An Early Exit Deep Neural Network for Fast Inference Intrusion Detection

João A. Simioni
joao.asimioni@ppgia.pucpr.br
Pontifical Catholic University of Paraná
Curitiba, Brazil

Eduardo K. Viegas
eduardo.viegas@ppgia.pucpr.br
Pontifical Catholic University of Paraná
Curitiba, Brazil

Altair O. Santin
santin@ppgia.pucpr.br
Pontifical Catholic University of Paraná
Curitiba, Brazil

Pedro Horhulhack
pedro.horhulhack@ppgia.pucpr.br
Pontifical Catholic University of Paraná
Curitiba, Brazil

Abstract

Deep Neural Networks (DNN) are currently state-of-the-art in intrusion detection literature, where authors typically escalate the network parameters to pave the way for accuracy improvements. However, in addition to increasing the inference computational costs, this can also render them unsuitable for resource-constrained devices, given their limited memory and processing capabilities. This paper introduces a new early exit neural network for fast inference and reliable intrusion detection. Our proposal partitions the DNN utilized for intrusion detection into branches, with the objective of classifying the majority of samples on the earlier branches, thereby reducing inference costs. Challenging samples that reach the final DNN branch are subsequently classified using a reject option, improving system reliability. In addition, the branches and rejection thresholds are selected as a multi-objective optimization task. Experiments on a new dataset with over 8TB of data and a year-long real network traffic showed the proposal's feasibility. Our scheme reduces the average inference computational costs by up to 82% while decreasing the average error rates by up to 3.3.

CCS Concepts

• **Security and privacy → Intrusion detection systems**.

Keywords

Intrusion Detection, Neural Networks, Early Exits

ACM Reference Format:
João A. Simioni, Eduardo K. Viegas, Altair O. Santin, and Pedro Horhulhack. 2025. An Early Exit Deep Neural Network for Fast Inference Intrusion Detection. In *The 40th ACM/SIGAPP Symposium on Applied Computing (SAC '25), March 31-April 4, 2025, Catania, Italy*. ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3672608.3707974

SAC '25, March 31-April 4, 2025, Catania, Italy

ACM ISBN 979-8-4007-0629-5/25/03
https://doi.org/10.1145/3672608.3707974

1 Introduction

The utilization of resource-constrained devices, such as those in the context of Internet of Things (IoT), has consistently risen in recent years. These devices typically consist of battery-powered embedded computing systems with restricted processing capabilities, often featuring network connectivity. Due to their widespread adoption, these devices have become prime targets for cyber attackers, with reports indicating a 40% surge in attack incidents over the past year alone [1].

To safeguard IoT devices, a common approach involves the use of Network Intrusion Detection Systems (NIDSs), which monitors the device's network traffic for malicious footprints [16]. Over the past years, several techniques have been proposed to conduct such a task, wherein proposed schemes can usually be categorized as *misuse-based* or *behavior-based* schemes [3]. On the one hand, *misuse-based* approaches identify intrusion attempts by comparing them to a database of well-known malicious activities, limiting their ability to detect only previously known attacks. On the other hand, *behavior-based* techniques identify irregularities based on deviations from a previously established system profile, potentially detecting unseen attack types.

In response to this feature, several works have proposed new *behavior-based* techniques for intrusion detection, where approaches based on Deep Neural Networks (DNNs) frequently achieve the highest accuracy levels [21]. To accomplish such an objective, researchers, in their vast majority, escalate the parameters of their designed DNN architecture to pave the way for better accuracies. Besides increasing the inference computational costs, they also render them unsuitable for resource-constrained devices, given their limited memory and processing capabilities [6].

At deployment, DNN-based intrusion detection entails forwarding input parameters throughout the entire network architecture until the output layer is reached. Several spatial non-linear patterns and features are extracted in this process, later used as indicators for the classification task conducted at the output layer. Therefore, regardless of the evaluated event's complexity, be it complex or simple, the decision can only be made once all indicators have been extracted, potentially leading to the inefficient use of computational resources [12]. Considering this, in recent years, several studies have explored the introduction of early exits in DNN architectures. Early exits add multiple termination points that split the network into branches, enabling the inference task to conclude prematurely

if the current extracted patterns and features can reliably lead to a decision [18].

While early exits typically reduce the DNN inference computational cost with minimal impact on accuracy, they are not readily applicable to network intrusion detection, particularly on resource-constrained devices [11]. Unlike other domains, the behavior of network traffic is notably dynamic and continually evolves. The prior necessitates advanced DNN model generalization capabilities, while the latter can usually only be fixed through model updates [5]. Conversely, existing intrusion detection strategies based on early exits often fail to consider the tradeoffs in model generalization that arise from prematurely terminating the inference task. In addition, the classification unreliability resulting from an outdated DNN model due to changing network traffic behavior is generally assumed to be rectified through model updates [9]. However, this procedure frequently involves an extended time frame, often spanning days or even weeks, necessitating the deployed model to have a prolonged lifespan to ensure its reliability.

