Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. **68**, *3*, pp. 301–306, Bucarest, 2023

INCREMENTAL LEARNING FOR EDGE NETWORK INTRUSION DETECTION

ALINA FLORINA GLAVAN[1], VICTOR CROITORU[2]

Keywords: Edge; Supervised learning; Incremental learning; Intrusion detection.

The paper presents incremental learning as a solution for adapting intrusion detection systems to edge network conditions. Extreme gradient boost trees are proposed and evaluated with the Network Security Laboratory - Knowledge Discovery in Databases (NSL-KDD) benchmark dataset. The accuracy of the XGBoost classifier model improves by 15 % with 1 % of the KDD-test+ data used for training. A mechanism based on unsupervised learning that triggers retraining of the XGBoost classifier is suggested. These results are relevant in model retraining on resource-scarce environments (relative to a cloud environment), such as the network edge or edge devices.

1. INTRODUCTION

The evolution of 5G networks is facilitated by technologies like network function virtualization (NFV), multi-access edge computing (MEC), software-defined networks (SDN), and network slicing (NS) [1]. MEC is considered a core 5G technology that can lighten the load on the mobile core network by serving requests locally [2], considering the growing number of connected devices. Paper [1] estimates Internet of Things (IoT) devices number will increase during 2019 – 2030. The number of IoT-connected devices in 2023 is estimated at 15.1 billion and is expected to reach as much as 29.4 billion devices by 2030 [1]. Figure 1 presents the positioning of edge computing in the network.

Nencioni et al. [3] appreciate MEC by 3 criteria: security, dependability, and performance. MEC is the closest network layer to customers that offers computing capabilities. At the same time, it represents a wide attack surface to control. The MEC vulnerabilities presented in papers [2,4] are related to:

- inherited vulnerabilities of the applications deployed at the edge and the shared host platform;
- physical security, due to geographically distributed locations;
- known vulnerabilities of micro-service architecture.

Regarding the types of attacks, paper [3] and [4] mentions eavesdropping, data capture, denial of service (DoS), partial or total physical damage, malware, credential theft, and rogue/fake entities.

The main cybersecurity controls at the network edge, with significant cost benefits, are the deployment of firewalls and intrusion detection controls [5]. As the number of deployed use cases rises, paper [5] reports an increased interest in implementing network cybersecurity controls. The top two are privileged access management (PAM) control and intrusion detection and prevention control. Paper [5] ranks industrial Internet of things/operational technology (IIoT/OT) edge at the top edge types for implementation in the next 3 years. In an edge setup, ensuring availability in industrial IoT/OT is critical, and distributed denial of service (DDoS) attacks are the first concern of respondents from report [5]. Figure 2 presents an IIoT configuration with a firewall and an intrusion detection system (IDS) deployed in the edge network.

Intrusion detection systems (IDSs) are used to monitor network traffic to detect anomalies and signs of misuse [6]. IDS is a commonly implemented countermeasure, and attackers constantly evolve their methods to compromise the network – with new attack methods, IDSs must also evolve.

Artificial intelligence/machine learning (AI/ML) for intrusion detection is being studied by many researchers in various topics [6–10]. Usually, training and evaluating a machine learning model requires high computational power and storage capacity. Considering the large volume of traffic entering the edge network, implementing machine learning-based IDSs at the edge can be challenging.

