Hierarchical Multiclass Continual Learning for Network Intrusion Detection

Jacopo Talpini, Fabio Sartori, Marco Savi
Department of Informatics, Systems and Communication (DISCo), University of Milano-Bicocca, Italy

***Abstract*—The evolution of Internet and its related communication technologies have consistently increased the risk of cyber-attacks. In this context, a crucial role is played by Intrusion Detection Systems (IDSs), which are security devices designed to identify and mitigate attacks to modern networks. In the last decade, data-driven approaches based on Machine Learning (ML) have gained more and more popularity for executing the classification tasks required by signature-based IDSs. However, typical ML models adopted for this purpose are trained in static settings while new attacks – and variants of known attacks – dynamically emerge over time. As a consequence, there is the need of keeping the IDS capability constantly updated, which poses peculiar challenges especially in resourced-constrained scenarios. To this end, we propose a novel hierarchical model based on a binary classification of benign and malicious traffic performed by a Bayesian Neural Network that is trained continuously and efficiently by exploiting Continual Learning. A generative multiclass classifier is then adopted to incrementally classify new kinds of attacks with respect to the malicious traffic. We prove the effectiveness of our approach showing that it removes the need of storing network traffic data samples related to historical data, representative of all the kinds of attacks, while ensuring good detection capabilities.**

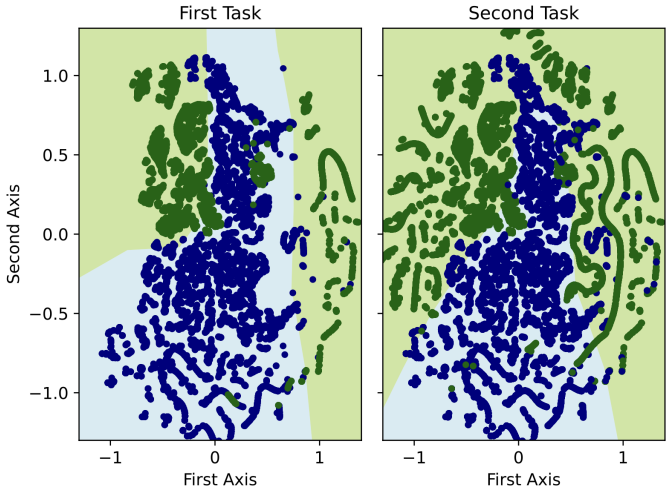

Fig. 1. A pictorial representation of Continual Learning on a sequence of two tasks. Blue dots represent benign traffic while dark green ones are attacks, light blue and green areas represent the decision contour of a binary classifier, namely an IDS, that should adapt to the appearance of new attacks in the second task, while keeping good performance on the older one.

I. Introduction

Network intrusions stand as a major scourge within modern communication networks. As the frequency and complexity of these incidents continues to rise [1], it becomes crucial to develop meticulous detection strategies and resilient counteraction measures. This is essential for the effective identification and mitigation of the threats posed by such intrusions. *Intrusion Detection Systems* (IDSs) are among the primary security measures in communication networks, to identify attacks, unauthorized intrusions, as well as malicious activities [2]. IDSs are generally divided into two families: *signature-based* and *anomaly-based* [3]. In this paper, we analyze the first category, which is based on pattern recognition and aims to compare signatures of well-known attacks and benign traffic with the current network traffic patterns. On the other hand, anomaly-based methods rely on a model only for the normal (i.e., benign) network traffic so that any pattern that deviates from the usual one is considered an intrusion. In contrast to signature-based IDSs, anomaly-based IDSs are able to detect also new types of attacks, but they typically suffer from a high rate of false positives [4].

Traditional approaches to IDSs rely on *knowledge-based* systems [5]: however, the increasing complexity of networks makes those systems more prone to errors [3] [6]. As a consequence, data-driven approaches based on *Machine Learning* (ML) have been widely considered in recent years for detecting attacks, demonstrating great performance in terms of classification scores [4].

