Energy-Efficient CPU+FPGA-based CNN Architecture for Intrusion Detection Systems

Lucas A. Maciel
Pontifical Catholic Univ. of Minas Gerais, Brazil

Matheus A. Souza and Henrique C. Freitas
Pontifical Catholic Univ. of Minas Gerais, Brazil

***Abstract*—Machine Learning applications have their viability strongly linked to the ability of computer architectures to offer high performance and energy efficiency. Although different architectures can offer high computational performance, they may lack energy efficiency, which is crucial for consumer-based applications. For instance, Intrusion Detection Systems can use Machine Learning techniques to monitor network traffic and identify possible malicious activities. These systems are constantly active on devices such as firewalls, reinforcing the need for energy efficiency, e.g., in smart homes and autonomous vehicles. Field-Programmable Gate Array (FPGA) can offer better energy efficiency than other architectures. Considering that, we designed and evaluated a CPU+FPGA-based Convolutional Neural Network for Intrusion Detection Systems. We deployed our strategy in a heterogeneous CPU (Intel Xeon) + FPGA (Arria 10) platform. Then, we compared the proposed architecture with its respective parallel software version for power, energy, and performance evaluation. The NSL-KDD data set was used for intrusion detection benchmarking. The energy efficiency results for CNN showed up to $4.5\times$ more operations per watt than its software version.**

THE FAST ADVANCE of Machine Learning (ML) techniques in several research areas, such as data classification [1] and intrusion detection [2], can produce feasible solutions for complex and heterogeneous consumer-based scenarios such as smart home [3] and autonomous vehicles [4]. Due to the increased traffic generated by traditional devices and Internet-of-Things (IoT) [5], energy-efficient ML architectures are required. Besides, increasing the accuracy of the results and generating them in time-restricted scenarios are two of the leading original goals of ML algorithms. The large volume of data requires attention concerning security, which intensifies investments in monitoring strategies. For instance, Intrusion Detection Systems (IDS) can identify possible attacks, inform their administrators that intrusions may occur, or even prevent the arrival of malicious packages [6]. In this context, IDS can also employ deep learning methods [7] to

Digital Object Identifier 10.1109/MCE.2023.Doi Number
Date of publication 00 xxxx 0000; date of current version 00 xxxx 0000

promote the evaluation of new records according to similarities between the already known data.

One of the most used algorithms for data classification, processing, and image recognition in supervised learning is the Convolutional Neural Network (CNN) [8], [9], a Deep Learning algorithm. It processes data in several layers of connected neurons where the hidden layers can represent convolutions, pooling, or fully-connected processes.

A CNN model generally consists of two stages. The former is *feedforward* for data recognition processes. The latter is *backward* for updating the network weights during training. In addition, it presents components responsible for its operations, such as (i) the feature extractor, which identifies and selects the attributes with the essential information within the *"feature maps"* of the network input data; and (ii) the classifier, which decides the probability of a given entry belonging to the evaluated categories.

CNN applications perform mathematical operations demanding high-performance computing with energy efficiency. Thus, efficient heterogeneous architectures, e.g., Central Processing Unit (CPU) + Field-Programmable Gate Array (FPGA), can achieve lower execution time and energy. In this paper, our goal is to design and evaluate a Convolutional Neural Network (CNN) for a hybrid CPU (Intel Xeon) + FPGA (Intel Arria 10) platform, using Open Computing Language (OpenCL) in comparison to a CPU-based multithreading OpenMP version. This hybrid processing fits well with next-generation firewalls, which employ deep package inspections. Those firewalls include devices such as the FPGA besides the traditional CPU in their boxes, which can quickly receive updates for novel techniques [10], [11]. Also, they can deploy Machine Learning algorithms to detect zero-day attacks, which standard detection systems cannot identify [12]. Furthermore, using OpenCL as a high-level synthesis (HLS) strategy allows for a more efficient and portable software development process for hardware acceleration, providing a standard programming interface for heterogeneous computing systems.

In summary, our contributions are:

- An energy-efficient strategy to deploy a CNN architecture for Network Intrusion Detection designed for a Hybrid Multi-Chip Package system.
- A more general and easy-to-use high-level synthesis code, based on OpenCL, with a CNN architecture that can be reused for FPGA systems[1].

