Al-Rafidain Journal of Computer Sciences and Mathematics (RJCSM)

www.csmj.uomosul.edu.iq

Review of a Unified Real-Time IDS and Mitigation Framework Using Apache Spark

Ammar Ahmed Abdullah[1] and Dhuha Basheer Abdullah[2]

[1,2] Department of Computer Science, College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq
Email: ammar.20ba1175@student.uomosul.edu.iq[1] and prof.dhuha_basheer@uomosul.edu.iq[2]

Article information

Article history:
Received 31 July, 2025
Revised 23 August, 2025
Accepted 25 September, 2025
Published 25 December, 2025

Keywords:
Cyber Threat Intelligence,
Big Data,
Apache Spark,
Machine Learning,
Intrusion Detection System,
DDoS,
Real-Time Analytics.

Correspondence:
Ammar Ahmed Abdullah
Email:
ammar.20ba1175@student.uomosul.edu.iq

Abstract

Traditional intrusion detection systems are being surpassed by the increasingly sophisticated cyber threats that modern networks face. The increasing scale and complexity of modern network environments, coupled with the evolving sophistication of cyber threats, have rendered traditional Intrusion Detection Systems (IDS) inadequate for real-time and large-scale protection. This paper presents a comprehensive review and design strategy for a unified, real-time IDS and mitigation framework leveraging Apache Spark. This paper proposes a unified real-time IDS framework that utilizes Apache Spark to address the aforementioned disparity. The design combines threat intelligence, distributed machine learning, and streaming data analytics to facilitate automated mitigation and scalable multi-vector threat detection. We have identified critical limitations (e.g., offline detection, limited attack scope, outdated datasets) and have developed a set of objectives to address them through a review of current Spark-based IDS research. The outcome is a definitive roadmap for a next-generation IDS that offers low-latency, adaptive, and transparent defense in high-throughput network environments.

DOI: 10.33899/rjcsm.v19i2.60320, ©Authors, 2025, College of Computer Science and Mathematics, University of Mosul, Iraq.

1. Introduction

The origins of intrusion detection can be traced to the seminal model proposed by Dorothy E. Denning in 1987, which introduced the formal comparison of audit records with user behavior profiles to identify anomalies[2]. Throughout the 1990s, intrusion detection systems were primarily host-centered and computationally lightweight. The release of Snort 1.0 in 1998 marked a pivotal shift, enabling community-driven signature sharing and a move toward network-centric inspection. Recent real-world studies show that open-source tools like Snort, Suricata, and Zeek are still the most popular choices. Each has its own pros and cons when it comes to detection coverage and performance overhead.[3]. The limits of signature-based techniques, on the other hand, were obvious during large worm outbreaks such as Code Red and SQL Slammer, which were able to circumvent static signature constraints. The introduction of Cyber Threat Intelligence (CTI), which is described as adversary-centered, evidence-based knowledge that enhances raw Indicators of Compromise (IoCs) with contextual information such as motive, capability, and intent, was hastened as a result of these problems[4]. Snort, Suricata, and Zeek are some of the most well-known open-source IDS tools. In scenarios with a lot of traffic, comparative studies show that Suricata, which has a multithreaded architecture, often works better than Snort and Zeek. These solutions use signature-based detection methods, which work well against known threats but may not work as well against new or quickly changing ones.[5]. To address the limitations of signature-based systems, the cybersecurity community has increasingly turned to Cyber Threat Intelligence (CTI). CTI involves the collection and analysis of information about potential threats, enabling organizations to anticipate and mitigate attacks more proactively[6].

Intrusion detection systems have seen a dramatic improvement in their capabilities due to the development of big data technology[7]. Apache Spark, utilizing in-memory processing and its Structured Streaming module, offers a scalable framework for real-time analysis of extensive data, which is essential for prompt danger identification in contemporary networks.[8]. In response to different processing requirements, architectural models such as Lambda, Kappa, and Delta have been designed to suit both batch and streaming workloads. Each offers different trade-offs in terms of complexity, latency, and system performance [9].

Even with these improvements, there are still big problems to deal with. For instance, big data platforms like Spark let you analyze data in real time, but they also make things more complicated when it comes to state consistency, stability, and system interaction[10]These enhancements have not eliminated the existence of significant issues. Spark and other big data platforms allow for real-time data analysis, but they also introduce new challenges with respect to system interaction, stability, and state consistency.[11].

The need for security solutions that are scalable, flexible, and have low latency has become critical since cyber threats are evolving at a faster rate than traditional intrusion detection systems can keep up with. Current Spark-based intrusion detection research indicates severe inadequacies, despite Apache Spark's strong big-data processing platform having the potential to provide real-time intrusion detection. Unfortunately, a lot of the current methods aren't very good. They use antiquated or fake datasets, don't automatically mitigate assaults, can't explain their results, and only offer binary classification of attacks. Their deployment is hindered in modern, high-throughput network contexts due to these restrictions, which are required for immediate and intelligent replies. In order to fill these gaps, this study examines current state-of-the-art implementations of IDS that use Spark, compares and contrasts their pros and cons, and then suggests a theoretical unified framework that combines CTI with advanced machine learning to direct future studies toward better real-time cybersecurity solutions. This article provides an in-depth review of Intrusion Detection Systems (IDS) developed on Apache Spark, emphasizing the use of Cyber Threat Intelligence (CTI) to improve detection efficacy. We highlight the strengths, weaknesses, and common design patterns of current approaches by carefully looking at recent studies.

Based on these findings, we present a conceptual unified IDS architecture that overcomes the stated deficiencies, including the absence of real-time streaming, insufficient attack diversity, obsolete datasets, and the lack of automated mitigation. This study makes three important contributions:

- Literature Synthesis – providing an up-to-date overview of Spark-based IDS research and CTI integration trends.
- Gap Identification – highlighting persistent shortcomings in scalability, attack coverage, automation, and explainability.
- Framework Recommendation – presenting a conceptual architecture that can guide future research toward more adaptive, transparent, and operationally viable IDS solutions.

2. Security Challenges of Big Data

Big data describes datasets that are excessively large, intricate, and swiftly produced, rendering conventional data processing technologies inadequate for efficient management [12]. The term "big data" has emerged as a result of the exponential development of digital data over the past two decades[13]. This term involved the complexity, velocity, and variety of the vast quantities of data that have been generated across a variety of domains[14]. It is frequently distinguished by the "5 Vs": volume, velocity, variety, veracity, and value[1]. Big data infrastructure needs distributed systems and parallel computing architectures that work across clusters of cheap hardware to process and analyze this kind of data [15]. The key components of a big data platform comprise data intake tools, such as Apache Flume[16] and Kafka [17], distributed storage systems, such as Amazon S3[18] and Hadoop Distributed File System (HDFS)[19], and frameworks for distributed processing, like Apache Hadoop[20] and Apache Spark[21]. Together, these systems can take in, store, and handle huge amounts of data while still being fault-tolerant and available [22], [23]. Big data has many benefits. Businesses can get useful information from large and varied datasets, improve decision-making based on predictive analytics, streamline operations, give customers more personalized experiences, and find trends or outliers in real time [24]. Big data, for instance, makes predictive diagnosis and patient monitoring possible in the healthcare industry by utilizing real-time sensor data and extensive electronic health records [25]. By evaluating streaming transactional data, it makes it easier to detect fraudulent activity and engage in algorithmic trading in the financial sector[26]. Big data is utilized by governments for the purposes of urban planning, traffic optimization, and public safety. Retailers, on the other hand, engage in consumer behavior analysis in order to enhance inventory management and marketing techniques [27][28].

