Arabian Journal for Science and Engineering (2025) 50:12071–12095
https://doi.org/10.1007/s13369-024-09663-6

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

Integrated Genetic Algorithm and Deep Learning Approach for Effective Cyber-Attack Detection and Classification in Industrial Internet of Things (IIoT) Environments

Nadia Alkhafaji[1,2] · **Thiago Viana**[1] · **Ali Al-Sherbaz**[3]

Received: 19 April 2024 / Accepted: 29 September 2024 / Published online: 21 October 2024
© The Author(s) 2024

Abstract
Cyber-attack detection within Industrial Internet of Things (IIoT) environments presents unique challenges due to the complex, resource-constrained, and real-time nature of these networks. Traditional detection techniques often struggle to adapt to the dynamic environment of IIoT. For instance, many existing methods rely on signature-based detection, which fails to identify evolving threats. Other approaches, such as anomaly-based detection, can generate a high rate of False Positives, leading to inefficiencies in threat management. To address these challenges, we propose a novel detection and classification model specifically tailored for IIoT environments. The proposed model integrates Genetic Algorithms (GA) and Deep Learning (DL) to enhance cyber-attack detection within IIoT environments. The GA component optimises feature selection from raw network data, ensuring the extraction of meaningful and relevant features. Leveraging these selected features, the DL component constructs a robust model capable of accurately detecting and classifying various cyber-attack patterns across IIoT devices. Through experimentation on real-world IIoT network traffic (UNSW-NB 15 dataset), the proposed approach demonstrates its efficacy in improving attack detection accuracy and adaptability. The integration of GA and DL offers a synergistic solution that addresses the complexities of IIoT cybersecurity, contributing to a more secure and resilient IIoT ecosystem. The integrated GA–DL classification model developed in this work achieved 98% precision, 96% accuracy, 94% recall, and 12% losses with only less than 50% of the features of the UNSW-NB 15 dataset. The reduction in features required for the identification and classification of cyber-attacks reduces the processing time by 50%.

Keywords IIoT · Genetic Algorithm · Deep Learning · Cyber-security · Cyber-attacks · Artificial intelligence · Prediction · Classification · Features selection

1 Introduction

The Industrial Internet of Things (IIoT) stands as a transformative force in the contemporary industrial landscape. Unlike the conventional IoT systems tailored primarily for consumer-oriented applications such as wearables, smart homes, and connected vehicles, IIoT focuses explicitly on industrial processes [1]. The core promise of IIoT lies in its capacity to drive unprecedented operational efficiencies, productivity enhancements, and revenue opportunities across various sectors. IIoT utilises interconnected sensors, instruments, and other devices networked together with computers' industrial applications. This networked ensemble facilitates the collection, exchange, and analysis of data, fostering intelligent decision-making in real time [2]. By embedding technology into physical assets, IIoT offers manufacturers a more detailed view of their plant operations and the lifecycle of their products.

Several industries are rapidly acknowledging the transformative potential of IIoT. For instance, in the manufacturing sector, IIoT promises smart factories where machinery and equipment can improve processes through self-optimisation and autonomously adapt to new production requirements [3]. The oil and gas industry can leverage IIoT for predictive

✉ Nadia Alkhafaji
nadia.alkhafaji@uwe.ac.uk

[1] School of Business, Computing and Social Sciences, University of Gloucestershire, Cheltenham GL50 2RH, UK

[2] School of Engineering, University of the West of England Bristol, Bristol BS16 1QY, UK

[3] Institute of Continuing Education, Cambridge University, Cambridge CB23 8AQ, UK

maintenance of pipelines and to monitor real-time conditions of reservoirs [4]. Agriculture, too, hasn't remained untouched, with IIoT enabling precision farming, ensuring efficient use of resources, and maximising yield [5]. The rise of IIoT is in tandem with the global push for Industry 4.0—the next phase in the digitisation of the manufacturing sector. Industry 4.0 emphasises the importance of smart machines that keep getting smarter as they access more data [6]. Consequently, the advent of IIoT is ushering in an era where industries can achieve more with less, driven by data-led insights.

However, with its explosive growth, IIoT also brings forth challenges, especially in the areas of cybersecurity, data privacy, and integration with legacy systems. Addressing these concerns is paramount to realise the full potential of IIoT across industries. IIoT exposes critical infrastructures to heightened risks of cyber-attacks. According to a report by CyberX, 53% of industrial sites surveyed were found to run outdated Windows systems, leaving them highly susceptible to ransomware attacks and other forms of malware [7]. Kaspersky Labs' research further reveals that at least 40% of IIoT systems were affected by cyber-attacks in the last half of 2019 alone [8]. The types of attacks can vary significantly from data breaches and information theft to more sophisticated forms like Denial of Service (DoS) attacks and Advanced Persistent Threats (APT), posing a threat not just to data integrity but also to physical safety and operational continuity [9]. A case in point is the 2015 Ukrainian power grid attack, which demonstrated that cyber-attacks on IIoT could have dire real-world consequences [10]. Hence, fortifying cybersecurity measures in IIoT environments has become a subject of paramount importance for both sustaining business competitiveness and ensuring national security.

In this work, an AI-based detection and classification model has been developed to detect and classify nine different cyber-attacks on IIoT devices. This paper reported the use of GA and DL to develop the detection and classification model. The major scientific contributions of this work are as follows:

- *Development of a novel detection model* The study proposes a Deep Learning-based detection and classification model specifically tailored for IIoT environments. The model is designed to operate efficiently within the resource constraints typical of IIoT networks, without compromising on accuracy.
- *Advanced feature selection and hyperparameter tuning* The study introduces a sophisticated feature selection process and hyperparameter tuning methodology that optimises the model's performance. This approach enhances the model's precision in identifying malicious activities while reducing False Positives.
- *Comprehensive evaluation using real-world data* The proposed model is rigorously evaluated using real-world IIoT network traffic, encompassing a variety of attack types and network conditions. This evaluation demonstrates the model's adaptability and robustness in diverse and dynamic environments.
- *Contribution to the understanding of feature importance* The work provides detailed insights into the importance of specific features within the IIoT context and their role in improving detection accuracy. These insights contribute to the broader understanding of feature significance in cybersecurity models.
- *Demonstration of model adaptability* The research highlights the model's ability to adapt to different types of cyber-attacks, showcasing its potential for deployment in real-world IIoT systems where adaptability is crucial.

The remainder of this paper is structured as follows: Sect. 2 (Related Work) provides a comprehensive review of the related work, highlighting existing methods and their limitations in the context of IIoT cyber-attack detection. Section 3 (Methodology) details the methodology employed in this study, including the design of the detection model, the feature selection process, and the hyperparameter tuning approach. In Sect. 4 (Results), the experimental setup and results were presented. This section includes a thorough analysis of the model's performance across various metrics, with a focus on accuracy, precision, and adaptability. Section 5 (Work Scope and Limitations) explores the implications of the findings, comparing the proposed model to existing approaches and considering potential applications in real-world IIoT environments. Finally, Sect. 6 (Conclusion and Future Work) concludes the paper by summarising the key contributions and findings, and it outlines directions for future research to further enhance the model's capabilities.

2 Related Work

2.1 Features Selection

In contemporary contexts, the management of Big Data for machine learning applications has become increasingly vital across multiple sectors, including cybersecurity. The burgeoning data within the cybersecurity domain necessitate both effective and efficient management strategies. For dealing with high-dimensional datasets, both data mining (DM) and machine learning (ML) techniques are geared towards feature reduction to extract valuable insights. One significant challenge to implementing these DM and ML methodologies is the so-called "curse of dimensionality", which alludes to the complications arising when data are dispersed in

a high-dimensional space, thereby affecting learning algorithms initially designed for lower-dimensional spaces [11]. Another pressing concern is the phenomenon of overfitting, which negatively impacts the accuracy of machine learning models, particularly when the datasets are feature rich. Excessive features also result in increased computational costs and memory requirements. A prevailing solution to the high-dimensionality challenge involves dimensionality reduction, which can be effectively achieved through feature selection techniques [12–15]. This strategy transforms a high-dimensional feature space into a lower-dimensional space by selecting a subset of the most relevant features. In real-world datasets, noise often introduces redundant and irrelevant features. The removal of such noise can substantially improve both the learning rate and the detection accuracy of classifiers, simultaneously reducing False Positives (FPs) and False Negatives (FNs) [16].

Feature selection methodologies can be broadly categorised into supervised and unsupervised techniques, including filter methods, wrapper methods, and embedded methods. Filter methods, such as information gain and Chi-square, are computationally efficient and easy to implement but may overlook feature interactions. Wrapper methods, like recursive feature elimination (RFE), offer higher accuracy by evaluating feature subsets using a learning algorithm, though they are computationally intensive. Embedded methods combine feature selection with model training, balancing between efficiency and accuracy by integrating selection directly into the learning process. The primary objective of these approaches is to pick a subset of features from the original feature set based on their capability to distinguish among different data classes or to make estimations in regression analysis. Conversely, unsupervised feature selection mechanisms are primarily employed for clustering tasks. Unlike supervised approaches, where the relevance of features is determined based on their correlation with class labels, unsupervised techniques utilise alternative criteria to ascertain feature relevance [16].

2.2 Genetic Algorithm

GA is a search heuristic inspired by natural selection and genetics principles, widely utilised for optimisation and search problems. In IIoT security, GA is particularly effective for feature selection in Intrusion Detection Systems (IDS). High dimensionality in IIoT network data often complicates the efficacy of IDS. Feature selection via GA can systematically reduce this dimensionality by selecting the most relevant attributes, thereby improving the classification accuracy of cyber-threats. This makes GA an invaluable tool for enhancing the security posture of IIoT systems by fine-tuning the capabilities of IDS.

In the research conducted by Stein et al. [17], a GA-based feature selection mechanism is introduced, wherein Decision Trees (DTs) are employed in conjunction with GA. The authors utilise the KDD dataset, implementing GA for feature selection and DTs for classification of the test data. An initial population is generated randomly and subjected to ranked-based selection and two-point crossover to yield a new population. Bit-level mutation is executed on the offspring, with the two elite parents retained and the remaining population replaced. The fitness of individual chromosomes is assessed using the sum of the validation error rates as the fitness function. The study identifies 32 optimum features from the 41 attributes available in the KDD dataset (22% reduction in features). Conversely, Ho endeavours to create an anomaly-based Intrusion Detection System (IDS) that employs an unsupervised learning approach, leveraging a bio-inspired stochastic clustering model known as the Ant Colony Clustering Model (ACCM) [16]. In the realm of supervised learning, a multi-objective genetic fuzzy intrusion detection mechanism is proposed. This technique serves as a genetic feature selection wrapper, searching for an optimal subset of features that encapsulate the majority of relevant information. The study evaluates detection accuracy based on 27 optimal features out of the 41 in the KDD dataset (34% reduction in features) and reports an accuracy rate of 99% for network traffic data.

In the study conducted by Ahmad et al. [18], a multifaceted feature selection methodology is introduced, incorporating GA, principal component analysis (PCA), and multilayer perceptron (MLP) for intrusion detection tasks. The primary objective of this research is to improve classifier detection rates in the context of intrusion detection. The authors benchmark their GA-based approach against a straightforward PCA implementation. Their GA application focuses on identifying the principal feature space that interacts optimally with the classifier. Their model's efficacy is assessed through three separate experiments, featuring subsets of 12, 20, and 27 features. The highest accuracy achieved was 99%, utilising a 12-feature subset. GAs are employed solely for feature selection in the research by Sindhu et al. [19]. The optimisation process commences with the generation of a random initial population. Subsequently, the fitness of each chromosome is calculated to produce the next generation of the population. The employed fitness function incorporates feature count, sensitivity, and specificity to evaluate the merits of the feature subset. When contrasted with alternative feature selection approaches, their methodology achieves an accuracy of 98% using a subset of 16 features, extracted from the 41 original attributes present in the KDD dataset.

