SSRG International Journal of Electrical and Electronics Engineering
ISSN: 2348-8379/ https://doi.org/10.14445/23488379/IJEEE-V12I7P123

Volume 12 Issue 7, 312-330, July 2025

Original Article

BSCSO-STNN: A Big Data-Driven IoT Intrusion Detection Model

S. Ravishankar[1], P. Kanmani[2]

[1]*Department of Computer Science, Sona College of Arts and Science, Salem, Tamil Nadu. India.*
[2]*Department of Computer Science, Thiruvalluvar Government Arts College, Rasipuram, Tamil Nadu. India.*

[1]*Corresponding Author : ravirohan83@gmail.com*

Received: 18 May 2025 Revised: 18 June 2025 Accepted: 18 July 2025 Published: 31 July 2025

Abstract *- The rapid expansion of the Internet of Things (IoT) and Big Data (BD) has led to security challenges. Securing IoT-BD against cyberattacks is necessary. An increasing number of applications are being implemented on BD platforms due to the rapid proliferation of data on the Internet. As the volume of data increases, the possibility of intrusions on the platform correspondingly increases. Conventional Intrusion Detection Systems (IDS) are ineffective for managing the extensive volume of historical data and unable to fulfil the security demands of BD platforms. This research aims to propose a novel intrusion detection model using Binary Sand Cat Swarm Optimization and Spatiotemporal Transformer Neural Network (BSCSO-STNN) model to address these issues. The CIC-IoT-23 and Bot-IoT datasets are collected and applied to train the model for evaluation. The developed BSCSO-STNN model is deployed in an Apache Spark (APS) framework. The datasets are initially preprocessed in this framework with data cleaning, oversampling, label encoding, and normalization. After preprocessing, the data is applied to the BSCSO for feature selection. Using the selected features, the STNN model performs binary and multiclass classification for both datasets. The BSCSO-STNN model attained 99.08% accuracy, 98.78% detection rate, 99.02% precision, and 98.94% F1-score using the CIC-IoT-23 dataset. The model attained 99.04% accuracy, 98.81% detection rate, 98.97% precision, and 98.95% F1-score for the BoT-IoT dataset in multiclass classification. The developed model outperformed all the current models in this research and demonstrated its accuracy in detecting intrusions.*

Keywords *- Intrusion detection, Big Data, IoT, Deep Learning, BSCSO, STNN, Apache spark.*

1. Introduction

The cybersecurity threat has significantly accelerated in the era of BD, surpassing the efficiency of current solutions, such as conventional IDS. The significant recent advancement of technology capabilities facilitates cyber threats aimed at individuals and organizations through diverse malicious behaviors. Governments and economic communities worldwide have implemented solutions to address emerging BD security challenges and to prepare for potential issues by improving data security governance [1]. As of February 2025, Statista reports that short-range IoT devices, including smart home assistants and wearable fitness trackers, constituted the most prevalent devices with internet connectivity globally, with 17.4 billion users. Mobile phones subsequently reached 8.65 billion connections, indicating their extensive adoption. Wide-area IoT devices, including linked automobiles and remote monitoring systems, comprised 4.93 billion connections.

In the current threat landscape, conventional security measures such as antivirus software, firewalls, and Virtual Private Networks (VPNs) are often inadequate. This necessitates the use of effective intrusion detection techniques. Consequently, the implementation of an IDS with conventional security measures is a necessary strategy. An IDS is automated software employed to analyze network data for the detection and/or prevention of harmful attacks. It is categorized into two major methods: Misuse-Based IDS (MIDS) and Anomaly-Based IDS (AIDS). The MIDS method classifies the attack based on recognized patterns, referred to as signatures. AIDS, conversely, aims to identify anomalous patterns or behaviors, facing the primary issue of accurately differentiating between abnormal and normal patterns with minimal error. Various studies examine the issues affecting IDS, particularly in the context of the emerging BD era, which introduces diverse forms of harmful attacks. Consequently, specific tools are necessary to manage the scalability of large data to detect cyber risks [2].

Conventional data processing techniques are insufficient for handling large data volumes. Consequently, substantial volumes of datasets cannot be effectively processed using conventional data processing techniques. Standard database systems are inadequate for managing the huge volume and

rapid velocity of big data; the development of new or enhanced data processing methods is required [3]. Figure 1 illustrates the steps of BD processing. BD exists in various formats and sizes and is typically not immediately suitable for analytics. Consequently, following data collection, BD undergo staging or preprocessing. Data preprocessing involves the creation of a functional dataset prepared for analysis. After preprocessing, data analysis is conducted on the resultant datasets. Data analysis methodologies encompass statistical approaches, soft computing, Deep Learning (DL), Machine Learning (ML), and data mining. The application analytics encompasses a query mechanism designed to meet the particular necessities of the application. The query interface functions as a control pane that determines the outcome that the user intends to generate from data. Data visualization is the last phase of processing. Visualization entails the creation of visual representations of data, processing outcomes, and data.

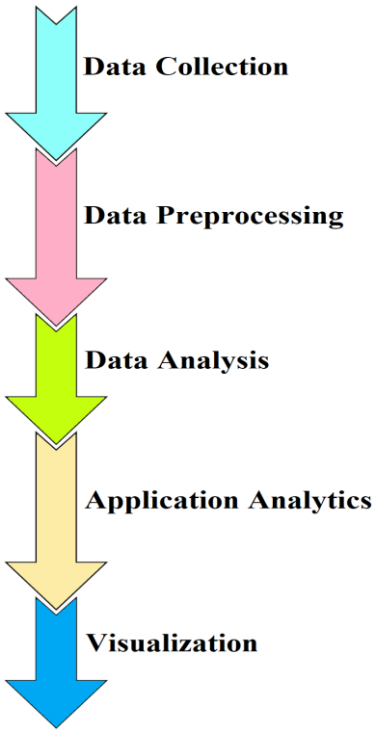

Fig. 1 BD processing stages

The IoT and big data share two primary relationships. IoT serves as a principal BD source, making it essential for BD analytics to enhance the quality of BD services. IoT data varies from traditional BD due to distinct characteristics such as large-scale operational data, heterogeneity, spatial and temporal correlations, and increased noise levels [4]. A primary concern in BD is data security, which has increased due to the growth in data volume following the application of encryption techniques. Consequently, many researchers implement different encryption algorithms in an effort to minimize data size. IDSs are strategically placed within a network to identify threats and monitor traffic. The IDS achieves this by aggregating data from various systems and network infrastructures and analyzing the data for potential threats [5]. Rapid improvements in IoT technology have created a comprehensive smart computing platform by merging smart things with sensing, communication, and processing functionalities [6]. The fundamental aspect of IoT is the intricate BD produced by several networked sources in real-time, which poses distinct processing and analytical issues. In software engineering, best practices have been consistently applied in IoT technologies to manage large data sets effectively across several domains [7].

1.1. Problem Statement

A significant security challenge for IDS is addressing the various types of malicious software that result in network security vulnerabilities and critical failures [8]. Cyber-attacks have become increasingly complex, complicating the detection of unidentified malware due to the advancement of sophisticated evasion techniques designed to exfiltrate sensitive information while bypassing intrusion detection systems. Moreover, cybersecurity dangers exist during inter-network connections. Consequently, novel methods and solutions are necessary for preventing attacks and facilitating the prompt detection of intrusions. Recent advancements in ML and DL methodologies have been implemented for IDS, the discovery of anomalous activities in networks, and their mitigation. DL methodologies outperform ML approaches when applied to big datasets [9]. Thus, this research proposes a novel hybrid DL model for intrusion detection.

1.2. Research Objectives

The use of Artificial Intelligence (AI) in IDS signifies a substantial progression in cybersecurity. This combination will handle the escalating complexity and prevalence of cyber threats. AI-driven IDSs employ ML and DL algorithms to analyse extensive network traffic, detect anomalies, and detect suspected intrusions in a BD environment [10]. The key contributions of this proposed work are defined in the following:

- This research develops a BD-based intrusion detection model using a hybrid DL model for detecting and classifying attacks.
- BD datasets like CIC-IoT-23 and BoT-IoT are applied to train and evaluate the developed model.
- A series of data preprocessing processes is performed to enhance the data's suitability for model training and analysis.
- The BSCSO technique is applied for selecting the optimal features, and the STNN algorithm is utilized to detect and classify the attacks.
- The developed BSCSO-STNN model is processed using Apache Spark on Google Colab.
- The research model is evaluated using performance

indicators like accuracy, detection rate, precision, and F1-score.
- The results of the developed BSCSO-STNN model are compared and validated with the current models discussed in this research for proper validation.

The paper is organized into the subsequent sections. Section II succinctly examines the current models relevant to the research study. Section III encompasses the implementation of the developed research methodology. Section IV highlights the experimentation findings of the research methodology and a comparison with current models. The final section concludes the research with an overview of the findings and recommendations for subsequent research initiatives.

2. Related Works

This section presents a review of current works applied to improving big data-based intrusion detection has been analyzed. All the reviewed current models are critically analyzed and presented in Table 1. The critical analysis includes their advantages and limitations. The review of the related works is as follows: By integrating temporal and spatial data in IDS models, the research in [11] proposed a fusion model based on CNN and C-LSTM. Fusion incorporates improved parallelism in the training process, yielding superior outcomes without necessitating an excessively deep network. The fusion resulted in reduced training duration, rapid convergence, and computational efficiency for limited resource-constrained network components. The model was better suited for anomaly detection in the big data context of the IoT. The model achieved superior recall, precision, and accuracy.

A decentralized methodology employing a fog computing layer integrated with a reptile group intelligence technique was proposed in [12]. The model minimized network traffic volume and analyzed in the cloud layer utilizing APS architecture. Essential network traffic characteristics were identified by a chameleon optimizer technique and a principal component reduction technique. Multi-layered artificial neural networks were utilized for traffic analysis within the fog layer. Experiments utilizing the NSL-KDD dataset demonstrated that the model attained superior accuracy in detecting intrusions.

An IDS model that employed big data analytics and a modified LSTM algorithm was proposed in [13]. This model was developed to overcome the limits of misuse and anomaly detection. Depending on distributed and parallel techniques, the model employed a BD analytics platform. The model detected anomalous behaviors within a network to identify harmful or unauthorized actions and facilitated a response during a breach of confidentiality. The distributed and parallel platforms enhanced both accuracy and training duration. The model exhibited a 96.11% accuracy. The study in [14] addressed the issue of decision-making based on multicriteria in the identification of anomalies in IoT. The study concentrated on the feature of BD analytics pertaining to data stream processing. To achieve this, the Event Strength Function (ESF) was implemented. An approach for multicriteria analysis was introduced, considering the dynamic characteristics of the Pareto set using data stream processing. Based on temporal graphs and multicriteria anomaly detection, an approach was proposed for detecting network anomalies. The outcomes have been demonstrated to be effective for IoT anomaly detection.