Big data poses substantial security and privacy challenges, despite its transformative potential. The attack surface is exacerbated by the inherent nature of distributed architecture, which involves the partitioning of data across multiple nodes and the concurrent access of data by a variety of applications and users [29]. In big data ecosystems, frequent vulnerabilities include unauthorized data access, insecure APIs, and the absence of standardized authentication mechanisms [30]. Additionally, the integration of numerous open-source tools is a common practice in big data environments. However, a significant number of these tools lack built-in security features or enforce feeble encryption standards. If protocols such as SSL/TLS or disk-level encryption are not implemented, data in transit and at rest is vulnerable to intrusions [31]. Data provenance and integrity are additional significant concerns. The continuous ingestion of data from a variety of sources makes it challenging to verify accuracy and ensure that it has not been tampered with[32]. Privacy is also a very important problem. Collecting sensitive personal data for big data analytics is common. If organizations don't have the right tools for anonymization or consent, they could be breaking privacy rules like the GDPR [33]or HIPAA[34]. Furthermore, typical security methods such as firewalls and perimeter-based access control are unsuitable in big data environments that are cloud-based and dynamic, where data is stored and processed across multiple sites that are geographically separated[35]. To address these problems, it requires both technical solutions, like encryption, access controls, and audit logs, and policy models that balance new ideas with good data management.

3. Strategy to Enhance Security in Big Data

Traditional data processing systems, including relational databases and single-node processing engines, are inadequately equipped to manage datasets that grow to terabytes and petabytes while simultaneously guaranteeing performance and fault tolerance[36]. To overcome these challenges, big data technologies were created, which allow parallel processing across clusters of machines through the use of distributed computing frameworks [37]. Apache Spark has become one of the most popular and powerful platforms in the big data ecosystem thanks to its fast in-memory computing and ability to work with a wide range of processing paradigms, such as machine learning, graph analytics, batch processing, and real-time streaming[38]. It is a general-purpose distributed computing engine designed for large-scale data processing. It was developed at UC Berkeley's AMPLab and later became an Apache top-level project in 2014. Spark distinguishes itself from earlier platforms like Hadoop MapReduce by emphasizing in-memory processing, which significantly reduces the latency associated with reading and writing intermediate results to disk. At the heart of Spark is the concept of the Resilient Distributed Dataset (RDD), an immutable distributed collection of objects partitioned across the nodes of a cluster. RDDs support two types of operations: transformations, which lazily define a new dataset based on an existing one (such as map, filter, or reduceByKey), and actions, which trigger the actual execution and return results to the driver program or write them to storage[39].

Spark applications follow a master–worker architecture. The driver program, which is the main process of a Spark application, converts user-defined code into a Directed Acyclic Graph (DAG) representing the execution plan. This DAG is then divided into stages, and tasks within these stages are dispatched to executor processes running on worker nodes. Executors are in control of running tasks and keeping memory and storage for cached or durable RDDs in order [40]. Spark is able to integrate without any problems into a wide variety of infrastructure setups because it supports several cluster managers. These cluster managers include its own built-in standalone manager [41], Apache Mesos[42], Hadoop YARN[43], and Kubernetes[44]. Spark implemented two significant optimization layers, Catalyst and Tungsten, to enhance performance further. Catalyst is a robust query optimizer utilized predominantly in Spark SQL and DataFrame APIs. It employs rule-based and cost-based optimizations, including predicate pushdown, constant folding, and join reordering, to formulate efficient physical execution plans [45]. In contrast, tungsten introduces strategies for optimizing memory and CPU performance, including as off-heap memory management, bytecode-level optimizations, and whole-stage code creation[46]. These parts work together to let Spark run complicated workloads with performance that is almost as good as native code, while still offering high-level APIs in Scala, Python, Java, and R[47]. Apache Spark comes with a number of built-in libraries that extend its capability to meet a variety of data processing requirements. These libraries are in addition to the core engine that Spark uses. The processing of structured data is made possible by Spark SQL through the utilization of both SQL queries and DataFrame/Dataset APIs[48]. MLlib provides a machine learning library that is scalable and contains methods for classification, regression, clustering, and recommendation systems[49]. GraphX offers application programming interfaces (APIs) for graph-parallel computation as well as algorithms such as PageRank and linked factors[50]. Structured Streaming is an extension of Spark SQL that allows developers to define continuous processing logic with the same semantics as batch queries. This extension helps developers create streaming applications. Because of its unified architecture, Spark is extremely well-suited for the construction of end-to-end data pipelines, which include everything from the ingestion and purification of data to the training and deployment of models [51].

Apache Spark's features have been used in many different areas. In banking, it helps find fraud and assess risk in real time. In e-commerce, it runs recommendation engines and analyzes customer behavior. In healthcare, it speeds up genomic sequencing and medical imaging analytics[52]. Scientific organizations utilize Spark to analyze astronomical data and climate models, capitalizing on its distributed architecture to exceed the limitations of single-node systems. Its connection with cloud platforms such AWS, Azure, and GCP, as well as storage systems such as HDFS, S3, and Cassandra, enhances its versatility[53]. Hence, Apache Spark is an important part of the big data environment because it gives people a quick, scalable, and adaptable way to work with huge datasets in many different fields and situations. Its design improvements—such as RDDs, DAG scheduling, in-memory computing, and unified APIs not only make it better than previous systems, but they also lay the groundwork for further progress in AI and data analytics in the future.[54].

