An Energy-Efficient Intrusion Detection Offloading Based on DNN for Edge Computing

João A. Simioni, Eduardo K. Viegas, Altair O. Santin

[1]Graduate Program in Computer Science (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
Curitiba – PR – Brazil

{joao.asimioni, eduardo.viegas, santin}@ppgia.pucpr.br

Abstract. *To address the computational limitations associated with implementing Deep Neural Network (DNN)–based intrusion detection on resource-constrained devices, this work proposes an energy-efficient edge architecture that integrates distributed early-exit DNN models to minimize processing overhead while preserving detection performance. Our approach employs multi-objective optimization to dynamically offload complex tasks to the cloud, thereby balancing the trade-off between accuracy and energy consumption under operator constraints. Furthermore, it incorporates a rejection mechanism and confidence calibration via temperature scaling to ensure reliability as network traffic evolves. Experiments on a 7TB year-long dataset demonstrate that the system reduces edge energy consumption to only* 1% *while offloading only* 10% *of events, all without compromising detection accuracy and even improving the F1-Score by* 0.02 *compared to traditional approaches.*

1. Introduction and motivation

To address ever-increasing network threats, operators often resort to Network-based Intrusion Detection Systems (NIDSs) [Espindola et al. 2026]. While rule-based NIDSs fall short in detecting new attack variants, behavior-based schemes identify anomalies based on deviations from a normal baseline. In recent years, Deep Neural Network (DNN)-based approaches have emerged as the most effective for high accuracy [Ahmad et al. 2020]. However, to adequately capture network traffic behavior, researchers often increase DNN complexity by adding parameters and layers [Hu et al. 2021]. This complexity incurs higher memory and processing requirements, hindering their application on resource-constrained devices [Tekin et al. 2024].

To overcome the high computational cost of deep models, Early Exits have been proposed as an efficient solution [Teerapittayanon et al. 2016]. These mechanisms introduce side branches at intermediate layers of the DNN, allowing the inference process to terminate prematurely if a sample can be classified with high confidence, thus avoiding the execution of the entire network depth. Edge Computing (EC) complements this approach by enabling the edge or an intermediate infrastructure to cooperate with the cloud [Simioni et al. 2025a]. While offloading all raw network traffic for classification is impractical due to bandwidth saturation and latency requirements [Wang et al. 2023], early exits enable a more granular strategy [Tekin et al. 2024]. By deploying the initial DNN layers at the edge, only the intermediate feature maps of complex samples (those that fail

the early exit) are offloaded to the cloud [Colocrese et al. 2024]. This strategy ensures that the computational work already performed by the edge device is not wasted and reduces network bandwidth usage. However, since network traffic behavior is dynamic and continually evolving [Angelucci et al. 2024], this distributed architecture must also support adequate generalization of DNN models.

Contribution. This paper proposes a new DNN-based NIDS employing early exits within an energy-efficient EC architecture, implemented threefold. First, we couple early exits with multi-objective optimization to achieve adaptive classification offloading. The model classifies traffic at the edge and offloads events to the cloud when additional processing is required, balancing accuracy and energy efficiency. Second, to ensure reliability under dynamic traffic, we implement a reject option at the final DNN branch alongside confidence calibration. This ensures only highly confident classifications are accepted, improving generalization. Third, we deploy this approach within a distributed edge-cloud infrastructure. Consequently, our scheme enables reliable NIDS offloading from edge devices while optimizing energy efficiency and latency. In summary, the main contributions of this article are as follows.

- A new DNN-based NIDS with early exits that operates within an energy-efficient architecture. Our proposed scheme can autonomously offload computation to the cloud when required, while also handling changes in network traffic behavior.
- A prototype that demonstrates our scheme's feasibility under a variety of energy-efficient EC deployment settings. Our scheme can reduce the energy consumption and processing costs of edge devices by up to 1%, while maintaining or improving the F1-Score by 0.02. This is achieved by offloading only 10% of network events to the cloud, thereby optimizing resources on both the edge and cloud.