Contribution. In light of this, this paper presents a new early exit DNN aiming for fast inference time while keeping intrusion detection reliability. The proposed model is implemented following two strategies. First, we address early exits in DNN-based intrusion detection as a multi-objective optimization task. Our insight is to reduce model inference time and enhance the system detection accuracies by optimizing the selection of exit thresholds. Second, we introduce a classification approach with a reject option at the final DNN branch, allowing the rejection of potentially unreliable classifications influenced by new network traffic behavior. As a result, we ensure classification reliability through the classification with a reject option while simultaneously reducing inference computational costs by employing early exits.

In summary, the main paper contributions are:

- An evaluation of the classification reliability of a widely used DNN architecture for the intrusion detection task. Experiments on a dataset spanning a year of real network traffic revealed that current approaches demand impractical computational processing and suffer a reduction in accuracy in the months following the training.
- A new early exit DNN for reliable intrusion detection implemented through multi-objective optimization coped with a classification with a reject option. Our proposal can reduce the average error rate by as much as 3.3 while lowering inference computational costs by up to 82%.

Roadmap. The remainder of this paper is organized as follows. Section 2 further describes the intrusion detection application with DNNs. Section 3 describes the related works. Section 4 presents our proposal, while Section 5 evaluates its performance. Section 6 concludes our work.

2 Background

2.1 Network-based Intrusion Detection for IoT

Behavior-based NIDSs have been extensively used by network operators for monitoring the device's communication and detecting malicious activities [10, 16]. In general, these tools implement DNN-based intrusion detection following a four sequential module implementation, encompassing *Data Acquisition*, *Feature Extraction*, *Classification*, and *Alert*. The first module is responsible for the real-time collection of network packets from a specified Network Interface Card (NIC). The behavior of the collected data is extracted by the *Feature Extraction* module, which typically compiles a feature vector summarizing the communication between network entities within a specified time window. The resulting behavioral vector serves as input to the *Classification* module, which employs a previously trained DNN to classify events as either *normal* or *intrusion*. Lastly, the *Alert* module signals events classified as *intrusion*.

The application of DNNs for network traffic classification on resource-constrained devices requires the prior execution of the training task [19]. The behavior of a training dataset is extracted during the training phase. Thus, it should ideally consist of millions of labeled network samples representative of what is expected in a production environment. The resulting model's accuracy is assessed during the testing phase, and the measured accuracy rates are expected to indicate its performance in a production deployment.

2.2 Early Exits

Over the past years, the accuracy of proposed DNN-based intrusion detection techniques has consistently improved [8]. To pave the way for more accurate detection, researchers usually escalate the number of the DNN parameters. Consequently, implemented solutions frequently demand impractical inference computational costs, hindering their deployment on resource-constrained devices. Early exits are designed to tackle this challenge by introducing side branches that enable the premature termination of network inference [18]. Each side branch typically consists of fully connected layers that classify the input at the current layer. If an input sample has high confidence at a branch, it can exit from that branch without traversing the entire network, effectively reducing the depth of the network that a portion of the samples needs to traverse.

To conduct the DNN training procedure with early exits, researchers often resort to a joint training rationale [20]. Let N be the number of exit branches, and $\tilde{y}^i$ be the classification output of each branch i on a given event with label y. We can compute the joint loss function as a weighted sum of losses of each branch through the following equation:

$$\mathcal{L}_{joint}(\tilde{y}^i, y^i) = \sum_{i=1}^{N} w_i \mathcal{L}(\tilde{y}^i, y^i) \tag{1}$$

where $\mathcal{L}_{joint}$ is the joint loss function, $\mathcal{L}$ the loss function, and w_i the branch i weight. Here, w_i can be used to fine-tune the accuracies at each branch, such as allowing for a preference towards more accurate earlier branches, resulting in a lower overall computational cost.

At the test phase, the network inference task traverses the input event until it reaches the first branch, prematurely stopping the inference based on whether the classification threshold surpasses a given confidence threshold t. Typically, the acceptance threshold is established based on the operator's judgment, considering the desired trade-off between accuracy and average inference time. This step is repeated for each branch in the model, and if no early exit reaches the required confidence level, the final branch is reached and the event's class is determined based on the decision made at the final branch.

Early exits provide a compelling mechanism to manage the trade-off between inference speed and accuracy in deep neural networks. By introducing intermediate classification branches, they allow for faster processing of simpler inputs [7]. These earlier exits capitalize on the observation that not all inputs require the full complexity of a deep network to achieve accurate classification. Since they involve traversing a smaller portion of the network, they naturally lead to faster inference and reduced energy consumption, making them particularly attractive for resource-constrained environments.