The research in [15] examined the newly developed UWF-ZeekData22 dataset, which analyzed data from Zeek's Connection Logs. The data were gathered via the network security monitor called Security Onion 2 and annotated utilizing the MITRE ATT&CK system. Spark was employed within the big data framework to execute classifiers like Naïve Bayes (NB), Random Forest (RF), Decision Tree (DT), support vector classifier, Gradient Boosted Trees (GBT), and Logistic Regression (LR) for the classification of reconnaissance and discovery tactics from this dataset. The findings demonstrated that DT, GBT, and RF have superior efficacy in intrusion detection.

A distributed combined DL-IDS model for the Internet of Vehicles utilizing the APS framework was proposed in [16]. The model integrated a CNN with an LSTM network to extract features and data for the detection of vehicle network intrusions from extensive vehicular network traffic and the identification of anomalous behavior. Experimental validation on the NSL-KDD and UNSW-NB15 datasets demonstrated that the CNN-LSTM model utilizing the Spark framework was effective.

An enhanced IDS model for IoT security was proposed in [17] through the integration of multimodal BD representation and transfer learning. PCAP files were analyzed to identify pertinent attack and data bytes. Spark-based methodologies for optimizing BD were employed to handle extensive volumes of data. The transfer learning was employed to provide semantically enhanced trained features. The classification of diverse cyberattacks was conducted by integrating multimodal features utilizing CNN-Gated Recurrent Unit (GRU), CNN-Recurrent Neural Network (RNN), and CNN-LSTM architectures. The findings indicated that the CNN-LSTM model had superior performance. The research in [18] improved the accuracy of network IDS models by removing duplicate and nonessential features. To address accuracy concerns arising from extraneous features, a model called Las Vegas Wrapper enhanced with Multiple Evaluation Criteria (LVW-MECO) has been developed. The LVW algorithm employed various evaluation criteria to detect relevant features from IoT network data and enhanced the accuracy of intrusion detection. The model improved intrusion

detection efficacy and secured IoT data integrity for a more secure IoT ecosystem. A Privacy-Preserve Statistically Learning method with an Optimizer Approach for Higher-Dimensional BD Environments (PPSLOA-HDBDE) was proposed in [19]. A Linear Scaling Normalization (LSN) was employed to normalize the given data. The SCSO was utilized for selecting features. For intrusion classification, an ensembled model of a Temporal Convolution Network (TCN), Multi-Layered Autoencoders (MAE), and Extreme Gradients Boosting (XGB) approaches were employed. The hyperparameter optimization of these approaches was achieved through the application of an Improved Marine Predators Approach (IMPA). The model demonstrated enhanced accuracy in identifying intrusions. Two ML methodologies, Random Forest (RF) and Multi-Layer Perceptron (MLP), were implemented in [20] by employing the Scikit and the Spark ML library for the identification of Denial of Service (DoS) attacks. Following the identification of DoS assaults, the models' performance was enhanced by reducing the prediction time. The study attained a comparable mean accuracy across the employed models. In terms of testing and training duration, the BD method surpassed the non-big data method since Spark executes computation processes in a distributed memory.

A secure anomaly detection methodology for BD platforms with quantum optimization clustering was proposed in [21]. A BD platform anomaly detection framework was constructed using a distributed software based on Spark and Hadoop. The model effectively detected anomalies in networks by acquiring and analyzing server records from BD platforms. A quantum ant colony optimizer clustering technique for offline anomaly identification was developed to identify diverse anomalies.

To enhance the precision of the optimal paths search within the ant colonies, quantum bits encoding was utilized for the positioning of the ants. The findings demonstrated that the model efficiently accomplished anomaly clustering identification in large datasets. A classification-based network attack detection model utilizing BD network traffic data was developed in [22]. The model employed a hybrid DL network, integrating CNN and LSTM for enhanced intrusion detection. Furthermore, data imbalance mitigation employing the Synthetic Minority Oversampling Techniques (SMOTE) and Tomek's s-Link Sampling Techniques (STL) was utilized to alleviate the impact of data inconsistency on system efficiency. The study utilized PySpark within the Google Colab environment, where the CNN-LSTM yielded superior accuracy compared to the LSTM-CNN approach. An IDS methodology utilizing BD analytics tools to integrate diverse data sources and identify intrusions was developed in [23]. The study employed the decision tree method, Semantically Similar Data Miner (SSDM) and the Bayesian network algorithm K2 to assess the integrated data. The study demonstrated the efficacy of data fusion utilizing the Hadoop ecosystem (MapReduce) and the Neo4j database for managing and processing large datasets. Experimental results demonstrated that, in both instances, data fusion significantly enhanced outcomes.

A BD technique framework was developed in [24] utilizing Hadoop–Spark to train and evaluate multi-class and binary classification. The framework was performed through a one-vs-rest method for intrusion detection using the BoT-IoT dataset. All the algorithms present in Hadoop Spark were tested regarding accuracy and processing duration. Due to the significant imbalance in the BoT-IoT dataset, the accuracy of minority class detection was enhanced by generating additional data samples with a Conditional Tabular Generative Adversarial Network (CTGAN). The model employed the RF method from Spark's multi-class algorithms for training. With its high accuracy, an analysis of each class revealed that minority groups were adversely impacted. A real-time IDS model designed to identify IoT threats via multiclass classification models using the PySpark framework was proposed in [25]. Diverse ML methods like DT, RF, LR, and Extreme Gradient Boosting (XGB) were utilized employing the One Vs Rest (OVR) methodology. The model used the IoT-23 dataset, and SMOTE was utilized for dataset preparation. Furthermore, SelectKBest features selection was utilized to select the best significant feature for classification. The findings demonstrated that XGB attained superior accuracy among the assessed algorithms.

Table 1. Critical analysis of analyzed related works

Ref	Models	Datasets	Application	Advantages	Disadvantages
[11]	CNN + C-LSTM Fusion	KDDCup99	Anomaly detection in IoT using spatial-temporal data	Improved parallelism, faster convergence, efficient training on limited resources	May underperform on deeply hierarchical attack patterns
[12]	Fog computing + Chameleon Optimizer + ANN	NSL-KDD	Decentralized intrusion detection with Spark	Reduces network traffic, enhances scalability, and provides real-time analysis	Complexity in deployment and integration
[13]	Modified LSTM with BD analytics	NSL-KDD	Distributed anomaly detection	High accuracy (96.11%), reduced	Limited to temporal anomaly patterns

				training time via parallelism	
[14]	Event Strength Function (ESF) + Multicriteria Analysis	Real-time Data	Stream processing-based anomaly detection in IoT	Dynamic anomaly detection with temporal graphs	High implementation complexity
[15]	DT, RF, GBT classifiers with Spark on UWF-ZeekData22	UWF-ZeekData22	Reconnaissance and discovery tactic detection	High detection accuracy with DT, RF, GBT	Limited generalization due to dataset specificity
[16]	CNN + LSTM with APS	NSL-KDD and UNSW-NB15	Internet of Vehicles (IoV) intrusion detection	Effective feature extraction, scalable processing	Needs high computational resources
[17]	CNN-GRU, CNN-RNN, CNN-LSTM + Transfer Learning	CIC-IoT 2022, Edge-IIoT, and CIC-IoT 2023	IoT attack classification using multimodal data	Semantic enrichment, efficient BD handling	Complex multimodal integration and tuning
[18]	LVW-MECO feature selection	IoT-23	IoT anomaly detection with essential feature refinement	Improved accuracy and data integrity	Not suitable for real-time detection
[19]	PPSLOA-HDBDE (TCN + MAE + XGBoost + IMPA)	BoT-IoT	High-dimensional intrusion detection	Strong accuracy, efficient feature selection via SCSO	High computational cost due to the ensemble
[20]	RF, MLP using Spark ML	DDoS dataset	DoS attack detection	Reduced prediction time, scalable with Spark	Focused only on DoS; lacks broader attack coverage
[21]	Quantum Ant Colony Clustering	KDDCup99	BD anomaly detection in networks	Enhanced clustering precision via quantum encoding	The offline method is unsuitable for real-time detection
[22]	CNN-LSTM with STL & SMOTE	CIDDS-001	Multiclass BD-based intrusion detection	Improved accuracy with imbalance handling	Requires tuning for balance optimization
[23]	SSDM + K2 with Hadoop & Neo4j	KDD99 and DARPA99	Integrated intrusion detection via data fusion	Improved insights from multi-source fusion	Complex deployment of Neo4j-Hadoop-Spark
[24]	CTGAN + RF on Spark	BoT-IoT	Intrusion detection using BoT-IoT dataset	Enhanced minority class detection, high accuracy	CTGAN adds processing overhead
[25]	PySpark + OVR with XGB, RF, DT, LR	IoT-23	Real-time IoT multiclass intrusion detection	High accuracy with XGB, feature selection via SelectKBest	The OVR approach may struggle with overlapping classes

2.1. Research Gap Analysis

It is clear that all these analyzed models utilized the BD framework to detect intrusions. The research gaps are identified by analyzing the reviewed current models in this research. Most of the models utilized the BD frameworks with the DL and ML algorithms. Many models have integrated optimization techniques for optimizing classification and feature selection. However, the limitations arise in the aspects of computational complexity, lack of generalization, and utilization of imbalanced datasets. Few studies used the feature selection process combined with the DL models, which are optimized for BD-based intrusion detection. These research gaps are addressed by developing a novel BSCSO-STNN model within an APS framework. This proposed IDS model will be scalable, effective, and accurate in detecting intrusions on both binary and multiclass classifications.

3. Materials and Methods

This research proposed an intrusion detection and classification model based on the BD framework using a metaheuristic and DL algorithm model. The model is developed and deployed on the APS-BD platform for processing and classification. This APS engine will be helpful in handling BD datasets like CIC-IoT-23 and BoT-IoT and processing them. Figure 2 depicts the workflow of the developed intrusion detection model. Both the CIC-IoT-23

and BoT-IoT datasets are applied for this research to train and evaluate the developed IDS model. The developed intrusion detection model has three primary stages of processing: data preprocessing, feature selection, and classification. Initially, the BoT-IoT dataset is collected and applied to the research. The collected datasets are processed in the preprocessing stage with various preprocessing methods like data cleaning, oversampling, label encoding, and normalization. After preprocessing, the datasets are split into an 80:20 ratio for training and evaluating the model. The preprocessed training set of the data is then processed to select the optimal features from the data sets individually using the BSCSO technique.

The selected optimal features are helpful in improving the classification performance. The STNN model performs attack detection and classification based on the selected optimal features. The STNN model effectively captures both network features based on spatial dependencies and temporal patterns. This capability is important to accurately detect attacks in a BD framework. The STNN's attention mechanism helps the model to focus on data segments, which results in better classification. Finally, the performance of the BSCSO-STNN model is evaluated based on performance indicators like accuracy, detection rate, precision, and F1-score. The results will be evaluated and compared with the current models for validation. The results highlight that this developed BSCSO-STNN model is highly accurate and effective in detecting intrusions on both binary and multiclass classifications.