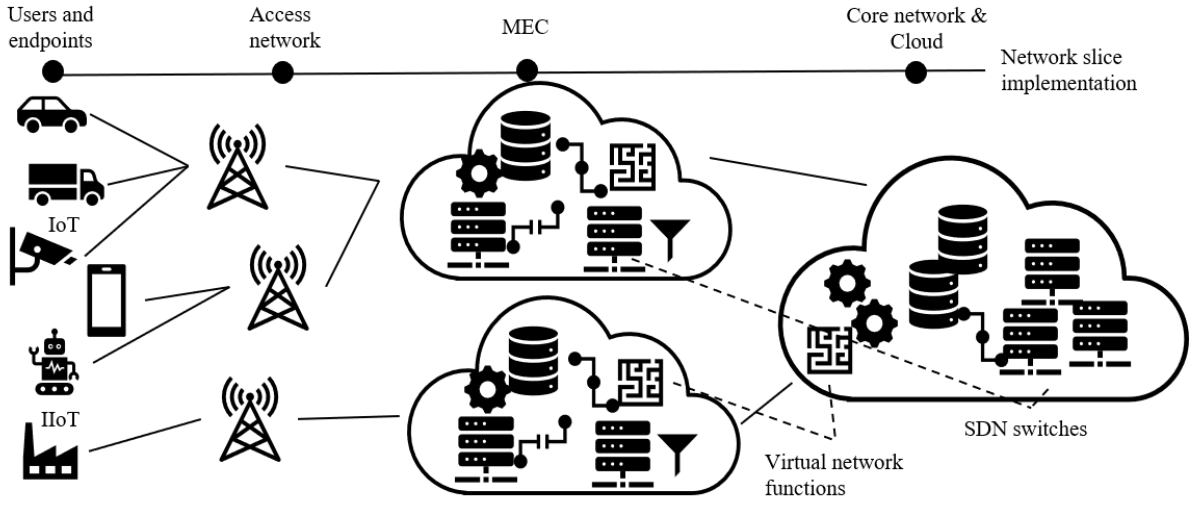

Fig. 1 – 5G network technologies.

[1] University Politehnica of Bucharest, Bucharest, Romania. E-mails: alinafglavan@gmail.com, croitoru@adcomm.pub.ro
DOI: 10.59277/RRST-EE.2023.68.3.9

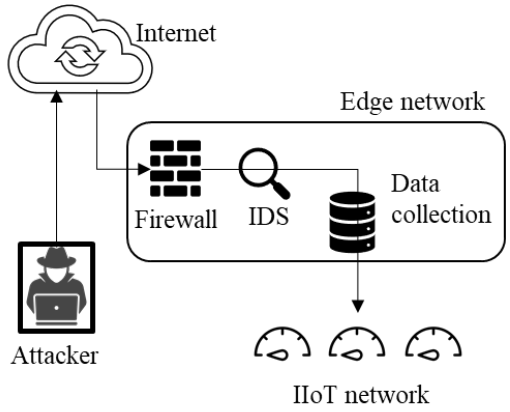

Fig. 2 – IoT network protected by firewall and IDS deployed at the edge.

Training and testing are mostly performed in high-performance environments like the cloud. Although the edge is marketed as a cloud, edge resources are limited [11].

This paper studies incremental learning as a solution for keeping edge-deployed detection models up to date with minimal resource consumption. The experimental setup considers the network security laboratory - knowledge discovery in database (NSL-KDD) dataset. The experiment aims to solve a classification task using supervised learning in an evolving environment. Considering the limited resources at the edge and the large amount of data, a scalable and performant model is considered for incremental learning. A gradient boost tree, specifically extreme gradient boost (XGBoost), is proposed as the machine learning algorithm. As incremental learning needs a trigger, an algorithm based on unsupervised learning is proposed to trigger retraining. K-means and silhouette coefficient are parts of the suggested triggering mechanism architecture.

The next part of the paper is organized as follows: the second chapter introduces incremental learning and NSL-KDD datasets, and a short review of related work is presented. Chapter three presents the experimental setting, providing details on the dataset preprocessing, machine learning algorithm, and environment. Chapter four is a dive into the training and testing results. The incremental learning results are explained by feature importance analysis. A machine learning-based mechanism to trigger the model updating is proposed in chapter five. Conclusions and future work are discussed in the final chapter.

2. CURRENT STATE OF AI AND INTRUSION DETECTION RESEARCH

AI/ML solutions are suggested to address a wide range of topics. Among them, disease prediction is studied in [12,13] ML-based diagnosis for battery cells is suggested in [14], and an evaluation of ML models for network intrusion detection is presented in [15,16]. The design and deployment of ML solutions for and at the network edge are studied in [17,18]. The last two topics are presented in more detail in the next subchapters.

2.1 NSL-KDD DATASET FOR INTRUSION DETECTION

The NSL-KDD is an intrusion detection dataset and represents the cleaned version of KDD CUP 99 dataset [19]. The NSL-KDD dataset is available in [20]—the authors of [19] present NSL-KDD as a benchmark dataset. One of the reasons is that the train and test data are from different probability distributions, and the test data has examples of new attack types (compared to the training dataset) [19]. Still, the examples can be classified into 5 categories: normal traffic, DDoS attack, probing attack, privilege escalation attacks (user to root – U2R), and remote access attacks (remote to local – R2L) [19].

The dataset was used in the training of various models. Paper [15] presents the dataset structure. The paper proposes feature selection for accuracy enhancement of SVM (support vector machine), J48, and Naïve Bayes.

The paper's authors [21] employ unsupervised autoencoders and 2 isolation forest modules to perform binary classification of examples for the fog environment.