4. Apache Spark's in scalable intrusion detection systems (IDS)

Apache Spark has grown to be an effective tool in the last few years for analyzing a lot of network data to identify security issues. Spark allows the fast process and analyze massive amounts of network traffic, sometimes in less than a second. This makes it valuable for finding threats in real time[55]. Initial studies showed that Spark Streaming could process massive amounts of network packets and significantly shorten attack detection times when compared to legacy systems[56]. A number of sophisticated machines learning models, including ensemble methods, hybrid approaches that incorporate multiple techniques, and deep learning models (such as LSTM and CNN networks) have replaced more simplistic ones as the area has progressed[57]. The amount and variety of data used in research has also grown from small, simple datasets to millions of network records from real or simulated environments. In follow the key studies that have been achieved for utilized Apache Spark with IDS. Gumaste et al. (2020)[58], proposed a real-time Spark Streaming pipeline for DDoS detection in an OpenStack-based private cloud. The system captures mirrored virtual network traffic using a packet sniffer and classifies it using a distributed 100-tree Random Forest model implemented in Spark MLlib. Their experiments, conducted on traffic generated during simulated ICMP flooding over a ~4000-second interval, produced a private dataset of moderate size (<1M records). The Random Forest model achieved 94.4% detection accuracy on the real-time dataset and 99.2% on the benchmark KDD Cup dataset, outperforming Decision Tree and Logistic Regression in both accuracy and false positive rate. Although the study demonstrates improved detection and training time with increased Spark cluster nodes, it does not report an exact latency figure such as 430 ms. The system logs suspicious IP addresses for administrator review but does not implement automated mitigation (e.g., iptables blocking). Additionally, it lacks runtime profiling for JVM or Spark resource performance. The authors conclude that Random Forest offers superior classifier accuracy in their Spark-based detection pipeline. Haggag et al. (2020)[59], introduced "DLS-IDS," a distributed deep learning intrusion detection system built on Apache Spark and trained using the NSL-KDD dataset. The authors implemented three neural models—Multilayer Perceptron (MLP), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM)—within Spark's distributed framework and compared their performance to non-distributed (single-node CPU) versions. Among these, the LSTM model achieved the highest test accuracy of 82.2%, indicating modest detection performance. Spark integration significantly improved training time compared to the standalone setup. However, all experiments were conducted in offline batch mode, without real-time streaming, latency analysis, or mitigation capabilities. The study also acknowledges that NSL-KDD is an outdated and limited dataset, lacking representation of modern traffic and attacks. Thus, while DLS-IDS demonstrates Spark's feasibility for distributed deep learning in IDS applications, its practical effectiveness is constrained by dataset limitations and absence of real-time design.

In 2020, Morfino & Rampone[60], developed a near-real-time intrusion detection system for IoT denial-of-service attacks utilizing Apache Spark. Leveraging the SYNDOS-2M synthetic dataset containing two million samples, they trained and evaluated several Spark MLlib classifiers—including logistic regression, decision tree, random forest, and others on both binary and multiclass detection tasks. All models achieved over 99% accuracy, with random forest attaining 100% accuracy on the synthetic test set. Training and detection were efficient; for instance, the decision tree model trained in approximately 23.2 seconds, and the detection time for 20,000 records was just 0.13 seconds, indicating low system latency. Their Spark Streaming pipeline processed live data, though the system did not include automated mitigation or response mechanisms. The primary limitation, as noted by the authors, is the reliance on synthetic data (SYNDOS-2M), which may not fully capture the characteristics of real-world IoT network traffic.

In 2022, Hagar & Gawali[61], compared three approaches for network intrusion detection on the CSE-CIC-IDS2018 dataset: a traditional machine learning pipeline using Apache Spark's MLlib, and two deep learning models (CNN and LSTM). The authors first applied random forest–based feature selection within Spark to reduce the dataset to 19 features. Both CNN and LSTM were trained as multi-class classifiers, but the Spark-MLlib model achieved the highest overall performance, with F1-scores of approximately 1.0 across all 15 traffic classes. Additionally, the Spark model demonstrated the fastest training and evaluation times (approximately 7.56 minutes and 39 seconds, respectively), outperforming both deep learning models. The study was conducted as an offline batch analysis; no real-time streaming or mitigation mechanisms were implemented. The authors also acknowledged important limitations, including the use of extensive over- and undersampling (which increases the risk of overfitting) and evaluation restricted to a single dataset, thereby limiting generalizability. In addition. Azeroual (2022)[62], evaluated the scalability and performance modeling capabilities of Apache Spark MLlib in a multi-node cluster environment, profiling the behavior of K-Means clustering, Random Forest regression, and Word2Vec on large synthetic datasets. The study reported that MLlib-based predictive models could achieve up to 98% accuracy in forecasting their own execution performance, resulting in reductions of CPU usage by approximately 30% and processing time by about 25%. Importantly, this research focused solely on system-level performance prediction and resource optimization, rather than on security or intrusion detection efficacy. No specific IDS or threat detection models were implemented, and no security metrics or mitigation actions were evaluated; all experiments were conducted offline and unrelated to attack detection quality.

In 2023, Chliah et al. [63], developed a hybrid anomaly detection system using Apache Spark, which combines unsupervised and supervised machine learning techniques. Their approach first applies K-means clustering to NetFlow network data collected from Ibn Zohr University, then uses a K-nearest neighbors (KNN) classifier within each cluster to detect anomalies. Evaluated with k-fold cross-validation on the entire dataset, their hybrid model achieved an overall accuracy of 99.94%. All analysis was conducted offline using batch machine learning, with no real-time latency measurements or automated mitigation mechanisms implemented. As noted by the authors, the use of a single private dataset and the extremely high accuracy suggest possible overfitting or data leakage, and generalizability to broader or more diverse network environments remain unproven.

In 2024, Talukder et al. (2024)[64], present a PySpark-based intrusion detection system evaluated on the UNSW-NB15 dataset using both binary and multilabel classification tasks. For binary classification, their Random Forest (RF) model using the proposed feature set achieved the highest accuracy of 99.59%, closely followed by Extra Trees (ET) at 99.59% and Decision Tree (DT) at 98.97%. XGBoost (XGB) achieved 98.81%. Precision, recall, and F1-scores for RF and ET exceeded 99%. In multilabel classification, both RF and ET reached an accuracy of 99.95% with the proposed feature set, while DT scored 99.79% and XGB 95.04%. The results indicate that, with careful feature engineering, ensemble models—particularly Random Forest and Extra Trees—can deliver near-perfect classification on this benchmark in a distributed Spark environment. All experiments were conducted in offline batch mode, with no real-time mitigation or streaming evaluation reported. Alrefaei & Ilyas (2024)[65], present a real-time intrusion detection system (IDS) for IoT networks utilizing the PySpark framework and multiclass machine learning classification on the IoT-23 dataset. Their approach integrates data cleaning, normalization, feature selection (SelectKBest, SelectFromModel with XGB/RF), and SMOTE oversampling to address class imbalance. Using the One-vs-Rest (OVR) multiclass scheme, they compare Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), K-Nearest Neighbors (KNN), and Extreme Gradient Boosting (XGB). Among these, XGBoost achieves the highest overall accuracy at 98.89%, while Random Forest offers the fastest prediction time (0.0311 s). All top models exceed 97% accuracy; for example, RF reaches 98.54% and KNN 98.87%. Detailed results are reported per-class and per-model, with precision, recall, and F1-scores exceeding 95% in most categories. The system operates in real-time with Spark Streaming but does not implement automated mitigation or blocking. The primary limitations are the use of a single, reduced IoT-23 dataset, reliance on synthetic balancing (SMOTE), and the absence of practical deployment or response actions. The authors conclude that ensemble models in PySpark are highly effective for accurate and efficient real-time IoT attack detection, but further validation on more diverse datasets and operational scenarios is needed. Also in same year, Mamdouh et al. [66], developed a real-time intrusion detection system for IoT networks using PySpark, leveraging multiclass classification on the IoT-23 dataset. Their approach incorporated Spark Streaming, MLlib, SMOTE oversampling, and feature selection, and utilized a One-vs-Rest scheme to compare algorithms including Decision Tree, Random Forest, Logistic Regression, and XGBoost. Among these, XGBoost achieved the highest classification accuracy at 98.89%, while Random Forest provided the fastest prediction time at approximately 0.031 seconds. All evaluated models attained overall accuracy above 98%, and the system demonstrated low detection latency, suitable for real-time streaming applications. However, the framework did not implement automated mitigation or blocking, and the authors note key limitations such as the use of a single IoT dataset and synthetic balancing techniques; no practical deployment or response actions were demonstrated. In addition, Alslman et al. (2024)[67], propose a distributed DDoS attack detection system utilizing a stacked ensemble of Random Forest and XGBoost models, trained and evaluated on the CIC-DDoS2019 dataset. The system employs Apache Spark for parallelized data processing and training, enabling scalability to millions of records and significant reduction in model training time. The ensemble model achieves 99.94% accuracy across four classes (SYN, UDP, MSSQL, and benign traffic), with Spark reducing training time for Random Forest from 32 to 14 minutes and for XGBoost from 87 to 46 minutes roughly halving execution time compared to non-distributed training. No streaming, real-time mitigation, or response mechanisms are implemented; all evaluation is conducted offline. Key limitations include the exclusive focus on DDoS attacks (no multi-vector, non-DDoS, or mixed traffic types) and reliance on a single, large, labeled dataset, with no validation in live operational settings. The authors conclude that Spark-based stacked ensembles are highly efficient for high-volume DDoS detection, but further work is needed for broader applicability and online deployment. **Table 1** compares recent studies that we discussed with describe its model structure, dataset used point of strength and limitations. This summary highlights prevalent deficiencies in the literature, including the absence of genuine real-time processing and the lack of automated mitigation functionalities.

