Impact of Training Set Size on Resource Usage of Machine Learning Models for IoT Network Intrusion Detection

Barikisu A. Asulba, Nuno Schumacher, Pedro F. Souto, and Luis Almeida
Faculdade de Engenharia & CISTER research center, Universidade do Porto
Porto, Portugal
{up202103270, nschumacher, pfs, lda}@fe.up.pt

Pedro M. Santos
CISTER research center, Inst. Sup. de Engenharia, Instituto Politécnico do Porto
Porto, Portugal
pss@isep.ipp.pt

Nuno Martins and Joana Sousa
NOS Inovação
Lisboa, Portugal
{nuno.mmartins, joana.sousa}@nos.pt

***Abstract*—Security is a critical concern in Internet-of-Things (IoT) environments, including industrial IoT and one solution to enhance security is to deploy Network Intrusion Detection Systems (NIDS) using machine learning (ML) models in edge such as gateway devices. However, the resource constraints of these devices can pose challenges for implementing ML models. This study examines the impact of different training set sizes on the performance and resource usage of One-Class ML models that seem particularly adequate to this use case. The results indicate that One-Class Support Vector Machine, Isolation Forest, and Elliptic Envelope models are suitable for resource-constrained devices due to their low model size, classification time, and consistent performance with increasing sample size. The Local Outlier Factor model exhibited a high detection rate and low false alarm rate at the cost of high model size and classification time. Our results can help develop more efficient and effective network intrusion detection for IoT systems.**

***Index Terms*—classification time, intrusion detection, IoT network, machine learning, computing resources usage.**

I. Introduction

The security of IoT systems is a growing concern, as they are vulnerable to various cyber attacks such as denial of service (DoS), replay attacks, password guessing attacks, spoofing attacks, and insider attacks. To ensure system security, it is important to use secure communication protocols, implement appropriate authorization mechanisms, and have an effective intrusion detection system in place [1].

A common technique to improve security in the IoT infrastructure is deploying anomaly-based network intrusion detection systems (NIDS). These systems can identify unanticipated events that may threaten the network privacy, security and operation [2]. Machine learning (ML) algorithms are widely used, including for network intrusion detection. Support vector machines (SVMs) are the most common technique among researchers, followed by Neural Networks and Decision-Trees. These models are considered supervised models, as they require labeled data for training [3].

However, it can be difficult to obtain precisely labeled data and balanced data between attack (malign) and normal (benign) traffic, which is where unsupervised machine learning can be effective and valuable for NIDS. Unsupervised ML allows training without labeled data [4]. One of the primary techniques of unsupervised models for anomaly detection is known as One-Class Classification (OCC). This approach involves training a model on a data set that contains only benign samples. Samples that deviate significantly from this normal behavior are considered anomalous or indicative of potential intrusions [5].

Arregoces et al. [6] study the UNSW-NB15 data set and discovered that OCC is an efficient solution for class imbalance and the detection of new attack types. They found that the One-Class Support Vector Machine (OCSVM) achieved a 100% detection rate. Another study [7] combined the OCSVM with a sliding window and Fuzzy C-means for the detection of intrusion in Wireless Sensor Networks (WSN) and showed improved performance with a 74.61% utility value. Previous research has only evaluated traditional metrics and has not considered computing time and memory. Recent studies suggest that the deployment of ML in edge devices improves privacy and reduces latency [8], but the capabilities of the devices must be taken into account.

This study extends the research conducted by Schumacher et al. [9], [10] on machine learning models for intrusion detection in home IoT environments using the respective Internet gateway, also known as Customer Premises Equipment (CPE). We believe that the findings from their study can be applied to industrial sites with IoT devices as well. However, the prior work on one-class machine learning models in IoT networks had limitations, such as small training data sets and the lack of consideration for classification time and model size as metrics. In contrast, our paper addresses these limitations by evaluating the impact of training data set size on model accuracy and resource requirements.

By providing insights into the relationship between training set size, model performance, and resource requirements, our research makes a valuable contribution to the field of network intrusion detection systems in IoT environments. This understanding is crucial for guiding the design and

implementation of efficient and effective intrusion detection solutions that take into account the constraints and capabilities of edge devices. Our findings reveal that different models have different recommended training set sizes for achieving optimal performance, highlighting the trade-off between training set size, performance, and resource utilization.

The paper is organized as follows: Next, Section II describes the data sets, models and evaluation metrics. Section III details the experiment and presents the results. Section IV reviews related research while Section V concludes the paper.

II. Methodology

This section describes the one-class machine learning technologies used in the work, the data set and data processing used, and the evaluation metrics.

A. One-Class Classification Models

OCC is an important unsupervised approach in ML-based anomaly detection. It detects anomalies using data intrinsic structure, e.g., determined with clustering, to identify observations that are far from the center of any cluster or density estimations to locate observations that are unlikely to have the same distribution as the rest of the data. This approach is different from traditional binary and multi-class classification methods and can be useful for data sets that have a large imbalance between classes [6]. In the context of ML-based network intrusion detection, obtaining samples of normal network traffic is relatively easy. However, obtaining samples of abnormal or malicious traffic can be more challenging. OCC can be an effective solution in these situations, as it requires only samples from the normal class for training. The OCC models used in this analysis include Elliptic Envelope, Isolation Forest, OCSVM, and Local Outlier Factor.