Paper [16] presents a solution for anomaly detection based on bidirectional long short-term memory and attention mechanism. long short-term memory (LSTM), convolutional neural networks (CNN), and recurrent neural networks (RNN) are compared in [22]. Synthetic minority oversampling technique (SMOTE) is proposed as a preprocessing step for accuracy improvement.

Wang et al. [18] use NSL-KDD for few-shot class-incremental learning evaluation in intrusion detection. The model aims to identify the new attack classes from the test dataset.

2.2 INCREMENTAL LEARNING

Incremental learning is a way of performing stateful training [23]. Stateful training means that once the model is trained on a data set, the model can enhance and adapt to new environmental conditions by continuing training on a recent relevant data set. Training continuation can be scheduled or triggered by a performance decrease or data distribution shift [23]. Web article [23] mentions a reduction of training costs by a factor of 45 for a specific use case with stateful daily retraining compared to retraining on the whole dataset. Continual learning is viewed as an evolution of incremental learning [23].

BrainyEdge [17] is a solution for keeping edge-deployed models relevant in evolving IoT environments. Transfer learning and incremental learning are considered for better adaptation to context and resource usage.

Paper [24] presents transparent learning as a solution for training deep learning models at the network edge. Each edge node's computing capability is considered to share the computing burden of model training.

Paper [18] proposes few-shot class-incremental learning to adapt to a rapidly evolving environment in case of intrusion detection.

3. MACHINE LEARNING MODEL WORKFLOW

The next section describes the dataset preprocessing and the machine learning algorithm as part of the model development workflow.

3.1 DATASET PREPROCESSING

Table 1 presents a summary of the train and test dataset. The table resulted from dataset exploration.

The preprocessing is performed on a train and test concatenated dataset. The feature “level” is not interesting and is dropped during preprocessing. The label will be encoded by categories (values from 0 to 4) and then by traffic

type (normal = 0, attack =1). Feature "su_attempted" has 3 values, although it is a binary feature [15]: value 2 is assumed as 0, and the change affects 62 rows. Feature "num_outbound_cmds" has one value only, which is dropped [25]. One-hot-encoding is used for encoding categorical features "protocol_type", "flag", and "service". This process adds 81 features to the dataset. Then, all features in the dataset (except for the label) are transformed with a *scikit* MinMax scaler. The default parameters are assumed for MinMax scaler. The dataset is split back into train and test, each containing the same examples as the original datasets. No other filter was performed on the datasets.

Table 1

Summary of NSL-KDD (source: designed by authors)

Dataset	Total number of				
	Records	Traffic types	Categories	Normal class	Attack class
KDD Train+	125972	23	5	67342	58630
KDD Test+	22543	38	5	9711	12832

3.2 MACHINE LEARNING MODEL

XGBoost is an open-source library [26] of supervised algorithms. XGBoost is based on gradient-boosting decision trees, which perform well on large amounts of data [27]. It is considered efficient and scalable and does not need to optimize parameters or tuning [27]. Still, the implementation permits the control of overfitting with the introduction of penalties on each tree's weights and biases. It is considered to perform better than other algorithms [27]. XGBoost classifier allows the usage of a parameter named *xgb_model*. The parameter value is given by the model that is to be loaded before training and allows training continuation [28]. XGBClassifier algorithm was considered in the current paper because it simplifies the machine learning pipeline from data exploration to data classification.

3.3 IMPLEMENTATION SETUP

Jupyter Lite [29] is used for quick environment setup. Python libraries are used for data handling (numpy, pandas) and figure generation (matplotlib, seaborn).

The xgboost library is used for algorithm implementation and sklearn for preprocessing, train-test split, and evaluation metrics.

Experiments were performed on a personal computer with the following specifications: intel I7 9700K processor and 32GB DDR4 1667mhz random access memory.

4. INCREMENTAL LEARNING RESULTS AND ANALYSIS

4.1 INITIAL TRAINING

XGBClassifier is loaded from the xgboost library – this is a classification class compatible with scikit-learn application programming interface (API) [30]. Sklearn library [31] is used for performance metrics extraction. The default parameters are considered for the implementation of the model.