The contributions of this work were published in the IEEE Internet of Things Journal [Simioni et al. 2025b], ACM/SIGAPP Symposium on Applied Computing (SAC) [Simioni et al. 2025a], Brazilian Symposium on Information Security and Computer Systems (SBSeg) [Simioni et al. 2024], and also resulted in a software registration at INPI (BR1020250142120), and a patent filling at INPI (BR512024004346-2). The collaboration of this work was also published in Brazilian Symposium on Computer Networks and Distributed Systems (SBRC) [Horchulhack et al. 2024a], and International Wireless Communications and Mobile Computing (IWCMC) [Horchulhack et al. 2024b].

2. Related Work

Deploying DNN-based NIDS on resource-constrained devices is challenging due to severe computational bottlenecks. Rather than relying on a single solution, recent research investigates various strategies to enable deep learning at the edge. To better position our work compared to the existing literature, Table 1 presents a summary of the related works based on six main characteristics.

First, we verify if the solution targets the *NIDS Domain* rather than general computer vision tasks. Next, *Resource Optimization* highlights works making models lighter for constrained hardware using lightweight architectures or dimensionality reduction. Since higher accuracy usually increases energy and latency, we evaluate *Multi-Objective Strategies* that balance these conflicting goals. Structurally, we note the use of *Early Exits Architectures* to let simple events finish processing early, alongside *Edge Offloading*

Table 1: A summary of related work and the characteristics of their implementations.

Work	NIDS Domain	Resource Optimization	Multi-Objective Strategy	Early Exits Architecture	Edge Offloading	Traffic Reliability
[Tekin et al. 2024]	✓	✓	×	×	×	×
[Hoang et al. 2022]	✓	✓	×	×	×	×
[Wang et al. 2023]	✓	✓	×	×	×	×
[Roopak et al. 2020]	✓	✓	✓	×	×	×
[Asgharzadeh et al. 2023]	✓	✓	✓	×	×	×
[Teerapittayanon et al. 2016]	×	✓	×	✓	×	×
[Li et al. 2020]	×	✓	✓	✓	✓	×
[Seifeddine et al. 2021]	×	✓	✓	✓	✓	×
[Colocrese et al. 2024]	×	✓	✓	✓	✓	×
[Angelucci et al. 2024]	×	✓	✓	✓	✓	×
[Bajpai et al. 2024]	×	✓	✓	✓	✓	×
Ours	✓	✓	✓	✓	✓	✓

Table 2: Average event detection throughput (events/sec).

DNN	Raspberry Pi	Desktop CPU	Desktop GPU
AlexNet	7.36	247.34	17,609
MobileNetV2	7.93	509.58	3,419

mechanisms that distribute processing across infrastructure levels. Finally, *Traffic Reliability* indicates whether the system handles real-world changes like concept drift using confidence calibration or rejection options to maintain robustness over time.

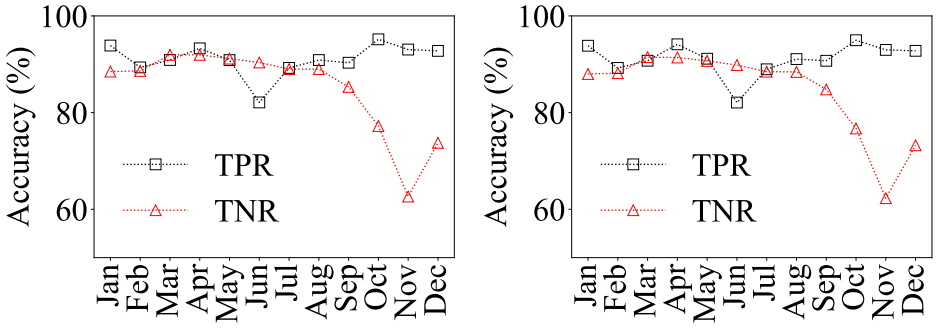
As shown in the summary table, despite recent progress, a clear research gap remains. Specifically, no existing work unifies multi-objective early exits with edge offloading to ensure reliable, continuous NIDS operation in dynamic network environments. Applying early exits in an EC setting is still an open challenge, as deciding when to terminate inference locally or offload to the cloud requires careful calibration under non-stationary traffic. Our work addresses this by optimizing the trade-off between energy, accuracy, and reliability in a distributed environment.