1) Elliptic Envelope : This is a method for fitting a multivariate Gaussian distribution to a data set, with the goal of identifying outliers. The method fits an ellipse (or, more generally, an ellipsoid) to the data, with the ellipse bounding the ”normal” region of the data, and points that lie outside are signaled. The size and shape of the ellipse are determined by the covariance matrix of the Gaussian distribution, which is estimated from the data. Points with a low Mahalanobis distance from the mean of the Gaussian distribution are considered to be within the elliptic envelope. The boundary between these two categories is determined by a threshold [11].

2) Isolation Forest (iForest): This method builds multiple Isolation Trees (iTrees). Each iTree is random binary tree. The root of each iTree is a sample of the training data set. The set of each node is split in two, by randomly selecting an attribute and a split value. The process is recursive and stops either when a singleton set is reached or the height of the tree exceeds the average tree height.

To classify a data point x, it is added to each of the trees of the iForest, and the length of the path from the data point to the root of the respective tree is used to compute the average path length to the root, which is then used to compute an anomaly factor, between 0 and 1. The higher the anomaly factor the more likely the data point is an anomaly.

TABLE I
Combined data set from multiple sources for either benign or malign classes

Source	Data	Description	Class
UNSW [13]	UNSW-IoT #1	Amazon Echo, Netatmo camera, Lixf light, WEMO power switch	**Benign**
	UNSW-IoT #2	iHome, Samsung camera, Hue bulb, TP Link plug	
IoT_23 [14]	IoT-23 - normal	Amazon Echo, Hue bulb, Somfy door lock	
NOS	NOS - IoT	Smart plug, temperature and humidity sensors	
	NOS - benign	Manual capture of web traffic	
Botnet_IoT [15]	Bot-IoT - TCP DDoS	TCP floods generated using hping3	**Malign**
	Bot-IoT - HTTP DDoS	HTTP request floods using Golden-eye	
	Bot-IoT - OS Scan	OS fingerprinting using Nmap and Xprobe2	
	Bot-IoT - Service Scan	Different port scans using Nmap and Hping3	
	Bot-IoT - Keylogging	Record keystrokes with Logkeys or exploits	
	Bot-IoT - Data Theft	Directory exfiltration through system exploits	
IoT_23 [14]	IoT-23 - DDoS	DDoS attack part of a botnet	
	IoT-23 - Port Scan	Port scans part of a botnet	
	IoT-23 - C&C	communication part of a botnet	

3) Local Outlier Factor (LOF): This is a data mining algorithm for identifying local outliers in data by measuring the deviation of the density of a data point to its neighbors. It uses the K-Nearest Neighbors method to calculate the reachability distance and local reachability density of each data point. The LOF score of a data point is calculated as the ratio of its local reachability density to the densities of its k nearest neighbors. Points with high LOF scores are considered outliers. The value of k, the number of nearest neighbors, must be chosen carefully to avoid overfitting or underfitting [12].

4) One-class Support Vector Machine: This model determines the minimum volume hypersphere containing all the normal data points. To allow for outliers, the method adds a positive parameter ν and one non-negative slack variable per training data-point. The solution of this minimization problem is an hyperplane that separates normal data points from anomalies.

B. Data Set

This study expands on previous research [10] and uses the same data composed of various public network data in `pcap` format. The data consists of network packets collected from sources such as IoT devices, IoT sensors and web captures and is available for academic purposes (except the *NOS* data set, which was produced by NOS, a portuguese ISP and cannot be publicly disclosed at the moment). Table I provides information on the sources and devices of the `pcap` files used in the study.

The features of the raw network data of interest for anomalous traffic detection are extracted using the TCP Statistics Analysis Tool (`Tstat`) [16]. More specifically, `Tstat` is able to extract from network packets key characteristics of each TCP connection, or flow, such as the number of packets exchanged, the duration of the connection, and counts for different connection flags; see Table II for the features extracted.

The TCP flow features extracted with `Tstat` characterize TCP connections in detail. However, these features do not provide contextual information that are important, for example, to detect DoS attacks. To address this, Schumacher [10] generated from the original `Tstat` features seven other features that relate TCP connections within a given time window, providing a deeper insight into the state of the network. Table III summarizes these extra features. More details with feature generation can be found in [10, p. 24].