This paper is organized as follows: Section RELATED WORK presents the related work. Next, Section PROPOSED CNN ARCHITECTURE shows our proposed machine learning architecture. Then, we describe the evaluation methodology in Section EVALUATION METHODOLOGY. Finally, we show the results obtained in Section RESULTS and our conclusions and future works in Section CONCLUSION.

RELATED WORK

Artificial Neural Networks can be applied in a broad set of applications [13], [14], [15]. Their use range from security purposes in the Internet of Things (IoT) area [16], [17] to autonomous vehicles [4], [18]. Regardless of the application or scenario, this section focuses on related works that describe CPU or FPGA-based CNN designs for network intrusion detection.

Due to the increase in Internet traffic, some research works focus on FPGA as an accelerator to improve Network Intrusion Detection (NID) performance. Jeune et al. [19] designed a real-time CNN without reducing performance based on using flow buckets to collect traffic features. The authors also used an FPGA-based design to evaluate the performance with high network speed in further work [20].

Despite not using the FPGA-based approach, Ding and Zhai [22] proposed an intrusion detection system with a Convolutional Neural Network using an NSL-KDD data set. They evaluated the proposed CNN in comparison to other machine learning algorithms, such as Random Forest, Support Vector Machine (SVM), Deep Belief Network (DBN) and Long Short Term Memory (LSTM). The results pointed out the improvement in the accuracy of detecting intrusions.

Likewise, two research works [21], [23] proposed and evaluated CNNs for intrusion detection systems with KDD99 and NSL-KDD data sets, respectively. The first one achieved up to 99.23% accuracy compared to other algorithms, such as support vector machine and deep belief network. The second work achieved a better accuracy, e.g., 97%, than other techniques, such as Recurrent Neural Network (RNN), Long short-term memory (LSTM), and Gated Recurrent Units (GRU).

The related work still leaves gaps for research on developing Convolutional Neural Networks for FPGA.

[1]https://github.com/cart-pucminas/machine-learning

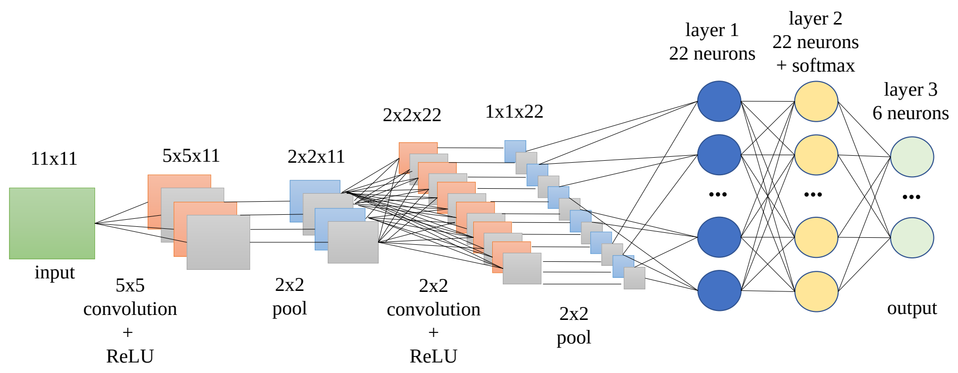

Figure 1. CNN architecture diagram

Some prior work focuses solely on using a CPU or FPGA to implement CNNs. In contrast, others aim to achieve high accuracy by comparing results obtained with different machine learning algorithms. However, these approaches do not prioritize performance and energy efficiency.

Our work differs from others in proposing a CNN architecture on a CPU+FPGA hybrid multi-chip package with high bandwidth to exploit performance with energy efficiency. Following task partitioning concepts, we designed execution flows to harness the power of highly parallel execution units on the FPGA for convolution operations. This hybrid multi-chip package is a standard system in next-generation firewalls, enabling flexibility and the deployment of ahead-of-the-curve techniques [10], [11]. Furthermore, implementations of CNN using OpenCL for FPGA-based multi-chip packages and intrusion detection are still few explored. Thus, our main advancement to the related work is an energy-efficient CPU+FPGA-based CNN architecture for network intrusion detection.

