knowledge

MDPI

Article

A Comparative Study of Two-Stage Intrusion Detection Using Modern Machine Learning Approaches on the CSE-CIC-IDS2018 Dataset

Isuru Udayangani Hewapathirana

Software Engineering Teaching Unit, Faculty of Science, University of Kelaniya, Kelaniya 11600, Sri Lanka; ihewapathirana@kln.ac.lk

Abstract: Intrusion detection is a critical component of cybersecurity, enabling timely identification and mitigation of network threats. This study proposes a novel two-stage intrusion detection framework using the CSE-CIC-IDS2018 dataset, a comprehensive and realistic benchmark for network traffic analysis. The research explores two distinct approaches: the stacked autoencoder (SAE) approach and the Apache Spark-based (ASpark) approach. Each of these approaches employs a unique feature representation technique. The SAE approach leverages an autoencoder to learn non-linear, data-driven feature representations. In contrast, the ASpark approach uses principal component analysis (PCA) to reduce dimensionality and retain 95% of the data variance. In both approaches, a binary classifier first identifies benign and attack traffic, generating probability scores that are subsequently used as features alongside the reduced feature set to train a multi-class classifier for predicting specific attack types. The results demonstrate that the SAE approach achieves superior accuracy and robustness, particularly for complex attack types such as DoS attacks, including SlowHTTPTest, FTP-BruteForce, and Infilteration. The SAE approach consistently outperforms ASpark in terms of precision, recall, and F1-scores, highlighting its ability to handle overlapping feature spaces effectively. However, the ASpark approach excels in computational efficiency, completing classification tasks significantly faster than SAE, making it suitable for real-time or large-scale applications. Both methods show strong performance for distinct and well-separated attack types, such as DDOS attack-HOIC and SSH-Bruteforce. This research contributes to the field by introducing a balanced and effective two-stage framework, leveraging modern machine learning models and addressing class imbalance through a hybrid resampling strategy. The findings emphasize the complementary nature of the two approaches, suggesting that a combined model could achieve a balance between accuracy and computational efficiency. This work provides valuable insights for designing scalable, high-performance intrusion detection systems in modern network environments.

Keywords: intrusion detection; stacked autoencoder; apache spark; machine learning; principal component analysis; cybersecurity; CSE-CIC-IDS2018

Academic Editor: Jose María Merigo

Received: 30 January 2025
Revised: 7 March 2025
Accepted: 10 March 2025
Published: 12 March 2025

Citation: Hewapathirana, I.U. A Comparative Study of Two-Stage Intrusion Detection Using Modern Machine Learning Approaches on the CSE-CIC-IDS2018 Dataset. *Knowledge* **2025**, *5*, 6. https://doi.org/10.3390/knowledge5010006

1. Introduction

With the expansion of computer networks and the massive increase in the usage of computer applications on these networks, network security is becoming more and more crucial. Every computer system has security flaws. Manufacturers find it challenging and expensive to close these gaps in practice. Intrusion detection systems now play a critical role in identifying network irregularities and attacks. These systems monitor network traffic

for suspicious activity, enabling organizations to respond swiftly and mitigate potential threats before they escalate into significant breaches. Modern intrusion detection systems utilize a combination of signature-based and anomaly-based detection methods. Signature-based methods rely on predefined patterns to identify known threats [1], while anomaly-based methods detect deviations from normal behavior, potentially identifying unknown threats [2]. The integration of both methods into hybrid systems has also shown promise in enhancing detection capabilities.

As cyber threats continue to evolve, the integration of machine learning and artificial intelligence into intrusion detection systems is becoming increasingly important, enhancing their ability to adapt and respond to new attack vectors in real time. There are many machine learning methods employed in previous research. Various supervised learning algorithms, including logistic regression, naive Bayes, KNN, decision trees, random forests, and SVM, have been applied to detect network intrusions [3]. These methods have demonstrated advantages such as higher detection rates and lower false alarm rates. Researchers have also explored ensemble learning and deep learning approaches for improved performance. Autoencoders, a type of unsupervised neural network, have shown promise in network intrusion detection systems (NIDS) due to their ability to learn compressed representations of normal network behavior and identify anomalies [4]. These models can effectively detect unknown attacks, including zero-day threats, by flagging traffic with high reconstruction loss [5]. Research has explored various aspects of autoencoder-based NIDS, including architecture optimization, data preprocessing, and feature engineering [6]. The performance of autoencoders in intrusion detection can be influenced by factors such as latent layer size and model complexity [6].

Various ensemble techniques have been explored to enhance detection accuracy and reduce false positives. Out of those, two-stage intrusion detection systems have shown promising results. These approaches typically involve a preliminary stage for traffic analysis or feature selection, followed by a more detailed classification stage. The work in [7] introduced a model that first distinguishes between normal and attack traffic, then identifies specific attack types in the second stage. Stacked autoencoders (SAEs) combined with deep neural networks or support vector machines have achieved high accuracy in multi-class classification tasks, ranging from 94.2% to 99.9% on different datasets [8,9].

Feature selection techniques such as correlation analysis and principal component analysis have been employed to improve model performance in most ensemble methods [10]. Overall, ensemble methods have consistently demonstrated superior performance compared to single classifiers in intrusion detection tasks [10,11].

NIDSs rely heavily on datasets for development and evaluation. Several benchmark datasets have been widely used in NIDS research, including KDD99, NSL-KDD, KYOTO 2006+, ISCX2012, UNSW-NB15, CIDDS-001, CICIDS2017, and CSE-CIC-IDS2018 [12]. These datasets vary in their characteristics, such as the number of instances, features, and attack types represented. While some datasets are simulated, others contain real network traffic data, which is crucial for developing effective NIDS. The CSE-CIC-IDS2018 dataset is one of the most recent datasets that is available for advancing intrusion detection research. It is a large, publicly available dataset covering 14 different types of attacks, making it suitable for training machine learning models [13]. However, this dataset presents various challenges due to its size and diversity. Big data analytics and machine learning offer promising solutions to the issues that arise due to the complexity and volume of the data. But still, difficulties are encountered due to the adversarial nature and dynamism of cyber threats [14]. Handling class imbalance and proper data cleaning and preprocessing are also mandatory when analyzing this dataset [15]. Recent research has explored the use of Apache Spark for efficient network intrusion detection systems (IDS) capable of processing

large volumes of data [16]. Multiple studies have employed machine learning algorithms on the Apache Spark platform, including logistic regression, support vector machines, random forest, gradient boosted decision trees, and naive Bayes, to detect attack traffic [17–19].

Despite significant advances in machine learning-driven IDS research, there remains a gap in comparative studies assessing deep learning-based feature extraction methods versus scalable, distributed computing-based feature selection techniques. The research question guiding this study is "How do the SAE-based two-stage deep learning model and the ASpark-PCA-based model compare in terms of classification performance, feature representation, and computational efficiency for network intrusion detection?" This research aims to compare two distinct two-stage intrusion detection models using the CSE-CIC-IDS2018 dataset: (i) a stacked autoencoder (SAE)-based model, which leverages deep learning for non-linear feature representation and a multi-layer perceptron (MLP)-based two-stage classifier for intrusion detection, and (ii) an Apache Spark (ASpark)-based model, which applies principal component analysis (PCA) for feature selection and employs Logistic Regression for binary classification and random forest for multi-class classification. While prior work has investigated the individual performance of deep learning and big data frameworks in IDS research, this study uniquely evaluates their relative trade-offs in a unified, controlled experimental setup. Unlike previous studies that focused on optimizing single models, this research aims to determine which approach is better suited for real-time, large-scale, or highly accurate intrusion detection applications.

This paper makes several key contributions to the field of network security and IDS research. First, it systematically evaluates two distinct two-stage classification frameworks—one using deep feature learning (SAE) and the other employing distributed feature selection (ASpark-PCA). Second, the study examines the differences in how SAE-based non-linear feature learning and PCA-based linear feature selection impact classification accuracy and robustness across diverse attack types. Third, the paper provides a detailed analysis of model performance on the CSE-CIC-IDS2018 dataset, offering insights into the effectiveness of these methods for both binary and multi-class classification tasks. Finally, the research highlights the importance of scalable architectures, such as Apache Spark, for real-time intrusion detection in dynamic network environments.