However, the vast majority of the proposed methods in the literature on ML-based IDSs are based on a *static* framework, where a model is trained once on a given training set, which is supposed to be representative of all possible network patterns encountered during the deployment. On the other hand, we may expect that in a real-world scenario intrusions do not emerge at once but gradually over time, so that the original training dataset will eventually become outdated. Thus, there is the need for a *continuous* training of an IDS to accommodate new intrusion variants, i.e., we require that a model should be able to learn from a subsequent collection of potentially different data distributions, referred to as *tasks*, and revealing new attacks. This process becomes particularly challenging when it is not feasible to store all potential historical data, and instead, older data are permanently discarded. This is the typical case occurring in *resource-constrained* scenarios such as when intrusion detection capabilities are deployed at the edge in shared virtualized environments, or in the far-edge (e.g. in IoT gateways) [7]. In this setting, the main issue is represented by the so-called *catastrophic forgetting* [8], i.e., the phenomenon according to which the performance on older data drops when the model is re-trained on the newest ones.

Figure 1 illustrates the impact of a distribution shift resulting from the emergence of novel attack types. The scatter-plots are generated using manifold learning through t-Distributed Stochastic Neighbor Embedding (t-SNE) [9] to reduce the dataset dimensionality and allow a 2D representation. This is

performed for two distinct task categories of the considered dataset (see Section IV), providing an intuitive insight into the necessity of retraining an IDS to maintain high predictive capabilities for both new and older data distributions.

In this paper, we exploit *Continual Learning* (CL) [10] capabilities to achieve this goal. The need for CL is particularly relevant when it may not be feasible to store large volumes of old data, which constitutes the primary setting explored in this paper. To this end, we suggest leveraging the inherent hierarchy in the intrusion detection problem (i.e., first predicting whether a flow is an attack and then determining its type) to create a model that can be updated continuously as new attacks emerge. In particular, we propose to rely on a parametric discriminative model (e.g. a Neural Network, NN) as a binary classifier for distinguishing between benign and attack flows. This model is based on Bayesian Inference [11] so that it can be continuously and effectively trained on new tasks. Additionally, we propose incorporating a straightforward generative classifier (e.g. Quadratic Discriminant Analysis) for modeling the residual classification problem of different attacks, as it occurs in our considered class-incremental setting.

Our illustrative results show that the proposed model is able to learn new data patterns, while preserving good classification abilities on the older ones, without storing historical data.

II. Related Work

The application of ML to network intrusion detection has been receiving considerable attention as it enables automated analysis of network traffic by leveraging past data [3] [12]. For instance, [13] [14] compare different classification algorithms to develop an IDS and, in general, the best performance is achieved by tree-based classifiers, like Random Forests, and Multi-Layer Perceptrons (MLPs). However, as already said, most existing works adopt a *static* approach, where training occurs once on a specific dataset, assuming that test data originates from the same distribution. This limitation is significant given the *dynamic* nature of communication networks, where new network traffic and attacks may emerge during deployment. As a consequence, the field of CL has gained more and more attention, also in the context of intrusion detection, for implementing both anomaly and signature-based IDSs. In the following, we analyze the most relevant work tackling this problem in the context of signature-based IDSs, as it is strictly coupled with multiclass classification.

In [15] the authors propose a comparison of different strategies for addressing the CL problem in the field of intrusion detection. The primary discovery indicates that models for CL based on *replay* surpass conventional statistical methods, as well as cutting-edge Boosted Decision Trees and Deep Neural Network (DNN) models, in effectively addressing the distribution shift problem. A similar conclusion was reached in [16]. In both these works the experiments are based on a setting where all the classes are a-priori known and then the model should adapt only to the covariate shift happening for each known class. In contrast, in our work we tackle the problem from a class-incremental perspective where the number of classes is not a-priori known, which we believe better sticks to the real world.