In the research by Kuang et al. [20], a novel Support Vector Machine (SVM) model incorporating Kernel Principal Component Analysis (KPCA) and GA is introduced. This model is designed for the identification of both normal and malicious

network traffic and is evaluated using the KDD Cup dataset. Within this framework, the GA is utilised to optimise the parameters of both the SVM and KPCA, effectively reducing the feature space's dimensionality. By employing 12 optimal features, they achieve a detection accuracy of 94% and observe rapid convergence alongside superior generalisation capabilities. Aslahi-Shahri et al. [21] assess the efficacy of the SVM classifier in intrusion detection tasks. The authors contend that SVMs, when operating in isolation, do not yield high accuracy rates. This limitation arises because SVMs require datasets exhibiting specific patterns and necessitate the selection of optimal, minimally redundant features. To address this issue, a GA is implemented to search for the optimal feature set to enhance the SVM's performance in intrusion detection tasks. Similarly, Das et al. [22] propose an ensemble feature selection methodology utilising a bi-objective Genetic Algorithm. This algorithm aims to resolve the challenges associated with optimal feature selection in data mining. The authors integrate two core principles: boundary region analysis based on rough set theory and multivariate mutual information. The effectiveness of their methodology is evaluated across several well-established datasets. Notably, when applied to the spam base dataset for email classification tasks, their method attains an impressive accuracy rate of 92%.

In the study by Gharaee and Hosseinvand [23], the authors put forth an IDS that employs GA for feature selection, complemented by a novel fitness function. Their methodology yields high predictive accuracy while maintaining a low False Positive rate. The effectiveness of their approach is validated using both the KDD and UNSW-NB 15 datasets, and they provide detailed accuracy metrics for each class. In addition, they create individual datasets for each class to apply their technique and report their findings accordingly. Yousefi-Azar et al. [24] propose the incorporation of autoencoders as generative models for feature learning. The authors elucidate the autoencoder's capability to grasp the latent features and semantic associations among dataset features. Their methodology is examined for both intrusion detection and malware classification tasks, utilising the KDD Cup dataset and the Microsoft Malware Classification Challenge (BIG 2015) dataset. In the context of intrusion detection, their approach, when combined with a Gaussian naïve Bayes classifier, reports an accuracy rate of 83.3%.

The work presented by Tsang Chi Ho [16] focuses on improving Intrusion Detection Systems using an Ant Colony Clustering Model (ACCM) and a genetic–fuzzy approach. ACCM addresses clustering challenges in network traffic, showing a detection rate of up to 99.24% on the KDD-Cup99 dataset. The genetic–fuzzy method automates intrusion signature development. Despite their effectiveness, these approaches face challenges like computational complexity and the need for balance between accuracy and interpretability. The study by Iftikhar Ahmad et al. [18] on intrusion detection using MLP and feature subset selection reports an enhanced detection rate with a reduced feature set. The research, using the KDD-cup99 dataset, shows that using just 12 features as opposed to the full set, an accuracy of 99% was achieved. However, the limitation of this work is that it did not fully represent all types of network intrusions, potentially affecting the generalisability of the findings. The authors in [25] explored the development of a lightweight Intrusion Detection System (IDS) for networks. The focus is on three aspects: data preprocessing, feature selection using a wrapper-based algorithm, and classification through a neural network ensemble Decision Tree. The IDS optimises specificity and sensitivity, emphasising soft computing techniques for fault tolerance and adaptability. However, their work has some limitations that include potential bias in the training process due to the prevalence of redundant records and the challenges of continually adapting to changing network environments. The study in [26] presented a hybrid Intrusion Detection System combining Kernel Principal Component Analysis (KPCA), Support Vector Machine (SVM), and Genetic Algorithms (GA). The model leverages KPCA for feature extraction, improving SVM's performance in classifying network intrusions. Enhanced with N-RBF kernel function, the SVM classifier is optimised using GA for its parameters, leading to reduced training time and improved detection accuracy. The study highlights the effectiveness of combining KPCA, SVM, and GA in intrusion detection, promising better generalisation and reduced training time. However, it acknowledges the need for further algorithm development for online intrusion detection and optimisation methods.

Recent advancements in the detection and classification of malicious activities within IIoT systems have increasingly used Deep Learning techniques due to their ability to manage the complexity and scale of industrial networks. Ibor et al. [25] introduced the AdacDeep model, which integrates an enhanced Genetic Algorithm with Deep Learning to predict cyber-attacks. Their approach is to anticipate potential threats before they manifest. The study claims applicability across various domains, but it does not provide sufficient empirical evidence to support this generalisation. The model's effectiveness may vary significantly across different environments, especially those with distinct data characteristics or operational constraints. Without extensive cross-domain validation, the model's generalisability remains questionable. Furthermore, Srivastava et al. proposed a hybrid model that combines Artificial Neural Networks with Genetic Algorithms (ANN-GA) for detecting cyber-attacks in IoT environments [26]. Their model focuses on optimising the hyperparameters of the ANN using Genetic Algorithms, thereby improving the detection accuracy, particularly in identifying Distributed Denial of Service (DDoS). Although the model shows high accuracy in detecting DDoS attacks,

Table 1 Comparison of current work with the literature

Ref	Dataset	Method	No. of features	Classifier	Measures (%)
[16]	KDD CPU	GA	72	Genetic Fuzzy Rule	Accuracy: 99.24 Precision: 81.91 Recall: 83.30
[18]	KDD CPU	GA + PCA	12	MLP	Accuracy: 99 Precision: - Recall: -
[27]	KDD CPU	GA	16	Decision Tree	Accuracy: 98.38 Precision: 96 Recall: 91.1
[28]	KDD CPU	GA	12	Multilayer SVM	Accuracy: 94.22 Precision: - Recall: -
[29]	KDD CPU	GA + SVM	10	SVM	Accuracy: 97 Precision: 94.1 Recall: 93.1
[30]	Spambase	GA	–	Decision Tree	Accuracy: 92.6 Precision: - Recall: 91
[31]	KDD CPU	Auto-Encoder	5	–	Accuracy: 96.3 Precision: - Recall: 91
This work	UNSW-NB 15	Integrated GA with Deep Learning (DL)	18	DL Model	Accuracy: 97 Precision: 98 Recall: 94

it may not perform as effectively against other types of cyber-threats that are prevalent in IoT environments, such as ransomware, data exfiltration, or insider threats. The study does not sufficiently explore the versatility of the ANN-GA model in identifying a wider range of attack vectors, which limits the comprehensiveness of its contribution to IoT security. Table 1 summarises the important aspects of these previous works in comparison with this work.

In the research conducted by Tahir et al. [32], a feature selection methodology predicated on GA is advanced, augmented by the integration of chaotic maps. This strategy is rigorously evaluated using data sourced from the fields of affective computing [33] and healthcare systems. Specifically, chaotic maps are utilised to enrich the initial population of the GA, which subsequently undergoes reproduction operations to yield an optimum feature set. The method's efficacy is assessed, particularly in the context of a seven-class emotion identification challenge.

Concurrently, the work of Viharos et al. [34] concentrates on the amalgamation of multiple feature selection techniques with the objective of achieving an optimal attribute identification strategy. Their proposal essentially embodies a synthesis of diverse feature selection paradigms aimed at generating a more generalised solution. To substantiate their approach, experiments are performed using datasets from the UCI repository as well as other real-world datasets. Nouri-Moghaddam et al. [35] propose a wrapper feature selection algorithm predicated on a multi-objective forest optimisation scheme, termed as Multi-Objective Wrapper Method based on Forest Optimisation (MOFOA). The Pareto front within their optimisation framework is meticulously maintained through an archival, grid, and region-based selection strategy. Their empirical evaluation is not only confined to UCI repository data but also extends to two specific microarray datasets [36].

Li et al. [37] introduce a feature selection technique tailored for network intrusion identification, leveraging the Krill Herd (KH) algorithm, a paradigm from the swarm intelligence domain. Their solution employs linear nearest neighbour lasso step optimisation to iteratively update the position of the krill herd within the search space, thereby facilitating the identification of a global optimal solution. Lastly, Dwivedi et al. [38] also contribute to the field of Intrusion Detection Systems (IDS) through a swarm intelligence-based approach. They incorporate the grasshopper algorithm, another instance of swarm intelligence, and integrate it with an ensemble feature selection method to bolster the system's performance. Sumaiya Thaseen et al. [39] delineate an integrated Intrusion Detection System that exploits a correlation-based attribute selection algorithm conjoined with an Artificial Neural Network. This system embodies a machine learning-based framework [40] tailored for Intrusion Detection Systems (IDS). Their correlation-based feature selection mechanism ranks attributes predicated on

the degree of correlation between each attribute and the ground truth. Lastly, Mahindru and Sangal [41] offer a feature selection technique designed for Android malware detection, grounded in machine learning principles. Their computational module relies on the Least Square Support Vector Machine (LSSVM), which is subsequently evaluated across three distinct kernel functions: linear, radial basis function, and polynomial. For empirical validation, the study utilises a corpus of two million unique Android applications.

2.3 Deep Learning

The application of Deep Learning algorithms for detecting and classifying cyber-attacks in Industrial Internet of Things (IIoT) systems is an area of burgeoning research interest, yet it remains fraught with challenges [42]. These algorithms offer notable advantages such as high accuracy rates and the ability to process complex and voluminous datasets. However, there exists a critical debate surrounding their efficacy and adaptability in IIoT environments. The computational overhead and latency involved in running Deep Learning models might be infeasible in real-time IIoT systems requiring immediate threat detection [43]. Additionally, these models are susceptible to adversarial attacks, thereby undermining their reliability. Lakshmanna et al further add to the scepticism by raising concerns about the model interpretability, an issue that can significantly impede trust in such systems [44]. With regard to IIoT security, Deep Learning offers significant advantages for the detection of cyber-attacks. Leveraging intricate computational models, Deep Learning algorithms are adept at discerning complex patterns in high-dimensional data, thereby enhancing detection accuracy and reducing False Positives. This capability is crucial for IIoT environments, where the identification of cyber-threats is paramount for safeguarding critical industrial processes.

Mudassir et al. explored the efficacy of multilayer Deep Learning models in detecting botnet attacks within IIoT systems [45]. Their research emphasises the use of Artificial Neural Networks (ANN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU) to classify IIoT malware. This study underscores the advantages of using Deep Learning to capture temporal dependencies and complex patterns in network traffic, which are often indicative of malicious activities. The use of these models in real-time detection scenarios demonstrates their potential in mitigating threats in critical industrial applications. Although the study focuses on botnet detection in IIoT systems, it does not sufficiently address the heterogeneity of IIoT networks. Different industrial sectors may employ distinct communication protocols, data structures, and operational constraints, which could significantly impact the performance of the proposed models.

Traditional cyber-attack detection techniques, such as signature-based detection, rely on predefined attack signatures and offer fast detection speeds but struggle with identifying novel threats. Anomaly-based detection can identify new types of attacks by flagging deviations from normal behaviour, yet it often suffers from higher False Positive rates. Deep Learning approaches, while offering higher accuracy and the ability to process complex, high-dimensional data, face challenges in IIoT environments due to their computational overhead and potential susceptibility to adversarial attacks.