5. Analysis of Recent studies

Although Apache Spark has enabled substantial advances in scalable intrusion detection systems (IDS), a review of recent literature reveals that significant limitations persist in the practical deployment and effectiveness of Spark-based IDS frameworks. Most current solutions are limited to offline or batch processing and do not support true real-time detection on streaming network data (Haggag et al., 2020; Hagar & Gawali, 2022; Talukder et al., 2024). As a result, these systems may fail to detect or

Table 1. A comparison of recent IDS solutions based on Apache Spark, focusing on the model structure, dataset, best performance, and main limitation of each study.

Authors [Ref]	**Model Structure**	**Dataset / Size**	**Best Metric**	**Train / Detect Time**	**Mitigation**	**Key Limitation**
Gumaste et al. [58]	Spark Streaming, 100-tree RF	Private NetFlow <1M	94.4% ACC (real); 99.2% (KDD)	Not stated (sub-second)	Log only	ICMP flood; offline; no mitigation/telemetry
Haggag et al. [59]	Spark MLP, RNN, LSTM	NSL-KDD ~125k	82.2% ACC (LSTM)	Offline batch	—	Old/small data; no real-time/streaming
Morfino & Rampone[60]	Spark MLlib (RF, DT, LR); Streaming	SYNDOS-2M (synthetic)	100% ACC (RF, synthetic)	Train 23s, detect 0.13s	—	Synthetic IoT data; no blocking; realism
Hagar & Gawali[61]	Spark RF feat. select → CNN, LSTM	CSE-CIC-IDS2018 ~2M	F1 ≈ 1.0 (RF)	Train 7.56 min, eval 39s	—	Resampling; offline; no mitigation
Azeroual [62]	Spark MLlib scalability (KMeans, RF, Word2Vec)	Big Data (>GB, synthetic)	≤98% ACC (job perf)	—	—	Not IDS; job/resource profiling only
Chliah et al. [63]	Spark KMeans → KNN hybrid	Private NetFlow (unknown)	99.94% ACC	Offline CV	—	Private data; possible overfit
Talukder et al. [64]	PySpark ensemble (XGB, RF, ET, DT)	UNSW-NB15, CIC-IDS2017/18	99.59% ACC (RF, binary)	Offline	—	Batch only; no real-time, KDD/NSL-KDD not used
Alrefaei & Ilyas [65]	PySpark Streaming + OvR (DT, RF, LR, XGB)	IoT-23 ~800k	98.89% ACC (XGB)	RF pred. 0.031s	—	Single IoT set; SMOTE; no mitigation
Mamdouh et al. [66]	Spark Random Forest	CIC-IDS2018 (large)	>99% ACC	Faster than baseline	—	Offline only; few details
Alslman et al. [67],	Spark ensemble (RF, XGB stack)	CIC-DDoS2019 ~2.9M	99.94% ACC (stacked)	Spark halves train time	—	DDoS only; offline; no mitigation

respond promptly to fast-evolving threats in dynamic environments. Integrated, end-to-end pipelines that unify data ingestion, real-time feature processing, model inference, and periodic model updating within Spark remain rare (Morfino & Rampone, 2020; Chliah et al., 2023).

Moreover, many Spark-based IDS implementations focus on a narrow subset of attack types or treat intrusion detection as a binary classification problem, limiting their ability to recognize diverse and complex multi-class or multi-vector threats (Morfino & Rampone, 2020; Alslman et al., 2024). The widespread use of outdated or synthetic datasets (e.g., NSL-KDD, SYNDOS-2M) further restricts the generalizability and robustness of these systems when confronted with modern, real-world traffic patterns (Haggag et al., 2020; Chliah et al., 2023). Issues such as class imbalance and absence of operational validation persist, even when newer datasets like UNSW-NB15 or CIC-IDS2018 are employed (Hagar & Gawali, 2022; Talukder et al., 2024). Furthermore, the majority of prototypes are restricted to detection and alerting, with the absence of automated response mechanisms—such as real-time blockage or reconfiguration of network policies—almost universally observed (Gumaste et al., 2020; Alrefaei & Ilyas, 2024). Other important aspects, including explainability of detection results, model fairness, and monitoring of Spark cluster resource utilization, have received little attention (Azeroual, 2022; Chliah et al., 2023). Therefore, there is a pressing need for a Spark-based IDS solution that goes beyond mere scalability and detection accuracy to provide integrated, real-time intrusion detection and automated mitigation across diverse attack classes, while addressing explainability, fairness, and resource monitoring in real production environments.

Based on the literature [58]- [67]), several key research gaps are identified in the domain of Spark-based IDS:

- **Lack of unified real-time processing pipelines:** There is no comprehensive end-to-end system that seamlessly combines data streaming, feature processing, model training, and live classification within a single Spark-based IDS. Most current implementations rely on offline or batch analysis of stored data, meaning they cannot operate in true real-time or adapt continuously to evolving threats. A unified framework that supports continuous learning and real-time detection remains absent ([58], [59], [60], [61], [63], [64]).
- **Inadequate multi-class and multi-vector**

detection at scale: Many existing solutions are designed to detect only a single attack type or treat intrusion detection as a simple binary (attack vs. normal) problem. There is limited capability for robust multi-class or multi-vector detection, especially in large-scale environments. Systems that perform well on specific attacks (e.g., SYN flood or DDoS) often fail to generalize to other threats. Achieving consistently high accuracy across diverse attack vectors in big data streams remains an unresolved challenge ([59], [60], [61], [63], [64], [65], [67]).