The findings of this study are significant for both academic research and practical cybersecurity applications. By evaluating the relative strengths and weaknesses of SAE-based deep learning models and Spark-based distributed computing frameworks, this research provides guidance on selecting the optimal intrusion detection methodology based on use-case requirements. The study's results will help industry practitioners and cybersecurity researchers design more efficient, scalable, and accurate IDS solutions. Moreover, this research highlights the complementary nature of these two approaches, suggesting that future IDS implementations could benefit from a combined framework that integrates the robustness of deep feature learning with the scalability of big data architectures. Such a solution could balance accuracy and efficiency, optimizing intrusion detection performance in dynamic, large-scale network environments.

2. Literature Review

2.1. The Evolution of Intrusion Detection Systems and Machine Learning Approaches

Intrusion detection systems (IDSs) play a critical role in network security, evolving alongside the increasing complexity of cyber threats. Initially, signature-based IDSs dominated the field, relying on predefined attack signatures to detect malicious activity. However, this method proved ineffective against zero-day threats, prompting a shift toward anomaly-based IDSs that establish a baseline of normal network behavior to detect deviations [20]. While anomaly-based detection improved the ability to identify unknown

attacks, it also introduced a higher false positive rate. As a result, hybrid IDSs combining signature-based and anomaly-based detection emerged as a more robust solution, enhancing detection accuracy and reducing false alarms [21]. The integration of machine learning (ML) techniques into IDSs has further advanced intrusion detection by enabling automated analysis of network traffic data, reducing reliance on manual rule-based detection. Supervised ML models train on labeled datasets, achieving high precision in classifying normal and malicious traffic [22]. Unsupervised methods, such as autoencoders (AEs), learn network behavior patterns without requiring labeled data, making them well suited for detecting novel attack types [23]. Additionally, ensemble models that combine multiple classifiers have been found to enhance detection accuracy and reduce false positives by leveraging the strengths of different models. Despite these advancements, IDSs still face challenges related to scalability, real-time detection, and high-dimensional feature spaces in network traffic data.

2.2. Two-Stage Classification in Intrusion Detection Systems

Two-stage classification frameworks have gained prominence in IDS research due to their ability to enhance accuracy and efficiency by separating detection and classification tasks. These frameworks typically involve an initial coarse classification stage that distinguishes between normal and malicious traffic, followed by a more refined second stage that identifies specific attack types. The structured methodology of two-stage models improves classification precision and reduces false positives by allowing models to process data in a hierarchical manner. Recent research has explored different combinations of machine learning and deep learning models in two-stage IDS architectures. The study in [24] proposed a three-layer hybrid model combining long short-term memory (LSTM) and random forest (RF) to improve IDS performance in imbalanced network traffic scenarios. The model utilized Nearmiss-2 class balancing, chi-square feature selection, and hyperparameter tuning, achieving over 99% accuracy, precision, recall, and F1-score for multi-class classification. Similarly, the random forest model combined with K-nearest neighbors (KNN) has demonstrated exceptional classification performance, achieving up to 99.998% accuracy [25]. Other studies have explored CNN-LSTM hybrids, achieving 99% accuracy, surpassing individual CNN and LSTM models [26]. Ensemble methods integrating random forest, gradient boosting, and multi-layer perceptron (MLP) have also been shown to enhance recall and precision, achieving 99.98% overall accuracy [27]. While two-stage IDS frameworks demonstrate significant advantages in classification accuracy, they often introduce higher computational costs due to increased model complexity. The trade-off between accuracy and computational efficiency must be carefully considered when designing IDS solutions, particularly for real-time and large-scale applications. Existing research has extensively explored various ML models for two-stage IDS architectures, but there is limited comparative analysis evaluating the feature selection strategies and computational trade-offs between deep feature learning models (such as SAEs) and distributed feature selection techniques (such as PCA in Apache Spark).

2.3. Stacked Autoencoders for Feature Learning in IDSs

Stacked autoencoders (SAEs) are an advanced class of unsupervised deep learning models designed to learn compressed feature representations by minimizing reconstruction error in network traffic data. SAEs consist of multiple encoder and decoder layers, enabling them to capture complex non-linear relationships in high-dimensional datasets. In IDS applications, SAEs have demonstrated strong anomaly detection capabilities by effectively learning the underlying structure of benign traffic and flagging deviations indicative of malicious activity. The first autoencoder layers extract high-level features from raw traffic

data, filtering out noise and enhancing detection precision [28]. The effectiveness of deep-stacked autoencoders in IDS research has been widely reported. Some studies have shown that SAE-based models achieve up to 98.22% classification accuracy in network intrusion detection [29] (S et al., 2024). Khan et al. in [30] proposed a two-stage deep learning model integrating an SAE with a Softmax classifier, achieving high detection accuracy on the KDD99 and UNSW-NB15 datasets. However, despite these advantages, SAEs are computationally intensive, requiring significant processing power to train and deploy in large-scale environments. This computational complexity limits their applicability for real-time intrusion detection in high-throughput network environments, creating a gap for alternative feature selection methods that balance accuracy with computational efficiency.

2.4. Principal Component Analysis for Feature Selection in IDSs

Feature selection and dimensionality reduction are critical for improving IDS performance by eliminating irrelevant or redundant features, thereby reducing model complexity and improving efficiency. Principal component analysis (PCA) is a widely used dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional space while preserving variance [31]. PCA-based feature selection has been shown to enhance IDS accuracy by retaining the most informative components of network traffic while discarding redundant features [32]. Autoencoders have also been explored for unsupervised feature extraction, offering advantages over PCA by capturing complex, non-linear dependencies in network traffic data [33]. While PCA provides computational efficiency, it may discard certain non-linear patterns critical for detecting advanced persistent threats (APTs). Previous studies have evaluated PCA and autoencoders separately, but comparative analyses assessing their trade-offs in a two-stage classification setting remain limited.

2.5. Apache Spark for Scalable IDS Solutions

The integration of big data frameworks such as Apache Spark has transformed IDS research by enabling scalable and real-time intrusion detection in high-volume network environments. Spark's distributed computing capabilities allow IDSs to process vast amounts of traffic efficiently, making it a viable alternative to traditional ML-based IDS models [34]. Spark's MLlib library supports a range of machine learning algorithms, including random forest and decision trees, which have demonstrated high F1-scores in intrusion detection tasks [35,36]. While previous Spark-based IDS models have leveraged deep learning techniques such as CNNs, LSTMs, and random forest, they often suffer from latency issues due to the computational complexity of training deep models on distributed frameworks [16]. Many of these approaches do not integrate feature selection techniques like PCA, instead relying on full feature sets, which increases computational cost. Unlike prior work, this study implements PCA for dimensionality reduction in an Apache Spark-based IDS model, reducing processing time while maintaining over 95% of the variance in network traffic data.

2.6. Intrusion Detection Datasets

Benchmark datasets, such as KDD99, NSL-KDD, and CSE-CIC-IDS2018, play a critical role in evaluating IDS performance. The CSE-CIC-IDS2018 dataset, used in this study, is particularly notable for its diversity and realism. This dataset encompasses multiple attack types, such as DoS, DDoS, and various intrusion attempts, which reflect real-world cyber threats [37]. This diversity allows researchers to test and validate machine learning models across different conditions, enhancing their robustness and adaptability [38]. Studies have demonstrated that identifying relevant features from the CSE-CIC-IDS2018 dataset can significantly enhance detection accuracy [39]. Techniques like principal component analysis (PCA) and random forest have been employed to optimize feature sets, leading to improved

processing speed and model efficiency [15]. The dataset has been instrumental in evaluating various machine learning classifiers, such as decision trees, random forest, and naive Bayes, revealing their strengths and weaknesses in detecting specific threats [15,37].