To sum up, in IIoT cybersecurity, the combination of Genetic Algorithms (GA) and Deep Learning (DL) presents a pivotal advancement. GAs, with their proficiency in efficiently sifting through complex feature spaces, significantly boost the accuracy of cyber-threat detection in IIoT environments. This enhanced detection is achieved with reduced computational power, as GAs facilitate the identification of critical features, thus streamlining the process. Concurrently, DL brings its advanced classification capabilities by discerning the nuanced characteristics of diverse cyber-threats. This approach is crafted to address the dynamic and evolving security challenges of IIoT. It leads to a paradigm shift towards AI-driven, robust cybersecurity frameworks, offering an industrially viable solution that navigates the varied and complex nature of cyber-threats in IIoT settings. This methodology is thus not only a response to current cybersecurity demands but also a proactive measure against future threats in the IIoT domain.

3 Methodology

3.1 Problem Formulation

In IIoT, Intrusion Detection Systems (IDS) stand as vital tools for recognising and counteracting cyber-vulnerabilities. These systems are critically important for differentiating the spectrum of cyber-threats that have the potential to disrupt industrial functionalities. Both Deep Learning (DL) and Genetic Algorithms (GA) have shown considerable promise for bolstering IDS capabilities within IIoT contexts. However, the intrinsic high dimensionality of IIoT network data can negatively influence the accuracy of machine learning models, particularly those based on DL and GA, utilised for these crucial classification tasks. A refined feature selection strategy is thus imperative, aimed not solely at mitigating the dimensionality but also at enhancing the classification of a range of cyber-threats. This work focuses on this challenge by proposing an advanced feature selection algorithm, which synthesises both Genetic Algorithms (GA) and Deep Learning (DL). The aim of this work is to elevate classification

performance, characterised by increased rates of True Positives (TP) and minimised instances of False Positives (FP), thereby solidifying the security robustness of IIoT networks.

3.2 Dataset and Preprocessing

3.2.1 Dataset

The UNSW-NB15 dataset [46], generated within the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS), employs the IXIA PerfectStorm tool to create a blend of genuine contemporary regular activities and synthesised modern attack patterns. Traffic capture, encompassing 100 GB of raw data (e.g. Pcap files), is facilitated by the Tcpdump tool. The UNSW-NB15 dataset was specifically designed to address the limitations of older datasets like KDD Cup. It includes a wider variety of attack types and more realistic traffic data, making it better suited for evaluating the effectiveness of Intrusion Detection Systems in current network environments. The dataset includes up-to-date and realistic traffic patterns and attack scenarios. The impact of choosing UNSW-NB15 dataset on the results in this work is as follows:

1. Accuracy and Precision
   The choice of the UNSW-NB15 dataset has a significant impact on the accuracy and precision of our detection model. Given that this dataset includes more recent and sophisticated attack patterns, the accuracy and precision metrics obtained from our model are more reflective of its performance in a real-world IIoT setting. The inclusion of modern attack types like infiltration, backdoors, and botnets, which are absent or underrepresented in the KDD Cup dataset, ensures that our model is rigorously tested against contemporary threats, leading to a more reliable assessment of its detection capabilities.

2. Loss
   The impact on the loss metric is also noteworthy. The diversity and complexity of the attacks in the UNSW-NB15 dataset result in a more challenging classification task for the model. This challenge is reflected in the loss values, which may be higher than those obtained with older datasets due to the increased difficulty of correctly classifying a broader spectrum of attack types. However, this also means that the model's performance metrics, including loss, are more indicative of its robustness and ability to handle real-world network traffic.

3. Generalisability
   By using the UNSW-NB15 dataset, we enhance the generalisability of our results. The dataset's modern and comprehensive nature means that the findings from our study are more likely to be applicable to current and future network environments, particularly in IIoT contexts.

This dataset categorises attacks into nine distinct families:

(i) *Fuzzers* Characterised by protocol fuzzing, fuzzers might act as proxies, modifying packets instantaneously and echoing them back. Often, fuzzers induce crashes or Denial of Service (DoS), enabling attackers to take control of IoT devices, irrespective of encryption measures in place. Examples encompass malformed packets and stack overflows.

(ii) *Backdoors* This refers to malicious software enabling hackers unauthorised access to systems. Typically installed via vulnerable network access points, like outdated plugins, these stealthy attacks often remain unnoticed.

(iii) *Analysis* Sometimes termed active surveillance, analysis attacks involve cybercriminals actively probing a target to gather intelligence about potential vulnerabilities. Analogous to military reconnaissance, in the cybersecurity realm, it often serves as a precursor to a more targeted attack. Port scanning is a typical method employed to identify open and susceptible ports.

(iv) *Denial of Service (DoS)* Such attacks saturate network resources, isolating IoT devices from their intended users. Distributed DoS (DDoS) incidents in the IoT context often involve numerous attackers targeting a singular IoT device.

(v) *Shellcode* Contrary to its name suggesting a relation to shell scripting, shellcode is a botnet attack vector wherein attackers aim to document code executed by a susceptible compiled program, subsequently initiating a remote terminal session. Once achieved, this remote access can further compromise the system.

(vi) *Reconnaissance Attacks* Regarded as broad informational gathering attacks, they can manifest either virtually or physically. Tactics such as traffic analysis and packet sniffing are typical of this category.

(vii) *Worms* Essentially, a computer worm is self-replicating malicious software that spreads across IoT devices autonomously. It can proliferate without attaching to any host program. An illustrative analogy would be an individual continually evading detection.

(viii) *Generic* Predominantly cipher-based, generic attacks predominantly focus on deciphering cryptographic secret keys. These attacks can often function independently without specific implementation details. Techniques such as linear and differential cryptanalysis and correlation attacks typify this category.

(ix) *Exploits* Representing malware variants that capitalise on known vulnerabilities in application software or operating systems, a classic example includes vulnerabilities within Microsoft Office.

To extract 49 features (shown in Table 2), inclusive of the class label, tools such as Argus and Bro-IDS are deployed alongside twelve custom-developed algorithms. Cumulatively, the dataset comprises 2,540,044 records. To evaluate the performance of our model, the UNSW-NB15 dataset, which consists of 2,540,044 records, was randomly split into a training set and a testing set. Specifically, 80% of the dataset, comprising 2,032,035 entries, was used for training the model, while the remaining 20%, consisting of 508,009 entries, was reserved for testing. These entries can be categorised into two primary classes: attack and benign.

Table 3 shows the number of instances of every attack in the dataset.

3.2.2 Dataset Preprocessing

The dataset went through several data preprocessing steps before using it for training and testing the attacks detection and classification model. The preprocessing steps are explained below:

3.2.2.1 Preprocess Empty Cells In this step, a systematic search through the data dataset was conducted to identify entries that are either blank strings or solely composed of space. These specific entries are then substituted with NaN (Not a Number) values. Following this initial transformation, the dataset might encompass NaN values. A function was developed to locate all instances of these NaN values and replace them with a numeric value, specifically 0.

3.2.2.2 Preprocess the Attacks Category Column This column represents the category of the cyber-attacks. However, it is not well document in the original dataset. A special code was developed to preprocess this column so that it is ready for model development as follows:

- *String trimming* this step is used to remove any leading or trailing white spaces from each entry within this column. This step is imperative for ensuring data consistency and mitigating any discrepancies arising from unintentional spacing.
- *Filling missing entries* subsequent to the string trimming operation, this column is scanned for any instances of missing values or NaN (Not a Number) entries. All discovered instances are then replaced with the string 'Normal'.

Table 2 Dataset features

N	Feature name	Type	Description
1	srcip	Nominal	Source IP address
2	sport	Integer	Source port number
3	dstip	Nominal	Destination IP address
4	dsport	Integer	Destination port number
5	proto	Nominal	Transaction protocol
6	state	Nominal	Indicates to the state and its dependent protocol, e.g. ACC, CLO, and (-) (if not used)
7	dur	Float	Record total duration
8	sbytes	Integer	Source to transaction bytes
9	dbytes	Integer	Destination to source transaction bytes
10	sttl	Integer	Source to destination time to live value
11	dttl	Integer	Destination to source time to live value
12	sloss	Integer	Source packets retransmitted or dropped
13	dloss	Integer	Destination packets retransmitted or dropped
14	service	Nominal	http, ftp, smtp, ssh, dns, ftp-data,irc and (-) if not much used service
15	Sload	Float	Source bits per second
16	Dload	Float	Destination bits per second
17	Spkts	Integer	Source to destination packet count
18	Dpkts	Integer	Destination to source packet count
19	swin	Integer	Source TCP window advertisement value
20	dwin	Integer	Destination TCP window advertisement value
21	stcpb	Integer	Source TCP base sequence number
22	dtcpb	Integer	Destination TCP base sequence number
23	smeansz	Integer	Mean of the packet size transmitted by the src

Table 2 (continued)

N	Feature name	Type	Description
24	dmeansz	Integer	Mean of the packet size transmitted by the dst
25	trans_depth	Integer	Represents the pipelined depth into the connection of http request/response transaction
26	res_bdy_len	Integer	Actual uncompressed content size of the data transferred from the server's http service
27	Sjit	Float	Source jitter (mSec)
28	Djit	Float	Destination jitter (mSec)
29	Stime	Timestamp	record start time
30	Ltime	Timestamp	record last time
31	Sintpkt	Float	Source interpacket arrival time (mSec)
32	Dintpkt	Float	Destination interpacket arrival time (mSec)
33	tcprtt	Float	TCP connection setup round-trip time, the sum of' synack' and' ackdat'
34	synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets
35	ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets
36	is_sm_ips_ports	Binary	If source (1) and destination (3) IP addresses equal and port numbers (2)(4) equal then this variable takes value 1 else 0
37	ct_state_ttl	Integer	No. for each state (6) according to specific range of values for source/destination time to live
38	ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service

Table 2 (continued)

N	Feature name	Type	Description
39	is_ftp_login	Binary	If the ftp session is accessed by user and password 1 else 0
40	ct_ftp_cmd	Integer	No. of flows that has a command in ftp session
41	ct_srv_src	Integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26)
42	ct_srv_dst	Integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26)
43	ct_dst_ltm	Integer	No. of connections of the same destination address in 100 connections according to last time
44	ct_src_ ltm	Integer	No. of connections of the same source address (1) in 100 connections according to the last time (26)
45	ct_src_dport_ltm	Integer	No. of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26)
46	ct_dst_sport_ltm	Integer	No. of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26)
47	ct_dst_src_ltm	Integer	No of connections of the same source (1) and the destination (3) address in 100 connections according to the last time (26)
48	attack_cat	Nominal	The name of each attack category

Table 2 (continued)

N	Feature name	Type	Description
49	Label	Binary	0 for normal and 1 for attack records

Table 3 UNSW-NB-15 dataset

Attack type	Number of records
Generic	61,878
Exploits	11,439
Fuzzers	5390
DoS	4907
Reconnaissance	3530
Analysis	670
Backdoor	666
Shellcode	378
Worms	44

3.2.2.3 Preprocess Hexadecimal Values In this step, a function was developed to adeptly identify and transfigure hexadecimal strings into their integer equivalents, while leaving other input values unaffected. Such a function is instrumental in this work where hexadecimal representations might be interspersed amidst other data types, necessitating uniformity for further analysis.

3.2.2.4 Encode Nominal Features Nominal features refer to categorical data that lacks an intrinsic order. Encoding such features is a crucial step in preparing data for various algorithms which require numerical input values. All the nominal features within the dataset have been efficiently converted from their original categorical form into a numeric format.

3.2.2.5 Normalise Dataset Data normalisation is an essential preprocessing step in data analysis and machine learning, ensuring that numerical attributes have the same scale, thus preventing features with larger scales from disproportionately influencing the outcomes of certain algorithms. The objective is to scale numerical attributes in the dataset to a uniform range, typically between 0 and 1.