On the other hand, [17] analyzes the problem of adapting a signature-based IDS to new kinds of attacks in a few-shot regime. However, the solution proposed by the authors relies on having access also to a certain number of historical data to adapt the classifier. In this work we instead analyze a scenario where historical data may no longer be available while executing the current task, due to privacy reasons or due to stringent resource constraints. Another contribution that analyzes a similar setting as the one considered in this paper is [18] which, however, exploits different models, not based on NNs. The authors propose an ensemble Incremental Learning algorithm based on Hoeffding Tree, Adaptive Boosting and Hard Sampling, which is capable of learning new attacks without forgetting the previously learned traffic patterns. This approach shows encouraging results but it still requires a list of samples from older tasks that are hard to classify.

III. Problem Formulation and Adopted Models

The field of Continual Learning is getting more and more attention since for several applications data comes in a sequence of datasets that are used for training and then permanently discarded, as it should occur for network traffic. More formally, the problem statement may be formulated as follows: given T subsequent and disjoint datasets $\mathcal{D}_t$, called *tasks*, i.e., $\mathcal{D}_t = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N_t}$ each with N_t input-output pairs, we want to train a parametric model f, characterized by a set of weights $\boldsymbol{w}$, sequentially on each task (i.e., $\boldsymbol{y}_i = f_{D_t}(\boldsymbol{x}_i, \boldsymbol{w})$) without forgetting the knowledge gained from the previously encountered tasks. In the context of this study, our objective is to develop a classifier that can categorize network traffic for each task, defined by the presence of benign traffic and only a specific type of attack. Since tasks vary with the introduction of new attack types, it is essential to consistently retrain the IDS to effectively identify diverse intrusion patterns.

At a high level, CL methods can be broadly classified into three main families [10]: *Replay* methods, *Parameter Isolation* methods, *Regularization*-based methods. Since a complete overview of the CL field is beyond the scope of this paper, we limit to describe the last approach which is the most suited for resourced-constrained scenarios. More specifically, the file-rouge of the models of this family is to rely on an extra regularization term introduced in the loss function of each task, in order to consolidate previous knowledge when learning on new data. Among the most commonly-employed methods it is worth mentioning *Elastic Weight Consolidation* [8], which imposes a quadratic penalty to regularise the update of model parameters that were important to previous tasks. The importance of parameters is approximated by the diagonal of the Fisher Information Matrix [8]. More recent and competitive approaches rely on the *Bayesian Inference* framework for mitigating forgetting. In fact, Bayesian Inference offers a straightforward and principled way of implementing CL, as shown in the following.

A. Background on Bayesian Neural Networks

This Section is devoted to introducing Bayesian Inference for Neural Networks and how we exploit it to define our model. The most distinguishing property of Bayesian NNs

(BNNs) is that the parameters of a model $\boldsymbol{w}$ are treated as random variables for which we aim to infer a posterior distribution $p(\boldsymbol{w}|\mathcal{D})$, given the training data $\mathcal{D}$, starting from a prior $p(\boldsymbol{w})$ and a likelihood $p(\mathcal{D}|\boldsymbol{w})$, through Bayes theorem. Predictions are then obtained by marginalizing over the posterior [19]. Unfortunately, the posterior distribution is intractable for NNs of practical size. To get around this problem, several approximate inference methods were proposed. One of the most common is represented by *Variational Inference*, which aims to find a tractable approximation to the Bayesian posterior distribution of the weights [20]. It is important to specify that variational inference can be exploited also for Continual Learning. For instance, in [11] the authors propose VCL, a method based on a Bayesian approach to learn parameters of a model, using the previous task's posterior as the new task's prior when new data are seen. More specifically, it is possible to approximate each task's posterior by a variational distribution, $q_t(\boldsymbol{w}) \sim p_t(\boldsymbol{w}|\mathcal{D}_{1:t})$ by maximising the following objective function for every task t [11]:

$$\mathcal{L}_t(q_t(\boldsymbol{w})) = \sum_{n=1}^{N_t} \mathbb{E}_{q_t(\boldsymbol{w})}[\log\ p(\boldsymbol{y}_n|\boldsymbol{x}_n, \boldsymbol{w})] - \mathrm{KL}(q_t(\boldsymbol{w})||q_{t-1}(\boldsymbol{w})). \tag{1}$$