However, this efficiency gain often comes at the cost of accuracy. Earlier exits have access to fewer features and less refined representations compared to the deeper layers of the network. This limitation can hinder their ability to discern subtle patterns or handle complex inputs, potentially leading to lower classification accuracy. Therefore, careful design and optimization of early exits are crucial to strike an effective balance between achieving faster inference and maintaining satisfactory accuracy levels. This often involves sophisticated training strategies and adaptive confidence thresholds to ensure that early exits are triggered only when a sufficient level of certainty is reached.

2.3 Multi-Objective Optimization

An optimization problem is a mathematical or computational problem where the goal is to find the best solution from a set of feasible solutions. The best solution is determined based on certain criteria, which are defined by an objective function. The objective function quantifies the performance or quality of a solution, and the optimization process aims to either maximize or minimize this function. Formally, an optimization problem can be defined as follows:

$$\begin{aligned} &Minimize\ or\ Maximize : f(x) \\ &x \in X \end{aligned} \tag{2}$$

where f(x) represents the objective function that needs to be optimized, x the decision variables, and X the set of possible values for x. The optimization problem may be subject to some constraint, which would limit the set X.

In traditional optimization problems, there is usually a single objective to be maximized or minimized, however, in many real-world scenarios, there are multiple criteria or goals that need to be considered, and these objectives may conflict with each other [2].

In the set of solutions, some of them are said to be dominated by another one, when another solution is equal to or better than the current one in all objectives. If the solution is not dominated by another one it is called Pareto optimal, and it means that it cannot be improved in any objective without compromising another one. The final step in the multi-objective optimization task is to find a Pareto optimal set, composed only of Pareto optimal options. In the Pareto optimal set, no solution can be said to be better than the other, without considering external factors. So, it depends on the system's operator to identify which solution to adopt from the set.

3 Related Work

Network-based intrusion detection through DNN-based approaches has been a popular research topic in the literature over the past years [16]. In general, the proposed schemes often prioritize improved detection accuracies while overlooking their practical applicability. For instance, Li Ma *et al.* [13] suggest the use of DNN for IoT intrusion detection via feature fusion. Although their approach enhances classification accuracy on an outdated dataset, it tends to overlook the associated inference costs and classification reliability issues. Another accuracy-focused approach was proposed by J. Zhang *et al.* [23], which resorts to a DNN ensemble for the detection task. Albeit the accuracy benefits, the utilization of multiple DNNs hampers their deployment on resource-constrained devices. M. Ge *et al.* [6] proposed the application of DNN to detect intrusions while considering the associated computational costs. Their proposal surpassed traditional shallow-based methods but overlooked the classification reliability. Some papers propose specific domain studies. M. Kang *et al.* [15] propose a DNN based intrusion detection system specifically designed for securing in-vehicle networks by analyzing network packets and identifying malicious activity in real-time. However, the inference computational cost is overlooked.

In recent years, recognizing the impracticality of applying these methods on resource-constrained devices, some research has shifted focus from accuracy to improving inference computational costs [3]. Y. Wang *et al.* [22] proposes the application of MobileNet, a light-weight DNN implementation, for intrusion detection. The authors showed that high accuracies can be reached while providing high detection throughput with minimal memory requirements. Unfortunately, the authors overlooked the evolving behavior of network traffic. While the application of early exits has been extensively researched in various domains [11], its adoption in intrusion detection is still in its early stages. En Li *et al.* [12] proposes the application of early exits on intrusion detection to improve detection throughput. Their approach notably enhances inference time, yet it tends to neglect the influence of network traffic behavior changes on the reliability of their solution. A similar approach was proposed by W. Seifeddine *et al.* [18] which creates inference branches to split the DNN on multiple devices. Although the authors can decrease inference computational costs, they overlook how the non-stationary behavior of network traffic can affect their scheme. As a result, there is a research gap in the literature for intrusion detection that considers the evolving behavior of network traffic while also addressing the computational inference costs.

4 Early Exit DNN for Fast Inference and Reliable Intrusion Detection

To tackle the evolving behavior of network traffic while minimizing processing demands, we introduce an intrusion detection model realized through early exits and classification with a reject option. The overview of the proposed model is shown in Figure 1 and is implemented following two main strategies, namely *Rejector*, and *Multi-objective optimization*.