We are implementing a binary classification for identification of normal and attack traffic. We consider training the model on the whole train dataset, followed by evaluation on the test dataset. The incremental actualization of

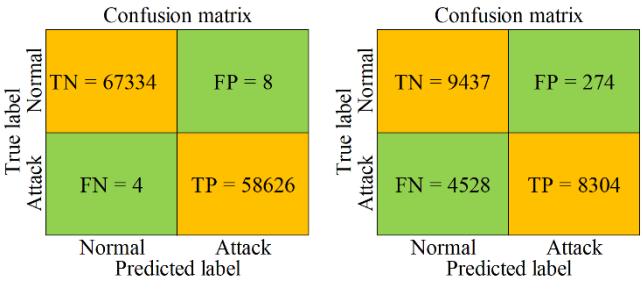

Fig. 3 – Model performance on KDD Train+ data (left) and KDD Test+ data (right) with confusion matrix metric.

the model is presented in subsection 4.2 and uses multiple batch sizes from the test dataset to update the model. The purpose is to detect the smallest number of samples needed to increase the old model's accuracy and observe accuracy evolution. The confusion matrix for the model was designed and presented in Fig. 3.

The matrix on the left represents the model's fit on train data. The matrix on the right represents the evaluation of the model on test data. The negative class is the normal traffic class, and the positive class is the attack class. True negative (TN) represents the negative examples (normal traffic) and is predicted negatively. True positive (TP) is the number of positive examples (attack traffic) and predicted positive. False negative (FN) is the number of positive examples predicted as negative, while false positive (FP) is the number of negative examples predicted as positive.

Accuracy is the chosen metric, offering an overall classification performance evaluation. Inaccurate predictions are considered to have the same cost. Accuracy is "the ratio between the number of correct predictions to the total number of predictions" [32].

The model performs well on train data, with an accuracy of 99 %. Still, the accuracy drops to 78.7 % at testing time, with many FN (attacks that the model omits). As this is a supervised learning algorithm, the model is expected to perform well on data from the same distribution. It is already known that the test data is generated with a different probability distribution function [19]. Since the probability distribution function has changed, the intrusion detection system should be able to adapt to the new environment. Dimensionality reduction through feature selection is proposed in [15] to enhance the accuracy of other models. Incremental learning is studied in the next part as a possible solution.

4.2 INCREMENTAL TRAINING

XGBoost classifier permits the usage of a parameter named *xgb_model*. The parameter makes the XGBoost classifier consider the old model and allows training continuation [28]. Dataset KDD Test+ will be used for model training and testing.

The accuracy testing methodology is used, and the number of examples for each retraining is empirically chosen. Table 2 presents the accuracy evolution over model updates with different train-test dataset sizes.

The model was trained at first with 22 (0.1 %) examples to create the baseline. Using 1 % of the dataset (225 examples) for training increases the test accuracy from 78.7 % to 93.04 %. Test data accuracy stabilizes after training with 20 % of the dataset (4508 examples). FN and FP values on test data are also listed in Table 2. Starting with

model no. 4, the average value of FP: FN ratio is 0.85. No improvement is associated with increasing the training dataset to more than 20 % of the available data. Once we start training with the new dataset, the new model's accuracy on the whole KDD Train+ (the dataset used for training the old model) decreases to 94.6 %. The decrease can be associated with catastrophic forgetting [33]. The issue of forgetting old tasks in favor of new learning tasks is a hot topic in continual learning, mostly studied in neural networks [33].

Table 2.

Incremental learning results with XGBoostClassifier. Model number 4 is considered the best performing model (source: designed by authors)

No.	Train size [%]	Test size [%]	Train accuracy [%]	Test accuracy [%]	FN (test)	FP (test)
1	0.1	99.9	72	78.7	4523	273
2	1	99	100	93.04	678	876
3	10	90	99.9	97.1	236	352
4	**20**	**80**	**99.8**	**98.04**	**161**	**192**
5	30	70	99.8	98	149	166
6	40	60	99.6	97.89	137	149
7	80	20	99.5	98.60	27	36

4.3 MODEL EXPLAINABILITY

With respect to model evolution, we can explain model predictions with feature importance analysis. From Table 2, we choose model number four, as this is an example of high accuracy achieved with relatively low train dataset. The method is available in the xgboost library.

Figure 4 and 5 present the feature importance generated on fitted trees. The figures present the importance of each feature in the construction of the trees. Figure 4 presents the 15 most important features in the old model (trained on KDD Train+). Figure 5 presents the 15 most important features in the new model (trained on 20% of KDD Test+). The src_bytes feature maintains an important role in both new and old models, but the dst_bytes feature is replaced by dst_host_srv_count in the new model. Also, the new model has feature srv_count on the 11th position and feature hot is eliminated when learning on the new dataset.