3. Problem Statement and Baseline Evaluation

This section analyzes the performance of current DNN-based NIDS techniques in the literature in terms of accuracy and processing requirements when deployed on edge devices. Specifically, the evaluation aims to answer two Research Questions (RQs):

- **RQ1**. *What is the intrusion detection performance of widely used DNN techniques?*
- **RQ2**. *What are the computational costs of evaluated techniques?*

To address these questions, we demonstrate the inadequacy of traditional DNN deployments using a realistic, large-scale dataset.



(a) AlexNet (b) MobileNetV2

Figure 1: Accuracy trends of commonly used classifiers over a one-year period, where models are trained in January and evaluated in subsequent months without updates.

3.1. MAWIFlow Dataset

Current datasets in the literature often assume static network traffic behavior, leading to schemes that perform poorly in production. To address this challenge, we utilize the *MAWIFlow* dataset, built upon the MAWI working group traffic archive (Samplepoint-F), a transit link between Japan and the USA. The dataset contains real, labeled 2016 network traffic collected at 15-second intervals, totaling over 70 billion packets (7 TB). We extracted 58 features based on Moore's work [Moore 2005] and assigned labels via MAWILab [Fontugne et al. 2010], averaging 105 million packets, 9 million flows, and 1.8 million anomalous flows daily.

3.2. Performance of Traditional DNN-based NIDS

To address *RQ1* and *RQ2*, we evaluated two widely used architectures: **AlexNet** (resource-demanding) and **MobileNetV2** (lightweight). To adapt the one-dimensional dataset to these DNNs, the 58-feature vector was reshaped into a two-dimensional 8×8 matrix (with zero padding) and configured to accept single-channel input. The models were trained for $1,000$ epochs using the Adam optimizer with a learning rate of 0.001.

RQ1 - The accuracy degradation (Fig. 1). The models were trained using data from January and evaluated over the remaining months without updates. The experiments revealed a significant decline in classification accuracy over time, driven by evolving network traffic behavior. Specifically, AlexNet experienced a decrease in its true-positive rate of up to 25% in November compared to the training period. This confirms that static models cannot reliably handle non-stationary traffic behavior.

RQ2 - The computational costs (Table 2). We assessed the computational costs of these models in two environments: a high-end Desktop (Intel Xeon E5-2640, Nvidia Tesla T4) and a resource-constrained Raspberry Pi 3 Model B (4-core Broadcom CPU, 1GB RAM). While a Desktop GPU can process over $17,000$ events/second, the Raspberry Pi achieved a throughput of only ≈ 7 events/second for both architectures. This throughput is insufficient to handle real-time network traffic in production environments, underscoring the need for optimization.

Baseline results demonstrate that directly deploying traditional DNNs on edge devices is impractical for real-world NIDSs, as computational bottlenecks severely limit in-

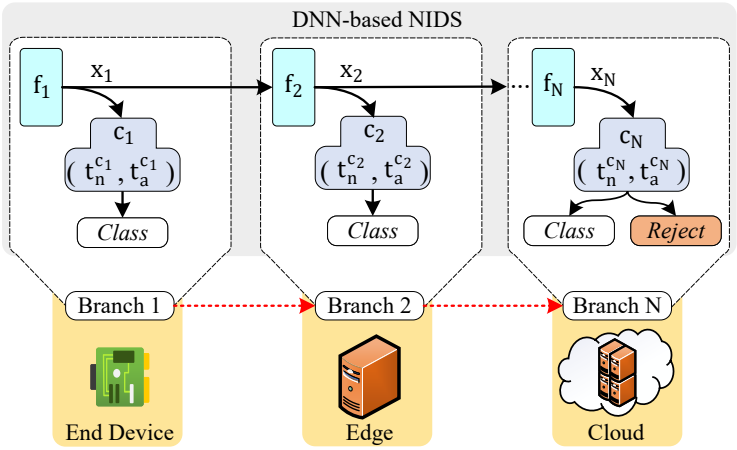


Figure 2: Workflow of the proposed DNN-based NIDS, illustrating early-exit–enabled intrusion detection offloading in EC environments, with DNN branches distributed across the end device, edge, and cloud.

ference throughput on constrained hardware (Table 2), and static models suffer significant degradation in true-positive detection accuracy over time due to concept drift (Fig. 1).

4. A DNN-based NIDS with Early Exits

To address these challenges of computational inefficiency and model deterioration, we propose a distributed NIDS architecture that intelligently splits the inference task across the End Device, Edge, and Cloud infrastructures, as shown in Figure 2.