The goal of the *Rejector*'s module is to identify unreliable classifications at the DNN final branch, even if outdated. Unreliable classifications are attributed to new network traffic behaviors that can cause an increase in misclassification rates. As a result, in light of the extended timeframe required for obtaining an updated model, we utilize the classification with a reject option to further extend the model's lifespan. We assume that classification confidence can

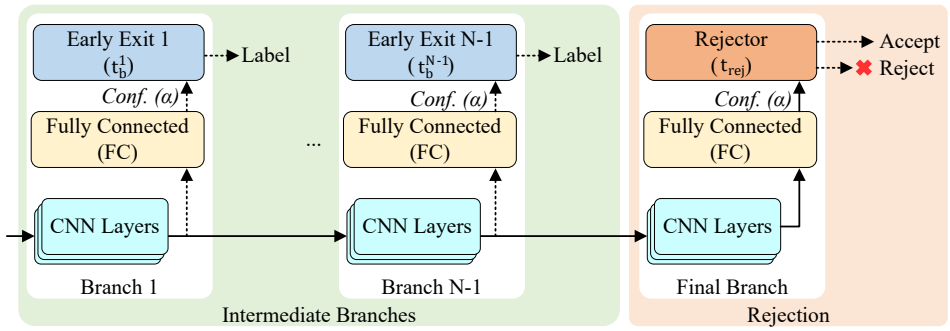


Figure 1: Overview of the proposed reliable intrusion detection model based on early exits for fast inference.

serve as a measure of classification quality, and thus, the rejection of low-confident classifications can aid in efficiently enhancing the system's reliability.

The goal of the *Multi-objective Optimization* is to proactively determine both the acceptance threshold for the DNN branches and the rejection threshold for the final branch (see Figure 1, denoted as t_b and t_{rej} thresholds). Our main insight is to optimize the early exit thresholds based on the network traffic behavior while simultaneously enhancing system reliability through classification with a reject option. As a result, our model can reduce the associated inference computational costs while ensuring system reliability.

To measure the exit confidence, be it for choosing the early exit or to decide to reject that sample, we used the softmax function on those outputs, which turns the model's raw output into probabilities. These probabilities tell us how sure the model is about its classification at that stage.

The following subsections provide a detailed description of our proposed model.

4.1 Classification with Reject Option

The network traffic behavior is highly variable and also evolves as time passes. Considering the non-stationary nature of network traffic behavior, DNN-based NIDSs must undergo regular updates, a procedure that frequently takes several weeks or even months to complete. In the meantime, the model deployed in the production environment, although outdated, must maintain its classification reliability.

To tackle such a challenge, we implement the classification procedure through a classification with a reject option to increase the model's lifespan. The proposed model adheres to the conventional DNN-based NIDS pipeline implemented through early exits (Fig. 1). Given a DNN model with N branches, where the final branch produces a classification confidence value $\alpha = \{\alpha_{normal}, \alpha_{intrusion}\}$ for each input event x. Here, α_{normal} denotes the event confidence to for a *normal* class, and $\alpha_{intrusion}$ the event confidence for *attack*, such that $\alpha \in \mathbb{R}[0,1]$. The *Rejector* module accepts or rejects the classification based on its associated confidence values α *vs*. a previously defined rejection threshold t_{rej} (see Fig. 1). To achieve such a goal, the module's implementation is coped with the rejection function, as determined by the following equation:

$$Rejector(\alpha, t_{rej}) \begin{cases} \emptyset & \text{if } \alpha \leq t_{rej} \\ \alpha & \text{otherwise} \end{cases} \tag{3}$$

where $\emptyset$ denotes events likely to be incorrect decisions the DNN model final branch performs. As a result, the *Rejector* module suppresses unreliable classification as measured by their associated classification confidence values at the final DNN branch. It is important to note that the rejection threshold should be determined based on the network operator's judgment. A higher rejection threshold will enhance system reliability but result in a higher proportion of rejected events. Conversely, a lower threshold will accept more events but also expose the system to potentially unreliable classifications.

4.2 Multi-objective Optimization

Network intrusion detection for resource-constrained devices should be performed with minimal processing requirements while maintaining classification accuracy and reliability. To address such a challenge, we conduct the model building as a multi-objective optimization task.

We consider a DNN model h implemented with N branches, coped with a *Rejector* module at the final branch (see Section 4.1). In such a case, the goal of the multi-objective optimization is to find $N-1$ associated t_b branches thresholds, along with a t_{rej} *Rejector* threshold. Hence, we consider two objectives based on the system's processing costs and accuracy. We assume that the first objective is a direct result of accepting an additional number of events at earlier branches, whereas the second objective relates to increasing the rejection rate, which, in turn, raises processing costs due to more events reaching the final branch. Therefore, we can conduct the multi-objective by solving the following equation.

