that the optimal arrangement can be obtained by reducing the complexity of the CNN design, specifying the scope of the detection task, and setting minimal operational controls. Cost efficiencies can also be gained in the training of a CNN and the export of the tactical knowledge gained to reduce the implementation time and operational controls.

The cost of failure when adopting AI algorithms for IoT intrusion detection is proportional to the impact of an undetected negative risk. In [64], the authors emphasized that it is critical to reduce false negatives in an intrusion-detection classification system. In our experiments, the KNN had no false negatives but exhibited a very high cost to achieve this result. By contrast, the CNN had 1.0% false negatives when autonomously classifying malicious and non-malicious PCAP files in the test sets. Although this number is low, the criticality of failure must be measured against impacts on the system. A false negative allows a malicious PCAP file into the IoT system, exposing it to the consequences of an attack. Further work is required to reduce the false negatives and classify all the malicious PCAP files by attack type. The IoT attack types are identified in Section 2, but the mapping of the impacts of the various types of attacks on IoT systems has not been completed. Improving a CNN offers residual benefits, and, while the simplicity of a KNN provides intangible benefits upon retirement, a CNN provides tangible benefits. A CNN boasts greater capacity for intrusion detection than a KNN, offering flexibility, adaptability, and retained learning. Consequently, a CNN can be reused or sold for residual benefits. Prospective users of AI intrusion detection capabilities must, therefore, evaluate the TCO of an algorithm across all the TCO dimensions before adoption. The research demonstrates that while KNN does not perform well for large datasets due to its high TCO, CNN, on the other hand, performs better on larger datasets with a more favorable TCO.

The second research question examines the trade-offs between these algorithms, highlighting their utility and application in real-world situations. Tables 6 and 8, and Table 9 are based on the model provided by Table 4, offering a thorough analysis of the financial, operational, and retirement phases related to the deployment of IoT algorithms, in addition to a performance score. This analysis includes the validation of the economic classifier model on the IoT-23 dataset. These statistics offer vital insights into the whole lifecycle costs of algorithm ownership by dissecting the TCO into individual features, such as procurement, availability, fit evaluation, implementation, testing, training, operational success rates, and eventual retirement

concerns. This method highlights how important it is to comprehend the functional and financial implications of implementing algorithms in IoT ecosystems, where cost-effectiveness and resource limitations play an essential role. It is a major step forward in directing decision-making processes for organizations aiming to optimize their IoT solutions because the thorough examination of these qualities illuminates real-world challenges and potential opportunities in deploying these algorithms.

6. Conclusions

The costs associated with utilizing AI applications for detecting IoT sensor attacks can be assessed by identifying each algorithm's attributes and the trade-offs of competing risks. Our literature review highlights the variety and significance of the vulnerabilities within IoT networks. The deployment of AI applications is pertinent to the big data context of IoT sensors, potential attack types, and the vast volumes of data requiring analysis. The research presented here demonstrates that each algorithm offers substantial benefits in distinguishing IoT sensor attacks from malicious and non-malicious PCAP files, but several mitigating factors must be evaluated to select the optimal solution. The research contribution includes the emphasis on the cost and benefit trade-offs of any artificial intelligence application for intrusion detection and the recommendation of a comprehensive TCO evaluation before the adoption of automated solutions for IoT intrusion detection.

This study has limitations that suggest avenues for future research. First, since the experiment focused on comparing two specific ML algorithms, KNN and CNN, expanding the scope of the economic classifier model to include additional relevant ML classifiers could provide a more comprehensive understanding of the trade-offs involved in selecting an appropriate ML technique for detecting intrusions in IoT sensors and associated systems. Second, the study was limited to one dataset and specific IoT sensors such as proximity, image, and motion sensors; future work could extend this research to include larger and/or multiple datasets and various sensor types to achieve triangulation and generalization. Third, there is a need to map different types of Industrial IoT attacks to their corresponding system implications, as these attacks target unique vulnerabilities in industrial ecosystems, which often involve a vast number of IoT devices. Further research could focus on developing economic classifier models that are replicable across various industrial settings, ensuring a well-balanced trade-off between detection accuracy and the costs associated with system failures.

Lastly, while the research presented here emphasizes the importance of TCO evaluation when considering the adoption of ML applications for detecting intrusions in IoT sensors and associated systems, it provides a model for conducting such an evaluation. Additionally, exploring multiple variables that influence the TCO of intrusion detection of lightweight classifiers in large scale and complex IoT ecosystems.