3.1. Dataset Details

The CIC-IoT 2023 data set comprises a range of recent IoT attacks. This collection contains communications from 105 authentic IoT devices and includes 33 distinct attack methods. To enhance classification performance, these attacks were categorized into seven distinct categories: DoS, DDoS, Spoofing, Brute Force, Recon, Mirai, and Web-based. This CIC-IoT-2023 dataset comprises 46 features.

Table 2. Data distribution of CIC-IoT-23 dataset

Attack Classes	No. of Records	Distribution %
Benign/Normal	1098195	2.35
Spoofing	486504	1.04
Reconnaissance	354565	0.76
Bruteforce	13064	0.03
WebBased	24829	0.05
DoS	8090738	17.33
Mirai	2634124	5.64
DDoS	33984560	72.79

As shown in Table 2, the attack category with the largest number of records is referred to as DDoS. The DDoS causes overflows in networks or devices with an overwhelming volume of traffic. This results in disruptions and renders services inaccessible. The next dominant class is DoS, which resembles DDoS but generally originates from a single source and similarly seeks to impair service availability. Brute Force seeks to obtain unauthorized access by systematically testing many password combinations.

Spoofing leads devices by disguising themselves as authentic entities, resulting in data exfiltration or virus distribution. Reconnaissance attacks collect network data to find vulnerabilities. Web-based attacks leverage vulnerabilities in web applications to gain unauthorised access unauthorized. The Mirai attack exploits IoT and converts devices into bots for extensive attacks like DDoS. This dataset offers a thorough analysis of the prevailing IoT attacks. The DDoS category contains a large number of instances, followed by DoS and Mirai. Remaining classes, like Recon, Spoofing, Brute Force, and Web-based, exhibit significantly limited samples [26].

Table 3. Data distribution of BoT-IoT dataset

Attack Classes	No. of Records	Category
Benign	9543	Normal
Data Theft	118	Information Theft
Keylogging	1469	
DoS-HTTP	29706	DoS
DoS-TCP	12315997	
DoS-UDP	20659491	
DDoS-HTTP	19771	DDoS
DDoS-UDP	18965106	
DDoS-TCP	19547603	
OS Fingerprinting	358275	Information Gathering
Service Scanning	1463364	

The Bot-IoT data set was developed by the Cyber Range Laboratory at UNSW Canberra within an actual network setting. This dataset comprised data collected from diverse smart home equipment, including refrigerators, lighting systems, thermostats, garage doors, and weather monitoring devices. This network architecture comprises a combination of botnet and benign traffic covering 73 million records. The Bot-IoT data set comprises 43 network traffic attributes and

three classifications of labels. Nevertheless, only 37 of the 43 features were considered significant for detecting botnet attacks in IoT networks. Table 3 presents the BoT-IoT data set details.

The two categories of attacks, specifically keylogging and data theft, have insufficient data samples. It can adversely affect the classification outcomes. Consequently, these two attack classes are excluded from the experiment [27].

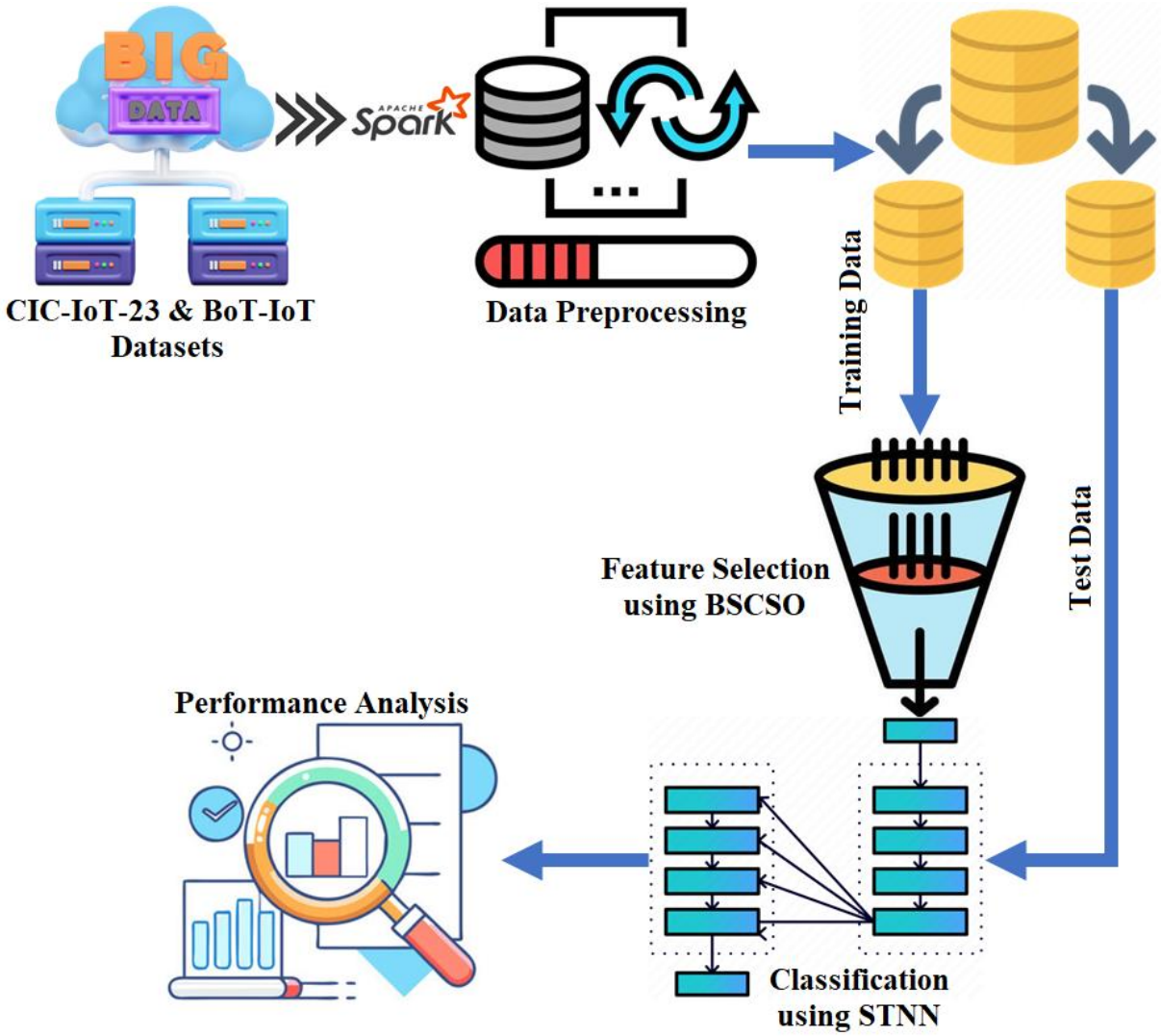

Fig. 2 Proposed BSCSO-STNN intrusion detection model

3.2. Apache Spark

In a big data environment, processing and storage of information are essential, where false alerts from benign attacks can impede operations. To mitigate these risks, it is imperative to use robust encryption mechanisms to secure critical data, deploy IDS to monitor and identify suspicious activity, and utilize DL to identify abnormalities and potential security concerns. The proposed research aims to develop an IDS model utilizing APS's big-data processing capabilities. The master node, equipped with software for drivers that invokes an application's primary program, constitutes the essential component of APS. The driver application, whether custom code or the operating system itself, is responsible for initializing the Spark environment in a dynamic shell environment. The Spark context serves as the gateway to all APS functionalities. It works with the cluster manager, which supervises the administration of several tasks. In a cluster, the application of the driver and Spark environment functions in combination to execute the operation. Resource allocation is initially managed by the cluster management. Then, the tasks are divided and allocated to either the slave or worker nodes. Subordinate nodes are responsible for fulfilling the tasks specified by the cluster manager. The Spark environment is then reinstated for these tasks. The executor is responsible for performing the tasks. The executors and Spark have a similar lifespan. It is vital to augment the total count of worker nodes to enhance the logical segmentation of tasks. Figure 3 illustrates the APS working architecture [28].

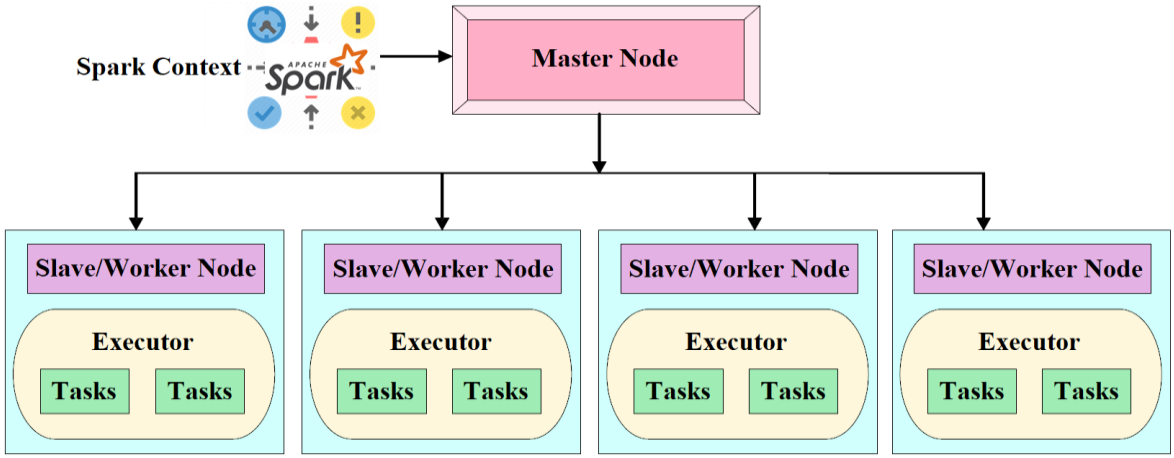

Fig. 3 Architecture of APS module

This research employs the CIC-IoT-23 and BoT-IoT datasets. The first processing of the big dataset 'InBD' is conducted utilizing the Spark framework, employing slave and master nodes for data processing. The Spark architecture typically yields exceptional processing capabilities due to the ability of worker node servers to operate in parallel. The partitioning of transformed data is performed, and the segmented information is distributed to many worker nodes. Additionally, worker nodes are utilized to execute processing and select features. The transformed data is divided into various subsets, as seen in Equation (1).

$$In^{BD} = U^{BD};\ (1 \le BD \le bd) \tag{1}$$

Here, U^{BD} Denotes segmented information, while bd represents the entirety of segmented information derived from In^{BD}. The complete segmented data corresponds to the total number of slave nodes. Every data segment is allocated to the slave node designated for preprocessing and selection of features. Consequently, the input of the s^{th} The slave node is represented as indicated in Equation (2).

$$U^{BD} = \{M_{s,q}\};\ (1 \le s \le S);\ (1 \le r \le R) \tag{2}$$

Here, $M_{s,q}$ denotes the BD^{th} segmented data within the s^{th} element of the r^{th} dataset. Each partitioned dataset is allocated to slave nodes for preprocessing and selection of features. Consequently, slave nodes execute preprocessing and feature selection tasks. The classification model gains access to the chosen features to identify intrusions within the system. Subsequently, the average of the outcomes from the STNN pertaining to the identification of intrusions will be computed. The proposed approach involves creating an architecture for an IDS designed for big data to identify intrusions within large datasets. The architecture of APS comprises a master node that is subdivided into multiple slave nodes. The data provided involves preprocessing through a min-max normalization technique. After the preprocessing phase, the most optimal features are identified. This methodology is employed for preprocessing and selecting features within each subordinate node. The features are extracted and input into the STNN model. A novel IDS model is developed by integrating the BSCSA-STNN. Finally, the intrusion detections are identified from the outputs.