2.7. Research Gap and Contribution of This Study

Despite extensive research on SAE-based IDS models and Apache Spark-based IDS frameworks, there is no direct comparative analysis evaluating deep learning-based feature extraction (SAEs) versus scalable distributed feature selection (PCA in Apache Spark). While SAE models offer high classification accuracy, they suffer from computational inefficiencies, making them less feasible for real-time IDS applications. In contrast, Apache Spark-based IDS solutions improve scalability but often fail to capture non-linear data representations effectively. This study bridges this gap by comparing an SAE-based IDS approach with an Apache Spark-PCA-based IDS framework in a two-stage classification setting. By evaluating the trade-offs in classification accuracy, feature representation, and computational efficiency, this research provides new insights into optimizing IDS performance in large-scale network security environments. Furthermore, these methodologies are evaluated on the most recently collected CSE-CIC-IDS2018 dataset.

3. Methodology

In this study, we propose a two-stage ensemble approach for intrusion detection using the CSE-CIC-IDS2018 dataset. The dataset initially undergoes several preprocessing steps. After preprocessing, the dataset is processed through two distinct approaches: a stacked autoencoder (SAE)-based approach and an Apache Spark (ASpark)-based approach. Each approach begins with feature selection—non-linear feature representation using an autoencoder in the SAE approach and dimensionality reduction using principal component analysis (PCA) in the ASpark approach. In both approaches, a binary classifier is first trained to classify instances as either benign or attack, producing a probability score for each instance. This probability score vector is then combined with the reduced feature set from the first step to form an enriched feature representation. Using this augmented feature set, a multi-class classifier is trained to predict the specific attack type for instances identified as attacks in the binary classification stage. Finally, the performance of each approach is evaluated using metrics such as accuracy, precision, recall, and F1-score. An overview of the methodology is provided in Figure 1.

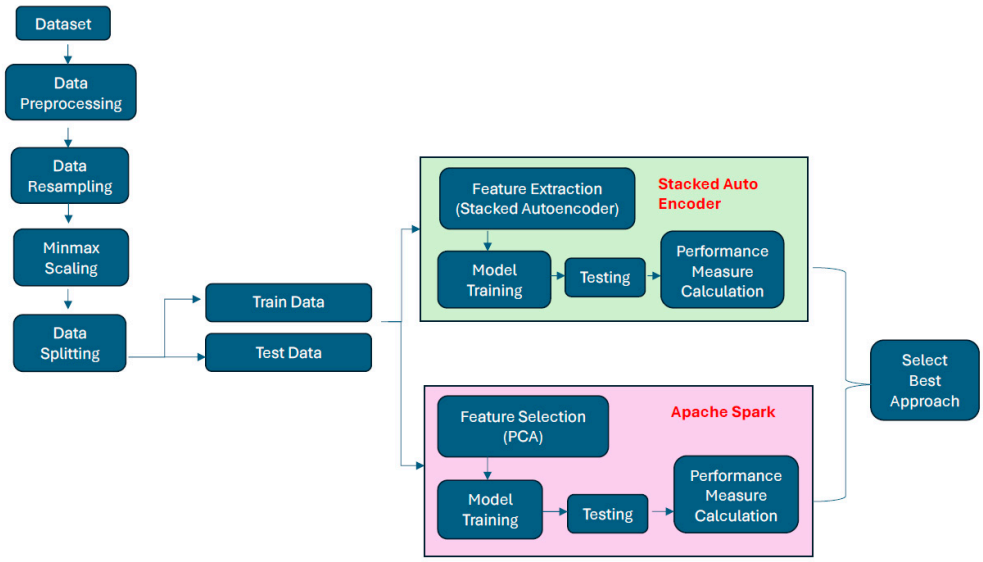

Figure 1. Overview of the research methodology.

3.1. Dataset

The CSE-CIC-IDS2018 dataset, created in collaboration between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC) in 2018, contains approximately 16 million records. The data are divided into two main categories: 83% normal traffic and 17% attack traffic. The dataset is split into ten separate files, with nine files featuring 79 attributes each and one file containing 83 attributes. Table 1 provides an overview of the class distribution across attack categories, showing that infiltration and web attacks such as SQL injection and SSH-Bruteforce are the least represented.

Table 1. Distribution of normal and attack class in network traffic distribution on CSE-CIC-IDS2018 dataset.

Class Name	Label Mapping	Original Size	Distribution (%)	New Size	New Distribution (%)
Benign	0	13,390,249	82.98	150,000	51.72
Bot	1	686,012	4.25	10,000	3.45
Brute Force-Web	2	576,191	3.57	10,000	3.45
Brute Force-XSS	3	461,912	2.86	10,000	3.45
DDOS attack-HOIC	4	286,191	1.77	10,000	3.45
DDOS attack-LOIC-UDP	5	193,354	1.20	10,000	3.45
DDoS attacks-LOIC-HTTP	6	187,589	1.16	10,000	3.45
DoS attacks-GoldenEye	7	160,639	0.99	10,000	3.45
DoS attacks-Hulk	8	139,890	0.87	10,000	3.45
DoS attacks-SlowHTTPTest	9	41,508	0.26	10,000	3.45
DoS attacks-Slowloris	10	10,990	0.07	10,000	3.45
FTP-BruteForce	11	1730	0.01	10,000	3.45
Infilteration	12	611	0.004	10,000	3.45
SQL Injection	13	230	0.001	10,000	3.45
SSH-Bruteforce	14	87	0.005	10,000	3.45

3.2. Data Preprocessing

At the initial stage of preprocessing, the ten datasets were merged, duplicate headers were removed, and timestamp features were removed. Then, ID columns such as "Flow ID", "Src IP", "Src Port", and "Dst IP" were removed. Next, features with NaN values were discarded, while individual data samples were removed for features with NaN values. Additionally, features with no variation and one of any two features with highly similar value distributions were eliminated. Correlations between features were calculated, and one of the features that correlated greater than or equal to 0.9 was removed. Following this preprocessing stage, the dataset was reduced from its original 84 features to 40 features, with 44 features and 95,760 data samples removed, leaving a total of 16,137,183 samples.

Before model fitting, the dataset presented two significant challenges: its large size and a severe class imbalance, where the ratio was approximately 4.79:1 between benign and attack samples. To address these issues, a hybrid resampling strategy was employed, resulting in a final dataset of approximately 290,000 samples, which represents about 20% of the dataset of 16,137,183 samples. This balancing process involved randomly downsampling the majority class (benign traffic) and either randomly downsampling or upsampling the attack classes based on their original sample sizes. Specifically, the benign class (label = 0), which originally comprised 13,390,249 instances, was randomly downsampled to 150,000 instances to reduce its dominance in the dataset. Each attack class (label $\neq$ 0) was adjusted to 10,000 instances: If an attack class had more than 10,000 samples, it was downsampled to 10,000 instances to maintain consistency across all attack types; if an attack class had fewer than 10,000 samples, it was upsampled using random replication (with replacement) until it reached 10,000 instances. After resampling, all the downsampled and upsampled subsets

were concatenated into a single balanced dataset, which was then shuffled randomly to prevent any ordering bias. The final dataset achieved a 1:1 ratio between attack and benign samples, ensuring equal representation for both categories and mitigating potential bias in model training. Balancing the dataset mitigates the issue of class imbalance, enabling the model to learn effectively from both benign and attack data and improving its ability to detect minority class samples. Table 1 provides a detailed comparison of the class distribution in the original and resampled datasets. After resampling, the dataset was split into 80% training and 20% testing subsets to ensure reliable model evaluation. To standardize the feature values and improve model convergence, min–max scaling was applied, transforming all features to a range between 0 and 1.

3.3. Stacked Autoencoder (SAE) Approach

The core model in the SAE approach is the autoencoder (AE), a type of feed-forward neural network designed for unsupervised learning. An autoencoder comprises three main components: an input layer, one or more hidden layers, and an output layer. The number of neurons in the input and output layers is typically identical. The AE learns a compressed representation of input data through two phases: the encoding phase, where input data are compressed into a latent representation, and the decoding phase, where the compressed representation is used to reconstruct the original input data. The objective is to minimize reconstruction error, thereby capturing key abstract features of the input data.