$$\begin{aligned} &\underset{t_{rej},\{t_b^1,\ldots,t_b^{N-1}\}}{\arg\min} \quad time(h(\mathcal{D}, t_{rej}, \{t_b^1, \ldots, t_b^{N-1}\})) \\ &\text{and} \\ &\underset{t_{rej},\{t_b^1,\ldots,t_b^{N-1}\}}{\arg\min} \quad error(h(\mathcal{D}, t_{rej}, \{t_b^1, \ldots, t_b^{N-1}\})) \end{aligned} \tag{4}$$

where h denotes the DNN model with multiple branches, coped with our *Rejector* (see Fig. 1). Here, the *time* measures the model h computational inference time on a given dataset $\mathcal{D}$ when using a rejection threshold t_{rej} and a set of t_b branches thresholds. In turn,

Figure 2: Representation of the tabular to matrix conversion used in the *MAWIFlow* Dataset samples.

the *error* measures the error rate with the same set of thresholds. As a result, our proposed scheme aims to identify the optimal system thresholds that enhance computational inference time while minimizing error rates.

5 Evaluation

In this section, we delve deeper into the performance of DNN techniques in relation to accuracy and processing requirements when confronted with changes in network traffic behavior. More specifically, we initially introduce the dataset utilized in the experiments conducted in this paper, followed by an assessment of the performance of DNN techniques on this dataset. Later, we evaluate the results from our proposed model, comparing with the results from the traditional approach.

The conducted set of evaluations aims to answer the following Research Questions (RQs):

- (***RQ1***) *What is the intrusion detection performance of widely used DNN techniques?*
- (***RQ2***) *How does our proposed multi-objective optimization improves system reliability?*
- (***RQ3***) *How can our rejector technique improve classification performance?*
- (***RQ4***) *What is the computational costs of our proposed scheme?*

5.1 MAWIFlow

To ensure a realistic evaluation, we utilize *MAWIFlow* dataset, a publicly available intrusion dataset containing real, valid, and labeled network traffic from production environments spanning an extended period. To achieve these characteristics, the dataset is built upon MAWI [14] working group traffic archive. It includes network traffic from MAWI samplepoint-F, a transit link between Japan and the USA, collected in 15-second intervals daily. The network data is collected daily, resulting in a network PCAP file for each day over the evaluation period, totaling over 7TB of data and encompassing more than 70 billion network flows. For this research, the network data collected throughout the entire year 2016 was employed. This collected data is organized into network flows based on the hosts and services involved in each communication. Each network flow represents a 15-second segment of client/service and server/service data, which is subsequently summarized into an associated feature set. For the purpose of this work, we extract 58 features from Moore [17] work. To assign labels, our study employs the MAWILab [5] unsupervised machine learning algorithms, which identify network anomalies subsequently labeled as attacks in our dataset.

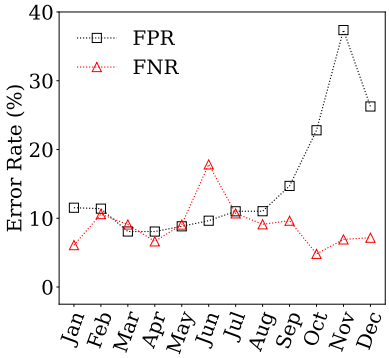


Figure 3: Accuracy trend over a year with AlexNet without periodic model updates. The classifier is trained in January and evaluated in subsequent months without updates.

5.2 Chasing a moving target

To evaluate the performance of our proposed method, we selected the widely used AlexNet DNN architecture. This architecture was originally designed for image classification tasks, requiring input data in a multi-channel image format. However, the *MAWIFlow* dataset is tabular. To address this, we preprocessed the data by transforming each 58-feature sample into an 8x8 single-channel image, padding with zeros as needed. Figure 2 demonstrates this transformation. We then employed average pooling to reshape the input into a 48x48x1 format suitable for the subsequent convolutional layers, which remained unmodified.

The DNN was trained using *adam* optimizer, running for $1,000$ training epochs. We utilized *categorical cross-entropy* as the loss function. The learning rate was 0.001 with a learning rate scheduler that stops training if there is no improvement in the validation accuracy over 50 epochs. These models were implemented using PyTorch API version 2.1.0. The classifier was evaluated in terms of:

- *False Negative (FN)*: number of samples that represent an attack, incorrectly classified as normal traffic.
- *False Positive (FP)*: number of samples with normal traffic which are incorrectly classified as an attack.

The initial experiment is designed to address *RQ1* and assess the classification performance of the chosen intrusion detection techniques on the *MAWIFlow* dataset when it encounters changes in network traffic behavior over time. To achieve this goal, we train the selected DNN architecture using the *MAWIFlow* data from January. We then evaluate the model's performance over the remaining year without periodic model updates. Figure 3 displays the monthly accuracies of the chosen DNN architecture. There is a significant decline in classification accuracies over time. As an example, it experiences an increase in its FP rate of up to 25% when compared to the training period (Jan. vs. Nov.). Evaluated intrusion detection techniques struggle to address the evolving network traffic patterns over time.