- **Absence of integrated mitigation and control:** Spark-based IDS research typically produces passive detectors that only log or alert on detected threats. Few, if any, systems integrate with network enforcement mechanisms such as firewalls (e.g., iptables), SDN controllers, or automated response modules that could block or mitigate malicious traffic in real time. Even state-of-the-art streaming classifiers usually stop at detection, without triggering any defensive actions ([58], [59], [60], [65], [67]).
- **Poor telemetry and resource monitoring:** There is a lack of focus on operational monitoring of IDS pipelines built on Spark. Most studies do not report on the performance or resource utilization (CPU, memory, network bandwidth) of their systems. Without unified dashboards, feedback loops, or real-time monitoring tools, it is difficult to diagnose bottlenecks, tune performance, or ensure reliability under production loads ([62]).
- **Evaluation on outdated, synthetic, or imbalanced datasets:** Many published studies validate their IDS solutions using datasets that are either synthetic or do not reflect the complexity of current network environments. The continued reliance on datasets like NSL-KDD (from the late 1990s) and artificial data (e.g., SYNDOS-2M) raises questions about the generalizability of these models. When more recent datasets are used, issues like class imbalance and insufficient diversity are often not fully addressed, leading to potential overfitting and limited real-world applicability ([59], [60], [61], [63], [64], [65], [67]).

Hence, even with recent developments, existing Apache Spark-based IDS solutions continue having limitations in actual implementation. Most systems function on offline or batch data and are incapable of executing genuine real-time detection on streaming network traffic. They often concentrate on a limited spectrum of attack vectors, frequently perceiving intrusion as a mere binary issue, and commonly depend on obsolete or artificial datasets, so compromising their efficacy on contemporary traffic. Significantly, nearly all of these systems lack automated mitigation and explainability; they identify and notify about dangers but do not engage in active response, nor do they elucidate the rationale for detection. This gap renders extensive, high-velocity network environments susceptible to rapidly developing threats. A unified IDS framework is essential that integrates Spark's large data processing with sophisticated multi-vector detection and proactive response. The proposed effort aims to enhance real-time cyber defense for modern networks by integrating scalability, threat diversity, and automation under a single platform.

4. The Suggestion Strategy to Overcome the Limitation of Recent Works

To address the above gaps, the following research questions (RQs) are proposed for investigation:

- How can we design a unified, real-time intrusion detection pipeline on Apache Spark that supports streaming data and ensures low-latency, scalable processing of network traffic? This question focuses on the architectural and system design needed to combine Spark Streaming, distributed feature extraction, and model inference/training into a seamless pipeline capable of handling high throughput data with minimal delay.
- What machine learning and deep learning techniques can enable accurate detection of multiple attack classes and vectors at scale in a Spark-based IDS, and how can we improve the model's generalization to new or imbalanced datasets? This question examines the algorithms and data aspects: it seeks to find suitable models (or ensembles of models) that can classify a broad range of intrusions in a multi-class setting, and explores methods (like feature selection, retraining with new data, or fairness-aware learning) to maintain performance across evolving, large-scale datasets.
- How can intrusion alerts from a Spark-based IDS be automatically translated into mitigation actions in real-world networks? In particular, what framework can integrate the IDS with control mechanisms (such as SDN controllers or firewall rules) to proactively contain, or block attacks once detected? This question addresses the bridge from detection to response, asking how to embed or attach the IDS into a larger security orchestration so that it not only flags threats but also triggers timely defensive measures.
- How can we incorporate explainability and monitoring into a large-scale Spark IDS to enhance its transparency and reliability? This question probes the addition of supporting features to the IDS: it aims to determine how we can provide

explanations for detection decisions (e.g., indicating which features or rules fired for a given alert) and how to continuously monitor the IDS's own performance (resource usage, processing delays, etc.) in order to ensure the system remains effective and does not introduce new blind spots or failure points.

Corresponding to the research questions, the objectives of this work are defined as follows:

- Unified Streaming Pipeline – Design and implement a unified IDS pipeline using Apache Spark that combines real-time stream processing with batch learning. The objective is to create an architecture capable of ingesting live network data (potentially via Kafka or similar), extracting features on the fly, and applying machine learning models in real-time, while also periodically updating or retraining those models with new data – all within the Spark ecosystem. This will directly tackle the need for a scalable, low-latency IDS solution.
- Scalable Multi-Attack Detection – Develop advanced detection algorithms and models suited for Spark's distributed environment that can classify multiple types of attacks with high accuracy. This involves exploring and evaluating techniques such as deep neural networks, ensemble methods, and hybrid approaches (e.g., combining anomaly detection with misuse detection) to improve detection rates. A key part of this objective is to ensure generalization: the models should be trained and tested on modern, diverse datasets (with appropriate handling of class imbalance) so that the IDS is effective against a wide array of known attack vectors and can adapt to new or emerging threats.
- Integration with Mitigation and Monitoring – Integrate the intrusion detection pipeline with real-world mitigation mechanisms and implement system telemetry. Concretely, this objective will create a link between the Spark IDS and network control interfaces: for example, developing a module that communicates with an SDN controller to dynamically reroute or block malicious traffic, or that updates firewall rules (iptables) when an attack is confirmed. In tandem, the system will include monitoring tools to track resource usage and performance metrics of the Spark cluster (such as throughput, latency per batch, CPU/memory utilization of executors). This ensures the IDS can not only detect but also respond to attacks in an automated fashion, and that it operates reliably under production conditions.
- Explainability and Fairness – Incorporate explainable AI techniques and fairness measures into the IDS. The goal is to augment the Spark-based IDS with components that can interpret the model's decisions (for example, generating human-readable explanations for why an alert was triggered, perhaps via feature importance or rule extraction from complex models). Additionally, the training process will include fairness considerations to avoid biased detection (ensuring, for instance, that detection performance is consistent across different attack categories or network segments). This objective will improve user trust in the IDS by making it more transparent and ensuring it behaves equitably on heterogeneous data.

Based on the previous key point we suggest an IDS system to overcome the limitations as illustrated in **Figure 1**.

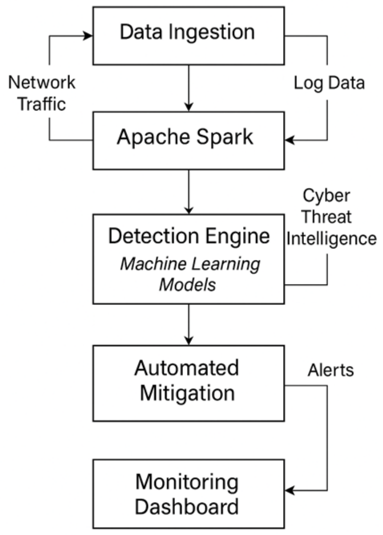

Figure 1. The suggested unified IDS framework

Figure 1. The suggested unified IDS framework architecture shows how data flows from the first step of data ingestion to threat detection and mitigation. Apache Spark's distributed processing lets the system analyze live network traffic and log streams in real time. Cyber Threat Intelligence feeds are used to add known threat indications to the detection engine. When the machine learning models find an intrusion, they trigger an automated mitigation module to stop or confine the danger. At the same time, a monitoring dashboard logs events and lets security staff know about them.