4.1. Distributed Workflow

The workflow comprises three integrated stages to ensure efficiency and reliability. First, the architecture implements an **Early Exit Inference** mechanism, in which the DNN is equipped with side branches distributed throughout the infrastructure. The first branch (f_1) is executed locally at the End Device. If the classification confidence is sufficiently high, the process terminates immediately to save resources. However, if the confidence is low, the intermediate feature vector x_1 is offloaded to the Cloud for processing by the subsequent branch (f_2), ensuring that only complex events consume cloud resources. Second, the system employs **Multi-Objective Optimization** to determine the ideal operation points. This stage optimizes thresholds for event acceptance and offloading, aiming to achieve the best balance between energy consumption and error rates. Finally, the architecture incorporates **Calibration and Rejection** strategies to address non-stationary network traffic. This involves a rejection option at the final branch to discard unreliable predictions and model calibration to adjust confidence scores, thereby improving generalization against concept drift.

4.2. Early Exit Inference

The mathematical formulation considers a feature vector $x \in \mathbb{R}^D$ and a label $y \in \{n, a\}$, where n denotes normal traffic and a denotes attack traffic. We introduce N classifiers $c_{\{1,\ldots,N\}}$ positioned at intermediate layers of the network. The inference process is designed to terminate at a specific branch i if the classification confidence surpasses a defined threshold t^{c_i}. To enable granular control over the system's sensitivity, this threshold

is defined as a tuple $(t_n^{c_i}, t_a^{c_i})$, thereby establishing distinct acceptance criteria for normal and attack events. During the training phase, the model utilizes a joint loss function to optimize all branches simultaneously:

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

where w_i represents the weight assigned to branch i. Hence, inference proceeds in multiple stages, with training optimized for each branch (Fig. 2, *Branch 1* to *N*).

4.3. Multi-Objective Optimization

Determining the optimal set of thresholds t^{c_i} is formulated as a multi-objective optimization problem, since minimizing resource usage and maximizing accuracy are often conflicting objectives. We aim to minimize the Energy Consumption (σ) and the Error Rate (ϵ) simultaneously by solving:

$$\arg \min_{\{t^{c_1},\ldots,t^{c_N}\}} \sigma(h(\mathcal{D}, \{t\})) \quad \text{and} \quad \arg \min_{\{t^{c_1},\ldots,t^{c_N}\}} \epsilon(h(\mathcal{D}, \{t\})) \tag{2}$$

where h denotes the DNN model with multiple branches, coupled with our rejection approach. Here, σ is a function that measures the model average inference energy consumption on a given dataset $\mathcal{D}$ when using the $\{t^{c_1}, ..., t^{c_N}\}$ thresholds, whereas ϵ is a function that measures the resulting error rate. This optimization is performed using the *NSGA-II* algorithm to generate a Pareto frontier of solutions. This approach allows the network operator to select a specific configuration from the Pareto set that best fits their current resource constraints and security requirements.

4.4. Addressing Generalization: Calibration and Rejection

To effectively handle the dynamic nature of network traffic, it is crucial that the model's confidence scores accurately reflect its probability of correctness. We implement a *Rejection Option* at the final branch N (located in the Cloud), defined by a rejection function:

$$rej(\hat{p}, t^{C_N}) = \begin{cases} \emptyset & \text{if } \hat{p} \leq t^{C_N} \\ \hat{p} & \text{otherwise} \end{cases} \tag{3}$$

where $\emptyset$ denotes events likely to be incorrect decisions the DNN model final branch performs, and t^{C_N} the acceptance thresholds at the final model branch. Events with confidence scores below the threshold t^{C_N} are rejected rather than being potentially misclassified, and are flagged for further review or manual inspection. Additionally, we employ *Calibration* via Temperature Scaling to minimize the Expected Calibration Error (ECE). The softmax function is adjusted by a learned temperature parameter T:

$$\text{softmax}(z_i) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}} \tag{4}$$