3.2.2.6 Split Dataset In this stage, two dataset splits have been performed as follow:

- *First, the dataset was split into two sets* the features set, which represents the input to the classification model, and the label set which contains the types of the attacks.
- Second, the dataset is randomly split into 80% training set and 20% testing set.

3.2.2.7 Encode Categorical Variables In the context of neural networks and Deep Learning models, the representation of target variables is paramount to achieving optimal training and prediction performance. One widespread technique is the utilisation of 'one-hot encoding', which converts categorical target variables into a binary matrix format, where each label is mapped to a unique binary vector. For instance, given three distinct classes A, B, and C, class A might be represented as [1, 0, 0], class B as [0, 1, 0], and class C as [0, 0, 1]. In essence, the conversion to one-hot encoding is pivotal, especially for multi-class classification tasks in neural networks. It facilitates the network's ability to distinctly classify instances into their respective classes, ensuring that each class is treated as an independent entity. By transforming the target variables in this manner, the neural network can be trained more effectively, providing more granular feedback during the backpropagation process and fostering improved model accuracy.

3.3 Integrating Deep Learning and Genetic Algorithm

This approach represents the integration of Genetic Algorithm (GA) operations with a Deep Neural Network (DNN) for the dual objectives of optimising feature selection and enhancing model accuracy. Initially, a DNN model is defined. The architecture of this model consists of several layers. The input layer has neurons equivalent to the number of selected features and utilises a Rectified Linear Unit (ReLU) activation function. This is followed by a number of hidden layers containing neurons, also adopting a ReLU activation function. The output layer employs a softmax activation function tailored to cater to multiple classes, ensuring the resultant probabilities range between 0 and 1 and sum to unity. The model is compiled using the categorical cross-entropy loss function, which is particularly suitable for multi-class classification, and the Adam optimiser, appreciated for its computational efficiency. During the fitness evaluation, each individual in the GA's population represents a binary-encoded feature selection. The function determines the features denoted by '1' in the individual and deploys the DNN model on them. The resulting model accuracy, when evaluated against a test dataset, serves as the fitness score.

The DEAP toolbox, a renowned framework in evolutionary computation, facilitates the GA operations. Each individual's genome length corresponds to the total number of features, and standard GA operations—such as two-point crossover, bit-flipping mutation, and tournament selection—are employed to evolve the population over defined generations. Upon algorithmic completion, the most optimal individual—i.e. the feature set providing the highest validation accuracy when paired with the DNN—is identified. This prime feature set, its cardinality, and the associated

DNN accuracy are reported as the algorithm's output. This holistic approach marries the global search prowess of the GA with the predictive strength of DNNs, thus paving the way for more informative and concise feature subsets and potentially enhanced classification outcomes. This integrated model leverages the strength of GAs in optimising feature selection and the prowess of DL in pattern recognition and classification.

(A) Genetic Algorithm (GA) for Feature Selection.
(A-1) Population Initialisation.

- Let $P(t)$ represent the population at generation t, where each individual $Ii \in P(t)$ is a binary vector of length N, representing the presence (1) or absence (0) of features in the dataset.

(A-2) Fitness Function.

- The fitness function $f(Ii)$ quantifies the effectiveness of the feature subset represented by individual Ii using the performance of the DL model.
- Mathematically, $f(Ii) = \text{Accuracy}(DNN(X_{Ii},Y))$, where X_{Ii} is the dataset restricted to features marked by '1' in Ii, and Y is the set of labels.

(A-3) Genetic Operations.

- *Crossover* $C(Ii,Ij) \rightarrow I_k$: A pair of parent individuals Ii,Ij produce an offspring I_k via a crossover mechanism (e.g. two-point crossover).
- *Mutation* $M(Ii) \rightarrow Im$: An individual Ii undergoes mutation to produce Im, typically through bit-flipping with a predefined probability.

(A-4) Selection.
A selection mechanism $S(P(t)) \rightarrow P(t+1)$ is employed to choose individuals for the next generation, often based on fitness values, such as tournament selection.

(B) Deep Learning (DL) Model for Classification.
(B-1) Model Architecture.

- The DL model is defined with an architecture suitable for classification tasks:
  - *Input layer* Receives input of dimension equal to the number of features GA selects.
  - *Hidden layers* Multiple layers with a predefined number of neurons, using Rectified Linear Unit (ReLU) activation functions.
  - *Output layer* Consists of neurons equal to the number of classes, employing a softmax activation function.

(B-2) Model Training.

- The DL model is trained on a subset of features X_{Ii}, where Ii is an individual from the GA population.
- Loss Function: The model uses categorical cross-entropy loss $L(Y, \hat{Y}) = -\sum_{c=1}^{M} Y_{o,c} \log\left(\hat{Y}_{o,c}\right)$, where Y and $\hat{Y}_{0,c}$ represent actual and predicted labels, and M is the number of classes.
- Optimiser: Adam optimiser is employed for its efficiency in converging towards optimal weights.

(C) The Integrated Model: GA–DL Algorithm
(C-1) Initialisation:

- Step 1: Initialise GA population $P(0)$ with binary vectors of length N.
- Step 2: Define the DL model architecture DNN.

(C-2) Evolutionary Process:

- Step 3: For each generation t, perform:
  - Step 3.1: For each individual Ii in $P(t)$:
    - Step 3.1.1: Construct dataset X_{Ii} based on selected features in Ii.
    - Step 3.1.2: Train DNN on X_{Ii} and evaluate performance on a validation set.
    - Step 3.1.3: Assign fitness score $f(Ii)$ based on the accuracy of DNN.
  - Step 3.2: Apply crossover and mutation to generate new individuals.
  - Step 3.3: Perform selection to form $P(t+1)$.

(C-3) Termination:

- Step 4: Repeat the process for a predetermined number of generations or until convergence criteria are met.
- Step 5: Identify the best individual I_{best} from the final generation.

(C-4) Output:

- Step 6: The feature subset represented by I_{best} and the accuracy of the DL model trained on this subset are presented as the output of the algorithm.

This integrated GA–DL model adeptly marries the global search capability of GAs in navigating through the high-dimensional feature space with the predictive strength of DNNs. The result is a sophisticated tool capable of identifying salient features and achieving high classification

accuracy, crucial for effective and reliable cyber-attack detection in IIoT systems. This model exemplifies a significant advancement in computational intelligence approaches applied to cybersecurity.

The pseudocode below represents the integrated approach:

FUNCTION CreateModel(input_dim: Integer)—> Model:

- INITIALISE a Sequential Neural Network model.
- ADD an Input Layer to the model with 'input_dim' neurons and ReLU activation.
- ADD Hidden Layers to the model with X neurons and ReLU activation.
- ADD an Output Layer to the model with the number of output classes and Softmax activation.
- COMPILE the model with 'categorical_crossentropy' loss, 'adam' optimiser, and 'accuracy' metric.
- RETURN the compiled model.

FUNCTION EvaluateIndividual(individual: List) -> Tuple:

- INITIALISE selected_features as the subset of features where the corresponding gene in individual is '1'.
- INITIALISE a model using CreateModel with input_dim equal to the length of selected_features.
- TRAIN the model on X_train[selected_features] and y_train_one_hot for XX epochs.
- EVALUATE the model on X_test[selected_features] and y_test_one_hot.
- RETURN the accuracy as the fitness score.

BEGIN MAIN:

1. ***INITIALISE*** *the Genetic Algorithm (GA) toolbox.*
2. ***DEFINE*** *data structures for Fitness and Individual, where Fitness is maximized.*
3. ***REGISTER*** *methods for initializing the population and defining genetic operators, including evaluation, crossover, mutation, and selection.*
4. ***INITIALISE*** *the population with N individuals.*
5. ***FOR*** *generation in 1 to G generations:*
   FOR *EACH individual in population:*
   EVALUATE *individual fitness using EvaluateIndividual function which:*
   - ***SELECTS*** *features based on individual genes*
   - ***BUILDS*** *and* ***TRAINS*** *a DNN model on selected features*
   - ***EVALUATES*** *the DNN model and* ***RETURNS*** *accuracy as fitness*

   APPLY *crossover on individuals with probability 0.5*
   APPLY *mutation on individuals with probability 0.2*
   SELECT *new generation from current population*
6. ***SELECT*** *the best_individual from the final population.*
7. ***PRINT*** *the features of best_individual, the total number of selected features, and the fitness (accuracy) of best_individual.*

END MAIN

3.4 Hyperparameter Tuning Process

The final step in this methodology is the hyperparameter tuning process which is implemented by using Keras Tuning library. This is a crucial process to maximise the model performance. This process includes the following.

1. Objective of Hyperparameter Tuning
   The core objective of hyperparameter tuning in this context was to calibrate the model parameters to optimise its performance, specifically aiming to maximise the accuracy on the validation set. This is particularly crucial in cybersecurity applications where the precision of predictions can significantly impact the effectiveness of threat detection and response.

Table 4 Training and testing configurations

Parameter	Value/setting
Dataset split	80% Training, 20% Testing
Learning rate	Determined by the hyperparameter tuner (within a range from 0.001 to 0.01)
Cross-validation	Fivefold Cross-Validation
Number of epochs	100
Batch size	64
Optimiser	Adam
Loss function	Categorical Cross-Entropy
Activation functions	ReLU, Tanh, ELU
Metrics	Loss, Accuracy, Precision, Recall
Evaluation metrics	Test Loss, Test Accuracy, Test Precision, Test Recall

2. Keras Tuner Implementation
   The tuning process was facilitated using Keras Tuner, a specialised Python library offering a powerful and flexible platform for hyperparameter optimisation. Keras Tuner provides several tuning algorithms; in this instance, a Random Search strategy was employed for its efficiency and effectiveness in exploring a wide parameter space.

3. Parameters Under Tuning

a. *Neurons in dense layers* The number of neurons in each dense layer was varied to determine the optimal size of the neural network. This impacts the model's capacity to learn complex patterns.
b. *Activation functions* Different activation functions ('relu', 'tanh', and 'elu') were considered. The choice of activation function affects how the model processes input data and can impact the training dynamics.
c. *Dropout rate* Dropout rates were tuned to identify the optimal level of regularisation, thus preventing overfitting while maintaining sufficient model complexity.
d. *L2 regularisation rate* The extent of L2 regularisation was also a subject of tuning, which helps in penalising large weights to avoid overfitting.

4. Search Strategy
   The random search method involved exploring random combinations of hyperparameters within specified ranges. This approach, as opposed to a grid search, offers a balance between comprehensiveness and computational efficiency.

5. Trials and Executions
   The tuner was set to conduct a set of trials, with each trial executing the model training twice. This setup was chosen to ensure a thorough exploration of the hyperparameter space while also accounting for the variability in model performance due to stochastic elements in neural network training.

6. The outcome of the Tuning Process
   Post the tuning process, the best-performing hyperparameter set was extracted. This set represents the combination of parameters that resulted in the highest accuracy on the validation dataset during the trials.

7. Implications of Tuning
   By systematically adjusting and evaluating different hyperparameters, the tuning process endeavours to enhance the model's ability to detect and classify cyber-threats accurately. It plays a pivotal role in tailoring the model to the specific nuances and intricacies of the dataset at hand, which is critical in cybersecurity's dynamic and often complex landscape. In essence, hyperparameter tuning stands as a cornerstone in the model development process, especially in fields like cybersecurity, where the stakes of model performance are high. This careful calibration of parameters is instrumental in striking the right balance between model complexity, generalisation ability, and training efficiency.