3.3. Preprocessing and Normalization

Data preparation serves as a phase that prepares unprocessed data for subsequent processing. The data can often be incomplete, inconsistent, and filled with errors. Data preparation is a technique employed to address such issues. Data cleaning involves finding and rectifying inaccuracies, inconsistencies, and errors within a dataset to maintain its reliability and integrity for analysis. Multiple strategies were employed to identify and rectify diverse errors, including missing data, duplicate entries, outliers, and anomalies in labeling or formatting.

SMOTE, as indicated by its terminology, is an over-sampling approach. This approach generates synthetic data by oversampling the minority class. SMOTE mitigates the overfitting issue associated with random oversampling techniques by producing synthetic data. SMOTE operates by choosing occurrences adjacent to the space of the feature. A sample at random is chosen from the minority class, and this occurrence has two nearest neighbors. Upon randomly selecting one of the nearest neighbors, the variation among the two sample features is multiplied by a factor ranging from 0 to 1 and then added to the chosen sample value. Based on the extent of over-sampling needed, neighbours from the k-NN are selected at random. SMOTE samples are combinations of two linear comparable patterns (s, s^R) from the minimal class and are delineated as in the following Equation (3).

$$n = s + d \cdot (s^R - s),\ 0 \le d \le 1 \tag{3}$$

Here, s^R Represents the randomly chosen sample of s based on the nearest neighborhood number, while d signifies the variation between the two samples [29].

In this work, prior to the implementation of the dataset to the classification algorithms, categorical variables were converted to numerical values using the Label Encoding procedure, followed by the normalization process as indicated in Equation (4). Therefore, the normalizing process transformed every numerical value in the dataset to a range of zero to one.

$$x' = \frac{x-\mu}{\sigma} \tag{4}$$

Here, x represents the actual value, x' Denotes the normalized value, and σ and μ signify the standard deviation and mean values, respectively [30]. The datasets are divided into training and test sets, with 80% assigned for training and 20% assigned to testing.

3.4. Binary SCSO-based Feature Selection

A metaheuristic technique known as SCSO is inspired by the natural behavior of sand cats (SCs). The SC possesses a unique hunting and foraging behavior. The capability of these creatures to detect prey underground or on the surface supports their extraordinary ability to locate prey. Consequently, they could quickly locate their prey. The SCSO mimicked this characteristic to identify the most efficient solution. The SCSO algorithm operates based on a unique basis. Following initialization, a search for prey is conducted to identify the optimal solution. The SC's capacity for lower-frequency noise generation was utilized. Every search agent (SA) possesses a predetermined range of sensitivity that begins at 2kHz [31]. In the SCSO technique, the $\vec{R}_G$ The variable minimizes linearly from two to zero using Equation (5).

$$\vec{r}_G = S_M - \left(\frac{S_M \times iter_c}{iter_{Max}}\right) \tag{5}$$

In this context, S_M is presumed to be 2, $iter_c$ denotes the present iteration count, and $iter_{Max}$ Represents the maximal count of iterations. Initially, the SC moves rapidly, but after 50% of the iterations, its movements get more deliberate. Similar to various metaheuristics techniques, the balance among exploitation and exploration stages is significant; thus, the SCSO employs a $\vec{R}$ Parameter.

$$\vec{R} = 2 \times \vec{r}_G \times rand(0,1) - \vec{r}_G \tag{6}$$

According to Equation (6), the transition between the two stages is equilibrated. Moreover, Equation (7) is formulated to prevent entrapment in local optima. The $\vec{r}$ The option defines the level of sensitivity range for every SA.

$$\vec{r} = \vec{r}_G \times rand(0,1) \tag{7}$$

The primary stage of the SCSO involves updating the location for every SA. According to Equation (8), the location update for every SA in every iteration relies on the optimal candidate positioning and its present position in conjunction with the level of sensitivity range.

$$\overrightarrow{Pos}(t+1) = \vec{r} \cdot \left(\overrightarrow{Pos}_{bc}(t) - rand(0,1) \cdot \overrightarrow{Pos}_c(t)\right) \tag{8}$$

In Equation (8), $\overrightarrow{Pos}_{bc}$, $\overrightarrow{Pos}_c$, and $\vec{r}$ Denote the optimal candidate position, present position, and range of sensitivity, respectively. Following the exploration stage of the prey search, the next stage in the SCSO is the exploitation stage, which involves attacking the prey.

$$\begin{aligned} \overrightarrow{Pos}_{rnd} &= \left|rand(0,1) \cdot \overrightarrow{Pos}_b(t) - \overrightarrow{Pos}_c(t)\right| \\ \overrightarrow{Pos}(t+1) &= \overrightarrow{Pos}_b(t) - \vec{r} \cdot \overrightarrow{Pos}_{rnd} \cdot \cos(\theta) \end{aligned} \tag{9}$$

The distance across the optimal and present locations of all the SAs in the respective iterations is computed utilizing Equation (9). The acute sensitivity of SC is employed to capture their prey. The level of sensitivity was presumed to be in a circular shape; hence, in all the movements, the directions will be based on an θ angle randomly derived from the roulette wheel selection in the SCSO. The random θ angle within 0 and 360 yields a cosine value ranging from -1 to 1.

$$\vec{X}(t+1) = \begin{cases} \overrightarrow{Pos}_b(t) - \overrightarrow{Pos}_{rnd} \cdot \cos(\theta) \cdot \vec{r} \\ \qquad |R| \le 1; exploitation \\ \vec{r} \cdot \left(\overrightarrow{Pos}_{bc}(t) - rand(0,1) \cdot \overrightarrow{Pos}_c(t)\right) \\ \qquad |R| > 1; exploration \end{cases} \tag{10}$$

The updation of SA's position is performed using Equation (10). A circular motion is thus accomplished. Here, $\overrightarrow{Pos}_b$ denotes the optimal position (optimal solution), while $\overrightarrow{Pos}_{rnd}$ Signifies a random position.

Optimization issues in a binary space are critical. Consequently, it is essential to use binary variants of metaheuristic techniques. In this method, the search space often encompasses 1 or 0 along with the movement of SAs within the binary region. The search space was organized into rows that define the solutions, which consist of an assortment of binary value for all the rows. The primary difference between the binary (discrete) and continuous versions of every metaheuristic technique is in the movements of the particle, wherein 0 transforms into 1 and vice versa. The SCSO technique utilizes a search space comprised of both real and continuous numbers, making it inapplicable for binary optimization problems. Hence, this research utilized the Binary SCSO (BSCSO) approach to address this issue. On updating the position of every SA, a V-shaped Transfer Function (TF) was employed to convert the acquired values into a range from zero to one. The solution to the issue is positioning every SA within a binary vector of 0s and 1s.

Every SC in the BSCSO approach identified sounds below 2 kHz, comparable to the SCSO algorithm. This approach adhered to the SCSO algorithm, although the SA performed within the interval [0, 1]. Employing Equation (11), every SA can modify its position. The V-shaped TF finally converted the output to either one or zero. The BSCSO approach employed the V-shaped TF as its primary rule. The search was carried out in a full search space of either one or zero. The upper and lower limits were one and zero. Following initialization, the positions of the SAs must be updated. Consequently, the SCSO algorithm facilitated the SC's foraging and hunting phases through its distinctive hearing capability. In every iteration, every SA acquired a position for updating, utilizing the V-shaped TF to convert the result to a range between one and zero.

$$V\left(x_i^n(t)\right) = \frac{2}{\pi}\arctan\left(\frac{\pi}{2}x_i^n(t)\right) \tag{11}$$

$$x_i^n(t+1) = \begin{cases} (x_i^n)^{-1} & if\ rand\ < V\left(x_i^n(t)\right) \\ (x_i^n) & Otherwise \end{cases} \tag{12}$$

Four V-shaped TFs and a positional update rule were executed to attain the objective. Here, $x_i^n(t)$ Denotes the position of the i^{th} SA in the n^{th} dimension at iteration t. The random number ($rand$) was a uniformly distributed *random number* between zero and one. Equation (12) is applied to update the position of SA opposite to the acquired position or similar to the acquired position. Figure 4 depicts the flowchart of the implemented BSCSO technique for feature selection.

The SA position was established through selecting or removing features, represented by binary vectors where '1' indicates the selection of a feature related to the SA position, and '0' indicates non-selection. Feature selection procedures aim to optimize classification accuracy while reducing the total count of features. The BSCSO method considered these two aims throughout its dynamic search to identify the best suitable conjunction of features for the applications. The binary SCSO employed a Fitness Function (FF) to determine the agent's positions using Equation (13):

$$FF = \alpha * ER + \beta\frac{|S|}{|C|} \tag{13}$$

The error rate, ER has been defined as the proportion of incorrectly classified instances to the total count of occurrences. S represented the size of the feature subset, C denoted the overall count of features, and β indicated the count of samples that were misclassified. The variables α and β were the vectors of weights for assessing the significance of the accuracy of classification and size of features [32].

A total of 15 optimal features have been selected from the CIC-IoT-2023 dataset using the BSCSO approach. The features selected are Duration, flow_duration, Protocol_Type, Header_Length, Srate, fin_count, urg_count, syn_count, HTTPS, rst_count, Min, IAT, Tot size, Weight, and Covariance. From the BoT-IoT dataset, a total of 13 features have been selected using the BSCSO technique. The selected features are Stime, saddr, daddr, ltime, dbytes, Rate, Srate, Drate, TnP_PerProto, AR_P_Proto_p_SrcIP, AR_P_Proto_p_DstIP, AR_P_Proto_p_Sport, and AR_P_Proto_p_Dport.

