This dataset is chosen for this research, because it is one of the most recent and essential datasets in the IDS field. It is widely used in other studies; hence, the comparison can be valid. Moreover, since the research focuses on using IDS with large datasets, this dataset is a convenient choice.

3.2 Random Forest

The first supervised machine-learning technique used in the proposed ensemble model is the random forest (RF). The RF is a robust machine-learning model that reduces overfitting and performs efficiently on large datasets. The RF randomly divides the dataset's features into sub-sets of features, where each sub-set is trained using the decision-tree model separately and independently of other sub-trees. In the training process, each dataset sample is trained in a particular sub-tree, while in the testing, the entire test data is trained in each sub-tree. The final result is aggregated by producing an average of these results. The decision-tree model consists of nodes and branches. At each node, evaluate the sub-set of features to generate and divide the observations into other nodes in the training set or to flow to a specific path when making a prediction.

3.3 eXtreme Gradient Boosting (XGBoost)

The second robust algorithm that handles the bias-variance trade-offs and provides a parallel tree boosting for classification is the XGBoost model. Like the RF algorithm, the XGBoost model uses gradient boosting, providing adaptation and generalization. It is considered an ensemble of multiple learners, such as decision trees, where the final decision is an ensemble of the subtrees' outputs. Consequently, the XGBoost prevents poor performance. XGBoost uses the gradient descent algorithm to optimize the model by updating weight and reducing cost value and the discrepancy between the expected and actual values. The mean squared error (MSE) cost function is used as an evaluation metric for classification tasks.

In the next section, the experiments are discussed by starting with the data pre-processing phase, then concluding the results of experiments with and without Spark, along with an evaluation and discussion of these results.

4. EXPERIMENT RESULTS

4.1 Data Pre-processing and Experiment Setup

Data pre-processing is crucial in the data analysis and machine-learning pipeline. It contributes to data quality, integrity and suitability for modeling, leading to more accurate and reliable results.

As aforementioned, the CSV files selected from the CIC-DDoS2019 dataset for evaluation in the proposed model were UDP.csv and SYN.csv. The UDP file consists of three classes: UDP attack, MSSQL attack and benign. At the same time, the SYN file consists of two classes: SYN attack and benign. These two files are under exploitation attacks, as shown in Figure 1. Table 2 represents the number of samples for each class before and after the pre-processing phase. The data pre-possessing consists of the following steps:

1) Eliminating duplicate records using drop_doplicate function provided by Python, where the number of samples for SYN and Benign classes is reduced from 4,284,751 to 3,806,356 and from 35,790 to 31,386, respectively, as shown in Table 2. However, there were no duplicate records for UDP and MSSQL. This step is essential, as it helps maintain data quality and consistency. Duplicate records can skew analysis results, leading to biased models or overfitting problems.
2) Eliminating attributes with summation and variances of zero can lead to more efficient, generalizable models and it is a common data pre-processing step. These attributes are: Bwd Packet Length Std, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, FIN Flag Count, PSH Flag Count, ECE Flag Count, Fwd Avg Bytes/Bulk, Fwd Avg Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk, Bwd Avg Bulk Rate and Similar HTTP. This step reduces the number of features from 88 to 74.
3) Replacing infinity and null values with zeros using replace(nan, 0) function provided by numpy module in Python.

4) Encoding categories including source IP, flow ID, destination IP and timestamp using LabelEncoder function provided by sklearn module in Python. This step is crucial, since many machine-learning algorithms work with numerical data; so, categorical variables must be encoded into a suitable numerical format.

Table 2. The number of records for each class.

Class	Samples count before pre-possessing	Samples count after pre-possessing
UDP attack	3754680	3754680
MSSQL attack	24392	24392
SYN attack	4284751	3806356
Benign (no attack)	38924	34520

The dataset was split randomly into 80% for training and 20% for testing. Consequently, the number of records in training and testing sets are 6095958 and 1523990, respectively. Furthermore, the default parameters of the Random Forest and XGBoost models were determined as follows: The number of sub-trees of both models is 100 and the weak learner in the XGBoost is the decision trees. Table 3 and Table 4 show the hyper-parameters of RF and XGBoost used to train the model.

Table 3. Random forest hyper-parameters.