The pseudocode below represents the hyperparamter tuning process:

the percentage of data allocated for training and testing, the

```
// Import necessary libraries
// Define the number of output classes based on the shape of the training labels
// Define the function to build the neural network model
FUNCTION build_model(hp: HyperParameters) -> Model:
    INITIALIZE model as a Sequential Neural Network

    // Define the first layer with dynamic units and activation
    ADD Dense Layer to model with units ranging from 128 to 512, selected activation function, and L2 regularisation
    ADD BatchNormalization layer
    ADD Dropout layer with rate ranging from 0.3 to 0.5

    // Define the second layer with dynamic units and activation
    ADD Dense Layer to model with units ranging from 64 to 256, selected activation function
    ADD BatchNormalization layer
    ADD Dropout layer with rate ranging from 0.3 to 0.5

    // Define the output layer for multi-class classification
    ADD Dense Layer to model with 'num_classes' units and Softmax activation

    // Compile the model with categorical cross-entropy loss and Adam optimiser
    COMPILE model using 'categorical_crossentropy' loss, 'adam' optimiser, and metrics including accuracy, precision, and recall

    RETURN model

//Initialise the hyperparameter tuner
INITIALIZE tuner as RandomSearch with the build_model function, objective to maximise validation accuracy, and defined number of trials and executions

// Perform hyperparameter tuning on the training data
// Retrieve the best hyperparameters from the tuning process
// Build the model using the best hyperparameters
// Retrain the model on the training data
// Evaluate the model on the test data
```

To provide a clear understanding of the operational steps involved in our proposed GA–DL (Genetic Algorithm–Deep Learning) approach, we have included a flowchart that visually represents the entire process. This flowchart outlines each stage of the methodology, starting from data preprocessing to the final evaluation and selection of the optimised model.

To ensure the reproducibility of our experiments and to provide clarity on the specific settings used during the training and testing phases of the proposed GA–DL approach, we have outlined the key configurations and parameters in Table 4. These parameters were carefully selected to optimise the performance of the Deep Learning model and to ensure that the Genetic Algorithm effectively selected the most relevant features from the dataset. The table provides details on learning rate used during model training, the cross-validation method employed, and other relevant settings. These configurations were instrumental in achieving the high levels of accuracy, precision, and recall reported in this study.

3.5 Evaluation Metrics

The evaluation of the results obtained in this work is conducted by using the following standard metrics:

3.5.1 Accuracy

In the context of a Deep Learning classification model, accuracy can be more specifically defined with reference to four

key concepts: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

- *True Positives (TP)* These are the instances where the model correctly predicts the positive class.
- *True Negatives (TN)* These are the instances where the model correctly predicts the negative class.
- *False Positives (FP)* These are the instances where the model incorrectly predicts the positive class.
- *False Negatives (FN)* These are the instances where the model incorrectly predicts the negative class.

Given these definitions, the accuracy of a classification model is the number of correct predictions (both positive and negative) divided by the total number of predictions made. This can be represented mathematically as:

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

This metric is commonly used in classification problems to determine the proportion of the total number of predictions that were correct.

3.5.2 Precision

Precision in the context of a Deep Learning classification model is a measure of the correctness achieved in positive prediction. It tells us out of all the positive classes we have predicted, how many are actually positive. Precision is a good measure to determine when the costs of False Positives are high. For example, in spam detection, it is rather good to have some spam messages get through (False Negatives) than have good emails categorised as spam (False Positives). The precision is calculated as follows:

$$\text{Precision} = \text{TP}/(\text{TP} + FP)$$

A high precision score relates to the low False Positive rate. Precision is particularly useful in a situation where False Positives are more concerning than False Negatives. It is important to note that precision alone is not a reliable metric; it should be considered in tandem with recall (also known as sensitivity) for a more comprehensive evaluation of a classification model's performance.

3.5.3 Recall

Recall, also known as sensitivity or the True Positive rate in the context of a Deep Learning classification model, measures the proportion of actual positives that are correctly identified by the model. It reflects the model's ability to find all the relevant cases within a dataset. The recall is calculated as follows:

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN})$$

A high recall score indicates that the model is capable of capturing a large proportion of the positive cases, which is particularly important in situations where failing to detect positives can have severe consequences, like in cyber-attacks detection in this work. However, a model with high recall can also have a high number of False Positives. Therefore, recall should not be used as the only performance metric but should be combined with other measures such as precision and accuracy to give a complete picture of the model's performance.

These measures are utilised in assessment of the final GA–DL classification model in this work.

4 Results

This section presents and discusses the conducted work and achieved results. The detection and classification accuracy, precision, recall, and loss of the GA–DL classifier are reported. The proposed solution is coded from scratch using Python programming language with the following libraries and editors:

- Jupyter (version: 1.0.0)
- Matplotlib (version: 3.3.2)
- Notebook (version: 6.0.3)
- Numpy (version: 1.18.1)
- Openpyxl version: 3.0.4)
- Pandas (version: 1.1.2)
- scikit-learn (version: 0.23.2)
- scipy (version: 1.4.1)
- seaborn (version: 0.11.0)

The machine used for simulations had Intel®Core TM i7 vPRO processor, Microsoft's Windows 10 operating system and RAM of 32.00 GB.

4.1 GA–DL Model Performance

4.1.1 Model Performance Without Hyperparameter Tuning

The GA–DL model performance has been analysed. As discussed in the previous section, the model performance is measured by calculating the accuracy, precision, recall, and the loss. Figure 1 shows that the model accuracy increased as the training was going. The model settled at about 97% accuracy after 100 epochs with a chance of slightly more than that accuracy if more epochs are introduced.

In IIoT systems and networks, maintaining high accuracy is vital because the cost of false predictions can be very

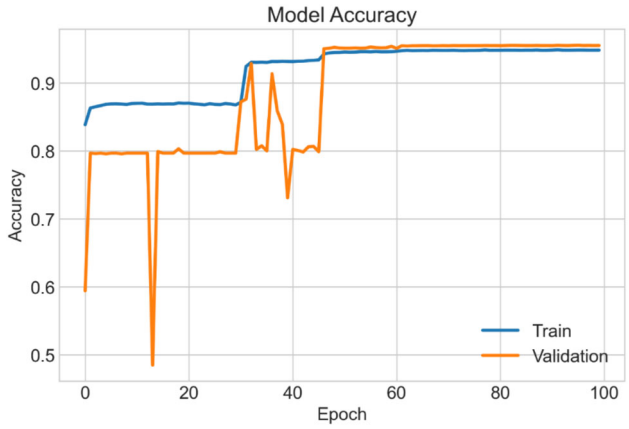


Fig. 1 GA–DL Model Accuracy

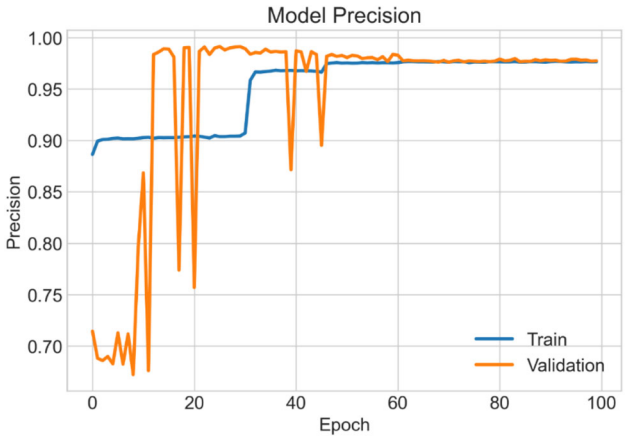


Fig. 3 GA–DL Model Precision

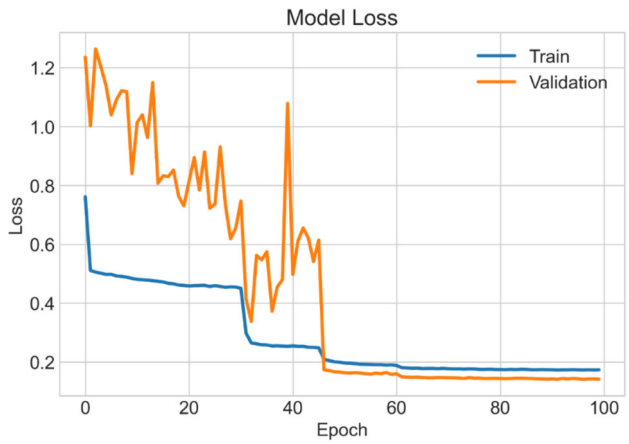


Fig. 2 GA–DL Model Loss

high. A False Negative might mean an undetected attack that could shut down critical infrastructure or even cause physical harm. Conversely, a high rate of False Positives can lead to "alert fatigue" where operators begin to ignore security alerts, potentially leading to a missed True Positive.

Figure 2 illustrates the model losses during the training and validation. In the context of a Deep Learning classification model, particularly this model that is designed for detecting and classifying cyber-attacks on IIoT systems, "loss" refers to a quantification of the difference between the model's predicted outputs and the actual ground truth values. It is a metric that summarises the model's errors made during training and validation phases. This loss function, also known as a cost function, is an essential component in training the Deep Learning model. It measures the inconsistency between the predicted values generated by the model and the actual values. In this work (IIoT cybersecurity), where the model's output can significantly impact the reliability and safety of the systems, the loss function becomes critical. The choice of a loss function depends on the nature of the problem, in this case, which is a multi-class classification (categorising different types of cyber-attacks), the function categorical cross-entropy is used.

In this specific case of IIoT, where the cyber-physical systems have various sensor readings and network traffic data as inputs, the loss function would capture how well the model is learning to differentiate between normal operational data and potential cyber-threats or anomalies. Since the cost of a False Negative (i.e. missing a real attack) could be catastrophic in a real-world industrial environment, the GA–DL model is optimised to minimise such errors, possibly at the expense of a higher False Positive rate (i.e. benign activities misclassified as attacks). The ultimate goal is to train a model with the lowest possible loss on unseen data, indicating it has a good generalisation ability and is adept at detecting and classifying cyber-threats accurately to ensure the security and integrity of IIoT systems.

Figure 3 shows the performance of the GA–DL model during the training and validation in terms of precision. Within the context of deploying the model for the discernment and classification of cyber-threats in IIoT networks, precision is a pivotal metric that gauges the veracity of the positive predictions made by the classifier. Precision is particularly imperative in scenarios where the cost of a False Positive—an erroneous alarm—is high, as such events can precipitate unnecessary and possibly disruptive responses. Precision, therefore, serves as an indicator of a model's reliability in asserting an incident as a cyber-attack and its adeptness in minimising false alarms. Given the intricate interplay between precision and recall (in this case 94%), it is paramount that precision is not evaluated in isolation. In the pursuit of a resilient and efficient IIoT security apparatus, achieving a balance between precision and recall is essential, ensuring that the Deep Learning model remains both trustworthy and comprehensive in its threat detection capabilities.

4.1.2 Model Performance with Hyperparameter Tuning

This model constructed using Keras, a high-level neural networks API, represents a sophisticated blend of machine learning techniques and hyperparameter tuning orchestrated to refine its predictive accuracy.

4.1.2.1 Model Architectural Highlights Sequential Model: The model is built using Keras's Sequential API, allowing for a linear stack of layers, which simplifies the architecture and enhances the model's interpretability.