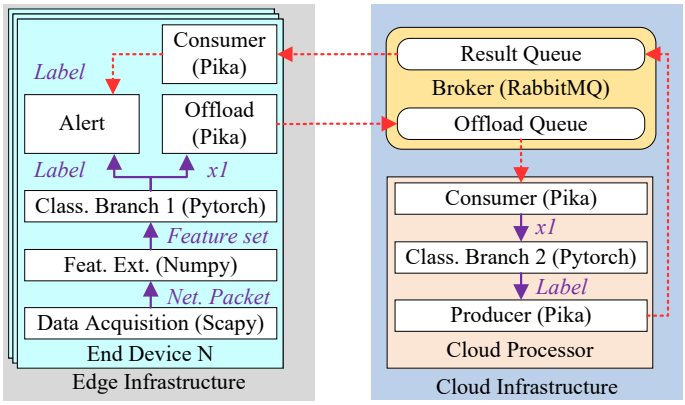


Figure 3: Prototype implementation overview.

where z_i is the sample i logit, and T the temperature vector. This adjustment ensures that the output confidence values more closely reflect the true probability of correct classification, thereby enhancing the reliability of early-exit decisions.

5. Prototype Implementation

We implemented a prototype to validate the architecture in a realistic distributed environment, encompassing data acquisition, processing, and offloading mechanisms, as shown in Figure 3.

5.1. Hardware and Software Stack

The setup uses a Raspberry Pi 3 Model B (Raspberry Pi OS) as the constrained end device and an IBM Cloud VM (Ubuntu) for cloud infrastructure. The software stack leverages PyTorch for DNN training/inference and RabbitMQ via the Pika library for edge-to-cloud communication.

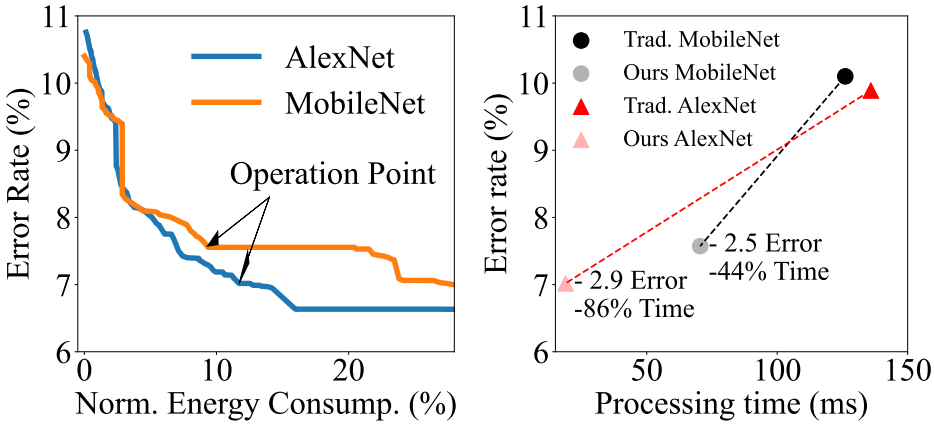
5.2. System Architecture

The system operates sequentially, starting at the End Device, which executes the *Data Acquisition* and *Feature Extraction* modules to process incoming network traffic. The device then runs *Branch 1* of the DNN locally. If the classification confidence does not meet the required threshold, the intermediate tensor is serialized and published to the `Offload Queue` managed by the RabbitMQ broker. On the cloud side, a *Cloud Processor* continuously consumes messages from this queue, executes *Branch 2* of the model, and publishes the final classification label to the `Result Queue`. This distributed loop allows the End Device to receive the final result asynchronously and trigger alerts if necessary.

6. Evaluation

We addressed three RQs regarding optimization, reliability, and edge trade-offs:

- **RQ3**. *How does the proposed multi-objective optimization improve system performance?*
- **RQ4**. *Does the proposed model improve classification reliability?*
- **RQ5**. *What are the system tradeoffs when implemented in an EC architecture?*



(a) Error (ϵ) vs. Energy (σ) (b) Ours vs. Traditional

Figure 4: Pareto curve of Error Rate vs. Normalized Energy Consumption and improvements from the proposed architecture