PROPOSED CNN ARCHITECTURE

Our hardware architecture was designed to support floating-point arithmetic to maintain consistency with the software-based CNN implementation, which also used floating-point arithmetic. For this reason, this paper does not exploit power consumption savings from quantization due to the potential loss in accuracy.

Although FPGAs offer high energy efficiency, they are limited by finite memory and logic blocks. Additionally, implementing an entire CNN on an FPGA can be challenging and time-consuming, requiring extensive optimization and tuning to achieve high performance and energy efficiency. Therefore, we decided to implement only the convolution operation on the FPGA. This operation is computationally intensive and benefits the most from FPGA acceleration. By offloading this operation to the FPGA, we achieved significant performance gains and energy efficiency improvements without overloading the FPGA with additional computations. Thus, we achieved an efficient and scalable CNN implementation that leveraged the CPU and FPGA's strengths.

Our CPU+FPGA-based CNN architecture was designed using OpenCL as a high-level synthesis (HLS) strategy, which provides a standard programming interface for heterogeneous computing systems. This approach allows us to develop portable code easily deployed on different hardware platforms, including hybrid CPU+FPGA systems. Additionally, we used the Tiny-DNN deep learning library-DNN [25] to construct our custom neural network. With the C++ and OpenCL approaches, the host (CPU) starts the FPGA and its objects and memory units and synchronizes the operations performed in parallel, using work items at the kernel level. This implementation approach allowed us to adopt a rationale centered around highly parallel execution units for the convolution operations on the FPGA. Using OpenCL, these execution units can be efficiently designed, such as through the use of indexed work items, and organized within an N-Dimensional Range (NDRange). This NDRange enables the execution of kernel instances for each work item, facilitating parallel processing.

The CPU+FPGA architecture (Figure 2) is based on the following interconnected blocks: (i) the Xeon is responsible for containing the connection interfaces with the FPGA, with the DRAM memory and storing the CNN programming logic; (ii) the CNN Code includes the modules responsible for executing the functions of the neural network and controlling the operations carried out in each flow of the algorithm, whether steps in the CPU or activating the device; (iii) the FPGA-FIU module has the bitstream responsible for making the interface between the interconnections and the bitstream provided by the CPU; (iv) the FPGA-CCI-P interface exposes communication channels to CPU; (v) and the FPGA-AFU module represents the convolution step of the CNN algorithm that will be executed on the FPGA.

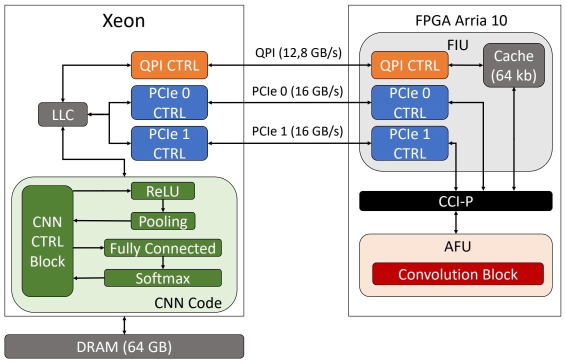

Figure 2. CPU+FPGA CNN architecture

Our design receives a previously trained CNN and a data set of test records as input for evaluation and simulation of the detection of network intrusions. As output, the class identified for each record and its accuracy is available.

Figure 1 shows a CNN architecture, receiving 121-feature input processed by two convolution layers, two pooling layers, and three fully-connected layers. The last layer is responsible for classifying the network intrusion records into six types of classes. The size of convolution layer kernels is [5*5] and [2*2], and the size of pooling layers is [2*2]. In addition, the architecture uses fully connected layers with 22 neurons in the first two layers and six neurons in the last. The CNN uses the Rectified Linear Unit (ReLU) activation function between each layer block, except in the last one, where it applies the Softmax function [26].

The size of the network layers was defined initially considering the number of types of attacks (22) and attributes (144 or 11x11) that would be evaluated. Based on this information, changes were made to the CNN hyperparameters to assess which one would have more acceptable accuracy.