Data availability statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

[1] Statista. (2023). Number of Internet of Things (IoT) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2033. Retrieved from https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[2] Security Brief. (2023). A sharp increase in cyberattacks on IoT devices: Check Point. Retrieved from https://securitybrief.co.uk/story/a-sharp-increase-in-cyberattacks-on-iot-devices-check-point.

[3] Ojo MO, Giordano S, Procissi G, et al. A review of low-end, middle-end, and high-end IoT devices. IEEE Access. 2018;6:70528–70554. doi:10.1109/ACCESS.2018.2879615

[4] Statista. (2024). Number of Internet of Things (IoT) connections worldwide from 2023 to 2024, with forecasts from 2024 to 2033. Retrieved from https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.

[5] ISO. (2019). Internet of things (IoT) — Interoperability for IoT systems Part 1: Framework https://www.iso.org/standard/71885.html.

[6] Alaba F, Othman M, Hashem I, et al. Internet of things security: A survey. J Netw Comput Appl. 2017;88:10–28. doi:10.1016/j.jnca.2017.04.002

[7] Surendran S, Nassef A, Beheshti B. A survey of cryptographic algorithms for IoT devices. In: 2018 IEEE long island systems, applications and technology conference (LISAT). New York: IEEE; 2018, May. p. 1–8.

[8] Hou KM, Diao X, Shi H, et al. Trends and challenges in AIoT/IIoT/IoT implementation. Sensors. 2023;23(11):5074), doi:10.3390/s23115074

[9] Qiu J, Zhang W, Lao I, et al. A survey of android malware detection with deep neural models. ACM Computer Surveys. 2021;53(6):1–36. doi:10.1145/3417978

[10] Rizvi S, Orr R, Cox A, et al. Identifying the attack surface for IoT network. Internet of Things. 2020;9:100162–100171. doi:10.1016/j.iot.2020.100162

[11] Hemalatha J, Roseline S, Greetha S, et al. An efficient densenet-based deep learning model for malware detection. Entropy . 2021;23(3):344–346. doi:10.3390/e23030344

[12] Lin K, Xu F, Xiao M. MFFusion: A multi-level features fusion model for malicious traffic detection based on deep learning. Comput Netw. 2022;202:108658–108665. doi:10.1016/j.comnet.2021.108658

[13] Aghapour Z, Sharifian S, Taheri H. Task offloading and resource allocation algorithm based on deep reinforcement learning for distributed AI execution tasks in IoT edge computing environments. Comput Netw. 2023;223:109577), doi:10.1016/j.comnet.2023.109577

[14] Deng X, Chen B, Chen X, et al. A trusted edge computing system based on intelligent risk detection for smart IoT. IEEE Trans Ind Inf. 2023;99:1–10.

[15] Vasan D, Alazab S, Wassan S, et al. IMCFN: image-based malware classification using fine-tuned convolutional neural network architecture. Comput Netw. 2020;171:107138–107148. doi:10.1016/j.comnet.2020.107138

[16] Venkatraman S, Alazab M, Vinayakumar R. A hybrid deep learning image-based analysis for effective malware detection. J Secur Appl. 2019;47:377–389.

[17] Kumar S. Meft-CNN malware classification with fine tune convolution neural networks using traditional transfer learning in the internet of things. Future Gener Comput Syst. 2019;125:334–351.

[18] Gilbert D, Mateu C, Planes J, et al. Using convolutional neural networks for classification of malware represented as images. J Comput Virol Hacking Tech. 2019;15:15–28. doi:10.1007/s11416-018-0323-0

[19] Kumara A, Jaidhar CD. Automated multi-level malware detection system based on reconstructed semantic view of executables using machine learning techniques at VMM. Future Gener Comput Syst. 2018;79:431–446. doi:10.1016/j.future.2017.06.002

[20] Yihua L, Vemun V. Use of K-nearest neighbor classifier for intrusion detection. Comput Secur. 2002;21(5):439–448. doi:10.1016/S0167-4048(02)00514-X

[21] Nicho M, Girija S. IoTVT model: A model mapping IoT sensors to IoT vulnerabilities and threats. In: 2021 20th international conference on ubiquitous computing and communications (IUCC/CIT/DSCI/SmartCNS). New York: IEEE; 2021, December. p. 123–129.

[22] Serror M, Hack S, Henze M, et al. Challenges and opportunities in securing the industrial internet of things. IEEE Trans Ind Inf. 2021;17(5):2985–2996. doi:10.1109/TII.2020.3023507

[23] Srivastava A, Gupta S, Quamara M, et al. Future IoT-enabled threats and vulnerabilities: state of the art, challenges, and future prospects. Int J Commun Syst. 2020;33(12):e4443), doi:10.1002/dac.4443

[24] Deogirikar J, Vidhate A. Security attacks in IoT: A survey. In: 2017 international conference on I-SMAC (IoT in social, mobile, analytics and cloud) (I-SMAC). New York: IEEE; 2017, February. p. 32–37.