The proposed SAE-based framework operates in two main stages: an autoencoder pretraining stage and a classification stage, designed for intrusion detection in network traffic. The model begins by pretraining an autoencoder, which consists of two components: (i) Encoder: Compresses input features into a lower-dimensional representation. (ii) Decoder: Reconstructs the original features from this compressed representation. The autoencoder is pre-trained using unsupervised learning with a hyperparameter-tuned deep neural network. Once pretrained, the compressed features generated by the encoder are passed to the classification stages. In the binary classification stage, the model uses the compressed features to classify network traffic as either "Benign" or "Attack". A hyperparameter-tuned MLPClassifier is trained to maximize classification accuracy for this task. The binary classifier outputs a probability score, which is then used as an additional feature in the multi-class classification stage. In the multi-class classification stage, the model extends the compressed feature set with the probability scores from the binary classifier and trains a second MLPClassifier. This stage distinguishes between normal traffic and specific attack categories, enabling the detection of diverse attack types. Both classifiers utilize supervised learning and leverage backpropagation to fine-tune their parameters. In the evaluation phase, the trained model is applied to previously unseen network traffic. The encoder first compresses the test features, which are then passed through the binary and multi-class classifiers. Performance metrics (Section 3.5) are computed for both classification tasks. Additionally, execution times are measured and logged to assess the model's efficiency.

3.4. Apache Spark (ASpark) Based Approach

The ASpark classification framework is designed in two stages, leveraging PCA-reduced features and Spark's distributed machine-learning capabilities. Apache Spark is a distributed cluster computing platform designed to process massive datasets efficiently. As an open-source framework, it leverages a multistaged in-memory processing technique. Spark provides robust tools for managing Hadoop clusters, accessing data from Hadoop-compatible sources, and executing distributed analytics workflows. Its core engine supports a wide range of capabilities, such as streaming, machine learning, graph processing, and SQL queries, making

it a versatile choice for big data analytics. In this research, Spark is utilized to perform both binary and multi-class classification tasks for network intrusion detection.

Principal component analysis (PCA) is a dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional subspace while retaining the most critical information. It identifies directions, or principal components, along which the variance in the data is maximized. These components are linear combinations of the original features and are orthogonal to one another. By discarding components with negligible variance, PCA reduces noise and computational overhead, enabling more efficient downstream processing. In this research, PCA is applied as a preprocessing step to reduce the feature dimensions of the dataset. The optimal number of principal components is determined by analyzing the explained variance ratio, ensuring that at least 95% of the original variance is retained. These reduced features serve as the input for subsequent classification tasks.

In the first stage of the Aspark classification framework, PCA is first applied to reduce the number of features. Then, a logistic regression model is trained on the PCA-reduced features to classify network traffic as either benign or an attack. Logistic regression, a widely used linear model, leverages the sigmoid function to predict probabilities for binary outcomes. Unlike more complex models, logistic regression performs well in high-dimensional spaces when combined with PCA, as PCA removes multicollinearity, ensuring that the linear decision boundary assumption holds. Furthermore, LR is computationally efficient, making it ideal for real-time applications where low latency is required. The output of this stage includes two key components: predicted labels and probability scores. The predicted labels classify each instance as either benign (class 0) or an attack (class 1). Meanwhile, the probability scores provide the model's confidence for each prediction, specifically indicating the likelihood of an instance belonging to the attack class (class 1). These probability scores are incorporated to enrich the feature set in the second stage. The second stage of the Aspark framework extends the binary classification by utilizing the probability scores generated in the first stage as additional features. These scores are concatenated with the PCA-reduced features to create an augmented feature set, which serves as input for the multi-class classification model. A random forest classifier is employed at this stage to classify network traffic into specific categories, including benign traffic and various attack types. Random forest was chosen for multi-class classification due to its robustness against overfitting, ability to handle complex decision boundaries, and superior performance on high-dimensional structured data. Unlike single decision tree classifiers, RF constructs multiple decision trees and aggregates their predictions, reducing variance and improving generalization. Additionally, RF is well suited for handling non-linear decision boundaries, which is crucial in distinguishing between multiple attack categories in the dataset. By combining the PCA-reduced features with the probability scores from the binary classifier, the augmented feature set provides richer information, thereby improving classification performance for multi-class tasks.

The Aspark approach takes full advantage of Apache Spark's distributed computing capabilities to efficiently process large-scale datasets. By parallelizing model training, and evaluation, the framework ensures scalability and speed, making it particularly well suited for real-time or batch intrusion detection in complex network environments. In the evaluation phase, key performance metrics are calculated.

3.5. Performance Evaluation

Several performance metrics are employed in this research to evaluate the effectiveness of the proposed intrusion detection approaches. These metrics include accuracy (Equation (1)), precision (Equation (2)), recall (Equation (3)), F1-score (Equation (4)), and

the time taken to process the test dataset (test time—Equation (5)). Consider TP as the number of correctly classified positive samples, TN as the number of correctly classified negative samples, FP as the number of incorrectly classified negative samples as positive, and FN as the number of incorrectly classified positive samples as negative. Then, the performance metrics can be defined as,

$$\text{Accuracy} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN}) \quad (1)$$

$$\text{Precision} = \text{TP}/(\text{TP} + \text{FP}) \quad (2)$$

$$\text{Recall} = \text{TP}/(\text{TP} + \text{FN}) \quad (3)$$

$$\text{F1-Score} = 2 \times (\text{Precision} \times \text{Recall})/(\text{Precision} + \text{Recall}) \quad (4)$$

$$\text{Test time} = \text{t\_end} - \text{t\_start}, \quad (5)$$

where t_start is the time when the model begins evaluating the test dataset and t_end is the time when the model completes evaluation.

Accuracy measures the overall ability of the model to correctly classify outcomes, offering a broad assessment of performance. Precision evaluates the reliability of positive predictions, focusing on minimizing false positives, which is particularly critical in intrusion detection to avoid unnecessary false alarms that can disrupt system efficiency. Recall, or the true positive rate, assesses the model's capability to detect actual intrusions, with high recall values being essential to reduce missed detections and ensure a robust system capable of identifying threats comprehensively. The F1-score, as the harmonic mean of precision and recall, provides a balanced measure of the model's overall performance, highlighting its ability to maintain both accuracy and completeness. A higher F1-score indicates the model's effectiveness in achieving an optimal trade-off between precision and recall. Additionally, the time taken to process the test dataset is calculated. The test time provides insights into the computational efficiency of the approaches, which is crucial for real-time intrusion detection applications.

4. Experimental Results

All experiments were conducted using Google Colab with the High RAM runtime environment, utilizing Python 3 for implementation. A T4 GPU was employed to accelerate the training of machine learning models. The implementation for the SAE approach included libraries such as Scikit-learn for preprocessing, feature selection, and model evaluation, and TensorFlow for constructing and training the neural network. This included key tools like MLPClassifier and GridSearchCV to optimize hyperparameters and fine-tune performance. The ASpark approach, on the other hand, utilized PySpark for distributed data processing and machine learning. Key modules such as VectorAssembler, StandardScaler, and PCA were employed for preprocessing and dimensionality reduction, while logistic regression and random forest classifiers were implemented through Spark MLlib for binary and multi-class classification, respectively. The cloud-based infrastructure of Google Colab provided scalability and computational efficiency, enabling the handling of large datasets and ensuring robust experimental outcomes.

Based on the results obtained from hyperparameter tuning, the stacked autoencoder (SAE) approach uses an optimized configuration across its components. The autoencoder, binary classifier, and multi-class classifier all employ a learning rate of 0.001, a single hidden layer with 64 neurons, and the ReLU activation function. These hyperparameters were selected to minimize reconstruction error in the autoencoder while effectively capturing non-linear patterns in the data. The Spark-based approach utilizes an optimized hyperparameter setup for both logistic regression and random forest to enhance performance. For logistic regression,

a regularization parameter (regParam) of 0.1 was applied to provide moderate regularization, while an elasticNetParam of 0.5 was used to combine L1 and L2 regularization. Additionally, the maximum number of iterations (maxIter) was increased to 200 to allow for more thorough optimization. For the random forest classifier, the number of trees (numTrees) was increased to 50 to strengthen ensemble learning, and the maximum tree depth (maxDepth) was set to 10 to capture complex patterns in the data. The subsampling rate (subsamplingRate) was reduced to 0.8, introducing randomness to prevent overfitting.