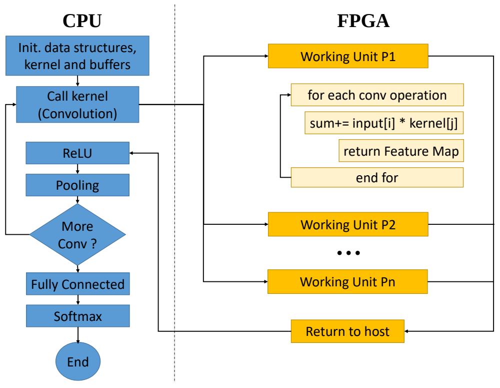

Figure 3. CNN architecture flow

The CNN operation (Figure 3) starts instantiating FPGA memories and loading data structures to store the records it will process. After, the previously trained model is loaded.

The CNN processes items in layers, executing only those related to convolution on the FPGA. The reason is to achieve high performance since the operations related to matrix multiplication are highly parallelizable and demand greater computational power. In addition, we also applied the NDRange strategy, allowing the FPGA to use several computing units with high power of parallelism in each available neuron.

The architecture sends processed records back to the host. Next, the host runs the other layers of *pooling*, *ReLU*, and *fully-connected*, using parallel loop structures, improving processing power. Lastly, the system informs the user of the final classification of the evaluated record and these steps are carried out iteratively for each existing record in the data set. The accuracy of the neural network is verified only at the end of the entire process, comparing the original label of the record with the one classified by the network.

EVALUATION METHODOLOGY

We compared our CPU+FPGA OpenCL-based architecture with CPU-only OpenMP-based software version with 24 threads, using the same input data. We used an Intel Xeon E5-2620 processor with 2.4GHz. The number of threads chosen is related to the best results achieved. Our heterogeneous architecture proposal runs on the Intel hybrid multi-chip package, with a 2.4GHz Intel Xeon CPU + Arria 10 GX1150 FPGA, with a frequency of up to 400MHz and 65.6MB of internal memory, allowing high data transfer.

The Convolutional Neural Network used two samples of NSL-KDD records [24] for training and testing. In addition, we developed a script in C language for reading a CSV file with 41 columns, where 40 are attributes and 1 is a record label. Their values have been normalized between 0-1, using the min-max technique [22]. The strategy converts the columns with categorical values ($protocol\_type$, $service$, $flag$) into new columns with binary values. For instance, $protocol\_type$ has three types of attributes: TCP, UDP, and ICMP, which form three columns [1,0,0], [0,1,0], and [0,0,1]. Thus, at the end of the process, we will have 122 columns, i.e., 121 attributes and one label. Finally, six classes group 22 types of attacks according to their characteristics (Dos, $U2r$, $R2l$, $Probe$, $Normal$, and $Unknown$).

Initially, we trained the CNN using a software version with 125000 input records and 64 iterations/epochs to generate a model with 90.59% accuracy for the CPU-Only and CPU+FPGA platforms.

With the model trained, we tested the CNN, varying the number of records according to the Fibonacci sequence times one thousand (1000, 2000, 3000, 5000, 8000, 13000, and 21000) due to the 22543 records limit in the NSL-KDD data set.

To compare the software and CPU+FPGA approaches, we evaluated the execution time and energy consumption. The Intel PowerPlay EPE returns the FPGA approach's power consumption. Meanwhile, the PowerTOP [27] tool does the same for the Xeon processor. We calculated the energy efficiency metric as the ratio between each workload's total number of operations and the consumed power. In other words, the metric is called MFLOPS (Millions of Floating Point Operations per Second) per watt, i.e., MFLOPS/W.

RESULTS

This section highlights our main results related to FPGA occupation and energy efficiency. The multichip package consists of a device with two units: the FPGA Interface Unit (FIU) and the Accelerated Function Unit (AFU). Table 1 shows the device occupation as the number of logical elements, Adaptive Lookup Tables (ALUTs), registers, memory blocks and Digital Signal Processing (DSP) Blocks. Arria 10 remains a large number of available resources for other potential kernels.

For a comprehensive CNN analysis, Figures 4, 5 and 6 show execution time, energy consumption and energy efficiency results for CPU+FPGA and CPU-only platforms. Thus, we executed 24 threads, with a variation of records (Rec) between 1000 and 21000. The execution time to process several network intrusion records by the CPU+FPGA approach is up to 77% lower than the CPU-Only version. In addition, there is a growing difference in the results as the number of records increases, which shows better scalability of the heterogeneous platform when working with large workloads due to the FPGA as an accelerator. The energy consumption shows that the CPU+FPGA platform consumes up to 78% less than the CPU-Only software version.