RF hyper-parameter	Value
Number of trees.	100
Function to measure the quality of a split.	gini
The minimum number of samples required to split an internal node.	2
The minimum number of samples required to be at a leaf node.	1

Table 4. XGBoost hyper-parameter.

XGboost hyper-parameter	Value
Number of boosting rounds.	100
Loss function	Squared error
Boosting model (weak learner)	Decision trees (gbtree)
The feature-importance type	Information gain
Maximum depth	6

4.2 Performance Metrics

To evaluate the proposed model, four metrics were utilized: accuracy, precision, recall and time. The assessment was conducted both with and without the Spark platform. Accuracy indicates the percentage of correctly classified instances out of the total number of cases evaluated, as per Equation 1, while time denotes the training time in minutes. Meanwhile, the ensemble model employing Apache-Spark was evaluated through a classification report that includes precision, recall and F1-Score. The formula for calculating the measures are presented in Equations 1-3.

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

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

$$\text{Recall} = \text{Precision.Recall/Precision+Recall} \quad (3)$$

TN stands for True Negative, FP for False Positive, TP for True Positive and FN for False Negative. These measures were utilized due to data imbalance, with F1-Score being commonly used to evaluate IDSs, as it combines precision and recall and provides valuable insights into the study outcomes.

The value of each measurement ranges from 0 to 1, with higher values indicating a more robust model that is better suited for detecting possible intrusions. Time measures were utilized to compare the performance of the proposed model with and without Spark, highlighting the value added by Spark usage. With Spark, the required times for building and training the model, as well as for running the detection model, are significantly reduced.

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 10, No. 02, June 2024.

4.3 Experiments and Results

Two experiments have been conducted. Apache-Spark's distributed computing capabilities make it a valuable tool for handling large datasets. Its scalability, in-memory processing and distributed model enable it to process vast amounts of data efficiently. However, one of its potential limitations is the resource requirements. As a result, both experiments have been implemented on Colab Pro with (25 GB) RAM; the first experiment was done without Apache-Spark by applying the RF and XGBoost ensemble model.

The same model with the same environment has been applied again but with the use of Apache-Spark using the PySpark library, considered an open-source interface for Apache-Spark. It allows SQL-like analysis on large amounts of structured or semi-structured data. Figure 4 and Table 5 illustrate the training time needed to conduct Random Forest and XGBoost separately, once under the Apache-Spark environment and the other without using it.

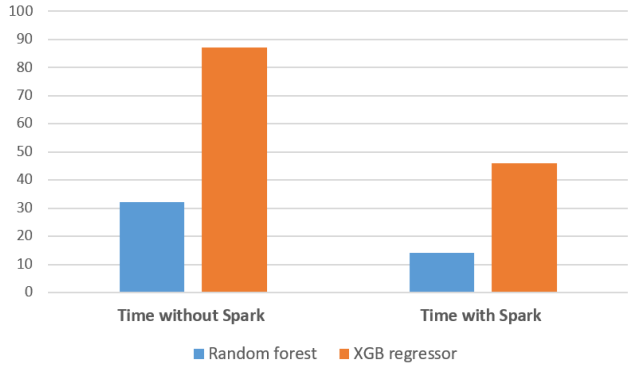

Figure 4. The training time required to train random forest and XGBoost models with/without Spark.

Table 5. The training time using the CIC-DDOS 2019 dataset.

Model	Training time without Spark in minutes	Training time with Spark in minutes
Random forest	32	14
XGB regressor	87	46

Table 6 illustrates the accuracies from conducting Random Forest, XGBoost and the proposed ensemble model, either with Spark or without it. It can be noticed how the results are close in both cases. As for Figure 5, the Recall, precision and F1-score measures are compared for classes SYN, MSSQL, UDP and Normal when conducting the proposed ensemble model.

Table 6. The accuracy using the CIC-DDOS 2019 dataset.