TABLE II
TSTAT FEATURES FOR TCP FLOWS (OR CONNECTIONS)

Feature	Unit
Packets from client/server	-
RST sent by client/server	0/1
ACK sent by client/server	-
PURE ACK sent by client/server	-
Unique bytes in the payload sent by client/server	bytes
Data packets with payload sent by client/server	-
Data bytes in the payload sent by client/server	bytes
Retransmitted packets sent by client/server	-
Retransmitted bytes sent by client/server	bytes
Out of sequence packets sent by client/server	-
SYN packets sent by client/server	-
FIN packets sent by client/server	-
Completion time	ms
Client first payload since start	ms
Server first payload since start	ms
Client last payload since start	ms
Server last payload since start	ms
Client first ACK since start	ms
Server first ACK since start	ms
Connection type as a bitmap	-
P2P protocol type	-
HTTP type	-

TABLE III
ADDITIONAL FEATURES GENERATED FOR TCP FLOWS

Feature	Unit
Rate of flows with same origin IP	flows/s
Rate of bytes with same origin IP	bytes/s
Rate of packets with same origin IP	pkts/s
Rate of flows with same destination IP and Port	flows/s
Rate of bytes with same destination IP and Port	bytes/s
Rate of packets with same destination IP and Port	pkts/s
Percentage of flows that are complete	-

In this study, the data set includes about one million data points (TCP flows) of normal traffic (benign) and about half a million data points of malign or attack traffic. Figure 1 shows the distribution of benign and malign points in the data.

1) Data Preprocessing: The data set was reduced in dimension by anonymizing features such as IP addresses and timestamps that had no statistical significance to the model. To ensure accurate and efficient results from machine learning models, the network data was preprocessed. Features' values were normalized using Min-Max scaling, which transformed data into a consistent range between 0 and 1, making all features uniform in scale. This normalization step is crucial because many ML algorithms are sensitive to the scale of the data and could produce poor results if the features had different ranges. Since the `Tstat`-generated features are numerical data types, feature mapping was not necessary for this data set.

C. Evaluation Metrics

The classification models performance was evaluated using the metrics listed below considering the following definitions:

- True positive (TP): Correctly marked malign data points;
- True negative (TN): Correctly marked benign data points;
- False positive (FP): Wrongly marked benign data points;
- False negatives (FN): Wrongly marked malign points.

a) Accuracy: Ratio of correctly identified benign and malign cases in the entire data set.

b) Detection rate (Recall) : Ratio of malign data points the model has correctly identified.

c) False Positive Rate: Ratio of benign data points misidentified as malign, resulting in a false alarm.

d) True Negative Rate (a.k.a. Specificity): Ratio of correctly identified benign traffic.

e) Classification time : This is the average time required for the model to classify a TCP flow as either normal or an anomaly. This metric is crucial in applications where fast responses are needed, such as real-time applications. It was computed by measuring the time required to process each test data set using Python's `timeit` function and then dividing the measured time by the number of TCP flows in the test data set. All values are reported in microseconds.

f) Model Size : This is the amount of memory required to store a model. Smaller models are more desirable, as it speeds up deployment and reduces cost of storing the model, a fundamental aspect when deploying ML on constrained resource devices. This metric was measured using the `getsizeof` function from Python's `sys` module.

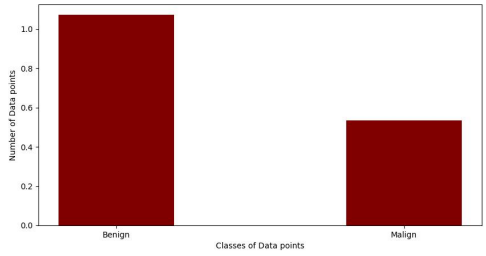

Fig. 1. Sample distribution between benign and malign classes

III. EXPERIMENT AND RESULTS

To evaluate the models we used the metrics described in Sec. II.C, and conducted two sets of experiments. In the first set of experiments, we study the effect on the model performance of using multiple disjoint training data sets of the same size. In the second set of experiments, we consider the effect of using training data sets of different sizes on the model performance. In both cases, the trained models are then used to classify the same test data set, and we compare the metrics obtained for each model.

All experiments were carried out on an Ubuntu machine with $8GB$ of *RAM*, a 4-core CPU Intel(R) Core(TM) i5-4570 with a clock speed of 3.20 GHz and a nominal clock-rate of 1.80GHz, using Python 3.8 and the `scikit-learn` library.

The combined data set used as input sample for all experiments contained 1,071,516 benign data points, i.e. normal network flows, and a little over 500,000 malign data points.

The models were trained using solely normal network flows. From the input data set we generated the following disjoint subsets:

- Training set: benign only, consisting of 50% of the whole benign sample (535,758 data points);
- Validation set: mixed benign/malign, containing 15% of the whole benign traffic sample (160,727) plus an equal number of malign data points (321,454 points total);
- Testing set: mixed benign/malign, containing 35% of the whole benign traffic sample (375,031) plus an equal number of malign data points (750,062 points total).

Hyperparameter tuning was performed with both training and validation sets. The best parameters for each model are discussed next and were found through grid search:

- Elliptic Envelope uses the *contamination* parameter to set the fraction of outliers in the data. Lower values indicate that the model is more sensitive to outliers detection. We set the *contamination* to 0.02857.
- The iForest model also uses the parameters *max-features* and *max-samples* to control the number of features and samples used for each split in the decision tree, respectively. In this case, *max-features* is set to 0.5 and *max-samples* is set to 0.3. The parameter *n-estimators* is used to set the number of decision trees in the model and is set to 10 and the *contamination* parameter used to set the proportion of outliers in the data was best at 0.01.
- LOF uses the *leaf-size* and *n-neighbors* parameters to control the number of points stored in each leaf node of the k-distance tree and the number of neighbors to use for density estimation, respectively. The parameter *novelty* is used to set the model to novelty detection mode, because the training data set contains no malign TCP flows. The *leaf size* is best at 3 and the *n-neighbors* is 1.
- OCSVM uses the parameters γ and ν to control the width of the decision boundary and the proportion of training errors, respectively. In this case, both γ and ν are set to 0.001.