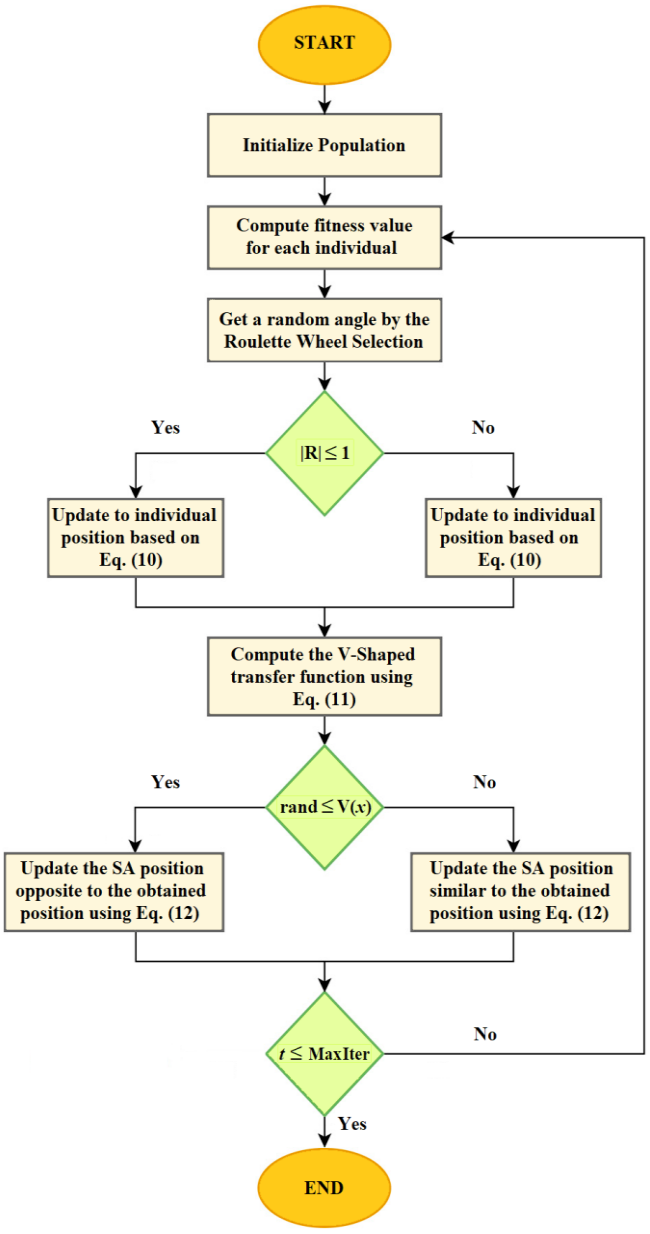

Fig. 4 Flowchart of BSCSO

3.5. STNN Attack Classification Model

This research implements the STNN model to detect and classify intrusions [33]. The STNN model focuses on effectively addressing the non-linear input data transformation using Equation (14). By analyzing the transformer, the model

specifically constructs $\Phi = [\Phi_1, \Phi_2, \dots \Phi_L]'$, which serves as a smooth diffeomorphism mapping.

$$\begin{bmatrix} \Phi\begin{pmatrix} x_1^1 \\ x_2^1 \\ \vdots \\ x_D^1 \end{pmatrix} \Phi\begin{pmatrix} x_1^2 \\ x_2^2 \\ \vdots \\ x_D^2 \end{pmatrix} \dots \\ \Phi\begin{pmatrix} x_1^M \\ x_2^M \\ \vdots \\ x_D^M \end{pmatrix} \end{bmatrix} = \begin{bmatrix} y^1 & y^2 & \cdots & y^M \\ y^2 & y^3 & \cdots & y^{M+1} \\ \vdots & \vdots & \ddots & \vdots \\ y^L & y^{L+1} & \cdots & y^{M+L-1} \end{bmatrix} \tag{14}$$

Here, D denotes the variable dimensions, L signifies the embedding dimensions, and M indicates the observed data steps.

$$\Phi(X^t, \bar{Y}^t) = Dec(Enc(X^t), \bar{Y}^t) = \hat{y}^t \tag{15}$$

The STNN model utilizes the data transformation equation alongside two distinct transformer units to provide multi-step-ahead prediction. According to the description given in Equation (15), one unit is the encoder, which simultaneously accepts the variables of D dimensions at $t(X^t)$ time as input. Subsequently, the encoder derives pertinent spatial data from the input data. The pertinent spatial data is thereafter transmitted to the decoder. The decoder receives an $L-1$ length data from the objective variable $Y(\bar{Y}^t)$. Then, the decoder retrieves the temporal data of the objective variable. The decoder forecasts the eventual values of the objective variable ($\hat{y}^t$) by integrating the spatial data of the input data (X^t) with the temporal data of the objective variable ($\bar{Y}^t$). Observe that y in Y constitutes a single variable within the observed variables X. The variable Φ in Equation (15) differs from that in Equation (14) because of $\bar{Y}^t$; nonetheless, Φ can be represented in a comparable manner by a suitable mathematical formulation. The encoder-decoder pair effectively resolves the nonlinear spatial data transformation Φ. $\bar{Y}^t = (0, y^t, y^{t+1}, \dots, y^{t+L-2})$ constitutes an L-dimensional time series, created by substituting the initial dimensions of $Y^{t-1} = (y^{t-1}, y^t, \dots, y^{t+L-2})'$ With 0. Hence, the link between the variables of the prediction is preserved.

3.5.1. Encoder

The encoder consists of two layers. One component is a Fully Connected (FC) layer, while another is a Continuous Spatially Self-Attention (CSSA) layer. This CSSA layer was utilized for obtaining pertinent spatial data from the highly dimensional variable inputs. X^t. The FC layer is utilized to derive an effective representation by normalizing the highly dimensional input variables. X^t And mitigating noise, constituting a forward propagating network defined by Equation (16).

$$X_{FFNN}^t = ReLU(W_{FFNN}X^t + b_{FFNN}) \tag{16}$$

In this context, FFFN denotes a feedforward neural network, W_{FFNN} represents the coefficient matrix, b_{FFNN} signifies the bias, and ReLU refers to the function of activation. The continuous spatially self-attention layer utilizes X_{FFNN}^t As its input, the self-attention layer simultaneously processes high-dimensional variables, enabling the encoder to obtain spatial data from the input. To acquire efficient spatial data $(S\acute{S}A^t)$A Continuous Attention Mechanism (CAM) was utilized for the Spatial Self-Attention (SSA) layer, in contrast to the traditional discrete probability-based attention method. Initially, three training weight matrix structures, such as W_E^Q, W_E^K, and W_E^V These were generated for the continuous SSA layer in the CAM. Equation (17) calculates the query matrix (Q_E^t), key matrix (K_E^t), and value matrix (V_E^t) for the continuous SSA layer through the multiplication of the output X_{FFNN}^t of the FC layer by all three of the previous weight matrices at each time step t.

$$\begin{cases} Q_E^t = X_{FFNN}^t W_E^Q \\ K_E^t = X_{FFNN}^t W_E^K \\ V_E^t = X_{FFNN}^t W_E^V \end{cases} \tag{17}$$

Equation (18) performs the matrix dot product to derive an equation for efficient spatial data. $(S\acute{S}A^t)$ for the given input variables X^t.

$$S\acute{S}A^t = \exp\left(\frac{1}{\sqrt{d_E}} \cdot Q_E^t \cdot K_E^{t\prime}\right) \cdot V_E^t \tag{18}$$

Here, d_E denotes the dimension of the Q_E^t, K_E^t, and V_E^t. The CAM ensures a seamless transfer of data for the encoder. The normalized representation of actual spatial data (SSA^t) was calculated utilizing residual connection and layer normalization (Equation (19)), which mitigates gradient vanishing and enhances model speed of convergence.

$$SSA^t = Norm\left(X_{FFNN}^t + S\acute{S}A^t\right) \tag{19}$$

3.5.2. Decoder

The decoder integrates efficient temporal and spatial evolution data and comprises dual FC layers: Continuous Temporal Self-Attention (CTSA) and Transformation Attention (TA) layers. The derivation of the efficient expression ($\bar{Y}_{FFNN}^t$) is obtained subsequent to the noise filtration of the input data ($\bar{Y}^t$) via an FC layer. Subsequently, the output ($\bar{Y}_{FFNN}^t$) is transmitted to the CTSA layer. The CTSA layer concentrates on the previous temporal evolution data across several time steps of the target variable ($\bar{Y}^t$). The irreversible impact on time necessitates that the present condition of the time series be determined utilizing previous data rather than next data. The CTSA layer employs a masked attention method to exclude future data. The comprehensive technique is outlined as follows.

Initially, three training weight matrices: W_D^Q, W_D^K, and W_D^V They were created for the Temporally Spatial Self-Attention (TSSA) layer. In addition, Equation (20) calculates the query matrix (Q_D^t), key matrix (K_D^t), and value matrix (V_E^t) for the TSSA layer.

$$\begin{cases} Q_D^t = \bar{Y}_{FFNN}^t W_D^Q \\ K_D^t = \bar{Y}_{FFNN}^t W_D^K \\ V_D^t = \bar{Y}_{FFNN}^t W_D^V \end{cases} \quad (20)$$

Furthermore, Equation (21) performs the matrix dot product to derive the equation for the temporal data ($T\acute{S}A^t$) concerning the variable of input ($\bar{Y}^t$).

$$T\acute{S}A^t = \exp\left(\frac{1}{\sqrt{d_D}} \cdot Q_D^t \cdot K_D^{t\prime} \cdot Mask\right) \cdot V_D^t \quad (21)$$

Here, d_D denotes the size of the Q_D^t, K_D^t, and V_E^t Inside the TSSA layer. Furthermore, Equation (22) was utilized to characterize the mask matrix with a d_M Dimension.

$$Mask = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 1 & 1 & \cdots & 1 & 1 \end{bmatrix} \quad (22)$$

In Equation (22), assigning 0 in the mask matrix prevented every position from attending to subsequent positions, preserving the past temporal evolution data associated with the objective variable. The normalized representation of the Temporal Evolution Information (TSA^t) in Equation (23) was calculated utilizing normalization and residual connection layer.

$$TSA^t = Norm\left(Y_{FFNN}^t + T\acute{S}A^t\right) \quad (23)$$

Additionally, the CTSA layer integrates the efficient spatial data (SSA^t) and the Temporal Evolution Data (TSA^t) to forecast the future values of the objective variable ($\acute{T}A^t$) as delineated in Equation (24). The variable d_{SSA^t} denotes the size of SSA^t.

$$\acute{T}A^t = \frac{1}{\sqrt{d_{SSA^t}}} TSA^t \cdot SSA^{t\prime} \cdot SSA^t \quad (24)$$

Finally, the TA^t is incorporated into the residual connection, applied the normalization layer, and arranged an FC layer appropriately to calculate the L-dimensional predictive result $\hat{Y}^t$.

$$\begin{cases} TA^t = Norm\left(TSA^t + \acute{T}A^t\right) \\ \hat{Y}^t = ReLU(W \cdot TA^t + b) \end{cases} \quad (25)$$

Here in Equation (25), W represents the coefficient matrix and b denotes the bias. The STNN model establishes the objective function in Equation (26) to reduce the loss.

$$\min \varepsilon = \sum_{t=1}^{M-L+1} \left\| \hat{Y}^t - Y^t \right\|_2^2 + \lambda \|W\|_2^2 \quad (26)$$

Here, M denotes the time-series steps observed, L signifies the predetermined window size, and $\hat{Y}^t$ and Y^t Indicate the anticipated and actual values of a target variable accordingly. $\|\cdot\|_2^2$ Denotes the Frobenius norm, λ regulates the significance of the penalty, and W represents the STNN's parameter space [34]. The pseudocode of the developed IDS model is presented below.