Model	Accuracy without Spark (%)	Accuracy with Spark (%)
Random forest	99.9995	99.9155
XGB regressor	99.9762	99.9942
Ensemble	99.94	99.9419

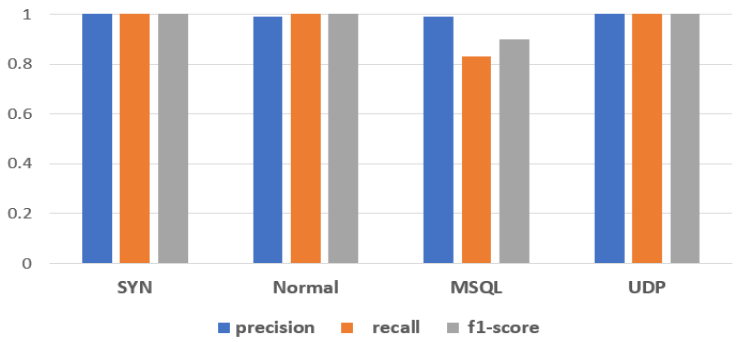

Figure 5. The performance of the proposed ensemble model for each class.

Table 7 conducts a comparison between selected proposed models in the literature. The comparison has been conducted regarding accuracy, F1-score, number of classes predicted and whether Spark is used or not. In reference [29], the F1-measure was calculated by averaging the F1-scores for the seven classes mentioned in that work.

Table 7. Comparison between the proposed model and literature.

Model	Accuracy (%)	F1-score	No. of classes	Spark
Extreme Gradient Boosting (XGBoost) [12]	99.7	99.79	two classes	-
XGBoost [13]	99.7	100	two classes	-
XGBoost [13]	91.26	92	multi-class	-
FEDFOREST (L+L) [16]	67.03	94.60	two classes	-
Random forest [28]	99.99	-	two classes	-
Parallel NB, DT and RF [20]	61.4, 97.9 and 97.3	51.3, 97.9 and 9.73	two classes	✓
Random forest [29]	89	89	seven classes	✓
CNN [32]	94.21	94.12	seven classes	-
Proposed (ensemble -RF and XGBoost-)	99.94	97.5	four classes	✓

The code that we have written and used throughout this study is publicly available in [36].

4.4 Discussion

The accuracy of the proposed model has been one of many concerns during the extraction of experiment results. The training time has been considered one vital criterion to concentrate on to prove our work's high performance and validity.

Table 3 and Figure 4 show that the time needed to train either a Random Forest or an XGBoost model has been reduced to about a half when using Spark. The required time to train Random Forest and XGBoost using Spark is reduced by 18 and 41 minutes, respectively. This time reduction makes the model more efficient, since detecting DDoS attacks is usually deployed in sensitive applications where time matters. Expanding the required time to train and use the model makes it less reliable and less usable in such applications.

From Table 4, it can be noted that the proposed ensemble model achieved a high accuracy regardless of whether or not Spark was used. Table 4 proves that even splitting the dataset in a way that Spark can use did not affect the accuracy of the proposed model. Thus, the model's performance has been enhanced while preserving its accuracy.

Figure 5 illustrates that both RF and XGBoost, working as a stacked ensemble model, can successfully distinguish the SYN, UDP, MSQL and benign. However, Recall and F1-score have been slightly reduced when using Spark in MSSQL attack. This is because of the number of its samples, which is considered small compared with the rest of the classes, as represented in Table 2. Thus, splitting and training small samples may yield less F1-score for such classes.

The proposed model achieves the best accuracy when compared with other models in the literature. Table 5 shows that the proposed model exceeds the existing models in terms of accuracy when the model is trained in more than two classes. It also outperformed the accuracy of models that classified only two classes, considering that binary classification is usually more accessible than multi-class classification.

Overall, the combination of stacked ensemble models under the Apache-Spark environment outperformed other models described in the literature.

5. CONCLUSION

This work proposed a distributed ML model for DDoS attack detection using Apache-Spark. Ensemble learning was used to build a robust and efficient IDS. The system can detect three DDoS attacks: UDP, MSSQL and SYN. The model comprises two trusted ML models: Random Forest and

XGB regressor. Hence, it can be considered a stacked ensemble model. Apache-Spark was used to train data distribution in parallel using the proposed model. Data distribution using Spark has enhanced the required time to train the model, as the time was reduced to around a half. At the same time, the accuracy was preserved at a level of over 99%. The required training time for the XGBoost regression model was reduced from 87 minutes to 46 minutes when Spark was used. The required training time for the Random Forest model was reduced from 32 to 14 minutes when Spark was used. The time reduction comes at the cost of increasing the used RAMs, which is considered the main limitation of the proposed approach in this work. This limitation can be avoided by using computers with large RAM capacity.