Both sets of experiments used the hyperparameter values just described because hyperparameter tuning is very time-consuming and the hyperparameter values affect less significantly the resources, i.e. the classification time and the model size, required for ML inference. On the other hand, they may have a significant effect on classification metrics such as detection rate and false alarm rate, therefore the values reported for these values should be interpreted with caution. To partially compensate for this limitation, we also carried out one experiment per classification algorithm, with a model that was trained using a data set of 4,000 data points and whose hyperparameters were tuned using a validation data set of 8,000 data points [10].

A. Training with different data sets of the same size

This set of experiments aimed at investigating how much sensitive the performance of the model is with respect to the specific training set while keeping its size constant. Thus, from the original training set we extracted five disjoint subsets, each with 100k data points. The models were then trained with each subset separately and finally tested with the full testing set described before.

Fig. 2 shows the box-plots for the classification time and model size. As shown, these metrics are insensitive to the training data sets. Overall, LOF has the highest model size, with an average of 37.4 MB, and classification time, with an average of 142 μs. The results of the other models are more than one order of magnitude smaller than those for LOF. Nevertheless, the LOF classification time is still sufficiently low to allow for real-time intrusion detection.

Fig. 3 shows the box-plots for the detection rate and false alarm rate. Again, these plots show that both metrics have low sensitivity to the training subsets. There is no clear winner: although LOF has a higher detection rate, between 1 and 2 percent, this comes at the cost of a false alarm rate of almost 20%, between 15 and 20 percent higher than the other models. Note that all models displayed a detection rate above 96%, which is very high.

B. Training with data sets of growing size

In this set of experiments we evaluate the models trained with sets of variable size using the same metrics. The training data sets were built extracting the desired number of points from the benign traffic in the input sample. This way we generate training subsets starting with 100k data points and then adding other 100k points until reaching 600k, the largest subset. In the construction of the training sets, we made sure there was no overlap with the testing set, which, we recall, is kept constant across all experiments.

a) Classification time: Fig. 4 shows the variation of the classification time with the four models trained as described. The Elliptic Envelope and iForest models exhibit low classification times across all training set sizes with negligible absolute variation. The former shows an average classification time of 0.76 μs while the latter shows an average classification time of 3.5 μs. On the other hand, the classification time of both OCSVM and LOF models clearly grow with the training set size. The former shows some fluctuations despite a general behavior close to linear, from 7.25 μs to 40.98 μs. The latter has a more linear behavior but exhibits a much larger magnitude, from 142 μs to 1109.1 μs. Although OCSVM and LOF are clearly slower than the other models, the latter by more than two orders of magnitude, the classification times are in the order of the network latency, thus being suitable for real-time classification.

b) Model size: The results concerning model size are reported in Fig. 5. The situation is somewhat similar to the results with classification time, but with an interesting switch between OCSVM and the Elliptic Envelope. In this case, OCSVM and iForest show the lowest sizes, the former varying approximately linearly from 50kB to 260kB and the latter varying between 252.3kB and 630kB but with fluctuations. On the other hand, Elliptic Envelope grows steadily from slightly above 1MB to 6.7MB and again LOF shows the largest values