The second experiment examines the computational costs associated with the selected techniques. We assess the average inference computational costs on a Raspberry. It is a Raspberry Pi 3 Model B, with a 4-core Broadcom CPU and 1 GB of memory running on top of Raspberry Pi OS with kernel version 6.1. It is possible to note

Table 1: Average event inference time of each introduced AlexNet exit.

Model	1st Exit	2nd Exit
AlexNet	2.67ms	135.46ms

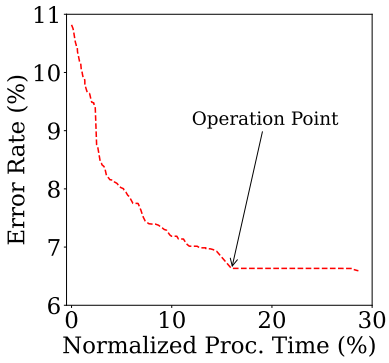


Figure 4: Error *vs* Proc Time in the multi-objective optimization for evaluated architecture. Processing time was measured on a Raspberry PI Model B. Error rate was measured on *MAWIFlow* Feb. and March.

a significant decrease in the detection throughput on a resource-constrained device. In this scenario, an average throughput of ≈ 7.3 events per second proves insufficient for handling the thousands of network events that a device may encounter when deployed on a real network. Hence, in addition to addressing changes in network traffic behavior over time, proposed schemes must also be capable of accomplishing this task while placing minimal demands on processing resources.

5.3 Early Exits for Intrusion Detection

We implement our proposed model using the same DNN architecture evaluated previously (see Section 5.2). Given that we make use of early exits, we added an intermediate branch, introduced between the 1st and 2nd convolutional layers. The early exit flattens the preceding layer output and applies a fully connected layer with 1600 input neurons with a 2 neurons output. Hence, the evaluated DNN architecture consists of two branches encompassing the added layers from the first branch, while the last branch comprises the traditional DNN output. The modified model is trained through the joint loss function (see Eq. 1) with *categorical* loss for each branch, with a 1.0 branch layer weight w. The learning rate was set at 0.001. This model was implemented using PyTorch API version 2.1.0.

With the selected placement for the early exit, we measured the inference time of each exit individually. The early exit has a shorter path, and, whatsoever, should have a faster response time. That can be verified by the data provided in table 1.

To answer *RQ2* we evaluate how our proposed multi-objective optimization can decrease the inference computational costs while maintaining the systems' accuracy. We implement our scheme through the *Non-dominated Sorting Genetic Algorithm* (NSGA-II) [4] on top of *pymoo* API. The NSGA-II uses a 100 population size, 1000 generations, a crossover of 0.9, and a mutation probability of 1.0.

Table 2: Threshold levels at the selected operation points for AlexNet (see Fig. 4).

Model	Normal (1st Exit)	Attack (1st Exit)	Normal (2nd Exit)	Attack (2nd Exit)
AlexNet	90.28%	83.74%	92.14%	87.76%

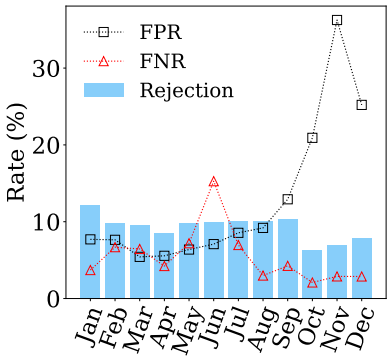
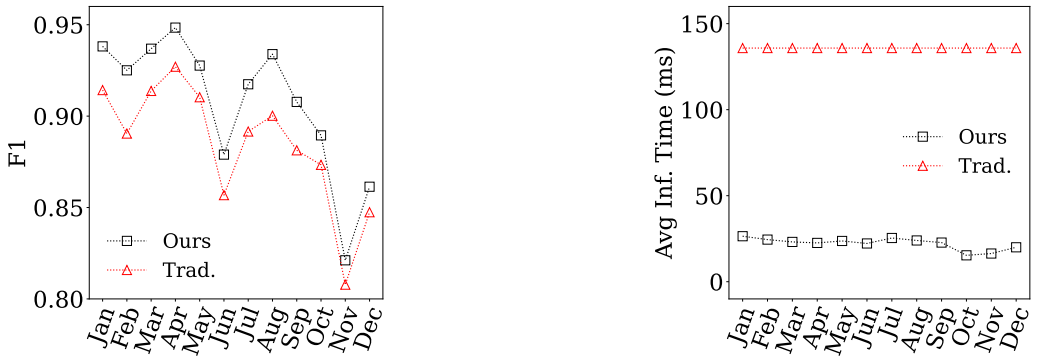


Figure 5: Accuracy trend over a year of our proposed scheme. The classifier is trained in January and evaluated in subsequent months without updates.