The binary classification results in Table 2 highlight the performance differences between the stacked autoencoder (SAE) approach and the Apache Spark-based (ASpark) approach. The SAE approach achieves significantly higher accuracy (0.96) compared to the ASpark approach (0.81), demonstrating its superior ability to correctly classify instances as benign or attack. SAE also outperforms ASpark in precision (0.98 vs. 0.78), indicating that the SAE approach produces fewer false positives and is more reliable when predicting attacks. In terms of recall, SAE achieves a value of 0.94, which is higher than ASpark's 0.83, suggesting that SAE is more effective at identifying true attacks and minimizing false negatives. Additionally, the F1-score of SAE (0.96) surpasses that of ASpark (0.81), indicating that the SAE approach balances precision and recall more effectively, providing an overall better performance for binary classification tasks.

Table 2. Binary classification performance measures.

Approach	Accuracy	Precision	Recall	F1-Score
SAE	0.96	0.98	0.94	0.96
ASpark	0.81	0.78	0.83	0.81

The multi-class classification results in Table 3 reveal notable differences in performance between the stacked autoencoder (SAE) approach and the Apache Spark-based (ASpark) approach across various classes. Both approaches achieve identical performance for benign traffic (class 0), with precision, recall, and F1-scores all at 0.95, 0.99, and 0.97, respectively. Similarly, for classes 1, 4, 5, 7, 8, 10, and 14, both approaches demonstrate perfect precision, recall, and F1-scores (1.00), indicating their ability to handle these specific traffic types with high accuracy. However, notable differences emerge for other classes.

For class 2, ASpark slightly outperforms SAE with a precision of 0.97 compared to 0.94 and a higher F1-score of 0.79 versus 0.77, though recall remains nearly identical. A similar trend is observed for class 3, where ASpark shows marginally better precision (0.77 vs. 0.75) and F1-score (0.84 vs. 0.83), with both approaches achieving equal recall. Class 9 highlights a significant disparity in performance; while ASpark has much higher precision (0.88 vs. 0.62), SAE achieves a considerably higher recall (0.93 vs. 0.55). As a result, ASpark's F1-score is slightly better at 0.74 compared to SAE's 0.67. For class 11, SAE exhibits better precision (0.86 vs. 0.67), whereas ASpark achieves higher recall (0.93 vs. 0.57), leading to an overall better F1-score for ASpark (0.78 vs. 0.57). Both approaches struggle with class 12, achieving low recall values (0.34 for SAE and 0.21 for ASpark), though ASpark slightly outperforms SAE in precision (0.80 vs. 0.77) and F1-score (0.33 vs. 0.34). For class 13, ASpark outperforms SAE across all metrics, achieving higher precision (0.86 vs. 0.82), recall (0.95 vs. 0.87), and F1-score (0.90 vs. 0.87). Despite these variations, the overall accuracy for both SAE and ASpark is the same at 0.93, suggesting similar success rates across all classifications.

Overall, SAE demonstrates more consistent performance across most classes and excels in recall for harder-to-detect classes like 9 and 11, while ASpark shows better precision and slightly higher F1-scores for challenging classes like 2, 9, 11, and 13. Both approaches

perform well for benign traffic and simpler attack types, but additional optimization may be needed for underperforming classes such as class 12.

Table 3. Multi-class classification performance results.

Class	Precision		Recall		F1-Score		Accuracy	
	SAE	ASpark	SAE	ASpark	SAE	ASpark	SAE	ASpark
0	0.95	0.95	0.99	0.99	0.97	0.97		
1	1.00	1.00	1.00	1.00	1.00	1.00		
2	0.94	0.97	0.66	0.67	0.77	0.79		
3	0.75	0.77	0.93	0.93	0.83	0.84		
4	1.00	1.00	1.00	1.00	1.00	1.00		
5	1.00	1.00	1.00	1.00	1.00	1.00		
6	0.99	0.99	0.99	0.99	0.99	0.99		
7	1.00	1.00	1.00	1.00	1.00	1.00	0.93	0.93
8	1.00	1.00	1.00	0.99	1.00	1.00		
9	0.62	0.88	0.93	0.55	0.74	0.67		
10	1.00	1.00	1.00	1.00	1.00	1.00		
11	0.86	0.67	0.57	0.93	0.57	0.78		
12	0.77	0.80	0.34	0.21	0.34	0.33		
13	0.82	0.86	0.87	0.95	0.87	0.90		
14	1.00	1.00	1.00	1.00	1.00	1.00		

The confusion matrix in Figure 2 further provides more details of the multi-class classification performance of the SAE approach. The SAE approach demonstrates excellent classification performance for several classes, with near-perfect results for classes 1, 4, 5, 7, 8, 10, and 14. These classes show minimal misclassifications, highlighting the model's ability to handle distinct traffic types effectively. Class 0, representing benign traffic, also performs well, with 29,597 out of 30,000 instances correctly classified. However, minor misclassifications occur in classes 12 and 13, as well as smaller errors across other classes. Despite its overall strong performance, the model struggles with specific classes, particularly classes 2, 3, 9, 11, and 12. For class 2, while 1315 instances are classified correctly, 415 are misclassified into class 3, and 183 into class 13, indicating feature overlap between these classes. Similarly, class 3 achieves a high correct classification rate of 1866 instances but shows confusion with classes 2 and 13. Class 9, although mostly accurate with 1857 correct classifications, exhibits significant misclassifications into class 11, with 143 instances incorrectly labeled. Class 11, in turn, shows considerable challenges, with 863 out of 2000 instances misclassified into class 9, suggesting a high degree of similarity between these two classes. Class 12 presents one of the most notable issues, as 1548 instances are misclassified into class 0, leaving only 441 instances correctly labeled. This indicates that the features of class 12 may closely resemble benign traffic, making separation difficult for the model. Class 13 performs well, with 1862 correctly classified instances and relatively minor confusion with classes 2 and 3.

Overall, while the SAE approach achieves outstanding results for distinct traffic types, it struggles with classes that exhibit overlapping features or less distinct characteristics, such as classes 2, 3, 9, 11, and 12. These observations suggest that further refinement in feature engineering or model design is necessary to improve the classification of these challenging classes.

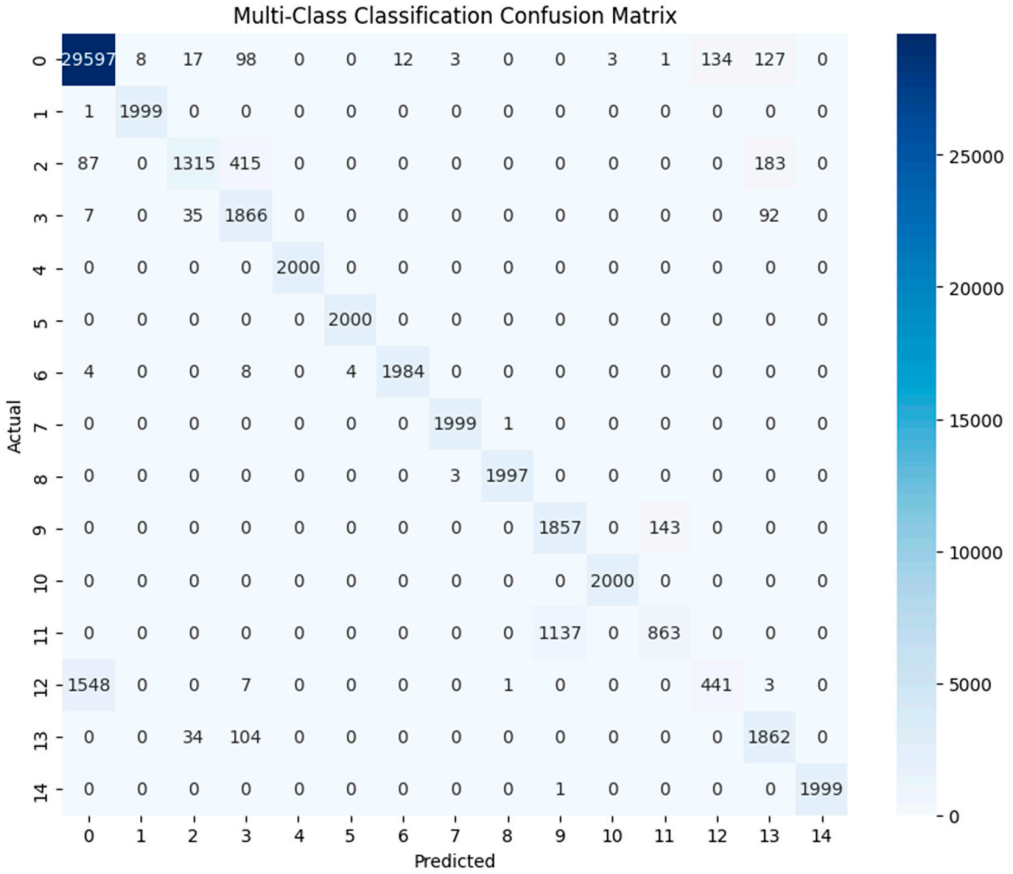

Figure 2. Confusion matrix for multi-class classification using SAE approach.