The first term in Eq. 1 is the expected log-likelihood of the current model over the data in the current dataset, and it contributes to updating the model knowledge for the current task. The second term is the Kullback-Leibler (KL)-divergence between the actual and the old variational posterior, and penalizes the difference between the current model and the approximate posterior of the previous task. The variational distribution $q_t(\boldsymbol{w})$ should be chosen to be easy to sample and is typically a diagonal multivariate Gaussian, as used in this paper. It is worth noting that this approach has the advantage of not requiring free parameters to be tuned on a validation set (w.r.t. [8] for instance) which can be non-trivial, especially in the considered setting. Last, we emphasized that in contrast to [11] we do not implement the fine-tuning phase for each task with a certain amount of data retained from the previous tasks, the so-called "coresets", and we solely rely on Eq. 1, aligning with the considered resource-constrained scenario.

B. Proposed model architecture

The proposed approach relies on a hierarchical model: first we exploit a BNN trained in a continual way as a binary classifier (benign/attack) and then we rely on a second generative classifier, to discriminate between known types of attacks, trained in a class-incremental way. Figure 2 summarizes the overall workflow at a high level, and in the following we describe the two models in more details.

The base model adopted in this paper is a MLP composed of two hidden layers of 32 and 16 neurons respectively employing ReLU as activation function. The last layer employs the sigmoid function so that the outputs of the model, given an input, can be interpreted as the probability of being an attack/benign traffic. This particular architecture was chosen to maximize the validation accuracy at a reasonable computational cost, where the training and validation were performed in the standard way, with all the training data present at training time. Then the question is if and how much Continual Learning impacts the overall classification performance. In order to exploit Bayesian Inference in the considered model and employ the previously-described approach for Continual Learning, we relied on TensorFlow probability infrastructure [21]. More specifically, we exploited Variational Inference with a multivariate diagonal Gaussian distribution as variational distribution. The model is trained for 50 epochs on each task with batches of 128 samples. In this way, it is possible to continuously train a binary classifier for benign/malicious flows, once new attacks are discovered, as previously described.

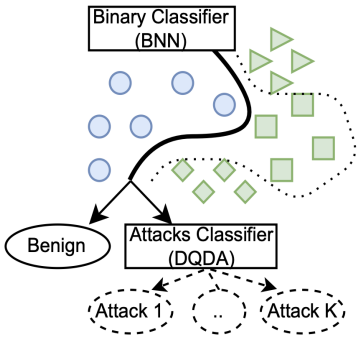

Fig. 2. A pictorial representation of the proposed hierarchical system. The binary classifier is constituted by a BNN while the multiclass attacks classifier is based on a simple generative classifier.

However, for choosing an effective countermeasure, a network administrator also needs to know what kind of attack is happening. To this end, we propose to rely on a generative multiclass classifier [19] for identifying the attack type. The advantage of an even simple generative classifier, like a Quadratic Discriminant Analysis (QDA), is that the parameters of each class-conditional density are estimated independently, meaning that there is no need to retrain the model from scratch when more classes are added. In contrast, in discriminative models (like NNs), all the parameters are entangled, so the whole model must be retrained if new classes are added. Here, we exploit a further simplified version of QDA, by approximating the class-conditioned distributions of the data as diagonal multivariate Gaussian; we refer to this model as Diagonal QDA (DQDA). It should be noted that a generative classifier may perform poorly if applied to the whole dataset (see Section V) but it may provide decent predictive performance on a subset of the data (e.g. represented by only the attack classes). The adopted DQDA classifier has the advantage of being a lightweight model with only $\mathcal{O}(CD)$ parameters, given C classes and D features; we exploited the implementation provided by Scikit-learn [22].