The multi-objective feature selection aims at solving equation 4 while optimizing the used first branch threshold (t_b^1) and the *Rejector* threshold (t_{rej}), by considering two objectives, namely *time* and *error*. We measure *time* by normalizing the processing time measured on a test dataset between the minimum (first branch) and the maximum (last branch) processing time, as determined by the DNN architecture. Similarly, we measure the *error* as the average between the measured FP and FN rates. We also consider an optimization constraint of at least 90% acceptance rate at the final branch (Fig. 1, *Rejector*).

Figure 4 shows the Pareto curve for the proposed multi-objective optimization technique. It is possible to note a direct trade-off between model accuracy *vs*. detection throughput. As an example, the multi-objective optimization enables the network operator to reach an error rate of only 6.6% while demanding only ≈ 16% of computational costs (Fig. 4, *Operation Point* for AlexNet). Our multi-objective optimization can reduce the system error rate by up to 3.3 while decreasing the average processing time by up to 82% (AlexNet). As a reference, table 2 shows the confidence thresholds for each exit at the operation points selected.

To answer *RQ3*, we investigate how our scheme with the added *Rejector* at the final branch performs on *MAWIFlow* dataset. In such a case, we set the used first branch threshold (t_b^1) and the *Rejector* threshold (t_{rej}) at the Operation Points highlighted on Fig. 4. The DNNs are then evaluated throughout *MAWIFlow* dataset without model updates. Figure 5 shows the accuracy performance of our scheme without periodic model updates with the *Rejector* module. Our proposed scheme provides better accuracy rates as time passes. Figure 6a further investigates the accuracy benefits of our model with AlexNet compared to the traditional approach. Our scheme can improve the F1 by an average of 0.03, with an improvement of up to 0.06 in August for the AlexNet DNN.



(a) F1 **(b) Processing Time**

Figure 6: Accuracy and processing time comparison of AlexNet with and without our scheme on MAWIFlow dataset.

To answer *RQ4* we investigate the processing benefits of our scheme. Therefore, we measure the average monthly event processing cost, recalling that it varies based on the ratio of events that are accepted at the early branch. Figure 6b shows the processing costs of our scheme *vs*. the traditional approaches on AlexNet. On average, our proposed model requires only $\approx$ 25% of processing compared to the traditional approach. In practice, our model can decrease the error rate while also significantly improving the average processing costs, paving the way for DNN-based network-based intrusion detection application on resource-constrained devices.

6 Conclusion

This work addressed the limitations of DNN-based intrusion detection techniques on resource-constrained devices. Traditional approaches struggle to maintain high accuracy and reliability when confronted with the evolving nature of network traffic, all while minimizing processing demands. To tackle this challenge, we introduced a novel intrusion detection model based on early exits and a classification with a reject option. The former reduces inference time by enabling premature termination of processing for confident classifications. The latter enhances reliability by rejecting potentially inaccurate classifications caused by evolving network traffic patterns.

Our evaluation on the *MAWIFlow* dataset demonstrated the effectiveness of our approach. Through multi-objective optimization, we achieved a significant reduction in processing costs, in some cases by up to 82%, while maintaining or even improving classification accuracy. This highlights the potential of our scheme to enhance the security of resource-constrained IoT devices without compromising performance.

Future work will investigate the incorporation of active learning strategies that leverage rejected events to continuously update the model and further adapt to evolving threats.

Acknowledgments

This work was partially sponsored by the Brazilian National Council for Scientific and Technological Development (CNPq), grants nº 304990/2021-3, 407879/2023-4, 302937/2023-4, and 442262/2024-8.

References

[1] August, 2023 (accessed October 5, 2023). *Mid-Year Update: 2023 SonicWall Cyber Threat Report.* https://www.sonicwall.com/medialibrary/en/white-paper/mid-year-2023-cyber-threat-report.pdf

[2] D. W. Coit A. Konak and A. E. Smith. 2006. Multi-Objective Optimization Using Genetic Algorithms: A Tutorial. *Reliability Engineering System Safety 91* (2006).

[3] Mohammed Ali Al-Garadi, Amr Mohamed, Abdulla Khalid Al-Ali, Xiaojiang Du, Ihsan Ali, and Mohsen Guizani. 2020. A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security. *IEEE Communications Surveys &amp Tutorials* 22, 3 (2020), 1646–1685.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (April 2002), 182–197.