The confusion matrix of the ASpark approach (Figure 3) demonstrates strong performance for several classes, particularly classes 1, 4, 5, 7, 8, 10, and 14, where classification is nearly perfect with minimal or no misclassifications. Class 0, representing benign traffic, is also handled effectively, with 29,773 out of 30,000 instances correctly classified. However, a small number of instances are misclassified into class 12 (105 instances) and class 13 (40 instances), reflecting some overlap in feature representations. For class 6, the model performs well, correctly classifying 1985 out of 2000 instances with only minor errors.

Despite its strong performance for distinct traffic types, the model struggles with certain classes. For class 2, only 1336 instances are correctly classified, with significant misclassifications into class 3 (400 instances) and class 13 (173 instances). Similarly, class 3 achieves good performance with 1866 correct classifications but exhibits confusion with classes 2 and 13. A more pronounced issue arises with class 9, where only 1090 instances are correctly classified, while 910 are misclassified into class 11, highlighting a major overlap in their feature space. This confusion is reciprocated in class 11, where 1850 instances are correctly classified, but 150 are misclassified into class 9. Class 12 presents the greatest challenge, as the majority of its instances (1574) are misclassified into class 0, leaving only 416 instances correctly identified. This indicates significant difficulty in distinguishing class 12 from benign traffic. Class 13 performs relatively well, with 1896 instances correctly classified, but there is moderate confusion with class 3, as 104 instances are misclassified.

Overall, the ASpark approach performs exceptionally well for distinct and well-separated classes but faces challenges with classes that exhibit overlapping features, such as classes 2, 3, 9, 11, and 12. The severe misclassification of class 12 into class 0 highlights the need for better feature representation or additional refinement in handling specific attack types. While the approach achieves strong overall performance, targeted improve-

ments in feature engineering or model design are necessary to enhance its accuracy for challenging classes.

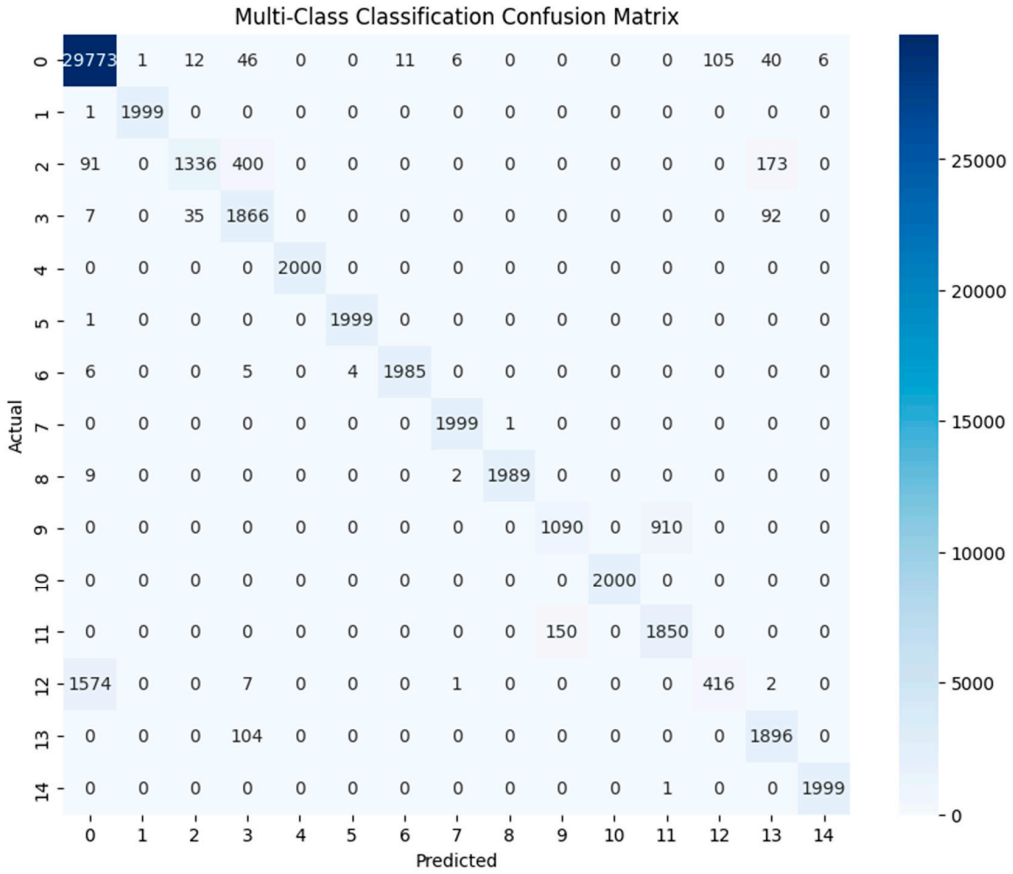

Figure 3. Confusion matrix for multi-class classification using ASpark approach.

The comparison of classification times for the test dataset with 58,000 data samples in Table 4 reveals that the ASpark approach is generally faster than the SAE approach for both binary and multi-class classification tasks. For binary classification, ASpark completes the task in 0.1145 s, slightly faster than SAE, which takes 0.1189 s. The difference here is minimal, indicating comparable efficiency between the two approaches for binary classification. However, the disparity becomes more pronounced for multi-class classification. ASpark processes the task in 0.0464 s, which is nearly half the time taken by SAE (0.0865 s). This demonstrates a significant advantage of the Spark-based framework in handling multi-class tasks, likely due to its distributed computing capabilities and optimizations for large-scale data processing.

Table 4. Time comparison for intrusion detection by SAE and ASpark approaches.

Classification	Approach	Time (Seconds)
Binary	SAE	0.1189
	ASpark	0.1145
Multi-class	SAE	0.0865
	ASpark	0.0464

5. General Discussion

The comparative evaluation of the stacked autoencoder (SAE) and Apache Spark-based (ASpark) approaches reveals key insights into their strengths, weaknesses, and

suitability for different use cases in intrusion detection. While both approaches show strong overall performance, their differences in feature representation, classification accuracy, and computational efficiency are notable.

The SAE approach, which leverages an autoencoder for feature representation, demonstrates superior accuracy and performance across most traffic types. It excels in multi-class classification, effectively distinguishing between complex attack types such as DDoS attacks-LOIC-HTTP (class 6), DoS attacks-SlowHTTPTest (class 9), and FTP-BruteForce (class 11). The SAE approach's ability to learn non-linear feature representations allows it to handle overlapping feature spaces better, which is evident in its relatively higher recall for challenging attack types like DoS attacks-SlowHTTPTest and FTP-BruteForce. However, the SAE approach struggles with Infilteration (class 12), where a significant number of instances are misclassified as benign (class 0). This suggests that the autoencoder's learned features may still lack sufficient separation for certain subtle or underrepresented attack types. Despite this, the SAE approach achieves higher overall F1-scores and precision, making it more suitable for scenarios requiring high accuracy and reliability in identifying diverse attack types.