5. TRIGGERING MECHANISM ARCHITECTURE

Figure 6 presents a possible edge-cloud setup that serves an IoT network. The edge is the first layer of the IIoT edge network that receives requests from both IIoT devices and the external world. The edge network can be seen as a sparse network of nodes. A network node might have more resources than the devices it serves, but it has less than the cloud. The collected data can be transmitted to the cloud to lessen the storage burden at the edge.

The edge node deploys a general model trained in a cloud environment in this setup. The purpose of the edge model is to perform predictions on incoming data from both sources. As devices evolve at the edge (e.g., get updated, patched, or even become malicious) and attackers change their methods, at some point, the cloud model will no longer fit the incoming data.

Incremental learning was proposed as a solution for adapting to a new dataset. Retraining on new data requires a degree of autonomy in deciding the retraining moment. A triggering mechanism is added in Fig. 6 to start updating the model with data gathered by the edge node. Drops in model performance can trigger updates, and data distribution changes or can be scheduled [23].

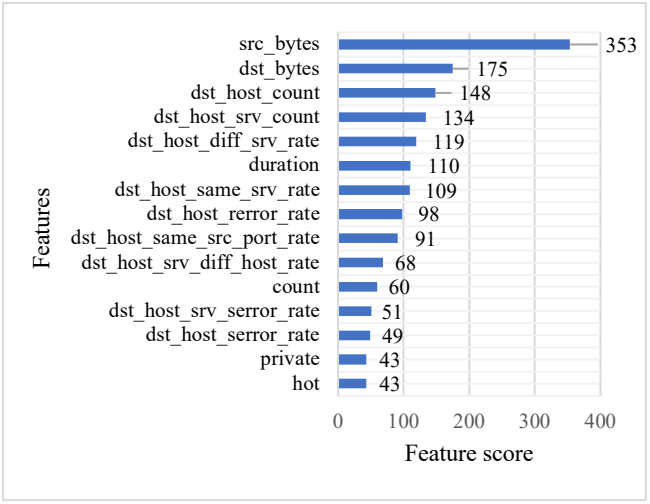

Fig. 4 – The 15 most important features in the old model construction.

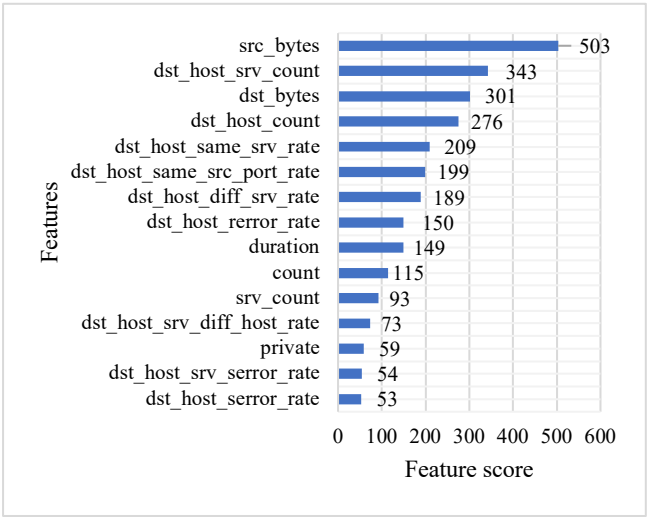

Fig. 5 – The 15 most important features in the new model construction.

Statistical methods can be used to identify data distribution shifts – but feature monitoring is a realistic solution for datasets with few features [23]. As the data at the edge node is not labeled, unsupervised methods could be employed for data shift identification. The unsupervised model should be able to distinguish outliers from valid data shifts.

Clustering algorithms are implemented for anomaly detection in [34–36]. In the case of K-means, the silhouette coefficient can be used for optimal cluster number identification [37]. A persistent change in the examples would determine optimal cluster number change or cluster reorganization.

Self-organizing Maps (SOM) is an unsupervised neural network based on competitive learning. The algorithm was applied for clustering microservices with similar resource consumption profiles in [38] and compared to K-means and K-shape. The algorithm proved faster than the K-clustering variations on a reasonable hardware setting. The effects of map size variation are studied in [39]. The purpose is to understand the dataset (old and new examples) to decide to retrain the model.