Input: Raw IoT datasets (CIC-IoT-2023, BoT-IoT)
Output: Intrusion classification
Initialization
Load CIC-IoT-2023 and BoT-IoT datasets into Spark DataFrames.
Merge and clean datasets.
Apply Label Encoding to categorical features.
Normalize numerical features using Normalization.
Apply SMOTE to balance the dataset for class imbalance.
Initialize population of sand cats with binary feature vectors.
For each iteration:
Evaluate fitness function.
Update positions using BSCSO position update rules.
Apply binary thresholding to get a feature selection mask.
Select the optimal feature subset with the highest fitness.
Split the selected feature dataset into training and testing sets.
Initialize the STNN model with encoder-decoder attention blocks.
Feed training data (features with temporal ordering preserved).
Train using backpropagation and optimization (e.g., Adam).
Validate the model on test data.
Predict test labels using the trained STNN.
End

Table 4. Hyperparameter Tuning of the Model

Model	Hyperparameter	Value
BSCSO	Population Size	30
	Maximum Iterations	50
	Sand Movement Coefficient (α)	1.5
	Threshold for Binary Conversion	0.5
STNN	Input Sequence Length	50
	Embedding Dimension	128
	Number of Attention Heads	4
	Number of Transformer Blocks	3
	Learning Rate	0.001
	Dropout Rate	0.2
	Epochs	50
	Batch Size	64
	Optimizer	Adam

Table 4 shows the hyperparameter values set for the developed research model. The hyperparameter tuning ensures effective learning and model convergence while minimizing overfitting and optimizing computational efficiency for BD in Spark-based environments.

4. Experimentation Analysis

4.1. Experiment Setup

This section highlights the experiments conducted on the extensive CIC-IoT-2023 and BoT-IoT datasets. The study employed the PySpark tool, which facilitates programming with Python on the Apache Spark BD platform within the Google Colab environment. The proposed system is implemented using PySpark. It is a library that connects Python with Apache Spark. All testing was conducted on Windows 10 64-bit, utilizing a Core i7 processor operating at 2.70GHz, 16 GB of RAM, and the programming language, Python. The dataset is divided into testing and training halves of 20% and 80%, respectively.

4.2. Result Metrics

The research assessed the efficiency of the BSCSO-STNN model developed for binary and multiclass classifications. Accuracy, Precision, Detection Rate, and F1-score parameters were computed to assess performance. The following are the formulas of these parameters presented in Equations (27) to (30). In these equations, TP-True Positive is the count of properly classified samples that are positive. TN-True Negative is the count of properly classified samples that were negative. FP-False Positive is the count of erroneously identified samples that were negative as positive. And FN-False Negative is the count of erroneously classified samples that were positive as negative.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (27)$$

Accuracy evaluates the model's entire capacity to categorize outcomes, accurately providing a comprehensive performance assessment.

$$Precision = \frac{TP}{TP+FP} \quad (28)$$

Precision assesses the dependability of positive prediction and aims to reduce the FPs. This metric is especially critical in IDS to prevent redundant false alarms, which could degrade the effectiveness of the system.

$$Detection\ Rate = \frac{TP}{TP+FN} \quad (29)$$

The detection rate, or recall, evaluates the BSCSO-STNN's ability to detect true intrusions. The DR with higher recall values is necessary to minimize missed detections and provide a resilient system efficient at thoroughly detecting attacks.

$$F1score = 2 \times \frac{Recall \times Precision}{Recall+Precision} \quad (30)$$

The F1-score is a harmonic average of recall and precision. It represents a balanced assessment of the classifier's total efficacy, which focuses on its capacity to uphold efficiency and accuracy. A high F1-score signifies the model's efficiency in attaining an ideal balance between recall and precision [11-25].

4.3. Performance Evaluation

This performance evaluation section discusses the results of the developed BSCSO-STNN model. The results are computed for both binary and multiclass classification on CIC-IDS-2023 and BoT-IoT datasets. The results are computed and discussed individually for both datasets in the following.

Table 5. Binary classification results on the CIC-IoT dataset

Metric (%)	Training	Testing
Accuracy	99.78	99.46
Detection Rate	99.62	99.38
Precision	99.75	99.41
F1-score	99.60	99.37

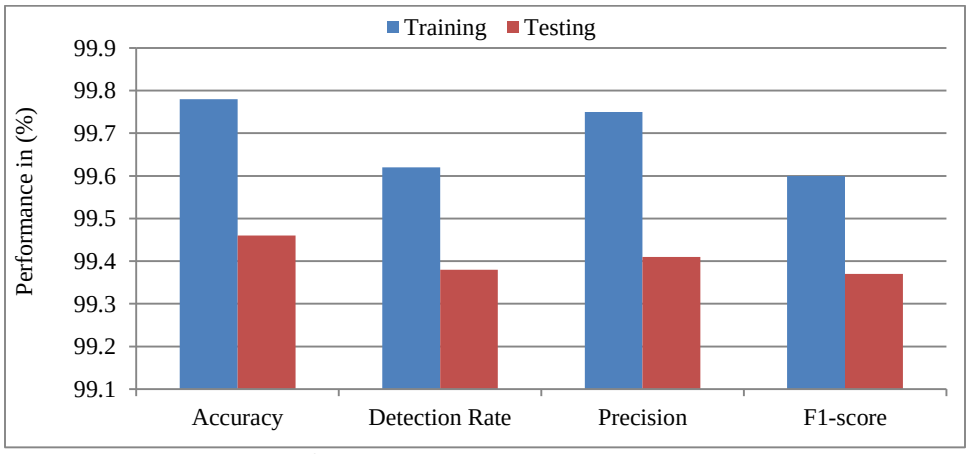

Fig. 5 Graphical illustration of BSCSO-STNN model's binary classification on CIC-IoT-23 dataset

Table 5 presents the results of the BSCSO-STNN model in classifying attacks from the CIC-IoT dataset based on binary classification. Using the selected optimal features using the BSCSO technique, the STNN model performed the classification with the best results. The developed model attained 99.78% accuracy in training and 99.46% in test data. The difference between the training and test accuracy is 0.32%. The BSCSO-STNN model achieved a 99.62% detection rate in training and 99.38% in testing. The difference between the training and test detection rate is 0.24%.

The precision score of the developed model is 99.75% in training and 99.41% in testing. The difference between the training and test precision score is 0.34%. The F1-score of the research model is 99.60% in training and 99.37% in testing. The difference between the training and test F1-score is 0.23%. Based on these obtained results, the BSCSO model demonstrated an effective and accurate performance in detecting attacks from the CIC-IoT-23 dataset based on binary classification. Figure 5 illustrates the graphical chart of the developed BSCSO-STNN model's binary classification results evaluated using the CIC-IoT-23 dataset. Table 6 presents the results of the BSCSO-STNN model in classifying attacks from the BoT-IoT dataset based on the binary classification. Upon selecting the best thirteen optimal features from the dataset using BSCSO, the STNN model classified the data based on whether the attack was present or not. The developed model attained 99.83% accuracy in training and 99.52% in test data. The difference between the training and test accuracy is 0.31%. The BSCSO-STNN model achieved a 99.75% detection rate in training and 99.39% in testing. The difference between the training and test detection rate is 0.36%. The precision score of the developed model is 99.80% in training and 99.48% in testing. The difference between the training and test precision score is 0.32%. The F1-score of the research model is 99.68% in training and 99.46% in testing. The difference between the training and test F1-score is 0.22%. The BSCSO-STNN model has performed consistently on all the parameters. According to these attained results, the BSCSO model demonstrated an effective and accurate performance in detecting attacks from the BoT-IoT dataset based on binary classification. Figure 6 depicts the graphical chart of the proposed BSCSO-STNN model's binary classification results evaluated using the BoT-IoT dataset.

Table 6. Binary classification results on the BoT-IoT dataset

Metric (%)	Training	Testing
Accuracy	99.83	99.52
Detection Rate	99.75	99.39
Precision	99.80	99.48
F1-score	99.68	99.46

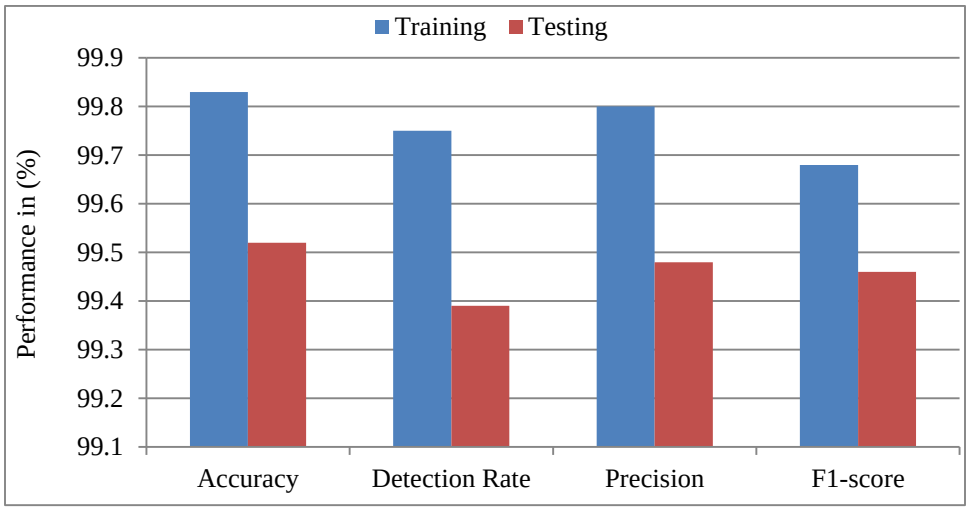

Fig. 6 Graphical illustration of BSCSO-STNN model's binary classification on BoT-IoT dataset

Table 7. Multiclass classification results on CIC-IoT-23 dataset

Classes	Accuracy	DR	Precision	F1-score
Benign	99.53	99.59	99.38	99.48
DDoS	99.27	99.04	99.59	99.29
Brute Force	98.95	98.71	98.84	98.78
Spoofing	98.61	98.30	98.37	98.21
DoS	99.02	98.96	99.06	98.89
Recon	99.10	98.98	98.87	98.93
Web-based	98.78	98.50	98.77	98.69
Mirai	99.35	99.16	99.28	99.22
Average	99.08	98.78	99.02	98.94

Table 7 presents the multiclass classification of the developed BSCSO-STNN when evaluating the CIC-IoT-23 data set. This research has performed multiclass classification on all the attack classes present in the dataset. A total of eight classes are classified, which include seven attack classes and one benign class. The BSCSO-STNN model has attained higher results on the benign or normal class. The model achieved an accuracy in the range from 98.61% to 99.53%. The accuracy score of the model for classes like benign, DDoS, DoS, Recon, and Mirai have reached over 99%.