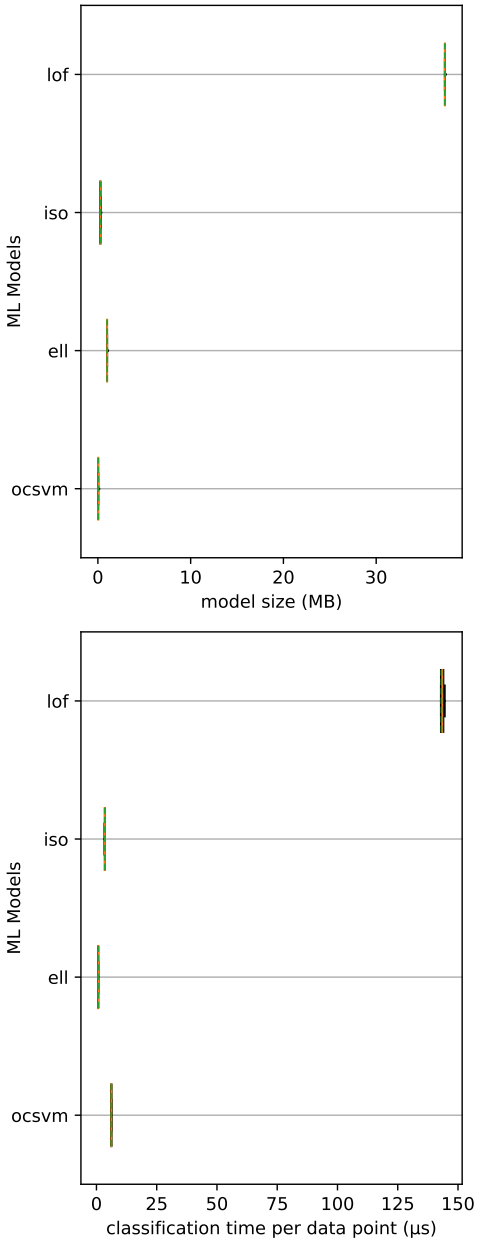

Fig. 2. Variability in model size and classification time for different models

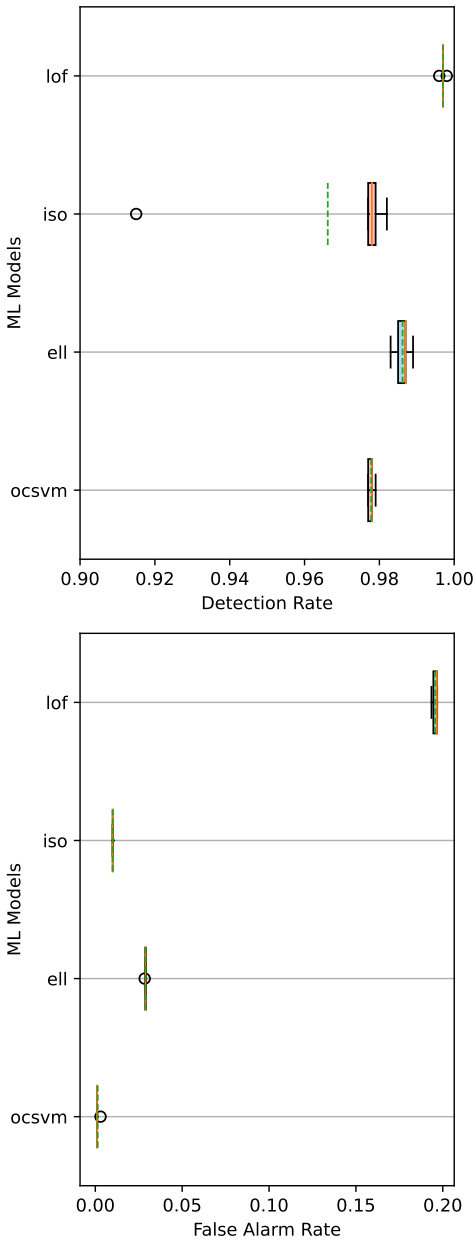

Fig. 3. Variability in detection rate and false alarm rate for different models trained with different sets of similar size (note the different scales)

varying steadily from approximately 50MB to over 260MB. The choice of algorithm can have a significant impact on the model size and developers must carefully consider the trade-off between model accuracy and memory usage when selecting an algorithm for a specific application when the available resources are scarce.

c) Detection Rate: Fig. 6 shows the detection rate of the models as the size of the training data set increases. The results show that all models offer high detection rates, between 0.977 and 1. Within this short range, OCSVM has the lowest detection rate and is practically insensitive to the training set size. Elliptic Envelope exhibits some fluctuations slightly above 0.98. iForest seems to indicate a tendency to increase detection rate with growing training set size, but the increase is not monotonic and it is rather small, from 0.977 to 0.984. Finally, LOF shows a negligible growth trend in detection rate at the top of the scale, from 0.997 to 1. These results suggest that the performance of each model varies with the size of the training set and the best training set size may differ for each model.

d) False Positive Rate: Fig. 7 shows the false positive rate against the size of the training data set. In this metric, all models but LOF present very low sensitivity to the training set size exhibiting low steady values across the whole tested