Conclusion

This review has methodically analyzed recent studies on Apache Spark-based Intrusion Detection Systems (IDS), using Cyber Threat Intelligence (CTI) and machine learning methodologies. Through an analysis of the strengths and limitations of current solutions, we discovered persistent limitations such as dependence on offline processing, limited attack coverage, outdated datasets, lack of automated mitigation, and lack of explainable results. Based on these discoveries, we outlined the conceptual framework of an integrated real-time Intrusion Detection System and mitigation strategy that tackles these challenges through the amalgamation of streaming analytics, multi-vector detection, and automated response. This conceptual architecture serves as a framework for future researchers to develop more adaptive, transparent, and scalable IDS solutions, thus enhancing the state of the art in Spark-based cybersecurity research.

Based on the gaps identified through the literature we reviewed, future researchers should cover and take in consideration when developing Spark-based IDS solutions the following aspects:

- Using Apache Spark Structured Streaming to analyze live traffic without latency is true real-time streaming integration.
- A more comprehensive multi-class attack detection system that goes beyond just normal/abnormal classification to encompass a wider range of intrusion types that are always changing.
- Integration with Software-Defined Networking (SDN) to make it possible to take quick action against threats, including rerouting or blocking traffic.
- Adding modern datasets and actual traffic traces (such CIC-IDS2018 and UNSW-NB15) to make sure the model is strong and useful.
- Adding explainable AI methods to make detection decisions more trustworthy and clearer.
- Automating the mitigation process so that systems can not only find dangers but also stop them before they happen.
- By implementing these suggestions, future work can fix the problems with current systems and help build real-time IDS frameworks that are more effective, scalable, and usable in the long run.

Acknowledgement

The authors are grateful to the College of Computer Science and Mathematics at the University of Mosul for their assistance in the conduct of this research.

Conflict of interest

None.

References

[1] Y. Demchenko, C. Ngo, C. De Laat, P. Membrey, and D. Gordijenko, “Big security for big data: Addressing security challenges for the big data infrastructure,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8425 LNCS, pp. 76–94, 2014, doi: 10.1007/978-3-319-06811-4_13.

[2] T. Bass, “Intrusion detection systems and multisensor data fusion,” *Commun ACM*, vol. 43, no. 4, pp. 99–105, 2000, doi: 10.1145/332051.332079.

[3] A. Gupta and L. Sen Sharma, “Performance Evaluation of Snort and Suricata Intrusion Detection Systems on Ubuntu Server,” *Lecture Notes in Electrical Engineering*, vol. 597, pp. 811–821, 2020, doi: 10.1007/978-3-030-29407-6_58.

[4] A. STĂNCIULESCU, C.-A. COPACI, and I. C. BACIVAROV, “Cyber Threats and Exploring the Sources of Cyber Threat Intelligence,” *Proceedings of the International Conference on Cybersecurity and Cybercrime (IC3)*, pp. 83–89, Nov. 2024, doi: 10.19107/CYBERCON.2024.11.

[5] A. Waleed, A. F. Jamali, and A. Masood, “Which open-source IDS? Snort, Suricata or Zeek,” *Computer Networks*, vol. 213, p. 109116, Aug. 2022, doi: 10.1016/J.COMNET.2022.109116.

[6] P. Alaeifar, S. Pal, Z. Jadidi, M. Hussain, and E. Foo, “Current approaches and future directions for Cyber Threat Intelligence sharing: A survey,” *Journal of Information Security and Applications*, vol. 83, p. 103786, Jun. 2024, doi: 10.1016/J.JISA.2024.103786.

[7] R. M .P . , P. V. B. Reddy, J. T. Thirukrishna, and C. Vidyadhari, “Intrusion detection in big data using hybrid feature fusion and optimization enabled deep learning based on spark architecture,” *Comput Secur*, vol. 116, p. 102668, May 2022, doi: 10.1016/J.COSE.2022.102668.

[8] T.-T.-T. Do, Q.-T. Huynh, K. Kim, and V.-Q. Nguyen, “A Survey on Video Big Data Analytics: Architecture, Technologies, and Open Research Challenges,” *Applied Sciences 2025, Vol. 15, Page 8089*, vol. 15, no. 14, p. 8089, Jul. 2025, doi: 10.3390/APP15148089.

[9] N. Su, S. Huang, and C. Su, “Elevating Smart Manufacturing with a Unified Predictive Maintenance Platform: The Synergy between Data Warehousing, Apache Spark, and Machine Learning,” *Sensors*, vol. 24, no. 13, Jul. 2024, doi: 10.3390/S24134237.

[10] A. C. Ikegwu, H. F. Nweke, C. V. Anikwe, U. R. Alo, and O. R. Okonkwo, “Big data analytics for data-driven industry: a review of data sources, tools, challenges, solutions, and research directions,” *Cluster Computing 2022 25:5*, vol. 25, no. 5, pp. 3343–3387, Mar. 2022, doi: 10.1007/S10586-022-03568-5.

[11] F. Ekundayo, I. Atoyeb, A. Soyele, and E. Ogunwobi, “Predictive Analytics for Cyber Threat Intelligence in Fintech Using Big Data and Machine Learning,” *International Journal of Research Publication and Reviews*, vol. 5, no. 11, pp. 5934–5948, Nov. 2024, doi: 10.55248/GENGPI.5.1124.3352.

[12] Y. Kumar, J. Marchena, A. H. Awlla, J. J. Li, and H. B. Abdalla, “The AI-Powered Evolution of Big Data,” *Applied Sciences 2024, Vol. 14, Page 10176*, vol. 14, no. 22, p. 10176, Nov. 2024, doi: 10.3390/APP142210176.

[13] Y. Shi, “Big Data and Big Data Analytics,” *Advances in Big Data Analytics*, pp. 3–21, 2022, doi: 10.1007/978-981-16-3607-3_1.

[14] A. De Mauro, M. Greco, and M. Grimaldi, “A formal definition of Big Data based on its essential features,” *Library Review*, vol. 65, no. 3, pp. 122–135, Apr. 2016, doi: 10.1108/LR-06-2015-0061.

[15] S. Salam Samaan and H. Awheed Jeiad, “A Review on Utilizing Big Data Techniques for Performance Improvement in Software Defined Networking,” *Iraqi Journal of Computers*, vol. 23, no. 3, 2023, doi: 10.33103/uot.ijccce.23.3.6.

[16] K. G. Srinivasa, S. G. M., and S. H., “Apache Flume,” pp. 95–107, 2018, doi: 10.1007/978-3-319-77800-6_6.

[17] J. Alwidian, S. A. Rahman, M. Gnaim, and F. Al-Taharwah, “Big Data Ingestion and Preparation Tools,” *Mod Appl Sci*, vol. 14, no. 9, p. 12, Aug. 2020, doi: 10.5539/MAS.V14N9P12.