The presented methodology can be considered an efficient IDS approach that can be used with DDoS attacks, such as the attacks the data of which is recorded in the CIC-DDOS2019 dataset. The presented distributed model is also robust and fast. Hence, it can be used when online intrusion-detection models are not affordable, complicated or down. Using an ensemble IDS of XGBoost and Random Forest with Apache-Spark has proved to be an easily built and trained model. The approach guarantees short training time and robustness against failure; when one distributed node in the Apache-Spark platform is down, the other nodes are available and a replacement node can take over the failed one.

In the future, more ML models will be added to the distributed ensemble model for the DDoS IDS by editing the used code and adding more slaves (nodes) to the ensemble model. Furthermore, the work will be expanded to detect other intrusions by training the model with datasets related to other attacks.

REFERENCES

[1] J. S. Ward and A. Barker, "Undefined by Data: A Survey of Big Data Definitions," arXiv preprint, arXiv: 1309.5821, 2013.

[2] M. I. Jordan and T. M. Mitchell, "Machine Learning: Trends, Perspectives and Prospects," Science, vol. 349, no. 6245, pp. 255-260, 2015.

[3] G. Kaur and M. Jain, (2020), "A Comparison of Two Blending-based Ensemble Techniques for Network Anomaly Detection in Spark Distributed Environment," International Journal of Ad Hoc and Ubiquitous Computing, vol. 35, no. 2, pp. 71-83, 2020.

[4] M. Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," Communications of the ACM, vol. 59, no. 11, pp. 56-65, 2016.

[5] I. Sharafaldin, A. H. Lashkari, S. Hakak and A. A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," Proc. of the 2019 IEEE Int. Carnahan Conf. on Security Technology (ICCST), pp. 1-8, Chennai, India, 2019.

[6] S. Manickam et al., "Labelled Dataset on Distributed Denial-of-Service (DDoS) Attacks Based on Internet Control Message Protocol Version 6 (ICMPv6)," Wireless Communications and Mobile Computing, vol. 2022, Article ID 8060333, DOI: 10.1155/2022/8060333, 2022.

[7] T. H. Chua and I. Salam, "Evaluation of Machine Learning Algorithms in Network-based Intrusion Detection Using Progressive Dataset," Symmetry, vol. 15, no. 6, p. 1251, 2023.

[8] B. I. Farhan and A. D. Jasim, "Performance Analysis of Intrusion Detection for Deep Learning Model Based on CSE-CIC-IDS2018 Dataset," Indonesian Journal of Electrical Engineering and Computer Science, vol. 26, no. 2, pp. 1165-1172, 2022.

[9] A. Elhanashi, K. Gasmi, A. Begni, P. Dini, Q. Zheng and S. Saponara, "Machine Learning Techniques for Anomaly-based Detection System on CSE-CIC-IDS2018 Dataset," Proc. of the Int. Conf. on Applications in Electronics Pervading Industry, Environment and Society (ApplePies 2022), Part of the Lecture Notes in Electrical Engineering Book Series, vol. 1036, pp. 131-140, Springer, 2022.

[10] I. F. Kilincer, F. Ertam and A. Sengur, "A Comprehensive Intrusion Detection Framework Using Boosting Algorithms," Computers and Electrical Engineering, vol. 100, p. 107869, 2022.

[11] R. Atefinia and M. Ahmadi, "Performance Evaluation of Apache Spark MLlib Algorithms on an Intrusion Detection Dataset," arXiv preprint, arXiv: 2212.05269, 2022.

[12] P. H. H. N. de Araujo et al., "Impact of Feature Selection Methods on the Classification of DDoS Attacks using XGBoost," Journal of Communication and Information Systems, vol. 36, no. 1, pp. 200-214, 2021.

[13] H. A. Alamri and V. Thayananthan, "Bandwidth Control Mechanism and Extreme Gradient Boosting Algorithm for Protecting Software-defined Networks against DDoS Attacks," IEEE Access, vol. 8, pp. 194269-194288, 2022.