IV. Dataset Description

We consider an open-source dataset, i.e., CICIDS2017[1] [23], to evaluate our proposed system. CICIDS2017 is a labeled per-flow dataset that covers both benign and intrusion traffic, and it is widely used in the literature (e.g. [2] [3]). Table I shows the per-class distribution of samples.

A. Data preprocessing

It is clear from Table I that the dataset exhibits a significant imbalance, with some classes including only a few tens or

[1]https://www.unb.ca/cic/datasets/ids-2017.html

TABLE I
SAMPLES DISTRIBUTION

Class	Number of samples
Benign	2095057
DoS Hulk	172846
DDoS	128014
PortScan	90694
DoS GoldenEye	10286
FTP-Patator	5931
DoS slowloris	5385
DoS Slowhttptest	5228
SSH-Patator	3219
Bot	1948
Web Attack-Brute Force	1470
Web Attack-XSS	652
Infiltration	36
Web Attack-Sql Injection	21
Heartbleed	11

hundreds of samples. This imbalance poses challenges for both model training and evaluation, particularly for classes that are underrepresented. Consequently, we decided to evaluate our method on the most represented, i.e., the first eight in the Table. This means that we consider seven different tasks, being benign traffic always present. However, we intentionally utilized the underrepresented classes for testing the generalization capabilities of the proposed approach, i.e., in a scenario involving new types of attacks.

The dataset including the eight classes is partitioned into 60% training, 20%, validation, and 20% testing sets. To reduce the unbalance across tasks and classes we randomly subsampled the training data according to the minority class. As a consequence, each task is composed of ~ 3100 benign flows and $\sim$ 3100 samples belonging to an attack class. Finally, data are standardized per-task, ensuring that each feature distribution has a zero mean and unit variance. The parameters for standardization (mean and standard deviation of each feature) are continuously updated on a task basis.

V. ILLUSTRATIVE NUMERICAL RESULTS

We initially analyze the performance of the proposed approach in binary classification (i.e., benign traffic vs. attack) and later we analyze the performance of the multiclass classifier in recognizing various types of attacks.

A. Binary classification

We compare our CL-based BNN classifier updated through Variational Inference with two baselines:

- **Full Dataset**: a traditional BNN classifier is trained from scratch on the full dataset including both current and historical data from past tasks.
- **Naïve**: a traditional BNN classifier is re-trained from scratch solely on the data of each task in the standard way. Basically, this training differs from our CL-based approach by neglecting the second term of Eq. 1. This baseline is useful for assessing the impact of distribution shifts on the traditional BNN classifier.

The main metric to assess the capabilities of the classifiers is the Area Under the ROC curve (*AUROC*) [24] for each model (the higher, the better). In this way it is possible to fairly compare the classifiers in a threshold-independent manner.

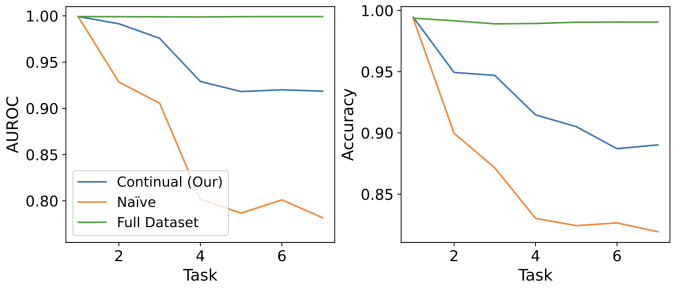

Fig. 3. Evolution of the AUROC and overall Accuracy over tasks for the different training strategies for the BNN model.