[18] H. M. Chen, C. J. Li, and B. S. Ke, “Designing a Simple Storage Services (S3) compatible system based on ceph software-defined storage system,” *ACM International Conference Proceeding Series*, pp. 6–10, Aug. 2017, doi: 10.1145/3145511.3145519.

[19] D. Hu, D. Chen, S. Lou, and S. Pei, "Research on reliability of hadoop distributed file system," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 11, pp. 315–326, 2015, doi: 10.14257/IJMUE.2015.10.11.30.

[20] P. R. Giri and G. Sharma, "Apache Hadoop Architecture, Applications, and Hadoop Distributed File System," *Semiconductor Science and Information Devices*, vol. 4, no. 1, pp. 14–20, May 2022, doi: 10.30564/SSID.V4I1.4619.

[21] E. Shaikh, I. Mohiuddin, Y. Alufaisan, and I. Nahvi, "Apache Spark: A Big Data Processing Engine," *2019 2nd IEEE Middle East and North Africa COMMunications Conference, MENACOMM 2019*, Nov. 2019, doi: 10.1109/MENACOMM46666.2019.8988541.

[22] M. S. Hadi, A. Q. Lawey, T. E. H. El-Gorashi, and J. M. H. Elmirghani, "Big data analytics for wireless and wired network design: A survey," *Computer Networks*, vol. 132, pp. 180–199, Feb. 2018, doi: 10.1016/j.comnet.2018.01.016.

[23] P. Raj, "The Hadoop Ecosystem Technologies and Tools," *Advances in Computers*, vol. 109, pp. 279–320, Jan. 2018, doi: 10.1016/bs.adcom.2017.09.002.

[24] Tochukwu Ignatius Ijomah, Courage Idemudia, Nsisong Louis Eyo-Udo, and Kikelomo Fadilat Anjorin, "The role of big data analytics in customer relationship management: Strategies for improving customer engagement and retention," *World Journal of Advanced Science and Technology*, vol. 6, no. 1, pp. 013–024, Jul. 2024, doi: 10.53346/WJAST.2024.6.1.0038.

[25] M. S. Akter, R. Islam, M. A. R. Khan, and S. Juthi, "Big Data Analytics In Healthcare: Tools, Techniques, And Applications - A Systematic Review," *Innovatech Engineering Journal*, vol. 2, no. 01, pp. 29–47, Jan. 2025, doi: 10.70937/JNES.V2I01.51.

[26] A. Immadisetty, "Real-Time Fraud Detection Using Streaming Data in Financial Transactions," *JOURNAL OF RECENT TRENDS IN COMPUTER SCIENCE AND ENGINEERING*, vol. 13, no. 1, pp. 66–76, Feb. 2025, doi: 10.70589/JRTCSE.2025.13.1.9.

[27] J. W. Lee, "Big data strategies for government, society and policy-making," *Journal of Asian Finance, Economics and Business*, vol. 7, no. 7, pp. 475–487, Jul. 2020, doi: 10.13106/JAFEB.2020.VOL7.NO7.475.

[28] A. Karami and F. Jafari, "Leveraging big data characteristics for enhanced healthcare fraud detection," *Cluster Comput*, vol. 28, no. 6, Oct. 2025, doi: 10.1007/S10586-024-05097-9.

[29] F. M. Awaysheh, M. N. Aladwan, M. Alazab, S. Alawadi, J. C. Cabaleiro, and T. F. Pena, "Security by Design for Big Data Frameworks Over Cloud Computing," *IEEE Trans Eng Manag*, vol. 69, no. 6, pp. 3676–3693, Dec. 2022, doi: 10.1109/TEM.2020.3045661.

[30] W. S. Admass, Y. Y. Munaye, and A. A. Diro, "Cyber security: State of the art, challenges and future directions," *Cyber Security and Applications*, vol. 2, p. 100031, Jan. 2024, doi: 10.1016/J.CSA.2023.100031.

[31] I. A. Atoum and I. M. Keshta, "Big data management: Security and privacy concerns," *International Journal of Advanced and Applied Sciences*, vol. 8, no. 5, pp. 73–83, May 2021, doi: 10.21833/IJAAS.2021.05.009.

[32] R. Nimesh Kumar Dhenia and M. Parikh Independent Researcher Senior IEEE Member Richmond, "Trust-Aware Data Pipelines for Verifiable AI Accelerators: A Data-Centric Frame Work," *International Journal of Engineering Research & Technology*, vol. 14, no. 6, Jun. 2025, doi: 10.17577/IJERTV14IS060187.

[33] N. Khaled, B. Pattel, and A. Siddiqui, "Cloud and IoT technologies," *Digital Twin Development and Deployment on the Cloud*, pp. 11–20, 2020, doi: 10.1016/B978-0-12-821631-6.00002-5.

[34] O. Isaac Abiodun, M. Alawida, A. Esther Omolara, and A. Alabdulatif, "Data provenance for cloud forensic investigations, security, challenges, solutions and future perspectives: A survey," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, pp. 10217–10245, Nov. 2022, doi: 10.1016/J.JKSUCI.2022.10.018.

[35] Sandeep Batchu, "Cloud Infrastructure Fortification: Advanced Security Strategies in the Era of Emerging Threats," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 11, no. 1, pp. 1407–1414, Jan. 2025, doi: 10.32628/CSEIT251112150.

[36] F. X. Diebold, "On the Origin(s) and Development of the Term 'Big Data,'" *SSRN Electronic Journal*, Oct. 2012, doi: 10.2139/SSRN.2152421.

[37] G. S. Bhathal and A. Singh, "Big Data Computing with Distributed Computing Frameworks," *Lecture Notes in Networks and Systems*, vol. 65, pp. 467–477, 2019, doi: 10.1007/978-981-13-3765-9_49.

[38] E. Shaikh, I. Mohiuddin, Y. Alufaisan, and I. Nahvi, "Apache Spark: A Big Data Processing Engine," *2019 2nd IEEE Middle East and North Africa COMMunications Conference, MENACOMM 2019*, Nov. 2019, doi: 10.1109/MENACOMM46666.2019.8988541.

[39] M. Zaharia *et al.*, "Apache spark: A unified engine for big data processing," *Commun ACM*, vol. 59, no. 11, pp. 56–65, Nov. 2016, doi: 10.1145/2934664.

[40] K. Fan and M. Wang, "DAG-Based Formal Modeling of Spark Applications with MSVL," *Information 2023, Vol. 14, Page 658*, vol. 14, no. 12, p. 658, Dec. 2023, doi: 10.3390/INFO14120658.

[41] A. Seabra and S. Lifschitz, "Advancing Polyglot Big Data Processing using the Hadoop ecosystem," Apr. 2025, Accessed: Jul. 16, 2025. [Online]. Available: http://arxiv.org/abs/2504.14322

[42] M. Frampton, "Apache Mesos," *Complete Guide to Open Source Big Data Stack*, pp. 97–137, 2018, doi: 10.1007/978-1-4842-2149-5_4.