In contrast, the ASpark approach, which uses PCA for feature representation, prioritizes computational efficiency and speed. By retaining 95% of the variance in the data, PCA provides a simpler, linear feature representation, enabling faster classification times, particularly for multi-class tasks. This advantage is evident in the significantly reduced time taken by ASpark for multi-class classification (0.0464 s compared to 0.0865 s for SAE). However, the reliance on linear transformations limits ASpark's ability to capture non-linear patterns, leading to lower performance in distinguishing complex or overlapping attacks. For example, ASpark shows higher confusion between DoS attacks-SlowHTTPTest and FTP-BruteForce, as well as significant misclassification of Infilteration instances as benign. These limitations are reflected in ASpark's lower F1-scores for certain attack types, despite achieving comparable or slightly better precision in some cases, such as Brute Force-Web (class 2) and SQL Injection (class 13).

The class-level analysis highlights the strengths and weaknesses of both approaches for specific attack types. Both SAE and ASpark perform exceptionally well for distinct traffic types, such as Bot (class 1), DDOS attack-HOIC (class 4), DDOS attack-LOIC-UDP (class 5), DoS attacks-GoldenEye (class 7), DoS attacks-Hulk (class 8), DoS attacks-Slowloris (class 10), and SSH-Bruteforce (class 14), where classification is near-perfect. However, challenges remain for subtle and underrepresented attack types like Infilteration and FTP-BruteForce, which require better feature separation and handling of overlapping patterns. The confusion between DoS attacks-SlowHTTPTest and FTP-BruteForce is particularly pronounced in the ASpark approach, suggesting a need for refined feature engineering or alternative feature representation techniques to enhance classification accuracy.

When considering the overall results, the SAE approach emerges as a robust solution for scenarios requiring high classification accuracy and reliability across diverse attack types. Its ability to adaptively learn non-linear feature representations gives it an edge in handling complex and overlapping traffic patterns. On the other hand, the ASpark approach offers significant advantages in terms of speed and computational efficiency, making it a better choice for real-time or large-scale environments where rapid decision making is critical. While ASpark shows slightly lower classification performance for complex attack types, its faster execution times and relatively high precision make it an appealing option for time-sensitive applications. Overall, the SAE approach is more suitable for accuracy-critical intrusion detection systems, while the ASpark approach is better suited for real-time, resource-constrained environments. Future work could focus on hybrid methods that combine the strengths of both approaches—leveraging the speed

of PCA-based representation with the adaptability of autoencoders—to achieve a balance between efficiency and accuracy. Additionally, addressing the challenges associated with subtle attack types like Infilteration and reducing confusion between similar attacks could further improve the effectiveness of both approaches.

6. Conclusions

This study explored two distinct approaches for intrusion detection using the CSE-CIC-IDS2018 dataset, a recent and comprehensive dataset capturing realistic network traffic patterns. A novel two-stage classification framework was employed in both approaches, where binary classification probabilities from the first stage were used as an additional feature for the multi-class classification in the second stage. This feature augmentation step was critical to enhancing the information available for distinguishing between specific attack types in the multi-class classification task.

In the stacked autoencoder (SAE) approach, an autoencoder was used in the first stage to generate non-linear, data-driven feature representations that compressed the original feature space while retaining critical information. A multi-layer perceptron (MLP) model was trained as the binary classifier to distinguish between benign and attack traffic, producing probabilities that were appended to the compressed features. These enriched features were then used to train another MLP model for multi-class classification. This approach excelled in handling attack types with overlapping feature spaces, such as DoS attacks-SlowHTTPTest, FTP-BruteForce, and Infilteration, achieving high accuracy, recall, and F1-scores across most classes. However, the deep learning-based feature extraction introduced slight computational overhead, making the SAE approach slower than the Spark-based approach.

The Apache Spark-based (ASpark) approach employed principal component analysis (PCA) in the first stage to reduce the dimensionality of the dataset while retaining 95% of the variance. A logistic regression model was trained as the binary classifier, and its probability outputs were combined with the PCA-reduced features to form an augmented feature set. In the second stage, a random forest classifier was trained on this augmented feature set for multi-class classification. This approach demonstrated superior computational efficiency, particularly in multi-class classification tasks, making it well suited for real-time and large-scale intrusion detection systems. However, its reliance on linear PCA transformations limited its ability to capture complex, non-linear relationships in network traffic data, resulting in higher misclassification rates for subtle attack types such as Infilteration. Despite this limitation, the ASpark approach performed well for distinct and well-separated traffic types, including Bot, DDOS attack-HOIC, and SSH-Bruteforce, where PCA effectively retained discriminative features. This study does not directly compare the ASpark-PCA approach with other Apache Spark-based IDS methodologies, as its primary objective is to evaluate the trade-offs between deep learning-based feature extraction (SAE) and distributed feature selection (ASpark). A direct comparison with alternative Spark-based implementations is beyond the scope of this work but presents an avenue for future research. Future studies could explore benchmarking ASpark-PCA against other Spark-integrated IDS approaches to further assess its performance in distributed network intrusion detection systems.

A key takeaway from this study is the complementary nature of the two approaches. The SAE approach, with its ability to learn non-linear feature representations using an autoencoder and MLP classifiers, is better suited for scenarios requiring high accuracy and robustness across diverse attack types. In contrast, the ASpark approach, which uses PCA, logistic regression, and random forest for computational efficiency, is more appropriate for real-time or large-scale applications where speed is critical. While both approaches exhibit strengths in different areas, their limitations highlight potential opportunities for improve-

ment. Future research could explore a combined framework that integrates the deep feature extraction capabilities of autoencoders with the efficiency of Apache Spark-based classifiers, potentially combining the binary classification power of SAE with a random forest-based multi-class classifier. Such an approach could strike a balance between accuracy and efficiency, optimizing both detection performance and real-time applicability. Additionally, further refinements in feature engineering and class imbalance handling could enhance the detection of challenging attack types, such as Infiltration, which were more difficult to classify in this study. Although a hybrid resampling strategy was employed to balance the dataset in the current study, incorporating advanced techniques such as SMOTE for rare attack types and near-miss downsampling for the benign class could further improve classification performance. Exploring alternative feature selection techniques beyond PCA and SAE could further improve model robustness. Furthermore, the models were evaluated using the CSE-CIC-IDS2018 dataset, which, while comprehensive, may not fully represent all possible real-world network conditions and attack variations. Future studies could explore the generalizability of these approaches by testing them on other contemporary intrusion detection datasets. This research provides actionable insights for network security professionals, researchers, and organizations developing intrusion detection systems. By leveraging modern datasets and advanced machine learning techniques, the findings contribute to designing systems that balance accuracy, scalability, and efficiency, addressing the growing complexity of cybersecurity threats in modern networks.

Funding: This research was supported by Research Grant RP/03/02/09/01/2022 from the University of Kelaniya.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Al-Jarrah, O.; Arafat, A. Network Intrusion Detection System Using Attack Behavior Classification. In Proceedings of the 2014 5th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 1–3 April 2014. [CrossRef]
2. Pillai, K. Network Intrusion Detection System—A Novel Approach. *J. Syst. Cybern. Inf.* **2013**, *11*, 6.
3. Musa, U.S.; Chhabra, M.; Ali, A.; Kaur, M. Intrusion Detection System Using Machine Learning Techniques: A Review. In Proceedings of the 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 10–12 September 2020. [CrossRef]
4. Khan, Z.M. Network Intrusion Detection Utilizing Autoencoder Neural Networks. *Commun. Appl. Nonlinear Anal.* **2024**, *31*, 336–354. [CrossRef]
5. Kamalov, F.; Moussa, S.; El Khatib, Z.; Mnaouer, A.B. Orthogonal Variance-Based Feature Selection for Intrusion Detection Systems. In Proceedings of the 2021 International Symposium on Networks, Computers and Communications (ISNCC), Dubai, United Arab Emirates, 31 October–2 November 2021; pp. 1–5.
6. Song, Y.; Hyun, S.; Cheong, Y.G. A Systematic Approach to Building Autoencoders for Intrusion Detection. In Proceedings of the Silicon Valley Cybersecurity Conference, San Jose, CA, USA, 17–19 December 2020; pp. 188–204.
7. Azzaoui, H.; Boukhamla, A. Two-Stages Intrusion Detection System Based on Hybrid Methods. In Proceedings of the 10th International Conference on Information Systems and Technologies, Lecce, Italy, 4–5 June 2020; pp. 1–7.
8. Muhammad, G.; Hossain, M.S.; Garg, S. Stacked Autoencoder-Based Intrusion Detection System to Combat Financial Fraudulent. *IEEE Internet Things J.* **2020**, *10*, 2071–2078. [CrossRef]
9. Zhang, R.; Chen, H. Intrusion Detection of Industrial Control System Based on Stacked Auto-Encoder. In Proceedings of the 2019 Chinese Automation Congress (CAC), Hangzhou, China, 22–24 November 2019; pp. 5638–5643.
10. Hossain, M.A.; Islam, M.S. Ensuring Network Security with a Robust Intrusion Detection System Using Ensemble-Based Machine Learning. *Array* **2023**, *19*, 100306. [CrossRef]