TABLE II
BINARY CLASSIFICATION METRICS

Model	Metric (Mean value ± Standard Error)			
	AUROC	Accuracy	F1 Weighted	F1 Macro
Continual (Our)	0.93 ± 0.02	0.89 ± 0.02	0.88 ± 0.02	0.85 ± 0.05
Naïve	0.77 ± 0.06	0.82 ± 0.02	0.79 ± 0.02	0.68 ± 0.04
Full Dataset	0.98 ± 0.01	0.98 ± 0.01	0.98 ± 0.01	0.97 ± 0.01

AUROC has also the advantage of being insensitive to the imbalance of classes [24], which makes this metric particularly well-suited to our scenario. We also exploited other common metrics, like the overall accuracy and the F1-Score. All the experiments were repeated 15 times by randomly permuting the order of the task, i.e., by changing the order of appearance of new attacks. Figure 3 reports the average evolution of the AUROC and Accuracy over different tasks for the considered models. The test set exploited to compute the AUROC for each task is composed of the union of all previous tasks' test sets, to check the ability of retaining past knowledge. In general, it is possible to state that the problem of distribution shifts badly affects the performance of an IDS: a classifier trained on a specific task is not able to generalize well to unseen kinds of attacks, despite they belong to the same macro-category (i.e., attack). On the other hand, these plots demonstrate that the proposed strategy based on Bayesian Inference allows the model to retain a certain amount of knowledge related to the older tasks, even though there is a performance gap compared to the model trained with the full dataset. This is also confirmed by the results of Table II, where the performance computed after the last task on the overall test-set is reported. Such a performance gap is expected; however, keeping training data related to older tasks is not feasible in many resource-constrained scenarios, as already thoroughly stressed.

Another aspect we investigated concerns the generalization capabilities of our binary classifier. For this purpose, we provided it with samples belonging to the six unrepresented classes (refer to the bottom of Table I) to simulate a *zero-day-attack* scenario. Our model reaches a True Positive Rate of 0.83 ± 0.03, the one trained on the Full Dataset 0.86 ± 0.02 and the Naïve model 0.77 ± 0.05. Thus, Continual Learning has a negligible impact on the generalization capabilities of a baseline BNN in a scenario with zero-days attacks.

TABLE III
MULTICLASS CLASSIFICATION METRICS

Scenario	Model	Metric (Mean value ± Standard Error) Accuracy	F1 Weighted	F1 Macro
Attacks Only	Continual (Our)	0.98 ± 0.01	0.98 ± 0.01	0.99 ± 0.01
Overall	Continual (Our)	0.88 ± 0.02	0.87 ± 0.02	0.84 ± 0.02
	BNN (Full)	0.89 ± 0.01	0.90 ± 0.01	0.69 ± 0.01
	DQDA (Full)	0.52 ± 0.01	0.55 ± 0.01	0.53 ± 0.01

B. Multiclass classification

Table III provides a summary of the overall performance in multiclass classification. For our model, we divided the analysis into two parts: the first row (*Attacks Only* scenario) reports the metrics computed on samples that are previously correctly classified by the binary classifier. Following that, we present the *Overall* performance of the multiclass classification computed across all classes, including the benign one (i.e., system-level evaluation). For comparison, we also considered DQDA and BNN stand-alone models trained on all available data in the typical static setting (i.e., *Full* dataset), where all classes are present during training. It is interesting to note that the DQDA classifier performs remarkably well for discriminating attacks (i.e., as done in our approach) but struggles in a multiclass setting when benign traffic is also present: in this case, the performance drops significantly. This fully justifies our choice of adopting a more flexible classifier (i.e., a BNN) for binary benign-attack classification, and relying on DQDA for the residual complexity of further classifying the attack among different attack types. Overall, we argue that the adoption of Continual Learning and the combination of two simple but specialized models can show decent overall performance compared to the same models trained conventionally, while removing the need of keeping historical data when re-training with new data is performed.