[14] H. A. Alamri and V. Thayananthan, "Analysis of Machine Learning for Securing Software-defined Networking," Procedia Computer Science, vol. 194, pp. 229-236, 2021.

[15] R. Zhou, X. Wang, J. Yang, W. Zhang and S. Zhang, "Characterizing Network Anomaly Traffic with

Euclidean Distance-based Multiscale Fuzzy Entropy," Security and Communication Networks, vol. 2021, Article ID 5560185, DOI: 10.1155/2021/5560185, 2021.

[16] T. Dong, S. Li, H. Qiu and J. Lu, "An Interpretable Federated Learning-based Network Intrusion Detection Framework," arXiv preprint, arXiv: 2201.03134, 2022.

[17] N. Ahuja, G. Singal, D. Mukhopadhyay and N. Kumar, "Automated DDOS Attack Detection in Software Defined Networking," Journal of Network and Computer Applications, vol. 187, p. 103108, DOI: 10.1016/j.jnca.2021.103108, 2021.

[18] M. I. Mohmand et al., "A Machine Learning-based Classification and Prediction Technique for DDoS Attacks," IEEE Access, vol. 10, pp. 21443-21454, 2022.

[19] T. G. Zewdie and A. Girma, (2022), "An Evaluation Framework for Machine Learning Methods in Detection of DoS and DDoS Intrusion," Proc. of the 2022 IEEE Int. Conf. on Artificial Intelligence in Information and Communication (ICAIIC), pp. 115-121, Jeju Island, Korea, 2022.

[20] A. Alsirhani, S. Sampalli and P. Bodorik, (2018), "DDoS Attack-detection System: Utilizing Classification Algorithms with Apache Spark," Proc. of the 2018 9th IEEE IFIP Int. Conf. on New Technologies, Mobility and Security (NTMS), pp. 1-7, Paris, France, 2018.

[21] A. Alsirhani, S. Sampalli and P. Bodorik, "DDoS Detection System: Utilizing Gradient Boosting Algorithm and Apache Spark," Proc. of the 2018 IEEE Canadian Conf. on Electrical & Computer Engineering (CCECE), pp. 1-6, Quebec, Canada, 2018.

[22] C. J. Hsieh and T. Y. Chan, (2016), "Detection DDoS Attacks Based on Neural-Network Using Apache Spark," Proc. of the 2016 IEEE Int. Conf. on Applied System Innovation (ICASI), pp. 1-4, Okinawa, Japan, 2016.

[23] K. Kato and V. Klyuev, "Development of a Network Intrusion Detection System Using Apache Hadoop and Spark," Proc. of the 2017 IEEE Conf. on Dependable and Secure Computing, pp. 416-423, Taipei, Taiwan, 2017.

[24] M. Jain and G. Kaur, "Distributed Anomaly Detection Using Concept Drift Detection Based Hybrid Ensemble Techniques in Streamed Network Data," Cluster Computing, vol. 24, pp. 2099-2114, 2021.

[25] 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, 2020.

[26] B. Zhou, J. Li, J. Wu, S. Guo, Y. Gu and Z. Li, "Machine-learning-based Online Distributed Denial-of-Service Attack Detection Using Spark Streaming," Proc. of the 2018 IEEE Int. Conf. on Communications (ICC), pp. 1-6, Kansas City, USA, 2018.

[27] M. J. Awan et al., "Real-time DdoS Attack Detection System Using Big Data Approach," Sustainability, vol. 13, no. 19, p. 10743, 2021.

[28] M. Alduailij, Q. W. Khan, M. Tahir, M. Sardaraz, M. Alduailij and F. Malik, "Machine-learning-based DdoS Attack Detection Using Mutual Information and Random Forest Feature Importance Method," Symmetry, vol. 14, no. 6, p. 1095, 2022.

[29] N. V. Patil, C. R. Krishna and K. Kumar, "SSK-DdoS: Distributed Stream Processing Framework Based Classification System for DdoS Attacks," Cluster Computing, vol. 25, no. 2, pp. 1355-1372, 2022.

[30] C. S. Shieh et al., "Detection of Unknown DdoS Attacks with Deep Learning and Gaussian Mixture Model," Applied Sciences, vol. 11, pp. 11, p. 5213, 2021.