A triggering mechanism algorithm is presented in Fig. 7. This is based on the K-means clustering algorithm. The old dataset (used for model training) is kept for K-means

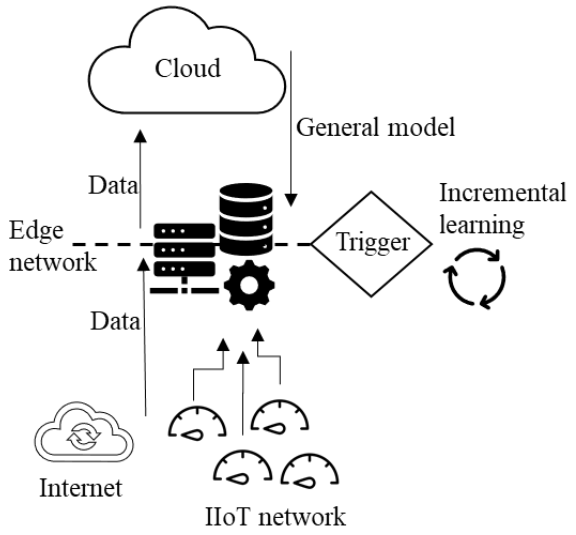

Fig. 6 – Edge-cloud setup for IIoT network.

training. The new examples are concatenated to the old dataset. The optimal number of clusters (OCN in Fig. 7) is calculated and will be used for fitting the K-means. If the OCN remains the same, then the new dataset is kept, and retraining of the model is triggered. A label propagation mechanism needs to be implemented to retrain the model XGBoost in a supervised manner. If OCN changes, then K-means is trained on the merged dataset. If the new examples are outliers of the newly created clusters, the data is discarded; otherwise, the new dataset is labeled and used for model retraining. After retraining, the merged dataset represents the old dataset, and a new execution of the Fig. 7 algorithm can begin.

6. CONCLUSIONS

This paper presents an end-to-end solution for intrusion detection using incremental learning. The authors propose a process involving three components: ML model, model explainability, and retraining trigger mechanism.

While related work (in chapter two) focuses on deep learning models to demonstrate incremental learning advantages, the authors suggest the XGBoost classifier, as it offers classification explainability, along with a balance between accuracy, performance, and resource consumption, while working with large amounts of data (if compared to neural or deep networks).

The paper's authors suggest the XGBoost classifier implementation for incremental learning on NSL-KDD intrusion detection dataset. To demonstrate the benefits of implementing incremental learning through XGBClassifier, the accuracy test methodology is used in chapter four. The accuracy on KDD-test+ dataset improves from 78 % to 93 % with only 225 train examples. The test accuracy reaches 98 % with 20 % of the KDD-test+ dataset used for retraining.

The authors suggest model explainability as a key advantage in implementing real-world ML-based intrusion detection solutions. The best-performing model from section four is detailed through feature importance in sub-section 4.3. Model explainability allows for evaluating solution correctness and applying adjustments in time.

To attain a degree of automation and ensure the evolution of the suggested intrusion detection solution in a dynamic environment, the authors propose a method to identify the need for retraining in chapter five. The authors suggest a logical diagram of the mechanism needed to trigger model retraining in Fig. 7. As the XGBoost classifier is a supervised model, retraining is performed on labeled examples. Nevertheless, the IDS is constantly exposed to new unlabeled data examples. Based on the authors' former research – published paper [38] – an unsupervised ML algorithm is suggested for a decision on retrain triggering and label propagation. The authors will further study the implementation and the details of the mechanism.

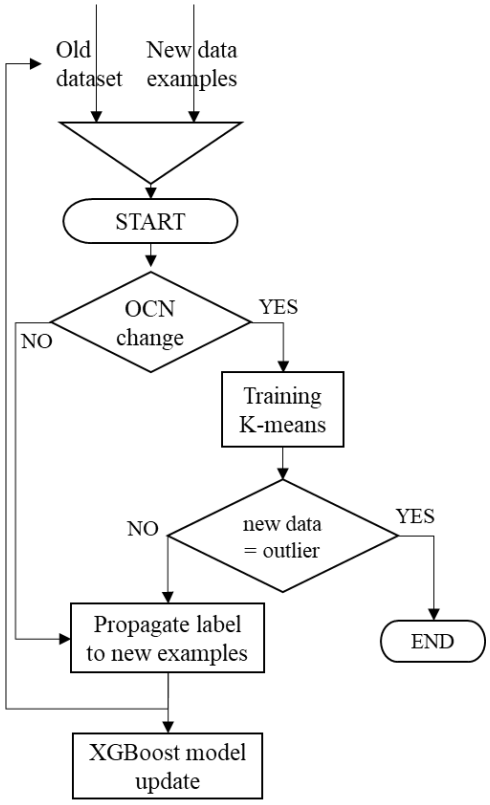

Fig. 7 – Triggering model update mechanism – logical diagram.