Table 1. FPGA occupation

Resources	FIU+CCI	AFU
Logical Elements	563 (49%)	58 (5%)
ALUTs	98256 (23%)	4272 (1%)
Registers	461376 (27%)	68352 (4%)
Memory Blocks	15466 (23%)	4035 (6%)
DSP Blocks	182 (12%)	15 (1%)

In terms of energy efficiency, measured in millions of floating-point operations per second (MFLOPS) per watt, the heterogeneous platform can run up to 4.5× more MFLOPS/watt than the CPU-only software version. The execution time impacts the energy results in both scenarios, even considering the average consumption of Xeon at 85W and CPU+FPGA at 81.45W.

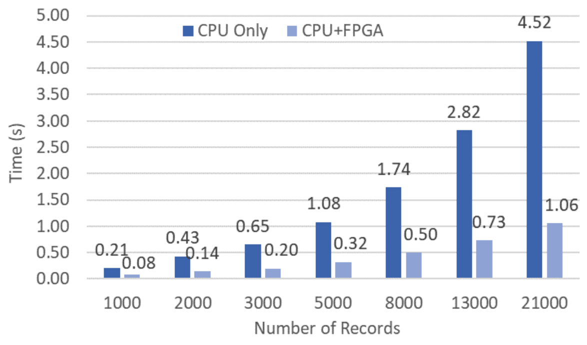

Figure 4. Execution time (seconds)

Our results show that we can implement CNN architectures on CPU+FPGA heterogeneous multichip packages for network intrusion detection systems. They can meet scenarios with varied types and amounts of data to promptly identify anomalies with energy efficiency.

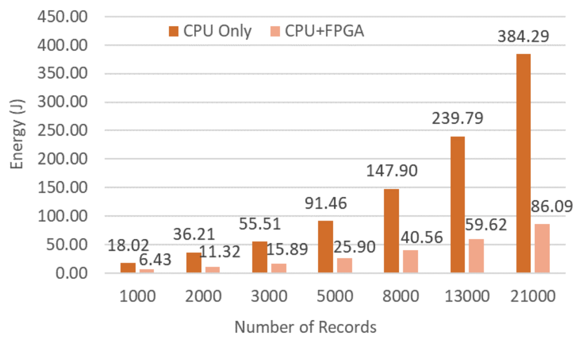

Figure 5. Energy consumption (Joules)

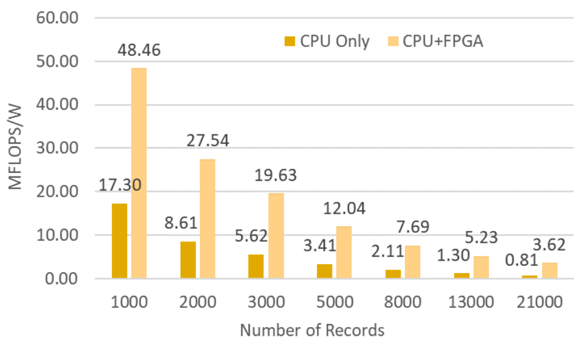

Figure 6. Millions of floating-point operations per second per watt (MFLOPS/W)

CONCLUSION

This work presented a heterogeneous CNN architecture for intrusion detection using OpenCL and C/C++ languages on the Intel HARPv2 platform (Intel Xeon + Arria 10). We implemented a Convolutional Neural Network (CNN) using the C++ OpenSource library called Tiny-DNN. This CNN version using parallel approaches achieved better performance on CPU+FPGA heterogeneous platform than the CPU-only version of up to 77% in execution time, up to 78% in energy consumption and up to $4.5\times$ in MFLOPS/watt. For future work, one can adopt memory sharing and load distribution strategies to explore a higher level of heterogeneity for more efficiency and scalability. Besides, we intend to compare this current high-level synthesis OpenCL-based CPU+FPGA solution with other heterogeneous architectures, e.g., CPU+GPU, focusing on flexibility, portability, performance, and efficiency. Finally, regarding the potential trade-off between processing and accuracy performances, our planning includes an FPGA design space exploration to be efficient in execution time, energy consumption and accuracy for different neural network algorithms.