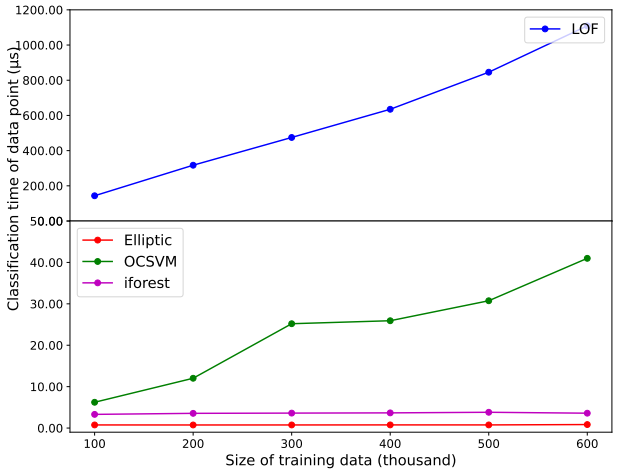

Fig. 4. Classification time against training set size

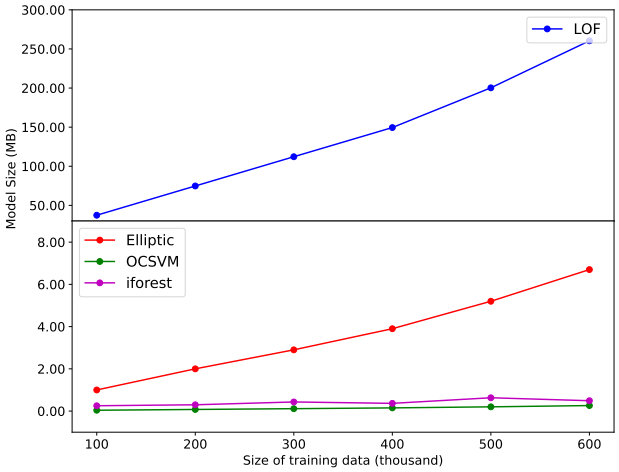

Fig. 5. Model size against training set size

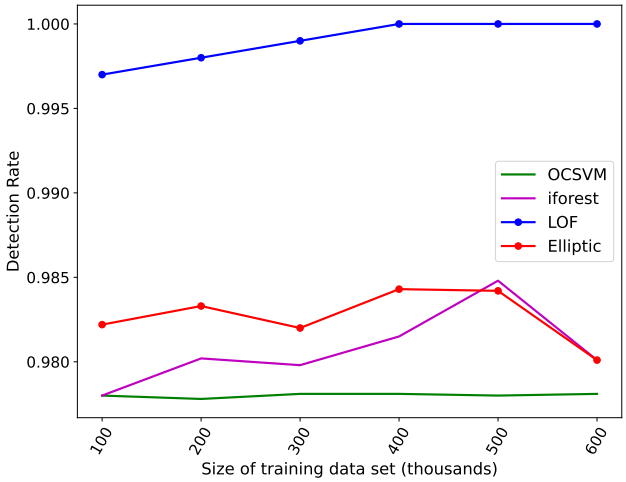

Fig. 6. Detection rate against training set size

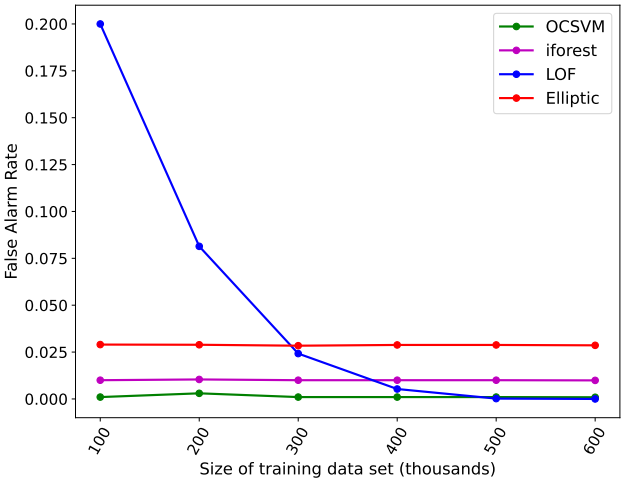

Fig. 7. False alarm rate against training set size

space. Elliptic Envelope has an FPR of 0.029, iForest 0.01 and OCSVM 0.001. Conversely, LOF starts with a relatively high value near to 0.2, decreasing quickly to 0, a value reached for the 500k training set size. This suggests that LOF benefits significantly from having a larger training set, and this finding highlights the importance of selecting an appropriate model and training set size when using one-class machine learning models for anomaly detection in IoT networks.

e) Training data set with 4k data points: All the results reported so far used models with hyperparameters tuned using a training set of about 500k data points, and a validation set of about 300k data points. To provide some insight on the effect of the sizes of the training and validation data sets in the hyperparameter values, we also run one experiment for each of the algorithms under analysis using the models developed in [10]. These models were developed using the methodology described in Sec. II, including the data sets shown in Table I. However, they were trained using a data set with 4k data points, and their hyperparameters were tuned using a validation data set with 8k data points. We tested these models using the test data set used for testing the other models. For easier comparison we included in Table IV the results of these experiments together with the others already reported in this section. The results with such small training set corroborate trends observed in our own results, but also uncover new trends that did not show up in the space we explored.

For all algorithms, the classification time and the model size present values consistent with our previous observations. Consequently, OCSVM's classification time becomes smaller than that of any of the other models, and its model size is still the smallest. In spite of that, OCSVM's detection rate and false alarm rate are similar to those obtained with larger training set sizes and better than those of all other algorithms, except for the detection rate of LOF, which is very close to 1. However, LOF's high detection rate is achieved at the cost of

a false alarm rate very close to 1, which is not acceptable for intrusion detection. On the other hand, both Elliptic Envelope and Isolation Forest present very small false alarm rates, at the cost of lower detection rates, which are, respectively, more than 10% and 20% lower than those obtained with larger training sizes. This suggests that, to achieve classification metrics similar to OCSVM, all other algorithms require larger training sets.