- *Layer configuration* Each layer is carefully structured, starting with a dense layer whose neurons range between 128 and 512. This is followed by a Batch Normalisation layer, which standardises inputs to the next layer, thus accelerating the training process and improving overall model performance.
- *Hyperparameter optimisation* The model employs Keras Tuner for hyperparameter tuning, an innovative approach that systematically explores a range of configurations to identify the most effective parameters. This includes the number of neurons, the type of activation function (ReLU, tanh, or ELU), and the degree of dropout and L2 regularisation, crucial for preventing overfitting.
- *Dropout regularisation* The inclusion of dropout layers, with rates varying between 0.3 and 0.5, is a strategic design choice to reduce overfitting, ensuring the model remains robust to unseen data.

4.1.2.2 Training and Evaluation The model undergoes a rigorous training regimen over 100 epochs with a batch size of 64, including a validation split of 20%. This comprehensive training is designed to thoroughly equip the model with the ability to discern intricate patterns indicative of various cyber-threats. Upon completion of the training phase, the model's performance is evaluated against the test set, with metrics such as loss, accuracy, precision, and recall being meticulously computed. These metrics provide a multifaceted view of the model's effectiveness, with accuracy measuring overall performance, precision focusing on the correctness of positive predictions, and recall highlighting the model's ability to identify all relevant instances.

4.1.2.3 Visualisation The model's training process is visually represented through a series of plots, illustrating the trajectory of accuracy, loss, and precision over successive epochs. These plots are not merely illustrative but serve as diagnostic tools, offering insights into the model's learning process and indicating areas for potential improvement.

4.1.2.4 Performance Comparison Performing the hyperparameter tuning approach (explained in section 0) resulted in enhancing the model performance in terms of the accuracy, precision, recall, and the model losses as shown in Fig. 4.

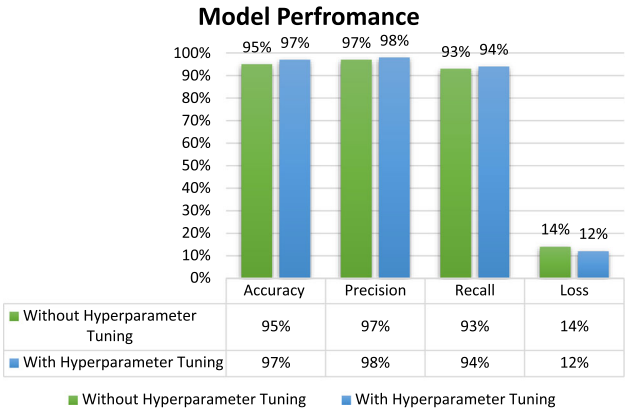


Fig. 4 Model Performance Comparison

The hyperparameter tuning resulted in:

- Increasing the model accuracy by 2% (from 95 to 97%)
- Increasing the model precision by 1% (from 97 to 98%)
- Increasing the model recall by 1% (from 93 to 94%)
- Reducing the model losses by 2% (from 14 to 12%)

Fine tuning the model and enhance its performance at these levels (e.g. 98%) is challenging, but the hyperparameter tuning is proven to be successful in enhancing the model performance on multiple levels. Figure 5 shows the model performance during the hyperparameter tuning process.

In Fig. 5, we observe that the plots related to hyperparameter tuning exhibit notable fluctuations, which are more pronounced than those seen in the plots without hyperparameter tuning. These fluctuations are a result of the complex interactions between various hyperparameters and the stochastic nature of the training process. These hyperparameters significantly influence the model's architecture and its ability to generalise from the training data. The fluctuations observed during hyperparameter tuning are indicative of the model's sensitivity to the hyperparameter settings. For instance, varying the number of units in the dense layers or changing the activation function (between relu, tanh, or elu) can lead to significant differences in how the model learns and generalises. Despite these fluctuations, the overall results after hyperparameter tuning show improved performance in terms of accuracy, precision, and recall, compared to the non-tuned model.

To address the fluctuations, we ensured that the final model was trained and evaluated multiple times to confirm the consistency of the results. Cross-validation and multiple trials help in averaging out the effects of random variability, providing a more reliable estimate of the model's true performance.

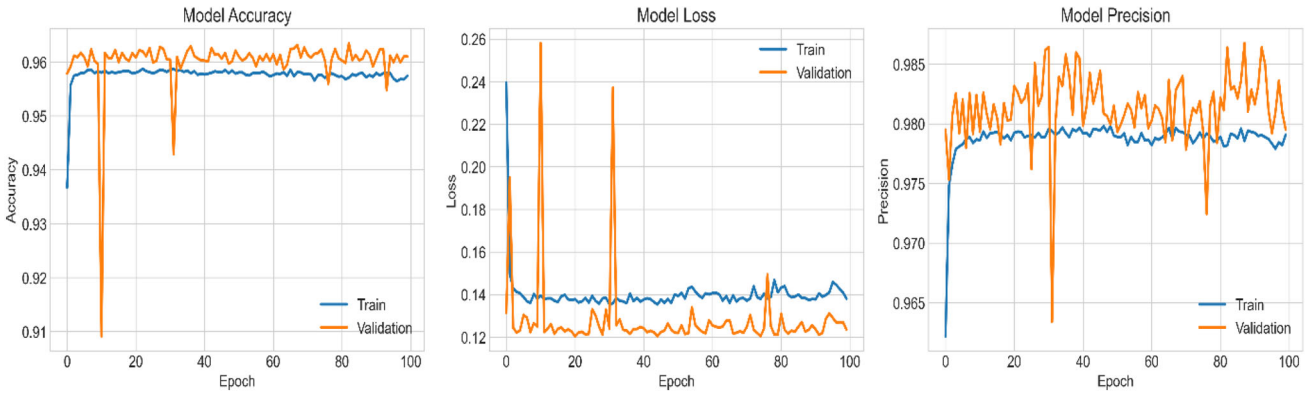


Fig. 5 Model Performance with Hyperparameter Tuning

4.2 Feature Importance and Confusion Matrix

4.2.1 Confusion Matrix

The Confusion Matrix has been employed as a pivotal analytical tool to assess the performance of the developed classification model, which is designed for detecting and classifying cyber-attacks within the IIoT environment. This matrix provides an intricate tableau, delineating not only the overall accuracy of the model but also its efficacy in correctly identifying various categories of cyber-threats. It achieves this by comparing the predicted outcomes against the actual labels, thereby elucidating four key components: True Positives, True Negatives, False Positives, and False Negatives.

The utilisation of the Confusion Matrix in our analysis has been instrumental in revealing the nuanced performance of the model across different attack types. This granular insight is crucial, given the diverse nature of cyber-threats. Importantly, the matrix has highlighted areas where the model is prone to misclassification, such as instances of false alarms and missed detections, which are critical in the context of cybersecurity. Furthermore, the Confusion Matrix has served as a guide in balancing the model's sensitivity and specificity, a balancing act of paramount importance in IIoT contexts. This balance is crucial to avoid operational disruptions due to false alarms and to ensure that real threats are not overlooked.

Two forms of confusion matrix have been produced: the first one is the classical matrix, as shown in Fig. 6 and Fig. 7), and the second one is the normalised which is primarily driven by the need for a clear, comparative understanding of the model's performance across different classes, particularly in scenarios where class imbalance is prevalent.

4.2.2 Features Importance

Following the performance evaluation of the detection and classification model, the features importance is calculated as well. The term 'features performance' refers to the efficacy and relevance of the input variables, or features, used by the classifier in making accurate predictions. Feature importance was assessed using a Random Forest algorithm, which provides a robust mechanism for evaluating the contribution of each feature to the model's predictive power. The feature importance scores were calculated using the Gini importance metric, which measures the total decrease in node impurity across all trees in the forest attributable to a given feature. The process is as follows:

1. Calculation of Feature Importance
   The Random Forest algorithm was applied to the training data, where it evaluated the contribution of each feature by calculating the Gini importance. This metric reflects the degree to which a feature reduces uncertainty (or impurity) when used to split the data at a node in the Decision Trees. The important scores were then extracted and sorted in descending order to identify the most influential features.

2. Visualisation of Feature Importance
   The feature importance scores were plotted using a bar plot, where the x-axis represents the features, and the y-axis shows their corresponding importance scores. The features were labelled, and the bars were annotated with the exact importance percentages to provide a clear and detailed visual representation.

3. Interpretation of Results
   The results indicated that "Source to transaction (sbytes)" were the most important for detecting and classifying malicious activity. These features significantly influenced the model's decision-making process, thereby enhancing its ability to differentiate between benign and malicious traffic. Understanding the importance of these features allowed us to streamline the model by focusing

Fig. 6 Confusion Matrix



on the most critical data points, reducing computational complexity while maintaining or even improving detection accuracy. This is particularly beneficial for IIoT environments, where computational resources are often limited, and real-time processing is crucial.

4. Impact on the Proposed Solution
   The identification of key features not only improved the efficiency and accuracy of the model but also informed the feature selection process during model development. By concentrating on the most impactful features, the model can operate more effectively in real-world IIoT environments, providing timely and accurate threat detection. Additionally, this insight allows for the potential reduction in feature dimensionality in future iterations of the model, which can further enhance processing speed without sacrificing performance.

Figure 8 shows the importance of each of the selected features for the detection and classification model.

Our analysis in Fig. 8 identifies two features as particularly relevant for the detection and classification model: sbytes and sttl.

1. sbytes (Source to Transaction Bytes)
   The sbytes feature represents the number of bytes transferred from the source to the destination during a transaction. This feature is crucial for detecting abnormal patterns in data transfer, which are often indicative of malicious activities. For example, an unusually high or low number of bytes in a transaction may signal an attempt to exfiltrate data or perform a Denial of Service attack. The model leverages this feature to distinguish between normal and suspicious data flows, making it a key factor in the classification process.

2. sttl (Source to Destination Time to Live Value)
   The sttl feature refers to the Time to Live (TTL) value for packets travelling from the source to the destination. TTL is a mechanism that limits the lifespan of a packet in a network by decrementing its value each time the packet passes through a router. The sttl feature can reveal important information about the network path and the behaviour of the source. Anomalies in TTL values, such as unusually low or high values, may indicate spoofing attacks, routing loops, or other network misconfigurations that could be exploited by attackers. The model uses sttl to detect these subtle but significant signs of malicious activity.