[43] J. C. Lin, I. C. Yu, E. B. Johnsen, and M. C. Lee, "ABS-YARN: A formal framework for modeling hadoop YARN clusters," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9633, pp. 49–65, 2016, doi: 10.1007/978-3-662-49665-7_4.

[44] V. Medel, O. Rana, J. Á. Bañares, and U. Arronategui, "Modelling performance & resource management in Kubernetes," *Proceedings - 9th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2016*, pp. 257–262, Dec. 2016, doi: 10.1145/2996890.3007869.

[45] G. M. Essertel, R. Y. Tahboub, J. M. Decker, K. J. Brown, K. Olukotun, and T. Rompf, "Flare: Native Compilation for Heterogeneous Workloads in Apache Spark".

[46] P. Beiersdorfer, J. Clementson, and U. I. Safronova, "Tungsten data for current and future uses in fusion and plasma science," *Atoms*, vol. 3, no. 2, pp. 260–272, Jun. 2015, doi: 10.3390/ATOMS3020260.

[47] R. Martha, "Efficient Data Processing with Apache Spark," Nov. 2024, doi: 10.2139/SSRN.5023456.

[48] S. Haines, "Transforming Data with Spark SQL and the DataFrame API," *Modern Data Engineering with Apache Spark*, pp. 93–115, 2022, doi: 10.1007/978-1-4842-7452-1_4.

[49] Z. Hasan, H. J. Xing, and M. Idrees Magray, "Big Data Machine Learning Using Apache Spark Mllib," *Mesopotamian Journal of Big Data*, vol. 2022, pp. 1–11, Dec. 2022, doi: 10.58496/MJBD/2022/001.

[50] A. A. Ali and D. Logofatu, "An Analysis on Graph-Processing Frameworks: Neo4j and Spark GraphX," *IFIP Adv Inf Commun Technol*, vol. 646 IFIP, pp. 461–470, 2022, doi: 10.1007/978-3-031-08333-4_37.

[51] K. El Bouchefry and R. S. de Souza, "Learning in Big Data: Introduction to Machine Learning," *Knowledge Discovery in Big Data from Astronomy and Earth Observation: Astrogeoinformatics*, pp. 225–249, Apr. 2020, doi: 10.1016/B978-0-12-819154-5.00023-0.

[52] E. Dritsas and M. Trigka, "Exploring the Intersection of Machine Learning and Big Data: A Survey," *Machine Learning and Knowledge Extraction 2025, Vol. 7, Page 13*, vol. 7, no. 1, p. 13, Feb. 2025, doi: 10.3390/MAKE7010013.

[53] A. El-Sayed, M. Abougabal, and S. Lazem, "Practical big data techniques for end-to-end machine learning deployment: a comprehensive review," *Discover Data 2025 3:1*, vol. 3, no. 1, pp. 1–18, Apr. 2025, doi: 10.1007/S44248-025-00029-3.

[54] P. Sewal and H. Singh, "A Critical Analysis of Apache Hadoop and Spark for Big Data Processing," *Proceedings of IEEE International Conference on Signal Processing,Computing and Control*, vol. 2021-October, pp. 308–313, 2021, doi: 10.1109/ISPCC53510.2021.9609518.

[55] O. Azeroual and A. Nikiforova, "Apache Spark and MLlib-Based Intrusion Detection System or How the Big Data Technologies Can Secure the Data," *Information (Switzerland)*, vol. 13, no. 2, Feb. 2022, doi: 10.3390/INFO13020058.

[56] "Enhancing Cyber Threat Detection through Real-time Threat Intelligence and Adaptive Defense Mechanisms," *International Journal of Computer Applications Technology and Research*, Jul. 2024, doi: 10.7753/IJCATR1308.1002.

[57] A. A. Hagar and B. W. Gawali, "Apache Spark and Deep Learning Models for High-Performance Network Intrusion Detection Using CSE-CIC-IDS2018," *Comput Intell Neurosci*, vol. 2022, p. 3131153, 2022, doi: 10.1155/2022/3131153.

[58] S. Gumaste, D. G. Narayan, S. Shinde, and K. Amit, "Detection of DDoS Attacks in OpenStack-based Private Cloud Using Apache Spark," *Journal of Telecommunications and Information Technology*, vol. 2020, no. 4, pp. 62–71, Dec. 2020, doi: 10.26636/JTIT.2020.146120.

[59] M. Haggag, M. M. Tantawy, and M. M. S. El-Soudani, "Implementing a deep learning model for intrusion detection on apache spark platform," *IEEE Access*, vol. 8, pp. 163660–163672, 2020, doi: 10.1109/ACCESS.2020.3019931.

[60] V. Morfino and S. Rampone, "Towards Near-Real-Time Intrusion Detection for IoT Devices using Supervised Learning and Apache Spark," *Electronics 2020, Vol. 9, Page 444*, vol. 9, no. 3, p. 444, Mar. 2020, doi: 10.3390/ELECTRONICS9030444.

[61] A. A. Hagar and B. W. Gawali, "Apache Spark and Deep Learning Models for High-Performance Network Intrusion Detection Using CSE-CIC-IDS2018," *Comput Intell Neurosci*, vol. 2022, p. 3131153, 2022, doi: 10.1155/2022/3131153.

[62] O. Azeroual and A. Nikiforova, "Apache Spark and MLlib-Based Intrusion Detection System or How the Big Data Technologies Can Secure the Data," *Information 2022, Vol. 13, Page 58*, vol. 13, no. 2, p. 58, Jan. 2022, doi: 10.3390/INFO13020058.

[63] H. Chliah, A. Battou, M. A. el hadj, and A. Laoufi, "Hybrid Machine Learning-Based Approach for Anomaly Detection using Apache Spark," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 4, pp. 870–878, 2023, doi: 10.14569/IJACSA.2023.0140496.

[64] Md. A. Talukder *et al.*, "Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction," Jan. 2024, Accessed: May 04, 2025. [Online]. Available: http://arxiv.org/abs/2401.12262

[65] A. Alrefaei and M. Ilyas, "Using Machine Learning Multiclass Classification Technique to Detect IoT Attacks in Real Time," *Sensors (Basel)*, vol. 24, no. 14, p. 4516, Jul. 2024, doi: 10.3390/S24144516.

[66] H. Mamdouh *et al.*, "Apache Spark Powered: Enhancing Network Intrusion Detection System Using Random Forest," *NILES 2024 - 6th Novel Intelligent and Leading Emerging Sciences Conference, Proceedings*, pp. 289–294, 2024, doi: 10.1109/NILES63360.2024.10753188.

[67] Y. Alslman, A. Khalil, R. Younisse, E. Alnagi, J. Al-Saraireh, and R. Ghnemat, "DDOS ATTACK-DETECTION APPROACH BASED ONENSEMBLE MODELS USING SPARK," *Jordanian Journal of Computers and Information Technology*, vol. 10, no. 2, pp. 123–137, Jun. 2024, doi: 10.5455/JJCIT.71-1694806966.