The accuracy score on other classes, like brute Force, spoofing, and web-based, is in the range of over 98%. In this dataset, the DDoS, DoS, and Mirai are the dominant classes, where the research model has attained the best accuracy scores for these classes. The average accuracy is computed as 99.08%. This highlights that the model has performed well in detecting attacks.

The BSCSO-STNN model has obtained the detection rate in the range from 98.30% to 99.59%. The detection rate is superior for classes like benign, DDoS, DoS, and Mirai, with 99%. The detection rate for the remaining classes is in the range of 98%. The average detection rate is computed as 98.78%. The precision score of the developed model is attained in the range from 98.37% to 99.59%. The best precision score is obtained for classes like benign, DDoS, DoS, and Mirai in the range of 99%. For other classes, the precision is in the range of 98%. The average precision score was 99.02%. The research model has achieved an F1-score in the range from 98.21% to 99.48% for all the classes. The best F1 scores are attained for classes like benign, DDoS, and Mirai, which range from 99%. For the remaining classes, the model achieved an F1-score in the 98% range. The average f1-score of the model was 98.94%. Figure 7 depicts the graphical chart of the proposed BSCSO-STNN model's multiclass classification results evaluated using the CIC-IoT-23 dataset. Table 8 presents the multiclass classification of the developed BSCSO-STNN model for evaluating the BoT-IoT dataset. A total of nine classes are classified, which includes eight attack classes and one benign class. The BSCSO-STNN model has attained higher results on the benign or normal class. The model achieved an accuracy in the range from 98.85% to 99.32%. The accuracy score of the model for classes like benign, DoS-HTTP, DoS-TCP, DDoS-HTTP, and DDoS-TCP have reached over 99%. The accuracy score on other classes is in the range of over 98%. In this dataset, the DDoS and DoS are the dominant classes, where the research model has attained the best accuracy scores for these classes. The average accuracy is computed as 99.04%.

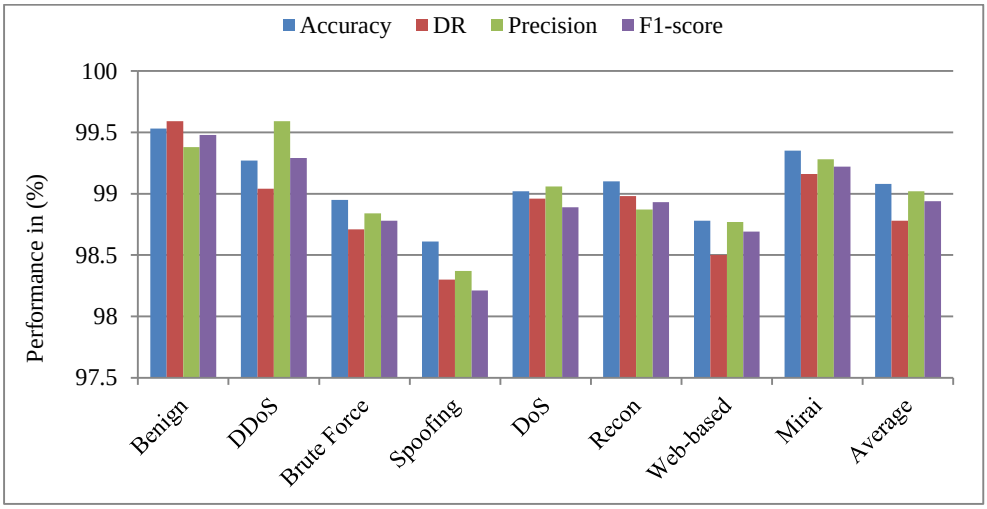

Fig. 7 Graphical illustration of BSCSO-STNN model's multiclass classification on CIC-IoT-23 dataset

Table 8. Multiclass classification results on BoT-IoT dataset

Classes	Accuracy	DR	Precision	F1-score
Benign	99.32	99.19	99.25	99.23
DoS-HTTP	99.19	99.03	99.10	99.08
DoS-TCP	99.07	99.00	99.03	98.99
DoS-UDP	98.85	98.77	98.82	98.80
DDoS-HTTP	99.18	99.03	99.09	99.05
DDoS-UDP	98.89	98.80	98.87	98.81
DDoS-TCP	99.08	98.93	98.99	99.02
OS FP	98.85	98.71	98.78	98.79
Service Scan	98.96	98.80	98.82	98.81
Average	99.04	98.81	98.97	98.95

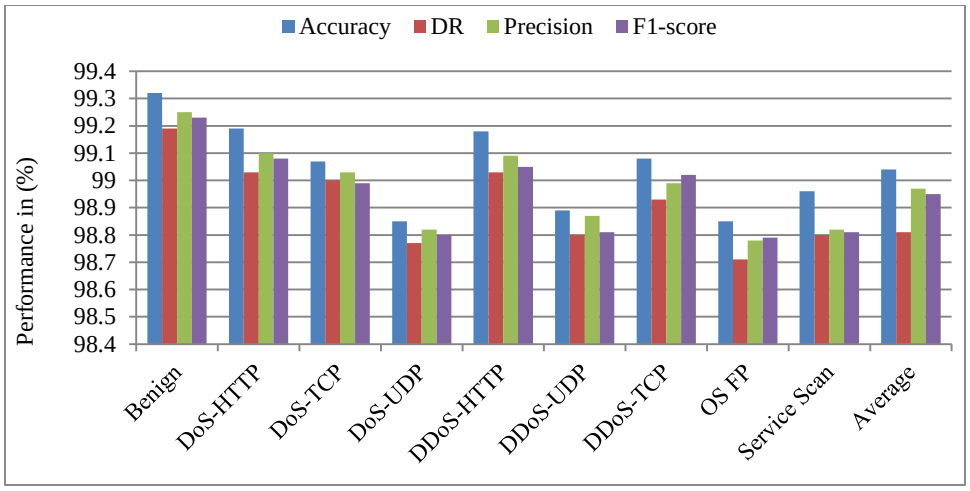

Fig. 8 Graphical illustration of BSCSO-STNN model's multiclass classification on BoT-IoT dataset

This highlights that the model has performed well in detecting attacks. The BSCSO-STNN model has obtained the detection rate in the range from 98.71% to 99.19%. The detection rate is superior for classes like benign, DoS-HTTP, DoS-TCP, and DDoS-HTTP, with 99%. The detection rate for the remaining classes is in the range of 98%. The average detection rate is computed as 98.81%. The precision score of the developed model is attained in the range from 98.78% to 99.25%. The best precision score is obtained for the classes like benign, DoS-HTTP, DoS-TCP, and DDoS-HTTP in the range of 99%. For other classes, the precision is in the range of 98%. The average precision score was 98.97%. The research model has achieved an F1-score in the range from 98.79% to 99.23% for all the classes. The best F1 scores are attained for classes like benign, DoS-HTTP, and DDoS-HTTP, which range from 99%. For the remaining classes, the model achieved an F1-score in the 98% range. The average f1-score of the model was 98.95%. Figure 8 depicts the graphical chart of the proposed BSCSO-STNN model's multiclass classification results evaluated using the BoT-IoT dataset.

Table 9. Performance comparison with current models

Models	Accuracy	Detection Rate	Precision	F1-score
ML-ANN [12]	98.87	98.46	98.97	98.80
LSTM [13]	96.11	NA	NA	NA
LR [15]	96.52	94.38	94.02	94.20
NB [15]	95.85	94.21	92.11	93.15
SVM [15]	97.36	94.02	97.08	95.53
CNN-LSTM [16]	99.00	NA	NA	NA
CNN-RNN [17]	96.10	96.30	95.80	95.80
LVW-MECO [18]	95.10	95.32	95.98	NA
PPSLOA-HDBDE [19]	98.89	96.64	96.43	96.53
CNN [22]	97.81	98.00	98.00	98.00
XGB [25]	98.89	98.12	96.70	97.41
Proposed Model (CIC-IoT)	99.08	98.78	99.02	98.94
Proposed Model (BoT-IoT)	99.04	98.81	98.97	98.95

Table 9 presents the proposed BSCSO-STNN model's average multiclass classification results with the current models discussed in the related works section. Both the average results of the model's multiclass classification performance were applied for this comparison. The research model has the average scores in the CIC-IoT-23 dataset with 99.08% accuracy, 98.78% detection rate, 99.02% precision, and 98.94% F1-score. For the BoT-IoT dataset, the model attained 99.04% accuracy, 98.81% detection rate, 98.87% precision, and 98.95% F1-score. The proposed model has the best accuracy of 99.08%, which is 0.21% to 3.98% higher than the compared models in this research. The CNN-LSTM model performs very closely to the developed model with 99% accuracy. Models like ML-ANN, PPSLOA-HDBDE, and XGB have close accuracy scores of 98.87% and 98.89%. The least performed model was LVW-MECO with 95.10%. The best detection rate of the BSCSO-STNN model was 98.81, which is 0.35% to 4.79%, which is an improvement over the current models. The models like XGB, CNN, and ML-ANN have achieved a detection rate in the range of 98%, which is closer to the proposed model's detection rate. The least performed model was SVM with a 94.02% detection rate. The

BSCSO-STNN model has its best precision score of 99.02%, which is 0.15% to 6.91% higher than the current models in this comparison. The ML-ANN and CNN models have attained precision scores of 98.97% and 98%, which are closer to the results of the proposed model. The least performed model was NB with 92.11%. The best f1-score of the model was 98.95%, which is 0.15% to 5.8% higher than the compared models.

The ML-ANN model has the closest range of f1-score with 98.80%, and the next best was CNN with 98%. The least performed model was NB with 93.15%. Figure 9 depicts the comparison of the results in a graphical chart. Overall, the developed BSCSO-STNN model has outperformed the other current models based on the obtained multiclass classification performance. It is highly applicable to the task of performing intrusion detection in BD applications.