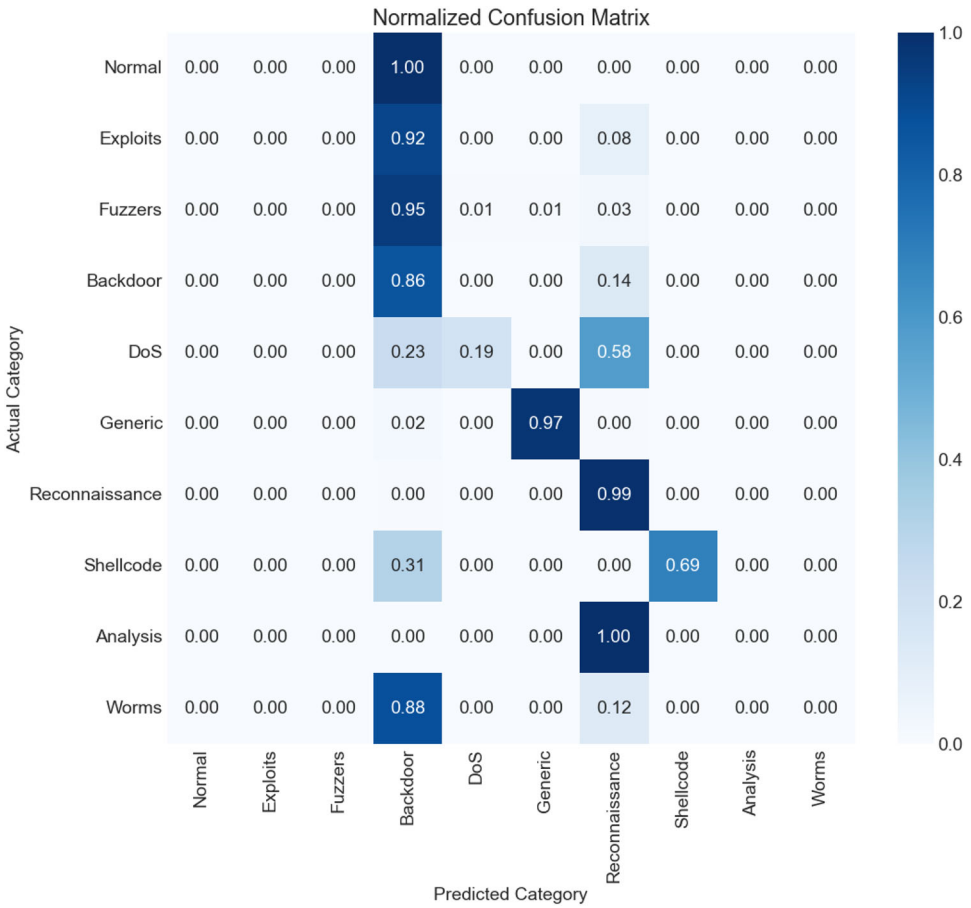


Fig. 7 Normalised Confusion Matrix

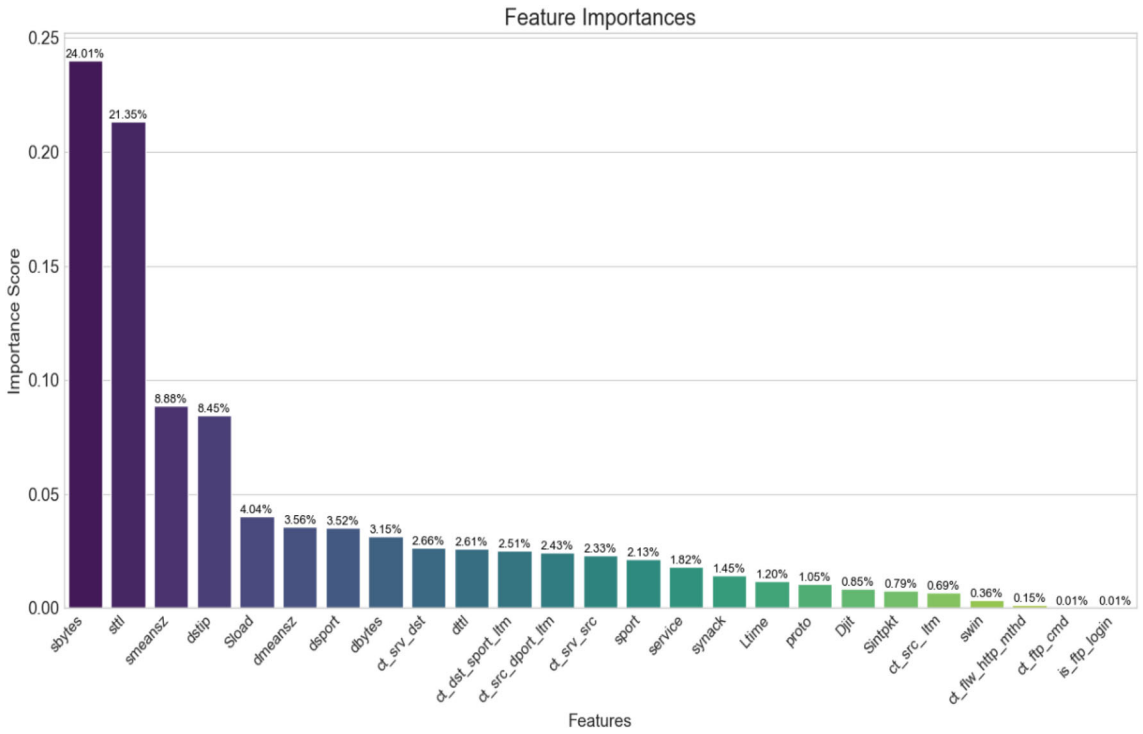


Fig. 8 Feature Importance

Both sbytes and sttl contribute significantly to the model's ability to detect and classify cyber-threats. The model's high reliance on these features underscores their effectiveness in capturing key aspects of network behaviour that differentiate between benign and malicious activities.

4.3 Novelty

The methodology presented embodies a novel integration of a Genetic Algorithm (GA) with a Deep Neural Network (DNN), a confluence that marks a significant stride in the field of cybersecurity, particularly in the context of IIoT environments. This innovative approach hinges on the synergy between the global search capability of the GA and the robust predictive power of the DNN. The GA's role in this methodology is pivotal, as it undertakes the task of feature selection, navigating through the high-dimensional feature space to identify the most significant features that enhance the model's accuracy. This selective process is intricately linked to the fitness evaluation of the GA, where each individual's fitness is determined by the accuracy of a DNN model trained on the selected features.

The DNN architecture, tailored for multi-class classification, is carefully designed with multiple layers employing ReLU activation functions, and a softmax output layer. The usage of categorical cross-entropy as the loss function optimises the model for multi-class scenarios, a common characteristic in cyber-attack datasets. The sophistication of this methodology is further elevated through the application of hyperparameter tuning, which fine-tunes the model to its optimal performance. This step is crucial in a field where precision and accuracy are paramount.

Overall, this methodology is not merely an incorporation of two distinct techniques but a harmonised system where each component complements the other, resulting in a model that is both precise in its predictions and efficient in handling the complex nature of cyber-threats in IIoT. The ingenuity of this approach lies in its capacity to leverage the strengths of both GA and DNN, thus paving the way for enhanced detection and classification outcomes in cybersecurity.

5 Work Scope and Limitations

While the presented study on integrating Genetic Algorithms (GA) and Deep Learning (DL) for cyber-attack detection in IIoT environments is robust, it is essential to acknowledge certain limitations. Firstly, the study's dependence on the dataset's quality and representativeness is crucial. The dataset's limitations in capturing the full spectrum of potential cyber-threats can impact the model's generalisability and effectiveness in real-world scenarios. Secondly, the computational intensity of GAs, combined with the inherent complexity of DL models, may pose challenges in terms of computational resources and time, particularly when scaling to larger and more complex IIoT networks.

Additionally, the evolving nature of cyber-threats means that the model may require continuous updates and retraining to maintain its effectiveness against new and emerging attack types. This retraining process can be resource-intensive and may not always be feasible in dynamic IIoT environments. Furthermore, the interpretability of DL models remains a challenge. While the integration with GA enhances feature selection, understanding the internal workings and decision processes of the DL model can be complex, which might pose challenges in terms of transparency and trustworthiness in security-critical IIoT applications.

Lastly, the study's scope is constrained by the current state of technology and understanding of IIoT cybersecurity threats. As the field rapidly evolves, future research may uncover new aspects that were not considered in this study. It's vital for future research to address these limitations, ensuring that the model remains effective and relevant in the ever-changing landscape of IIoT cybersecurity.

6 Conclusion and Future Work

This work presented an integrated approach that combines Genetic Algorithms (GA) with Deep Learning (DL) to enhance cyber-attack detection in Industrial Internet of Things (IIoT) environments. The proposed GA–DL model achieved a precision of 98%, accuracy of 96%, recall of 94%, and loss of 14%, while requiring less than half of the features from the UNSW-NB 15 dataset. This feature reduction not only streamlined the detection process but also halved the processing time, significantly improving system efficiency. Comparative analysis with existing literature highlights the superiority of the GA–DL approach, particularly in precision and feature reduction. The model's adaptability across diverse IIoT devices and cyber-attacks further underscores its potential to contribute to a more secure IIoT ecosystem.

Implications: The findings suggest that the GA–DL approach can significantly improve the efficiency and accuracy of cyber-attack detection in IIoT environments. By reducing feature dependency and processing time, the model can be more easily integrated into real-world IIoT systems, enhancing their resilience against cyber-threats. The model's adaptability also indicates its potential for broad application across various industrial settings.

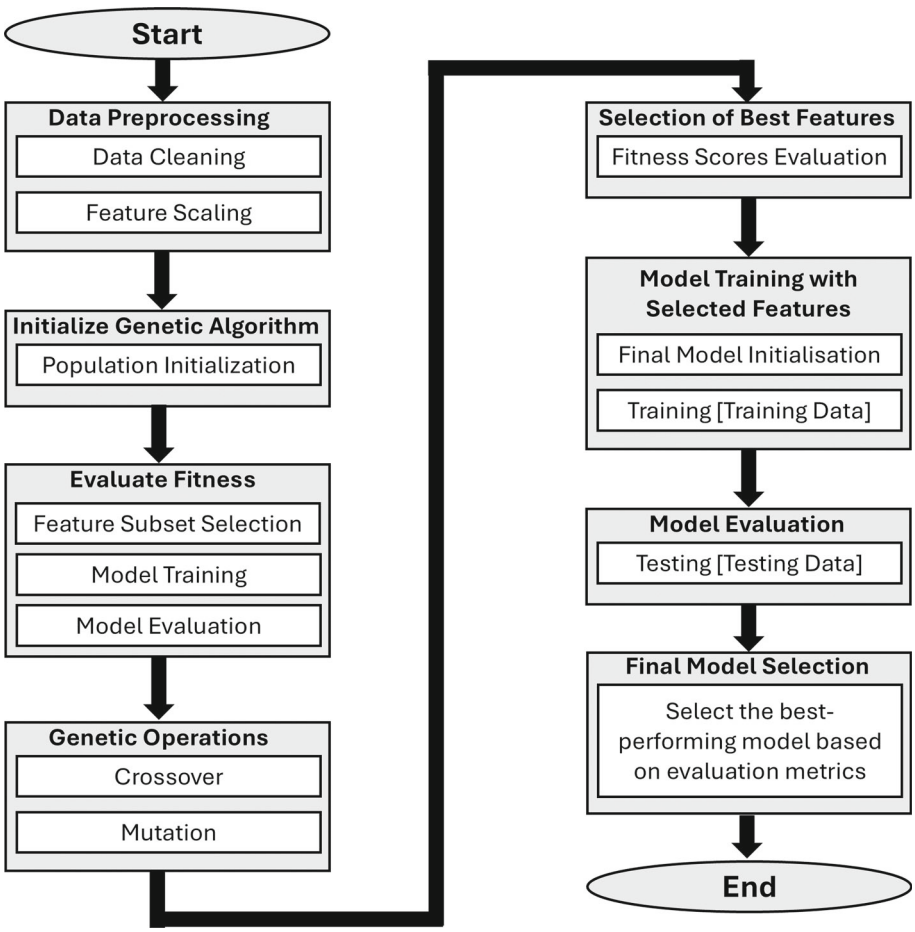
Limitations: Despite the promising results, the study has limitations that warrant further investigation. The model's performance was evaluated using a specific dataset, and while it showed adaptability, its scalability, and effectiveness across larger, more diverse datasets need validation.

Future Work: Future research will focus on testing the scalability of the model against larger datasets and integrating it with real-time IIoT systems to assess performance in dynamic environments. The incorporation of additional AI techniques, such as reinforcement learning and federated learning, will be explored to enhance predictive capabilities and facilitate distributed, privacy-preserving detection. Improving the model's interpretability and resilience against adversarial attacks will also be key areas of focus to ensure its practical applicability and reliability in industrial settings.

In conclusion, the integrated GA–DL approach represents a significant advancement in IIoT security. The results obtained pave the way for further research, with the goal of establishing a new benchmark for cyber-attack detection in Industry 4.0.