ACKNOWLEDGEMENT

This study was partly financed by CAPES (Code 001), CNPq, FAPEMIG, and Intel HARP.

REFERENCES

1. M. A. Souza et al., “Energy Efficient Parallel K-Means Clustering for an Intel Hybrid Multi-Chip Package,” in *30th Int. Symposium on Comp. Architecture and High Performance Computing*, pp. 372-379, 2018, doi: 10.1109/CAHPC.2018.8645850.
2. L. A. Maciel, M. A. Souza and H. C. Freitas, “Reconfigurable FPGA-Based K-Means/K-Modes Architecture for Network Intrusion Detection,” in *IEEE Trans. Circuits Syst. II Exp. Briefs*, vol. 67, no. 8, pp. 1459-1463, 2020, doi: 10.1109/TCSII.2019.2939826.
3. M. O. Farooq, I. Wheelock and D. Pesch, “IoT-Connect: An Interoperability Framework for Smart Home Communication Protocols,” in *IEEE Consumer Elect. Magazine*, vol. 9, no. 1, pp. 22-29, 1 Jan. 2020, doi: 10.1109/MCE.2019.2941393.
4. I. Ahmed, G. Jeon and A. Ahmad, “Deep Learning-Based Intrusion Detection System for Internet of Vehicles,” in *IEEE Consumer Elect. Magazine*, vol. 12, no. 1, pp. 117-123, 1 Jan. 2023, doi: 10.1109/MCE.2021.3139170.
5. M. Krishna S and T. Perumal, “Making Buildings Smarter and Energy-Efficient—Using the Internet of Things Platform,” in *IEEE Consumer Elect. Magazine*, vol. 10, no. 3, pp. 34-41, 1 May 2021, doi: 10.1109/MCE.2021.3053182.
6. A. Halimaa A. and K. Sundarakantham, “Machine Learning Based Intrusion Detection System,” in *3rd Int. Conf. on Trends in Elect. and Informatics*, pp. 916-920, 2019, doi: 10.1109/ICOEI.2019.8862784.
7. Y. Li, “Research on Application of Convolutional Neural Network in Intrusion Detection,” in *7th Int. Forum on Elect. Eng. and Automation*, 2020, pp. 720-723, doi: 10.1109/IFEEA51475.2020.00153.
8. E. Chen et al., “Application of Improved Convolutional Neural Network in Image Classification,” *Int. Conf. on Machine Learning, Big Data and Business Intelligence*, pp. 109-113, 2019, doi: 10.1109/MLBDBI48998.2019.00027.
9. J. Li et al., “An FPGA-Based Energy-Efficient Reconfigurable Convolutional Neural Network Accelerator for Object Recognition Applications,” in *IEEE Trans. Circuits Syst. II Exp. Briefs*, vol. 68, no. 9, pp. 3143-3147, 2021, doi: 10.1109/TCSII.2021.3095283.
10. M. S. Brunella et al., “HXDP: Efficient software packet

processing on FPGA NICs," in *Commun. ACM*, Vol. 65, no. 8, pp. 92–100, Aug. 2022, doi: 10.1145/3543668