C. Concluding comments

In summary, all models showed a high detection rate with very little dependence on the training set size, except for very small sizes that degraded this metric. Similarly, all models consistently showed very low false positives rate, except LOF that was negatively affected by small training sets. Model size grows with the training set size for all models. In fact, models construct their decision boundaries and set parameters based on the training data. As the size of the training data grows, the models become larger and more complex. This also explains the variability observed in iForest that uses random data partitioning, thus affecting the model size in random ways. Concerning the classification time, the growing internal model complexity does not affect Elliptic Envelope and iForest. However, it affects OCSVM and LOF causing the classification time to grow, even if the testing set is kept constant. These results also show that LOF is the most resource consuming model requiring large training sets for low false alarm rate, while OCSVM performs well with very small training sets, revealing very small memory requirements and classification time, showing good suitability to deployment in resource-constrained devices. For similar detection rate and false alarm rate, both Elliptic Envelope and iForest require larger training sets with higher resource requirements. However, the latter two models show a more deterministic classification time, which is practically independent from the training set size. This can be of interest in deployments within real-time systems. Finally, we recall that we did not retune the models hyperparameters for the different training set sizes, which could mitigate some of the effects we observed. Studying the resource impact of hyperparameters tuning will be addressed in future work.

IV. Related Work

To the best of our knowledge, our work is the first to focus on resource utilization metrics in the ML inference phase of OCC ML models. This is particularly important for applications running on edge devices, which have limited resources, whereas ML training can be done in more resourceful platforms. Virtually all other works focus on the classification metrics, and a few of them on resource utilization during the ML training phase. In this section, we review other works that applied ML for anomaly detection in IoT networks.

A network anomaly detection method based on deep learning was presented in [17] using autoencoders. The method calculates anomaly scores using a combination of reconstruction losses and Mahalanobis distances and detects anomalies by applying a threshold to these scores. The proposed model was tested using UNSW-NB15 data and outperformed other models in all performance evaluations, with a precision of 0.85 and a false positive rate of 0.042. In [5], the authors used the OCSVM classifier for anomaly detection in a home network of IoT devices. The model was found to be effective in detecting DDoS threats, as demonstrated by F1 scores between 91.58% and 99.67%. In a study for home IoT intrusion detection using OCC models [10], the average performance for identifying benign data was 95.2% and the average detection rate for malign data was 89.9% after combining the results from 5 OCC models through majority voting.

The work in [18] evaluated seven supervised ML models for the detection of network intrusions using the TON-IOT data set. The results showed that the decision tree (CART) model had the highest accuracy score of 0.88 with a classification time of 0.022 seconds, while the support vector machine (SVM) had the lowest accuracy score of 0.61 with a longer training and testing time of 3525 and 558 seconds, respectively. However, the study only considered the classification time as a metric and did not take into account the memory requirements of the models, which is crucial for edge deployments.

V. Conclusions

When selecting an appropriate machine learning technique to deploy on an edge device, it is important to consider the relationship between the size of the model and the computational resources to ensure that the model can be trained and deployed efficiently without overloading the system and provide an adequate quality of service, particularly detection time. In this study we examined the effectiveness of various one-class machine learning techniques for detecting network intrusions in a home IoT environment. Our aim was to determine which method provided the best balance between detecting attacks and minimizing false alarms while being efficient in terms of classification time and memory requirements. Our results showed that the OCSVM method was highly effective in detecting attacks with a low false positive rate. The use of resources, despite growing with the training set size, was always relatively small and the detection performance good even with very small training sets. Both the Isolation Forest and the Elliptic Envelope methods led to larger models with larger training sets, particularly the latter, but exhibited steady and low classification times. Finally, the LOF method provided the best detection performance at the cost of a rather large model and long classification times that increased with the training set size, thus a poor fit for our purposes.

In future work, we will investigate the impact of hyperparameter tuning on resource usage, as well as the behavior of the models with training set sizes below 100k data points. Additionally, we plan to explore the potential of neural network models, such as autoencoders, in edge computing. To this end, testing these models on edge devices and replicating the experiments with real-world datasets will be valuable in verifying if the conclusions remain valid.

TABLE IV
SUMMARY OF MODELS PERFORMANCE WITH VARYING TRAINING SET SIZE

Training data set size	Accuracy	Detection rate	FPR	TNR	Classification time per data point (μs)	Model size
OCSVM						
4,000 [10]	0.9878	0.9762	0.0006	0.9994	0.431	3.3kB
100,000	0.988	0.978	0.001	0.9989	7.252	40.1kB
200,000	0.9874	0.9778	0.003	0.997	12.0256	76.4kB
300,000	0.9885	0.9781	0.001	0.999	25.196	113.6kB
400,000	0.9885	0.9781	0.001	0.999	25.917	150.7kB
500,000	0.9885	0.978	0.001	0.999	30.744	201.1kB
600,000	0.9886	0.9781	0.0009	0.9991	40.989	260.9kB
Elliptic Envelope						
4,000 [10]	0.9367	0.8798	0.0064	0.9936	0.755	90.9kB
100,000	0.9788	0.9826	0.029	0.970	0.758	1MB
200,000	0.9772	0.9833	0.0289	0.9711	0.742	2MB
300,000	0.9768	0.982	0.02839	0.9716	0.748	2.9MB
400,000	0.9778	0.9843	0.0288	0.9712	0.761	3.9MB
500,000	0.9777	0.9842	0.0288	0.9712	0.757	5.2MB
600,000	0.9757	0.9801	0.0286	0.9713	0.846	6.7MB
Isolation Forest						
4,000 [10]	0.8823	0.7651	0.000499	0.9995	2.749	119.7kB
100,000	0.9863	0.9827	0.01	0.99	3.517	252.3kB
200,000	0.9848	0.9802	0.0104	0.9895	3.551	297.9kB
300,000	0.9848	0.9798	0.01	0.9899	3.610	431.6kB
400,000	0.9857	0.9815	0.01	0.99	3.666	365.6kB
500,000	0.9873	0.9848	0.01	0.9899	3.810	630.1kB
600,000	0.9851	0.9801	0.0099	0.9901	3.588	490.5kB
Local Outlier Factor						
4,000 [10]	0.502	0.9985	0.9945	0.0055	5.654	1.5MB
100,000	0.90	0.9978	0.9706	0.197	142.1	37.4MB
200,000	0.9586	0.9987	0.0814	0.9186	317.053	74.8MB
300,000	0.9878	0.999	0.0242	0.9758	474.767	112.2MB
400,000	0.9973	1.0	0.0053	0.9946	634.987	149.5MB
500,000	0.9999	1.0	0.00019	0.99998	845.610	200.3MB
600,000	1.0	1.0	0.0	1.0	1109.033	260.4MB