Appendix

The flowchart of the GA–DL (Genetic Algorithm–Deep Learning) method is shown below:



Authors Contributions Conceptualisation was performed by Nadia Alkhafaji (N.A.); methodology by N.A.; software by N.A.; validation by N.A.; formal analysis by N.A.; investigation by N.A.; resources by N.A., Ali Al-Sherbaz (A.A.), and Thiago Viana (T.V.); data curation by N.A.; writing—original draft preparation by N.A.; writing—review and editing by N.A, A.A., and T.V.; supervision by A.A. and T.V. All authors have read and agreed to the published version of the manuscript.

Funding This research received no external funding.

Data Availability The data used for this work are available at: https://research.unsw.edu.au/projects/unsw-nb15-dataset. (accessed on 10 October 2023).

Declarations

Conflict of interest The authors declare no conflict of interest.

Informed Consent Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Atzori, L.; Iera, A.; Morabito, G.: The Internet of Things: a survey. Comput. Networks **54**(15), 2787–2805 (2010). https://doi.org/10.1016/j.comnet.2010.05.010
2. Lee, I.; Lee, K.: The Internet of Things (IoT): applications, investments, and challenges for enterprises. Bus. Horiz. **58**(4), 431–440 (2015)
3. Hermann, M.; Pentek, T.; Otto, B.: Design principles for industrie 4.0 scenarios. In: 2016 49th Hawaii international conference on system sciences (HICSS), pp. 3928–3937. (2016)
4. Sayed, K.; Gabbar, H.A.: SCADA and smart energy grid control automation. In: H. A. B. T.-S. E. G. E. Gabbar, Ed. Academic Press, pp. 481–514. (2017). https://doi.org/10.1016/B978-0-12-805343-0.00018-8
5. Wolfert, S.; Ge, L.; Verdouw, C.; Bogaardt, M.-J.: Big data in smart farming: a review. Agric. Syst. **153**, 69–80 (2017)
6. Lasi, H.; Fettke, P.; Kemper, H.-G.; Feld, T.; Hoffmann, M.: Industry 4.0. Bus. Inf. Syst. Eng. **6**, 239–242 (2014)
7. CyberX Labs: 2019 Global ICS & IIoT Risk Report (2019)
8. Kaspersky Lab: Threat Landscape for Industrial Automation Systems in H1 2020," ICS Cert (2020)
9. Stouffer, K.; Falco, J.; Scarfone, K.: GUIDE to industrial control systems (ICS) security. In: The Stuxnet Computer Worm and Industrial Control System Security (2011)
10. Zetter, K.: Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid|WIRED. Wired (2016)
11. Liu, X.; Tang, J.: Mass classification in mammograms using selected geometry and texture features, and a new SVM-based feature selection method. IEEE Syst. J. (2014). https://doi.org/10.1109/JSYST.2013.2286539
12. Xue, Y.; Tang, Y.; Xu, X.; Liang, J.; Neri, F.: Multi-objective feature selection with missing data in classification. IEEE Trans. Emerg. Top. Comput. Intell. **6**(2), 355–364 (2022). https://doi.org/10.1109/TETCI.2021.3074147
13. Xu, H.; Fu, Y.; Fang, C.; Cao, Q.; Su, J.; Wei, S.: An improved binary whale optimization algorithm for feature selection of network intrusion detection. In: Proceedings of the 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems, IDAACS-SWS 2018, pp. 10–15 (2018). https://doi.org/10.1109/IDAACS-SWS.2018.8525539
14. Xue, B.; Zhang, M.; Browne, W.N.; Yao, X.: A survey on evolutionary computation approaches to feature selection. IEEE Trans. Evol. Comput. **20**(4), 606–626 (2016). https://doi.org/10.1109/TEVC.2015.2504420
15. Xue, Y.; Xue, B.; Zl, M.: Self-Adaptive particle swarm optimization for large-scale feature selection in classification. ACM Trans. Knowl. Discov. Data (2019). https://doi.org/10.1145/3340848
16. Tsang, C.H.: Network-based anomaly intrusion detection using ant colony clustering model and genetic-fuzzy rule mining approach (2006)
17. Stein, G.; Chen, B.; Wu, A.S.; Hua, K.A.: Decision tree classifier for network intrusion detection with GA-based feature selection. In: Proceedings of the 43rd Annual Southeast Regional Conference, Vol. 2, pp. 136–141 (2005)
18. Ahmad, I.; Abdullah, A.; Alghamdi, A.; Alnfajan, K.; Hussain, M.: Intrusion detection using feature subset selection based on MLP. Sci. Res. Essays **6**(34), 6804–6810 (2011). https://doi.org/10.5897/SRE11.142
19. Sivatha Sindhu, S.S.; Geetha, S.; Kannan, A.: Decision tree based light weight intrusion detection using a wrapper approach. Expert Syst. Appl. **39**(1), 129–141 (2012). https://doi.org/10.1016/j.eswa.2011.06.013
20. Kuang, F.; Xu, W.; Zhang, S.: A novel hybrid KPCA and SVM with GA model for intrusion detection. Appl. Soft Comput. J. (2014). https://doi.org/10.1016/j.asoc.2014.01.028
21. Aslahi-Shahri, B.M., et al.: A hybrid method consisting of GA and SVM for intrusion detection system. Neural Comput. Appl. (2016). https://doi.org/10.1007/s00521-015-1964-2
22. Das, A.K.; Das, S.; Ghosh, A.: Ensemble feature selection using bi-objective genetic algorithm. Knowl.-Based Syst. (2017). https://doi.org/10.1016/j.knosys.2017.02.013
23. Gharaee, H.; Hosseinvand, H.: A new feature selection IDS based on genetic algorithm and SVM (2017). https://doi.org/10.1109/ISTEL.2016.7881798
24. Yousefi-Azar, M.; Varadharajan, V.; Hamey, L.; Tupakula, U.: Autoencoder-based feature learning for cyber security applications. In: Proceedings of the International Joint Conference on Neural Networks (2017). https://doi.org/10.1109/IJCNN.2017.7966342.
25. Ibor, A.E.; Oladeji, F.A.; Okunoye, O.B.; Uwadia, C.O.: Novel adaptive cyberattack prediction model using an enhanced genetic algorithm and deep learning (AdacDeep). Inf. Secur. J. A Glob. Perspect. **31**(1), 105–124 (2022)
26. Srivastava, A.; Sharma, H.S.; Rawat, R.; Garg, N.: Detection of cyber attack in IoT based model using ANN model with genetic algorithm. In: 2024 IEEE International conference on computing, power and communication technologies (IC2PCT), vol. 5, pp. 1198–1201 (2024)
27. Sindhu, S.S.S.; Geetha, S.; Kannan, A.: Decision tree based light weight intrusion detection using a wrapper approach. Expert Syst. Appl. **39**(1), 129–141 (2012)
28. Kuang, F.; Xu, W.; Zhang, S.: A novel hybrid KPCA and SVM with GA model for intrusion detection. Appl. Soft Comput. J. **18**, 178–184 (2014). https://doi.org/10.1016/j.asoc.2014.01.028
29. Aslahi-Shahri, B.M., et al.: A hybrid method consisting of GA and SVM for intrusion detection system. Neural Comput. Appl. **27**(6), 1669–1676 (2016). https://doi.org/10.1007/s00521-015-1964-2
30. Das, A.K.; Das, S.; Ghosh, A.: Ensemble feature selection using bi-objective genetic algorithm. Knowl.-Based Syst. **123**, 116–127 (2017). https://doi.org/10.1016/j.knosys.2017.02.013
31. Yousefi-Azar, M.; Varadharajan, V.; Hamey, L.; Tupakula, U.: Autoencoder-based feature learning for cyber security applications. In: Proceedings of the International Joint Conference on Neural Networks, vol. 2017, pp. 3854–3861. (2017). https://doi.org/10.1109/IJCNN.2017.7966342.
32. Tahir, M.; Tubaishat, A.; Al-Obeidat, F.; Shah, B.; Halim, Z.; Waqas, M.: A novel binary chaotic genetic algorithm for feature selection and its utility in affective computing and healthcare. Neural Comput. Appl. (2022). https://doi.org/10.1007/s00521-020-05347-y
33. Halim, Z.; Ali, O.; Khan, M.G.: On the efficient representation of datasets as graphs to mine maximal frequent itemsets. IEEE Trans. Knowl. Data Eng. (2021). https://doi.org/10.1109/TKDE.2019.2945573
34. Viharos, Z.J.; Kis, K.B.; Fodor, Á.; Büki, M.I.: Adaptive, HHybrid feature selection (AHFS). Patt. Recognit. (2021). https://doi.org/10.1016/j.patcog.2021.107932
35. Nouri-Moghaddam, B.; Ghazanfari, M.; Fathian, M.: A novel multi-objective forest optimization algorithm for wrapper feature selection. Expert Syst. Appl. (2021). https://doi.org/10.1016/j.eswa.2021.114737
36. Uzma Al-Obeidat, F.; Tubaishat, A.; Shah, B.; Halim, Z.: Gene encoder: a feature selection technique through unsupervised deep learning-based clustering for large gene expression data. Neural Comput. Appl. **34**(11), 8309–8331 (2022). https://doi.org/10.1007/s00521-020-05101-4
37. Li, X.; Yi, P.; Wei, W.; Jiang, Y.; Tian, L.: LNNLS-KH: a feature selection method for network intrusion detection. Secur. Commun. Netw. (2021). https://doi.org/10.1155/2021/8830431

38. Dwivedi, S.; Vardhan, M.; Tripathi, S.: Building an efficient intrusion detection system using grasshopper optimization algorithm for anomaly detection. Cluster Comput. (2021). https://doi.org/10.1007/s10586-020-03229-5
39. Sumaiya Thaseen, I.; Saira Banu, J.; Lavanya, K.; Rukunuddin Ghalib, M.; Abhishek, K.: An integrated intrusion detection system using correlation-based attribute selection and artificial neural network. Trans. Emerg. Telecommun. Technol. (2021). https://doi.org/10.1002/ett.4014
40. Halim, Z.; Waqar, M.; Tahir, M.: A machine learning-based investigation utilizing the in-text features for the identification of dominant emotion in an email. Knowl.-Based Syst. (2020). https://doi.org/10.1016/j.knosys.2020.106443
41. Mahindru, A.; Sangal, A.L.: FSDroid: a feature selection technique to detect malware from android using machine learning techniques: FSDroid. Multimed. Tools Appl. (2021). https://doi.org/10.1007/s11042-020-10367-w
42. Lalos, A.S.; Kalogeras, A.P.; Koulamas, C.; Tselios, C.; Alexakos, C.; Serpanos, D.: Secure and safe IIoT systems via machine and deep learning approaches. In: Security and quality in cyber-physical systems engineering, pp. 443–470. (2019). https://doi.org/10.1007/978-3-030-25312-7_16.
43. Wang, X.X.X., et al.: A survey of machine and deep learning methods for Internet of Things (IoT) Security. IEEE Commun. Surv. Tutor. **22**(4), 1646–1685 (2020)
44. Lakshmanna, K.; Kavitha, R.; Geetha, B.T.; Nanda, A.K.; Radhakrishnan, A.; Kohar, R.: Deep learning-based privacy-preserving data transmission scheme for clustered IIoT environment. Comput. Intell. Neurosci. **2022**, 8927830 (2022)
45. Mudassir, M.; Unal, D.; Hammoudeh, M.; Azzedin, F.: Detection of botnet attacks against industrial IoT systems by multilayer deep learning approaches. Wirel. Commun. Mob. Comput. **2022**(1), 2845446 (2022)
46. Moustafa, N.; Slay, J.: "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6. (2015). https://doi.org/10.1109/MilCIS.2015.7348942