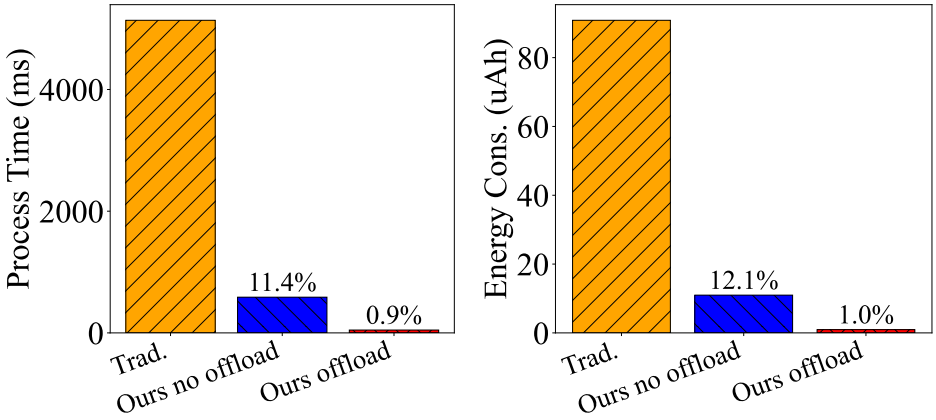
RQ3 - Multi-Objective Optimization Performance (Fig. 4(a)) We trained the models using NSGA-II with a population of 100 and 100 generations. The results reveal a direct trade-off: increasing acceptance on the first branch significantly reduces energy consumption but increases the error rate. Specifically, the system can achieve a ≈**7% error rate** while consuming only ≈**11% of the energy** required by the traditional on-device approach. Adopting a more aggressive strategy reduces energy consumption to **1%** if a 10% error rate is deemed acceptable.

RQ4 - Classification Reliability (Fig. 4(b)) We evaluated the model over the full year of data to assess its robustness against concept drift, focusing on calibration, rejection, and accuracy. Regarding calibration, the application of temperature scaling reduced the ECE on the first MobileNetV2 branch by **43%**, and by **40%** overall. In terms of rejection, the model rejected an average of **8%** of events (AlexNet) and **6%** of events (MobileNetV2) throughout the year, thereby filtering out uncertain predictions. Consequently, the proposed scheme improved the **F1-Score by an average of 0.02** compared to the baseline, confirming that the hybrid approach is more effective at handling non-stationary traffic.

RQ5 - Edge Computing Trade-offs (Fig. 5) We measured processing time, energy consumption, and network overhead across different Cloud zones (Brazil vs. the USA) to assess the feasibility of the architecture. Local processing (Branch 1 only) reduced processing time by **89%** for AlexNet and **63%** for MobileNetV2. Even when offloading is taken into account, the average processing time dropped to **1%** and **32%**, respectively. Regarding latency, deploying the cloud in Brazil (23ms RTT) resulted in an average inference time of **28ms** for AlexNet, compared to **64ms** in the US (157ms RTT); both scenarios are vastly faster than the **1,385ms** required for local-only inference on the Pi. Finally, offloading requires sending ≈7.2 KB (AlexNet) or ≈14.5 KB (MobileNet) per event, but since only ≈10% of events are offloaded, the total bandwidth usage remains minimal.

7. Conclusion

In this paper, we introduce a novel energy-efficient Edge Computing architecture for NIDS, driven by early-exit DNNs. Our framework successfully bridges the gap between the high computational demands of modern deep learning and the resource constraints



(a) Comp. Costs (AlexNet) (b) Energy Cons. (AlexNet)

Figure 5: Average per-event inference time and energy consumption at the End Device comparing full on-device DNN execution (Traditional), proposed early-exit with no offloading, and proposed early-exit with second-branch offloading to the Cloud.

of edge devices. We compared our approach against conventional strategies, demonstrating that it can reduce edge energy consumption by up to 99% while requiring only 10% of events to be offloaded to the cloud. Beyond efficiency, the system exhibits superior robustness, improving the F1-Score by 0.02 and effectively managing non-stationary traffic behavior through calibrated confidence and rejection options. These findings suggest that intelligent, distributed inference is a viable approach to securing the edge ecosystem without compromising device longevity.

Future work will focus on optimizing the physical placement of early exits within the DNN layers to further minimize data transfer and leveraging rejected samples for online incremental learning.

References

Ahmad, Z., Khan, A. S., Shiang, C. W., Abdullah, J., and Ahmad, F. (2020). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1).