11. J. Lázaro et al., "Embedded firewall for on-chip bus transactions," in *Comp. & Elect. Eng.*, Vol. 98, 2022, doi: 10.1016/j.compeleceng.2022.107707
12. Y. Guo, "A review of machine learning-based zero-day attack detection: Challenges and future directions," in *Comp. Commun.*, Vol. 198, 2023, doi: 10.1016/j.comcom.2022.11.001
13. D. Wang, K. Xu and D. Jiang, "PipeCNN: An OpenCL-based open-source FPGA accelerator for convolution neural networks," 2017 Int. Conf. on Field Prog. Technology, pp. 279-282, 2017, doi: 10.1109/FPT.2017.8280160.
14. M. R. Vemparala, A. Frickenstein and W. Stechele, "An efficient fpga accelerator design for optimized cnns using opencl," in *Int. Conf. on Architecture of Comp. Systems*, Springer, Cham, 2019, doi: 10.1007/978-3-030-18656-2_18.
15. J. Jiang et al., "A CPU-FPGA Heterogeneous Acceleration System for Scene Text Detection Network," in *IEEE Trans. Circuits Syst. II Exp. Briefs*, vol. 69, no. 6, pp. 2947-2951, 2022, doi:10.1109/TCSII.2022.3167022.
16. L. Ioannou and S. A. Fahmy, "Network Intrusion Detection Using Neural Networks on FPGA SoCs," *2019 29th Int. Conf. on Field Prog. Logic and Applications*, Barcelona, 2019, pp. 232-238, doi: 10.1109/FPL.2019.00043.
17. D. -M. Ngo et al., "FPGA Hardware Acceleration Framework for Anomaly-based Intrusion Detection System in IoT," *2021 31st Int. Conf. on Field Prog. Logic and Applications*, Dresden, 2021, pp. 69-75, doi: 10.1109/FPL53798.2021.00020.
18. L. Zhang, X. Yan and D. Ma, "Accelerating In-Vehicle Network Intrusion Detection System Using Binarized Neural Network," *Int. J. Adv. & Curr. Prac. in Mobility*, 4(6):2037-2050, 2022, doi: 10.4271/2022-01-0156.
19. L. Le Jeune, T. Goedemé, N. Mentens. "Towards Real-Time Deep Learning-Based Network Intrusion Detection on FPGA," in *Applied Cryptography and Network Security Workshops*, vol 12809. Springer, Cham, 2021, doi: 10.1007/978-3-030-81645-2_9.
20. L. Le Jeune et al., "SoK - Network Intrusion Detection on FPGA," in *Security, Privacy, and Applied Cryptography Eng.*, vol 13162. Springer, Cham, 2022, doi: 10.1007/978-3-030-95085-9_13.
21. R. U. Khan et al., "An Improved Convolutional Neural Network Model for Intrusion Detection in Networks," in *Cybersecurity and Cyberforensics Conf.*, pp. 74-77, 2019, doi: 10.1109/CCC.2019.000-6.
22. Y. Ding and Y. Zhai. "Intrusion Detection System for NSL-KDD Dataset Using Convolutional Neural Networks," in *ACM Int. Conf. on Comp. Science and Artificial Intelligence*, New York, pp.81–85, 2018, doi: 10.1145/3297156.3297230.
23. S. Al-Emadi, A. Al-Mohannadi and F. Al-Senaid, "Using Deep Learning Techniques for Network Intrusion Detection," in *IEEE Int. Conf. on Informatics, IoT, and Enabling Tech.*, pp. 171-176, 2020, doi:10.1109/ICIoT48696.2020.9089524.
24. L. C. Hong. (2015) Nsl-kdd dataset. [Online]. Available: https://github.com/defcom17/NSL_KDD.
25. E. Riba. (2016) Tiny-Dnn. [Online]. Available: https://github.com/tiny-dnn/tiny-dnn.
26. M. Abadi et al., "TensorFlow: a system for large-scale machine learning," in *USENIX Conf. on Operating Systems Design and Implementation*, pp. 265–283, 2020, doi: 10.5555/3026877.3026899.
27. Intel. (2007) PowerTOP. [Online]. Available: https://01.org/powertop.

Lucas Andrade Maciel is an IT Engineer Lead at Localiza&Co and an Adjunct Professor at the Pontifical Catholic University of Minas Gerais (PUC Minas), Belo Horizonte, Brazil. He received his B.Eng. in Computer Engineering (2017) and M.Sc. in Informatics (2020) from PUC Minas. His research interests are Computer Architecture, High-Performance Computing and Intrusion Detection Systems. Contact him at lucasmaciel@pucminas.br.

Matheus Alcântara Souza is an Assistant Professor at the Pontifical Catholic University of Minas Gerais (PUC Minas), Belo Horizonte, Brazil. He received his Tech. (2007) in System Development from Fabrai, M.Sc. (2015) and Ph.D. (2021) in Informatics from PUC Minas. His research interests are Computer Architecture and High-Performance Computing. Contact him at matheusalcantara@pucminas.br.

Henrique Cota de Freitas is an Associate Professor at the Pontifical Catholic University of Minas Gerais (PUC Minas), Belo Horizonte, Brazil. He received his B.S. in Computer Science (2000) and M.Sc. in Electrical Engineering (2003) from PUC Minas and Ph.D. in Computer Science (2009) from the Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil. His research interests are Computer Architecture and High-Performance Computing. Contact him at hcfreitas@ieee.org.