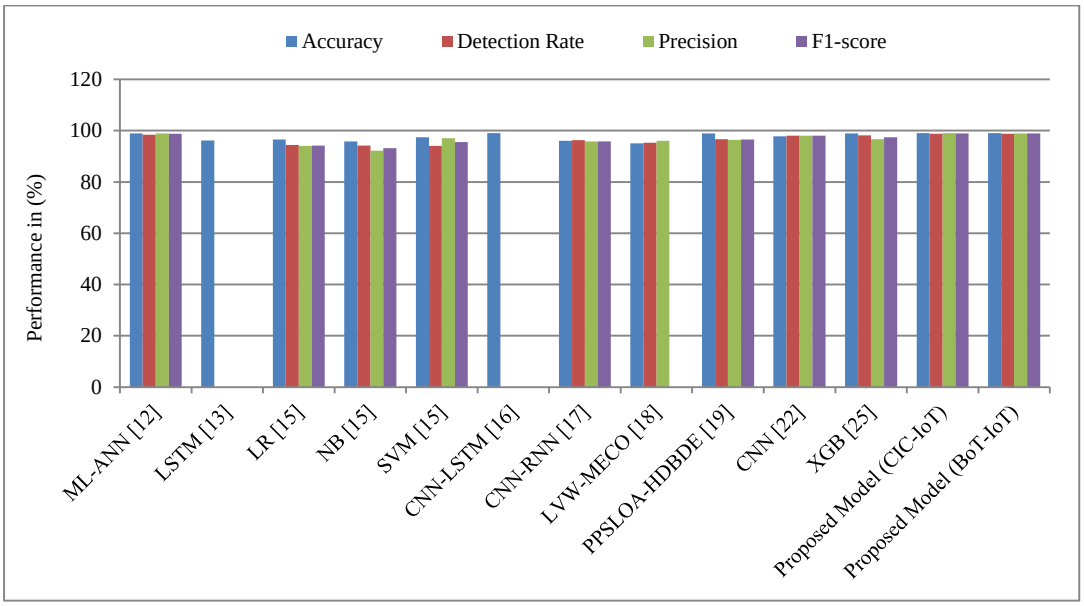

Fig. 9 Graphical illustration of results comparison

This BSCSO-STNN model has many advantages, including its integration of BSCSO for selecting the best optimal features and implementing STNN for classification.

The BSCSO model effectively minimizes the dimensionality and improves the associated feature selection. This efficient feature selection process results in improved classification performance. The STNN model's higher accuracy and detection rate over multiple attack classes are also significant.

The BSCSO-STNN model's adaptability with the APS framework enables the model to handle large datasets and perform efficient processing. However, the model has its own limitations. The STNN has higher computational complexity due to the transformer architecture. Moreover, the duration of training is longer than that of the few current models.

5. Conclusion

This research developed an IDS model named BSCSO-STNN for detecting and classifying intrusions in a BD framework. The model was developed by integrating the BSCSO technique with STNN. The model was designed to perform intrusion detection based on data collection, preprocessing, feature selection, and classification. The CIC-IoT-23 and Bot-IoT datasets were collected and applied to this model for training and evaluation. The developed BSCSO-STNN model was deployed in an APS framework. The datasets were initially preprocessed in this framework with data cleaning, oversampling, label encoding, and normalization. After preprocessing, the data was applied to the BSCSO for feature selection. Based on the BSCSO's fitness values, the technique individually selected the optimal features from both datasets. Using the selected features, the STNN model performed the classification for both datasets. The STNN model was effective in capturing temporal and spatial patterns, which were helpful in this research to detect the network attack patterns for detecting intrusions.

Finally, the classification's performance on binary and multiclass classification was assessed using the performance indicators. The BSCSO-STNN model attained 99.08% accuracy, 98.78% detection rate, 99.02% precision, and 98.94% F1-score using the CIC-IoT-23 dataset. The model attained 99.04% accuracy, 98.81% detection rate, 98.97% precision, and 98.95% F1-score for the BoT-IoT dataset in multiclass classification. The developed model outperformed all the current models in this research and demonstrated its accuracy in detecting intrusions. In future, the research will focus on improving the model by integrating continual

learning methods for handling multiple intrusions and unknown attacks. Additionally, the limitations of the model, like computational complexity and training Duration, will be effectively reduced by applying a lightweight transformer model. Furthermore, the research will be evaluated by applying recent datasets and real-time network traffic to increase the model's generalization.

Acknowledgment

The authors would like to thank the Sona College of Arts and Science, Salem, and Thiruvalluvar Government Arts College, Rasipuram, for their support in completing this project.

References

[1] Liyuan Sun, Hongyun Zhang, and Chao Fang, "Data Security Governance in the Era of Big Data: Status, Challenge, and Prospects," *Data Sciences and Management*, vol. 2, pp. 41-44, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[2] Fadi Salo et al., "Data Mining with Big Data in Intrusion Detection Systems: A Systematic Literature Review," *arXiv Preprints*, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Mohammed M. Alani, "Big Data in Cybersecurity: A Survey of Applications and Future Trend," *Journal of Reliable Intelligent Environment*, vol. 7, no. 2, pp. 85-114, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[4] Mohanad G. Yaseen, and A.S. Alabahri, "Mapping the Evolutions of Intrusion Detections in Big Data: A Bibliometric Analysis," *Mesopotamian Journals of Big Data*, vol. 2023, pp. 138-148, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Imane Laassar, and Moulay Youssef Hadi, "Intrusion Detection System for Internet of Things-Based Big Data: A Review," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 12, no. 1, pp. 87-96, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[6] Sanaa Mouhim, and Fadwa Lachhab, "Toward a Contexts Awareness System Using IoTs, AI, and Big Data Technologies," *IEEE Access*, vol. 13, pp. 40302-40315, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[7] Dina Fawzy, Sherin M. Moussa, and Nagwa L. Badr, "The Internet of Things and Architecture of Big Data Analytics: Challenge of Intersections at Different Domains," *IEEE Access*, vol. 10, pp. 4969-4992, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] KarthikKumar Vaigandla, Nilofar Azami, and Radha Krishna Karane, "Investigations on Intrusion Detection Systems (IDS) in IoTs," *International Journal of Emerging Trends in Engineering Research*, vol. 10, no. 3, pp. 158-166, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[9] Amjad Rehman Khan et al., "Deep Learning for Intrusion Detection and Security of Internet of Things (IoTs): Current Analysis, Challenge, and Possible Solution," *Security and Communications Networks*, vol. 2022, no. 1, pp. 1-13, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] Muhammadu Sathik Raja Sathik Raja M.S, "The Rise of AI-Driven Networks Intrusions Detections System: Innovation, Challenge, and Future Direction," *International Journal of AI, Big Data, Computational and Management Studies*, vol. 6, no. 1, pp. 1-9, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[11] Chao Li et al., "An Anomaly Detections Approach Based on Integrated LSTMs for IoTs Big Data," *Security and Communications Networks*, vol. 2023, no. 1, pp. 1-10, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[12] Abdelaziz Al Dawi et al., "An Approach to Botnet Attacks in the Fog Computing Layers and Apache Spark for Smart Cities," *The Journal of Supercomputing*, vol. 81, no. 4, pp. 1-30, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[13] Muhammad Babar et al., "An Improved Big Data Analytics Architecture for Intruder Classifications using Machine Learning," *Security and Communications Networks*, vol. 2023, no. 1, pp. 1-7, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[14] Janusz Granat et al., "Big Data Analytics for Event Detection in the IoTs-Multicriteria Approach," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4418-4430, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[15] Sikha Bagui et al., "Detecting Reconnaissance and Discovery Tactic from the MITRE ATT&CK Frameworks in Zeek Conn Log using Spark's Machine Learning in the Big Data Frameworks," *Sensors*, vol. 22, no. 20, pp. 1-25, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[16] Ali Alferaidi et al., "Distributed Deep CNN-LSTM Models for Intrusion Detection Methods in IoT-Based Vehicle," *Mathematical Problems in Engineering*, vol. 2022, no. 1, pp. 1-8, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[17] Farhan Ullah et al., "Enhanced Networks Intrusion Detection Systems for Internet of Things Security using Multimodal Big Data Representations with Transfer Learning and Game Theory," *Sensors*, vol. 24, no. 13, pp. 1-31, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[18] Jie Wang et al., "Multicriteria Features Selection Based Intrusions Detections for Internet of Things Big Data," *Sensors*, vol. 23, no. 17, pp. 1-17, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[19] Fatma S. Alrayes et al., "Privacy-Preserving Approach for IoTs Network using Statistical Learning with Optimizations Algorithms on High-Dimension Big Data Environments," *Scientific Report*, vol. 15, no. 1, pp. 1-27, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[20] Mazhar Javed Awan et al., "Real-Time DDoS Attacks Detection Systems using Big Data Approach," *Sustainability*, vol. 13, no. 19, pp. 1-19, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Lijuan Deng, Long Wan, and Jian Guo, "Research on Security Anomaly Detections for Big Data Platform based on Quantum Optimizations Clustering," *Mathematical Problems in Engineering*, vol. 2022, no. 1, pp. 1-10, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[22] Samed Al, and Murat Denner, "STL-HDL: A New Hybrid Network Intrusion Detection System for Imbalanced Datasets on Big Data Environments," *Computers & Security*, vol. 110, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[23] Farah Jemili, "Toward Data Fusion-Based Big Data Analytics for Intrusion Detection," *Journal of Information and Telecommunications*, vol. 7, no. 4, pp. 409-436, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[24] Ricardo Alejandro Manzano Sanchez et al., "Toward Developing A Robust Intrusion Detection Model using Hadoop Spark and Data Augmentations for IoT Network," *Sensors*, vol. 22, no. 20, pp. 1-17, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[25] Ahmed Alrefaei, and Mohammad Ilyas, "Using Machine Learning Multiclass Classification Techniques to Detect IoT Attacks in Real Time," *Sensors*, vol. 24, no. 14, pp. 1-19, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[26] Viet Anh Phan, Jan Jerabek, and Lukas Malina, "Comparisons of Multiple Feature Selection Techniques for Machine Learning-Based Detections of IoT Attacks," *ARES '24: Proceedings of the 19th International Conferences on Availability, Reliability and Security*, Vienna Austria, pp. 1-10, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[27] Xiangyu Liu, and Yanhui Du, "Toward Effective Feature Selections for IoT Botnet Attack Detection using a Genetic Algorithm," *Electronics*, vol. 12, no. 5, pp. 1-12, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[28] L. Madhuridevi, and N.V.S. Sree Rathina Lakshmi, "Metaheuristics-Assisted Hybrid Deep Classifier for Intrusion Detection: A Big Data Perspective," *Wireless Networks*, vol. 31, no. 2, pp. 1205-1225, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[29] Thi-Thu-Huong Le et al., "Toward Unbalanced Multiclass Intrusion Detection with Hybrid Sampling Method and Ensembled Classifications," *Applied Soft Computing*, vol. 157, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[30] Nojood O. Aljehane et al., "Optimizing Intrusion Detections using Intelligent Feature Selections with Machine Learning Models," *Alexandria Engineering Journal*, vol. 91, pp. 39-49, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[31] Amir Seyyedabbasi, "Binary Sand Cats Swarm Optimizations Algorithm for Wrapper Features Selections on Biological Data," *Biomimetics*, vol. 8, no. 3, pp. 1-19, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[32] Elnaz Pashaei, "An Efficient Binary Sand Cats Swarm Optimizations for Feature Selection in High-Dimensional Biomedical Data," *Bioengineering*, vol. 10, no. 10, pp. 1-17, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[33] Junzhong Ji et al., "Spatio-Temporal Transformers Networks for Weather Forecasting," *IEEE Transactions on Big Data*, vol. 11, no. 2, pp. 372-387, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[34] Yujie You et al., "Spatiotemporal Transformers Neural Networks for Time-Series Forecasting," *Entropy*, vol. 24, no. 11, pp. 1-18, 2022. [CrossRef] [Google Scholar] [Publisher Link]