Angelucci, S., Valentini, R., Levorato, M., Santucci, F., and Chiasserini, C. F. (2024). Edge computing with early exiting for adaptive inference in mobile autonomous systems. In *ICC 2024 - IEEE International Conference on Communications*, pages 2980–2985.

Asgharzadeh, H., Ghaffari, A., Masdari, M., and Gharehchopogh, F. S. (2023). Anomaly-based intrusion detection system in the internet of things using a convolutional neural network and multi-objective enhanced capuchin search algorithm. *Journal of Parallel and Distributed Computing*, 175:1–21.

Bajpai, D. J., Jaiswal, A., and Hanawal, M. K. (2024). I-splitee: Image classification in split computing dnns with early exits. In *ICC 2024 - IEEE International Conference on Communications*, pages 2658–2663.

Colocrese, M., Koyuncu, E., and Seferoglu, H. (2024). Early-exit meets model-distributed inference at edge networks. In *2024 IEEE 30th International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 39–44.

Espindola, A. d. S., Santin, A. O., Casimiro, A., Ferreira, P. M., and Viegas, E. K. (2026). Understanding the adversary: A survey of adversarial machine learning in network intrusion detection. *Computer Science Review*, 62:100995.

Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. (2010). MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *ACM CoNEXT '10*, Philadelphia, PA.

Hoang, T.-M., Pham, T.-A., Do, V.-V., Nguyen, V.-N., and Nguyen, M.-H. (2022). A lightweight dnn-based ids for detecting iot cyberattacks in edge computing. In *International Conference on Advanced Technologies for Communications (ATC)*, pages 136–140.

Horchulhack, P., Viegas, E. K., Santin, A. O., and Simioni, J. A. (2024a). Fortalecendo a segurança de redes: Um olhar profundo na detecção de intrusões com cnn baseada em imagens e aprendizado por transferência. In *Anais do XLII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, SBRC 2024, page 449–460. Sociedade Brasileira de Computação.

Horchulhack, P., Viegas, E. K., Santin, A. O., and Simioni, J. A. (2024b). Network-based intrusion detection through image-based cnn and transfer learning. In *2024 International Wireless Communications and Mobile Computing (IWCMC)*, page 386–391. IEEE.

Hu, X., Chu, L., Pei, J., Liu, W., and Bian, J. (2021). Model complexity of deep learning: a survey. *Knowledge and Information Systems*, 63(10):2585–2619.

Li, E., Zeng, L., Zhou, Z., and Chen, X. (2020). Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, 19(1):447–457.

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

Roopak, M., Tian, G. Y., and Chambers, J. (2020). An intrusion detection system against ddos attacks in iot networks. In *10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 562–567.

Seifeddine, W., Adjih, C., and Achir, N. (2021). Dynamic hierarchical neural network offloading in iot edge networks. In *10th IFIP International Conference on Performance Evaluation and Modeling in Wireless and Wired Networks (PEMWN)*.

Simioni, J., Viegas, E. K., Santin, A., and Horchulhack, P. (2025a). An early exit deep neural network for fast inference intrusion detection. In *Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing*, SAC '25, page 730–737. ACM.

Simioni, J. A., Viegas, E. K., Santin, A. O., and de Matos, E. (2025b). An energy-efficient intrusion detection offloading based on dnn for edge computing. *IEEE Internet of Things Journal*, 12(12):20326–20342.

Simioni, J. A., Viegas, E. K., Santin, A. O., and Horchulhack, P. (2024). Detecção de intrusão através de redes neurais profundas com saídas antecipadas para inferência rápida e confiável. In *Anais do XXIV Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg 2024)*, SBSeg 2024, page 242–255. Sociedade Brasileira de Computação - SBC.

Teerapittayanon, S., McDanel, B., and Kung, H. (2016). BranchyNet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE.

Tekin, N., Acar, A., Aris, A., Uluagac, A. S., and Gungor, V. C. (2024). Energy consumption of on-device machine learning models for iot intrusion detection. *Internet of Things*, 21.

Wang, Y., Qin, G., Zou, M., Liang, Y., Wang, G., Wang, K., Feng, Y., and Zhang, Z. (2023). A lightweight intrusion detection system for internet of vehicles based on transfer learning and mobilenetv2 with hyper-parameter optimization. *Multimedia Tools and Applications*.