11. Belouch, M.; Hadaj, S.E. Comparison of Ensemble Learning Methods Applied to Network Intrusion Detection. In Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing, Cambridge, UK, 22–23 March 2017; pp. 1–4.
12. Ghurab, M.; Gaphari, G.; Alshami, F.; Alshamy, R.; Othman, S. A Detailed Analysis of Benchmark Datasets for Network Intrusion Detection System. *Asian J. Res. Comput. Sci.* **2021**, *7*, 14–33. [CrossRef]
13. Leevy, J.L.; Khoshgoftaar, T.M. A Survey and Analysis of Intrusion Detection Models Based on CSE-CIC-IDS2018 Big Data. *J. Big Data* **2020**, *7*, 104. [CrossRef]
14. Wang, L. Big Data in Intrusion Detection Systems and Intrusion Prevention Systems. *J. Comput. Netw.* **2017**, *4*, 48–55. [CrossRef]
15. Songma, S.; Sathuphan, T.; Pamutha, T. Optimizing Intrusion Detection Systems in Three Phases on the CSE-CIC-IDS-2018 Dataset. *Computers* **2023**, *12*, 245. [CrossRef]
16. Hagar, A.A.; Gawali, B.W. Apache Spark and Deep Learning Models for High-Performance Network Intrusion Detection Using CSE-CIC-IDS2018. *Comput. Intell. Neurosci.* **2022**, *2022*, 3131153. [CrossRef]
17. Kulariya, M.; Saraf, P.; Ranjan, R.; Gupta, G.P. Performance Analysis of Network Intrusion Detection Schemes Using Apache Spark. In Proceedings of the 2016 International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, 6–8 April 2016; pp. 1973–1977.
18. Saravanan, S. Performance Evaluation of Classification Algorithms in the Design of Apache Spark-Based Intrusion Detection System. In Proceedings of the 2020 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 10–12 June 2020; pp. 443–447.
19. Yogesh, K.; Karthik, M.; Naveen, T.; Saravanan, S. Design and Evaluation of Scalable Intrusion Detection System Using Machine Learning and Apache Spark. In Proceedings of the 2019 5th International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 19–21 September 2019; pp. 1–7.
20. Singh, R.; Kumar, H.; Singla, R.K.; Ketti, R.R. Internet Attacks and Intrusion Detection System: A Review of the Literature. *Online Inf. Rev.* **2017**, *41*, 171–184. [CrossRef]
21. Amarudin, R. A Systematic Literature Review of Intrusion Detection System for Network Security: Research Trends, Datasets and Methods. In Proceedings of the 2020 4th International Conference on Informatics and Computational Sciences (ICICoS), Semarang, Indonesia, 10–11 November 2020. [CrossRef]
22. Mishra, A.; Yadav, P. Anomaly-Based IDS to Detect Attack Using Various Artificial Intelligence & Machine Learning Algorithms: A Review. In Proceedings of the 2nd International Conference on Data, Engineering and Applications (IDEA), Bhopal, India, 28–29 February 2020. [CrossRef]
23. Suthishni, D.N.P.; Kumar, K.S.S. A Review on Machine Learning Based Security Approaches in Intrusion Detection System. In Proceedings of the 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 23–25 March 2022. [CrossRef]
24. Harwahyu, R.; Ndolu, F.H.E.; Overbeek, M.V. Three Layer Hybrid Learning to Improve Intrusion Detection System Performance. *Int. J. Electr. Comput. Eng.* **2024**, *14*, 1691–1699. [CrossRef]
25. Java, M.I.; Shabrina, U.I.; Wiliyanti, W.; Fahmi, R.N.; Pratomo, B.A. Enhancing Cybersecurity: Two-Phase Detection Approach for Intrusion Network for Anomaly Data. In Proceedings of the 2024 IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS), Bandung, Indonesia, 21–23 February 2024. [CrossRef]
26. Alshattnawi, S.; Alshboul, H.R. Combined Deep Learning Approaches for Intrusion Detection Systems. *Int. J. Interact. Mob. Technol.* **2024**, *18*, 144–155. [CrossRef]
27. Pansare, S.; Malik, A.; Batra, I. Hybrid Machine Learning Algorithm for Intrusion Detection Systems. In Proceedings of the 2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE), Gautam Buddha Nagar, India, 9–11 May 2024. [CrossRef]
28. Moukhafi, M.; Tantaoui, M.; Chana, I.; Bouazi, A. Intelligent Intrusion Detection Through Deep Autoencoder and Stacked Long Short-Term Memory. *Int. J. Electr. Comput. Eng.* **2024**, *14*, 2908–2917. [CrossRef]
29. S, P.K.; Shriyans, A.; Rajkumar, A.N.; Hariharan, S.; Kanthimathi, S. Improving Intrusion Detection System by Utilizing Stacking Based Convolutional Neural Network Ensemble Classifier. In Proceedings of the 2024 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 12–14 July 2024. [CrossRef]
30. Khan, F.A.; Gumaei, A.; Derhab, A.; Hussain, A. A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection. *IEEE Access* **2019**, *7*, 30373–30385. [CrossRef]
31. Cacchiarelli, D.; Kulahci, M. Hidden Dimensions of the Data: PCA vs Autoencoders. *Qual. Eng.* **2023**, *35*, 741–750. [CrossRef]
32. Thakkar, A.; Kikani, N.; Geddam, R. Fusion of Linear and Non-Linear Dimensionality Reduction Techniques for Feature Reduction in LSTM-Based Intrusion Detection System. *Appl. Soft Comput.* **2024**, *154*, 111378. [CrossRef]
33. Manjunatha, B.A.; Shastry, K.A.; Naresh, E.; Pareek, P.K.; Reddy, K.T. A Network Intrusion Detection Framework on Sparse Deep Denoising Auto-Encoder for Dimensionality Reduction. *Soft Comput.* **2023**, *28*, 4503–4517. [CrossRef]

34. Elmoutaoukkil, A.; Hamlich, M.; Khatib, A.; Chriss, M. Network Intrusion Detection in Big Datasets Using Spark Environment and Incremental Learning. *IAES Int. J. Artif. Intell.* **2024**, *13*, 4414–4421. [CrossRef]
35. Reddy, B.; Reddy, G.; Lokesh, K.; Venugopalan, M. SecureNet: A PySpark-Based Approach to Enhanced Network Intrusion Detection Using Machine Learning and Feature Engineering. In Proceedings of the 2024 4th International Conference on Intelligent Technologies (CONIT), Bangalore, India, 21–23 June 2024. [CrossRef]
36. Sarwath, A.; Gulmeher, R.; Sultana, Z. Spark-MLlib Intrusion Detection Mechanism Using Machine Learning Models. *Ind. J. Electr. Eng. Comput. Sci.* **2024**, *33*, 1235–1242. [CrossRef]
37. Ibrahimi, K.; Jouhari, M.; Jakout, Z. Enhancing Intrusion Detection Systems Using Machine Learning Classifiers on the CSE-CIC-IDS2018 Dataset. In Proceedings of the 2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM), Leeds, UK, 23–25 July 2024. [CrossRef]
38. K, R.S.; P, A.S.; Kumar, J. A Survey on Different Machine Learning Algorithms That Are Compatible with CSE-CIC IDS 2018 Dataset. *Int. J. Multidiscip. Res.* **2024**, *6*. [CrossRef]
39. Göcs, L.; Johanyák, Z.C. Identifying Relevant Features of CSE-CIC-IDS2018 Dataset for the Development of an Intrusion Detection System. *Intell. Data Anal.* **2024**, *28*, 1527–1553. [CrossRef]