The results presented in this paper were achieved using a personal computer. As such, the results of this paper can prove helpful when model retraining is necessary for resource-scarce environments (relative to a cloud/enterprise environment) such as the network edge, edge devices, or personal devices.

ACKNOWLEDGEMENTS

The results presented in this article have been funded by the Ministry of Investments and European Projects through the Human Capital Sectoral Operational Program 2014-2020, Contract no. 62461/03.06.2022, SMIS code 153735.

Received on 15 June 2023

REFERENCES

1. ***5G Americas, *Becoming 5G-advanced: the 3GPP 2025 roadmap* (2022). Accessed: May 26, 2023. https://www.5gamericas.org/wp-content/uploads/2022/12/Becoming-5G-Advanced-the-3GPP-2025-Roadmap-InDesign.pdf
2. ***European Union Agency for Cybersecurity, ENISA Threat Landscape for 5G Networks Report, ENISA (2020). Accessed: Feb. 15, 2023. https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-for-5g-networks.
3. G. Nencioni, R.G. Garroppo, R. F. Olimid, *5G multi-access edge computing: security, depedability, and performance*, arXiv preprint

arXiv:2107.13374 (2021).
4. A.F. Glavan, D. Gheorghica, V. Croitoru, *Multi-access edge computing analysis of risks and security measures*, Rev. Roum. Sci. Techn.– Électrotechn. et Énerg., **68**, *2*, pp. 206–211 (2023).
5. ***AT&T Cybersecurity Insights, AT&T Cybersecurity 2023 Edge Ecosystem* (2023). Accessed: May 27, 2023. https://cdn-cybersecurity.att.com/docs/industry-reports/cybersecurity-insights-report-edge-ecosystem.pdf
6. M. Almseidin, M. Alzubi, S. Kovacs, M. Alkasassbeh, *Evaluation of Machine Learning Algorithms for Intrusion Detection System*, IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), pp. 000277–000282, IEEE (2017).
7. A. Kumari, A.K. Mehta, *A hybrid intrusion detection system based on decision tree and support vector machine*, 2020 IEEE 5th International Conference on Computing Communication and Automation, ICCCA 2020, pp. 396–400 (2020).
8. K.K. Nguyen, D.T. Hoang, D. Niyato, P. Wang, D. Nguyen, E. Dutkiewicz, *Cyberattack detection in mobile cloud computing: A deep learning approach*, IEEE Wireless Communications and Networking Conference, WCNC, vol. 2018-April, pp. 1–6 (2018).
9. P. Wang, K.M. Chao, H.C. Lin, W.H. Lin, C.C. Lo, *An efficient flow control approach for sdn-based network threat detection and migration using support vector machine*, Proceedings - 13th IEEE International Conference on E-Business Engineering, ICEBE, pp. 56–63 (2017).
10. S. Khamaiseh, E. Serra, Z. Li, D. Xu, *Detecting saturation attacks in SDN via machine learning*, 2019 4th International Conference on Computing, Communications and Security, ICCCS 2019, pp. 1–8 (2019).
11. ***Edge Computing vs. Cloud Computing: Major Differences – Unite.AI*, Accessed: May 29, 2023. https://www.unite.ai /edge-computing-vs-cloud-computing-major-differences/
12. A. Lakshmi, *A disease prediction model using spotted hyena search optimization and bi-lstm*, Rev. Roum. Sci. Techn. – Électrotechn. Et Énerg, 68, 1, pp. 113–118 (2023).
13. H. Gupta et al., *Category boosting machine learning algorithm for breast cancer prediction*, Rev. Roum. Sci. Techn.– Électrotechn. et Énerg, **66**, *3*, pp. 201–206 (2021).
14. N. Sabri, A. Tlemçani, A. Chouder, *Real-time diagnosis of battery cells for stand-alone photovoltaic system using machine learning techniques*, Rev. Roum. Sci. Techn.– Électrotechn. et Énerg, **66**, no. *2*, pp. 105–110 (2021).
15. L. Dhanabal, S. P. Shantharajah, *A study on NSL-KDD dataset for intrusion detection system based on classification algorithms*, International Journal of Advanced Research in Computer and Communication Engineering, **4**, *6*, pp. 446–452 (2015).
16. T. Su, H. Sun, J. Zhu, S. Wang, Y. Li, *BAT: deep learning methods on network intrusion detection using NSL-KDD dataset*, IEEE Access, **8**, pp. 29575–29585 (2020).