[25] Ali W, Dustgeer G, Awais M, et al. Iot based smart home: security challenges, security requirements and solutions. In: 2017 23rd international conference on automation and computing (ICAC). New York: IEEE; 2017, September. p. 1–6.

[26] Ali Abdu NA, Basulaim KO. Machine learning in concept drift detection using statistical measures. Int J Comput Appl. 2024;46(5):281–291. doi:10.1080/1206212X.2023.2289706

[27] Bonab AB, Rudko I, Bellini F. (2021). A review and a proposal about socio-economic impacts of artificial intelligence. In Business Revolution in a

Digital Era: 14th International Conference on Business Excellence, ICBE 2020, Bucharest, Romania (pp. 251–270). Springer International Publishing.
[28] Chen N, Christensen L, Gallagher K, et al. Global econ. Impacts Assoc. Artif. Intell. 2016;23.
[29] Furman J, Seamans R. AI and the economy. Innov. Policy Econ. 2019;19:161–191. doi:10.1086/699936
[30] Abrardi L, Cambini C, Rondi L. (2019). The economics of artificial intelligence: A survey. Robert Schuman Centre for Advanced Studies Research Paper No. RSCAS, 58.
[31] Sultana R, Grover J, Meghwal J, et al. Exploiting machine learning and deep learning models for misbehavior detection in VANET. Int J Comput Appl. 2022;44(11):1024–1038. doi:10.1080/1206212X.2022.2099122
[32] Magán-Carrión R, Urda D, Díaz-Cano I, et al. Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches. Appl Sci. 2020;10(5):1775), doi:10.3390/app10051775
[33] Nandanwar H, Katarya R. Deep learning enabled intrusion detection system for industrial IOT environment. Expert Syst Appl. 2024;123808.
[34] Awotunde JB, Misra S. Feature extraction and artificial intelligence-based intrusion detection model for a secure internet of things networks. In: Illumination of artificial intelligence in cybersecurity and forensics. Cham: Springer International Publishing; 2022. p. 21–44.
[35] Saied M, Guirguis S, Madbouly M. Review of artificial intelligence for enhancing intrusion detection in the internet of things. Eng Appl Artif Intell. 2024;127:107231), doi:10.1016/j.engappai.2023.107231
[36] Butun I, Osterberg P, Song H. Security of the internet of things: vulnerabilities, attacks, and countermeasures. IEEE Commun Surv Tutorials. 2020;22(1):616–644. doi:10.1109/COMST.2019.2953364
[37] Kizza JM. System intrusion detection and prevention. In: Guide to computer network security. Cham: Springer International Publishing; 2024. p. 295–323.
[38] Omitola T, Wills G. Towards mapping the security challenges of the internet of things (IoT) supply chain. Procedia Comput Sci. 2018;126:441–450. doi:10.1016/j.procs.2018.07.278
[39] Alladi T, Chamola V, Sikdar B, et al. Consumer IoT: security vulnerability case studies and solutions. IEEE Consum Electron Mag. 2020;9(2):17–25. doi:10.1109/MCE.2019.2953740
[40] Frustaci M, Pace P, Aloi G, et al. Evaluating critical security issues of the IoT world: present and future challenges. IEEE Internet Things J. 2018;5(4):2483–2495. doi:10.1109/JIOT.2017.2767291
[41] Mourtzis D, Angelopoulos K, Zogopoulos V. Mapping vulnerabilities in the industrial internet of things landscape. Procedia CIRP. 2019;84:265–270. doi:10.1016/j.procir.2019.04.201
[42] Luo E, Bhuiyan A, Wang G, et al. Privacyprotector: privacy-protected patient data collection in IoT-based healthcare systems. IEEE Commun Mag. 2018;56(2):163–168. doi:10.1109/MCOM.2018.1700364
[43] Caminha J, Perkusich A, Perkusich M. A smart trust management method to detect on-off attacks in the internet of things. Security and Communication Networks. 2018;1:6063456.
[44] Chen K, Zhang S, Li Z, et al. Internet-of-things security and vulnerabilities: taxonomy, challenges, and practice. J Hardware Syst Secur. 2018;2(2):97–110. doi:10.1007/s41635-017-0029-7
[45] White G, Nallur V, Clarke S. Quality of service approaches in IoT: A systematic mapping. J Syst Softw. 2017;132:186–203. doi:10.1016/j.jss.2017.05.125