[5] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. 2010. MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *Proc. of the 6th Int. Conf. on emerging Networking EXperiments and Technologies (CoNEXT).*

[6] Mengmeng Ge, Naeem Firdous Syed, Xiping Fu, Zubair Baig, and Antonio Robles-Kelly. 2021. Towards a deep learning-driven intrusion detection approach for Internet of Things. *Computer Networks* 186 (Feb. 2021), 107784. https://doi.org/10.1016/j.comnet.2020.107784

[7] Jhonatan Geremias, Eduardo K. Viegas, Altair O. Santin, Alceu Britto, and Pedro Horchulhack. 2023. Towards a Reliable Hierarchical Android Malware Detection Through Image-based CNN. In *2023 IEEE 20th Consumer Communications amp; Networking Conference (CCNC).* IEEE, 242–247. https://doi.org/10.1109/ccnc51644.2023.10060381

[8] Pedro Horchulhack, Eduardo K. Viegas, Altair O. Santin, Felipe V. Ramos, and Pietro Tedeschi. 2024. Detection of quality of service degradation on multi-tenant containerized services. *Journal of Network and Computer Applications* 224 (April 2024), 103839. https://doi.org/10.1016/j.jnca.2024.103839

[9] Pedro Horchulhack, Eduardo K. Viegas, Altair O. Santin, and João A. Simioni. 2024. Network-based Intrusion Detection Through Image-based CNN and Transfer Learning. In *2024 International Wireless Communications and Mobile Computing (IWCMC).* IEEE, 386–391. https://doi.org/10.1109/iwcmc61514.2024.10592364

[10] Sai Srinadhu Katta and Eduardo Kugler Viegas. 2023. Towards a Reliable and Lightweight Onboard Fault Detection in Autonomous Unmanned Aerial Vehicles. In *2023 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 1284–1290. https://doi.org/10.1109/icra48891.2023.10161183

[11] Stefanos Laskaridis, Alexandros Kouris, and Nicholas D. Lane. 2021. Adaptive Inference through Early-Exit Networks. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning.* ACM. https://doi.org/10.1145/3469116.3470012

[12] En Li, Liekang Zeng, Zhi Zhou, and Xu Chen. 2020. Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing. *IEEE Transactions on Wireless Communications* 19, 1 (Jan. 2020), 447–457.

[13] Li Ma, Ying Chai, Lei Cui, Dongchao Ma, Yingxun Fu, and Ailing Xiao. 2020. A Deep Learning-Based DDoS Detection Framework for Internet of Things. In *IEEE International Conference on Communications (ICC).* IEEE.

[14] MAWI. 2023. MAWI Working Group Traffic Archive - Samplepoint F. https://mawi.wide.ad.jp/mawi/

[15] Je-Won Kang Min-Joo Kang. 2016. Intrusion detection system using deep neural network for in-vehicle network security. *PLoS ONE 11* (2016). https://doi.org/10.1371/journal.pone.0155781

[16] Borja Molina-Coronado, Usue Mori, Alexander Mendiburu, and Jose Miguel-Alonso. 2020. Survey of Network Intrusion Detection Methods From the Perspective of the Knowledge Discovery in Databases Process. *IEEE Transactions on Network and Service Management* 17 (Dec. 2020), 2451–2479.

[17] A. Moore. 2005. Discriminators for use in flow-based classification. In *Dept. Comput. Sci., Univ. London, London, U.K., Rep. RR-05-13*.

[18] Wassim Seifeddine, Cedric Adjih, and Nadjib Achir. 2021. Dynamic Hierarchical Neural Network Offloading in IoT Edge Networks. In *2021 10th IFIP International Conference on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN)*. IEEE.

[19] Samridha Shrestha, Saurabh Pathak, and Eduardo K. Viegas. 2023. Towards a Robust Adversarial Patch Attack Against Unmanned Aerial Vehicles Object Detection. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3256–3263. https://doi.org/10.1109/iros55552.2023.10342460

[20] Surat Teerapittayanon, Bradley McDanel, and H.T. Kung. 2016. BranchyNet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE. https://doi.org/10.1109/icpr.2016.7900006

[21] Eduardo Viegas, Altair Santin, Nuno Neves, Alysson Bessani, and Vilmar Abreu. 2017. A Resilient Stream Learning Intrusion Detection Mechanism for Real-Time Analysis of Network Traffic. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. IEEE, 1–6. https://doi.org/10.1109/glocom.2017.8254495

[22] Yingqing Wang, Guihe Qin, Mi Zou, Yanhua Liang, Guofeng Wang, Kunpeng Wang, Yao Feng, and Zizhan Zhang. 2023. A lightweight intrusion detection system for internet of vehicles based on transfer learning and MobileNetV2 with hyper-parameter optimization. *Multimedia Tools and Applications* (June 2023).

[23] Jielun Zhang, Fuhao Li, and Feng Ye. 2020. An Ensemble-based Network Intrusion Detection Scheme with Bayesian Deep Learning. In *IEEE International Conference on Communications (ICC)*. IEEE.