17. K.H. Le, K.H. Le-Minh, H.T. Thai, *BrainyEdge: an AI-enabled framework for IoT edge computing*, ICT Express, **9**, *2*, pp. 211–221 (2023).
18. T. Wang, Q. Lv, B. Hu, D. Sun, *A few-shot class-incremental learning approach for intrusion detection*, Proceedings – Int. Conf. on Computer Communications and Networks, ICCCN, 2021-July (2021).
19. A.A. Ghorbani, M. Tavallaee, E. Bagheri, W. Lu, *A detailed analysis of the KDD CUP 99 data se*t, IEEE symposium on computational intelligence for security and defense applications, pp. 1-6 (2009).
20. ***NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB*, Accessed: May 27, 2023. https://www.unb.ca/cic/datasets/nsl.html.
21. K. Sadaf, J. Sultana, *Intrusion detection based on autoencoder and isolation forest in fog computing*, IEEE Access, **8**, pp. 167059–167068 (2020).
22. M. Haggag, M.M. Tantawy, M.M.S. El-Soudani*, Implementing a deep learning model for intrusion detection on apache spark platform*, IEEE Access, **8**, pp. 163660–163672 (2020).
23. ***Real-time machine learning: challenges and solutions*, Accessed: May 30, 2023. Available: https://huyenchip.com/2022/01/02/real-time-machine-learning-challenges-and-solutions.html#towards-continual-learning.
24. K. Guo, Z. Liang, R. Shi, C. Hu, Z. Li, *Transparent learning: an incremental machine learning framework based on transparent computing*, IEEE Network, **32**, *1*, pp. 146–151 (201.
25. A.F Glavan, V. Croitoru, *Autoencoders and AutoML for intrusion detection*, in ECAI 2023 15th Edition International Conference on Electronics, Computers and Artificial Intelligence, pp. 1-4 (2023).
26. ***XGBoost Documentation — xgboost 1.7.5 documentation*, Accessed: May 28, 2023. Available: https://xgboost.readthedocs.io/en/stable/.
27. ***What is XGBoost? An Introduction to XGBoost Algorithm in Machine Learning | Simplilearn*, Accessed: May 28, 2023: https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article.
28. ***Python API Reference — xgboost 1.7.5 documentation*, Accessed: May 28, 2023. https://xgboost.readthedocs.io/en/stable/python/python_api.html
29. ***Project Jupyter | Try Jupyter*, Accessed: May 29, 2023. https://jupyter.org/try.
30. ***DataTechNotes: Classification Example with XGBClassifier in Python*, Accessed: May 28, 2023. https://www.datatechnotes.com/2019/07/classification-example-with.html.
31. ***scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation*, Accessed: May 28, 2023. https://scikit-learn.org/stable/
32. ***Accuracy, Precision, and Recall in Deep Learning | Paperspace Blog*, Accessed: May 28, 2023. https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/.
33. Z. Chen, B Liu, *Continual learning and catastrophic forgetting, lifelong machine learning*, Cham: Springer International Publishing, pp. 55–75. (2018).
34. V. Sridharan, M. Gurusamy, A. Leon-Garcia, *Anomalous rule detection using machine learning in software defined networks*, IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN 2019 – Proceedings, pp. 1–6 (2019).
35. S. Dong, Y. Xia, T. Peng, *Network abnormal traffic detection model based on semi-supervised deep reinforcement learning*, IEEE Transactions on Network and Service Management, **18**, *4*, pp. 4197–4212 (2021).
36. E. Raff, B. Filar, J. Holt, *Getting passive aggressive about false positives: patching deployed malware detectors*, IEEE International Conference on Data Mining Workshops, ICDMW, 2020-November, pp. 506–515 (2020).
37. ***K Means Clustering | Method to get most optimal K value*, Accessed: May 30, 2023. https://www.analyticsvidhya.com/blog/2021/05/k-mean-getting-the-optimal-number-of-clusters/.
38. A.F. Glavan, V. Croitoru, *Cloud environment assessment using clustering techniques on microservices dataset*, 14th International Conference on Communications, COMM 2022 – Proceedings, pp. 1–6 (2022).
39. Y. Liu, R.H. Weisberg, C.N.K. Mooers, *Performance evaluation of the self-organizing map for feature extraction*, Journal of Geophysical Research: Oceans, **111**, *C5*, p. 5018 (2006).