ACKNOWLEDGEMENTS

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDB/04234/2020), and by the Portuguese National Innovation Agency (ANI) through the Operational Competitiveness Programme and Internationalization (COMPETE 2020) under the PT2020 Partnership Agreement, through the European Regional Development Fund (ERDF), within project(s) grant nr. 69522, POCI-01-0247-FEDER-069522 (MIRAI).

REFERENCES

[1] M. Azrour, J. Mabrouki, A. Guezzaz, and A. Kanwal, "Internet of things security: Challenges and key issues," *Secur. Commun. Networks*, vol. 2021, pp. 5 533 843:1–5 533 843:11, 2021.

[2] G. Abdelmoumin, D. B. Rawat, and A. Rahman, "On the performance of machine learning models for anomaly-based intelligent intrusion detection systems for the internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 6, p. 4280 – 4290, 2022, cited by: 5.

[3] S. Kumar, S. Gupta, and S. Arora, "Research trends in network-based intrusion detection systems: A review," *IEEE Access*, vol. 9, pp. 157 761–157 779, 2021.

[4] J. Simon, N. Kapileswar, P. K. Polasi, and M. A. Elaveini, "Hybrid intrusion detection system for wireless iot networks using deep learning algorithm," *Computers and Electrical Engineering*, vol. 102, p. 108190, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790622004323

[5] J. White and P. Legg, "Unsupervised one-class learning for anomaly detection on home iot network devices," in *2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. IEEE, 2021, pp. 1–8.

[6] P. Arregoces, J. Vergara, S. A. Gutiérrez, and J. F. Botero, "Network-based intrusion detection: A one-class classification approach," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–6.

[7] Y. Becerikli, H. Qu, L. Lei, X. Tang, and P. Wang, "A lightweight intrusion detection method based on fuzzy clustering algorithm for wireless sensor networks," *Advances in Fuzzy Systems*, vol. 2018, p. 4071851, 2018. [Online]. Available: https://doi.org/10.1155/2018/4071851

[8] A. E. Eshratifar and M. Pedram, "Energy and performance efficient computation offloading for deep neural networks in a mobile cloud computing environment," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 2018, pp. 111–116.

[9] N. Schumacher, P. M. Santos, P. F. Souto, N. Martins, J. Sousa, J. M. Ferreira, and L. Almeida, "One-Class Models for Intrusion Detection at ISP Customer Networks," in *IFIP Advances and Innovations in Artificial Intelligence*, León, Spain, Jun. 2023.

[10] N. Schumacher, "Anomaly detection models for cloud-edge intrusion detection in customer networks," Dissertation, Universidade do Porto, 2022. [Online]. Available: https://hdl.handle.net/10216/142591

[11] S. Shriram and E. Sivasankar, "Anomaly detection on shuttle data using unsupervised learning techniques," in *2019 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*, 2019, pp. 221–225.

[12] Z. Gan and X. Zhou, "Abnormal network traffic detection based on improved lof algorithm," in *2018 10th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 01, 2018, pp. 142–145.

[13] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *Military Communications and Information Systems Conference (MilCIS), 2015 IEEE*. IEEE, 2015, pp. 1–6.

[14] S. Garcia, A. Parmisano, and M. J. Erquiaga, "Iot-23: A labeled dataset with malicious and benign iot network traffic," 2020. [Online]. Available: http://doi.org/10.5281/zenodo.4743746

[15] J. Ashraf, M. Keshk, N. Moustafa, M. Abdel-Basset, H. Khurshid, A. D. Bakhshi, and R. R. Mostafa, "Iotbot-ids: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities," *Sustainable Cities and Society*, vol. 61, p. 103041, 2021.

[16] "Tstat," http://tstat.polito.it/, accessed on Feb 1, 2023.

[17] D. Yang and M. Hwang, "Unsupervised and ensemble-based anomaly detection method for network security," in *2022 14th International Conference on Knowledge and Smart Technology (KST)*. IEEE, 2022, pp. 75–79.

[18] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems," *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020.