[31] D. Alghazzawi, O. Bamasag, H. Ullah and M. Z. Asghar, "Efficient Detection of DdoS Attacks Using a Hybrid Deep Learning Model with Improved Feature Selection," Applied Sciences, vol. 11, no. 24, p. 11634, 2021.

[32] A. Chartuni and J. Márquez, "Multi-classifier of DdoS Attacks in Computer Networks Built on Neural Networks," Applied Sciences, vol. 11, no. 22, p. 10609, 2021.

[33] Y. Yilmaz and S. Buyrukoglu, "Development and Evaluation of Ensemble Learning Models for Detection of Distributed Denial-of-Service Attacks in Internet of Things," Hittite Journal of Science & Engineering, vol. 9, no. 2, pp. 73-82, 2022.

[34] M. Seydali, F. Khunjush and J. Dogani, "Streaming Traffic Classification: A Hybrid Deep Learning and Big Data Approach," Cluster Computing, DOI: 10.1007/s10586-023-04234-0, 2024.

[35] S. M. S. Bukhari et al., "Secure and Privacy-preserving Intrusion Detection in Wireless Sensor Networks: Federated Learning with SCNN-Bi-LSTM for Enhanced Reliability," Ad Hoc Networks, vol. 155, p. 103407, 2024.

[36] "ColabCode," [Online], Available: Https://Colab.Research.Google.Com/Drive/1oZu2czCK9tJSwcjEfyvLiZnrI0JqYW62?Usp=Sharing, December 22, 2023.

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 10, No. 02, June 2024.

ملخص البحث:

إنّنا نعيش في عَصْرٍ يُعَدّ فيه الوقت المورِد الأغلى. لذا، فإنّ التّعاملَ مع الكمّ الهائل من البيانات التي تُجمّع من مصادر مختلفة لأغراض مختلفة يتطلّب إيجاد أنظمةٍ يمكنها معالجة البيانات بشكلٍ صحيح يجعلها ذات معنى. وإنّ استخدام البيانات الضّخمة في نماذج تعلُّم الآلة والذّكاء الاصطناعي من شأنه أن يُحسِّن فعالية تلك النّماذج ومتانتها.

تقترح هذه الورقة نموذجاً لكشْف الهجمات الموزّعة المتعلّقة برفض الخدمة (DDoS) باستخدام مجموعة بيانات عامّة معروفة لتدريب النّموذج. ويتمّ تدريب النّموذج بحيث يُمكنه توقُّع نوع الهجمة من بين أنواع متعدّدة من الهجمات يستخدمها المخترقون. وتُستخدم اثنتان من الخوارزميات الواردة في أدبيات الموضوع بوصفهما أساسَ النّموذج المقترح. وتستمدّ هاتان الخوارزميتان فعاليتهما ومتانتهما من الطبيعة المجمَّعة لتركيبهما، حيث تتكون كلٌّ منهما من عددٍ من شجرات القَرار (decision trees) بمتغيرات مختلفة. وقد جرى بناء نظام مجمَّع باستخدام الخوارزميتين من أجل تحسين دقّة كشْف الهجمات؛ واتّضح أنّ استخدام تلك التركيبة المجمَّعة يعطي أفضل النتائج.

من ناحيةٍ أخرى، فإنّ طول زمن التّنفيذ المترتِّب على تدريب النّموذج باستخدام مجموعة بيانات ضخمة يُعدّ مسألةً أخرى لا بدّ من أخذها بعين الاعتبار. ولتسريع عمل النّموذج، فقد تمّ استخدام "الشّرارة" (Apache-Spark) الّتي يؤدي استخدامها إلى تجزئة مجموعة البيانات ومعالجة تلك الأجزاء بالتّوازي، الأمر الّذي يقلّل زمن التّنفيذ مع المحافظة على دقّة كشْف الهجمات.

لقد حقق النّموذج المقترح دقّةً وصلت إلى 99.94%، وقلّل استخدام الشّرارة زمن التّنفيذ إلى ما يقرب من النّصف مقارنة بعدم استخدام الشّرارة. وبالمقارنة مع عددٍ من نماذج كشْف الهجمات الواردة في أدبيات الموضوع، تبيّن أنّ تلك النّماذج حققّت دقّة كشفٍ أقلّ من النّموذج المقترح في هذه الدّراسة.