VI. CONCLUSION

In this work, we introduced a novel approach based on Continual Learning for continuous training of an IDS upon discovering new attacks. Our method leverages the inherent hierarchy of the problem by utilizing a binary BNN continously updated by means of Variational Inference, and a generative DQDA multiclass classifier only for further attack classification. Numerical results on a real-world dataset demonstrate that our proposed approach effectively manages class-incremental training without the need of storing historical data on past attacks. For this reason, it is well-suited for scenarios with stringent resource constraints, where the demand for thin models and limited storage size of training datasets is prevalent.

One notable limitation of the proposed solution is its inability to automatically recognize (i.e., without an external intervention) new types of attacks. As part of future work, we plan to enhance the model by incorporating Out-of-Distribution detection capabilities.

ACKNOWLEDGEMENT

The research leading to these results has been partially funded by the Italian Ministry of University and Research (MUR) under the PRIN 2022 PNRR framework (EU Contribution – NextGenerationEU – M. 4,C. 2, I. 1.1), SHIELDED project, ID P2022ZWS82.

REFERENCES

[1] European Union Agency for Cybersecurity (ENISA), "ENISA Threat Landscape 2022," https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022, 2022, [Accessed: 03-October-2023].

[2] S. Tsimenidids, T. Lagkas, and K. Rantos, "Deep learning in iot intrusion detection," *Journal of Network and Systems Management*, vol. 30, 01 2022.

[3] Z. Tauscher, Y. Jiang, K. Zhang *et al.*, "Learning to detect: A data-driven approach for network intrusion detection," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 2021, pp. 1–6.

[4] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin *et al.*, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[5] V. Hassija, V. Chamola, V. Saxena *et al.*, "A survey on iot security: Application areas, security threats, and solution architectures," *IEEE Access*, vol. 7, pp. 82 721–82 743, 2019.

[6] N. Shone, T. N. Ngoc, V. D. Phai *et al.*, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[7] M. Zang, C. Zheng, L. Dittmann, and N. Zilberman, "Towards continuous threat defense: In-network traffic analysis for iot gateways," *IEEE Internet of Things Journal*, pp. 1–1, 2023.

[8] J. Kirkpatrick, R. Pascanu, N. Rabinowitz *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[9] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[10] G. I. Parisi, R. Kemker, J. L. Part *et al.*, "Continual lifelong learning with neural networks: A review," *Neural networks*, vol. 113, pp. 54–71, 2019.

[11] C. V. Nguyen, Y. Li, T. D. Bui *et al.*, "Variational continual learning," in *International Conference on Learning Representations*, 2018.

[12] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE Symposium on Security and Privacy*, 2010, pp. 305–316.

[13] A. Verma and V. Ranga, "Machine learning based intrusion detection systems for iot applications," *Wireless Personal Communications*, vol. 111, pp. 2287–2310, 2020.

[14] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran *et al.*, "Deep learning approach for intelligent intrusion detection system," *Ieee Access*, vol. 7, pp. 41 525–41 550, 2019.

[15] S. Prasath, K. Sethi, D. Mohanty *et al.*, "Analysis of continual learning models for intrusion detection system," *IEEE Access*, vol. 10, pp. 121 444–121 464, 2022.

[16] C. Oikonomou, I. Iliopoulos, D. Ioannidis, and D. Tzovaras, "A multiclass intrusion detection system based on continual learning," in *2023 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2023, pp. 86–91.

[17] T. Wang, Q. Lv, B. Hu, and D. Sun, "A few-shot class-incremental learning approach for intrusion detection," in *2021 International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2021.

[18] M. Data and M. Aritsugi, "An incremental learning algorithm on imbalanced data for network intrusion detection systems," in *Proceedings of the 10th International Conference on Computer and Communications Management*, 2022, pp. 191–199.

[19] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[20] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*. PMLR, 2015, pp. 1613–1622.

[21] J. V. Dillon, I. Langmore, D. Tran *et al.*, "Tensorflow distributions," *arXiv preprint arXiv:1711.10604*, 2017.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[24] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.