[46] Zhang K, Liang X, Lu R, et al. Sybil attacks and their defenses in the internet of things. IEEE Internet Things J. 2014;1(5):372–383. doi:10.1109/JIOT.2014.2344013
[47] Kolias C, Kambourakis G, Stavrou A, et al. DDos in the IoT: mirai and other botnets. Computer (Long Beach Calif). 2017;50(7):80–84. doi:10.1109/MC.2017.201
[48] Dhanabal L, Shantharajah S. A study on NSL-KDD dataset for intrusion system based on classification algorithms. Int J Adv Res Comput Commun Eng. 2015;2015(4):446452.
[49] Sesmero M, Ledezma A, Sanchis A. Generating ensembles of heterogeneous classifiers using stacked generalization. Wiley Interdiscip Res Data Min, Knowl Discovery. 2015;5:21–34. doi:10.1002/widm.1143
[50] Hossin M, Sulaiman M. A review on evaluation metrics for data classification evaluations. Int J Data Min Knowl Manage Process. 2015;5(2):01–11. doi:10.5121/ijdkp.2015.5201
[51] Sebesto S, Baranov E, Biondi F, et al. Optimizing symbolic execution for malware behavior classification. Comput Secur. 2020;93:101775–101789. doi:10.1016/j.cose.2020.101775
[52] Yakura S, Shinozaki R, Nishimura Y, et al. Neural malware analysis with attention mechanism. Comput Secur. 2019;87:101592–101607. doi:10.1016/j.cose.2019.101592
[53] Jahromi S, Hashemi A, Dehghantanha K, et al. An improved two-hidden-layer extreme learning machine for malware hunting. Comput Secur. 2019;89:101665101696.
[54] Tahsien S, Karimipour H, Spachos P. Machine learning based solutions for security of internet of things (IoT): A survey. J Netw Comput Appl. 2020;161:102630), doi:10.1016/j.jnca.2020.102630
[55] Roda I, Macchi M, Albanese S. Building a total cost of ownership model to support manufacturing asset lifecycle management. J Production Planning Control: Manage Oper. 2020;31(1):19–37. doi:10.1080/09537287.2019.1625079
[56] Nicolescu R, Huth M, Radanliev P, et al. Mapping the values of IoT. J Inf Technol. 2018;33(4):345–360. doi:10.1057/s41265-018-0054-1
[57] Cui Z, Du L, Wang P, et al. Malicious code detection based on CNNs and multi-objective algorithm. J Parallel Distrib Comput. 2019;129:50–58. doi:10.1016/j.jpdc.2019.03.010
[58] Bhupendra K, Ankar M, Payan K. Deep CNN-based damage classification of milled rice grains using a high-magnification image dataset. Comput Electron Agric. 2022;195:106811–106823. doi:10.1016/j.compag.2022.106811
[59] Abbasi M, Shahraki A, Taherkordi A. Deep learning for network traffic monitoring and analysis (NTMA): A survey. Comput Commun. 2021;170:19–41. doi:10.1016/j.comcom.2021.01.021
[60] Wu Y, Wei D, Feng J. Network attacks detection methods based on deep learning techniques: a survey. Security and Communication Networks. 2020;2020(1):8872923.
[61] Borrego-Carazo J, Castells-Rufas D, Biempica E, et al. Resource-constrained machine learning for ADAS: A systematic review. IEEE Access. 2020;8:40573–40598. doi:10.1109/ACCESS.2020.2976513
[62] Al-Yaseen WL, Idrees AK. MuDeLA: multi-level deep learning approach for intrusion detection systems. Int J Comput Appl. 2023;45(12):755–763. doi:10.1080/1206212X.2023.2275084
[63] Singh J, Singh J. Assessment of supervised machine learning algorithms using dynamic API calls for malware detection. Int J Comput Appl. 2022;44(3):270–277. doi:10.1080/1206212X.2020.1732641
[64] Datta P, Rohilla R. An autonomous and intelligent hybrid CNN-RNN-LSTM based approach for the detection and classification of abnormalities in brain. Multimed Tools Appl. 2024: 1–27.
[65] Mijalkovvic J, Spognardi A. Reducing the false negative rate in deep learning based network intrusion detection systems. Algorithms. 2022;15(8):258–267. doi:10.3390/a15080258
[66] Garcia S, Parmisano A, Erquiaga M. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Data set]. Zenodo. doi:10.5281/zenodo.4743746
[67] Cui Z, Xue F, Cai X, et al. Detection of malicious code variants based on deep learning. IEEE Trans Ind Inf. 2018;14:3187–3196. doi:10.1109/TII.2018.2822680