FPGA-Enabled Efficient Framework for High-Performance Intrusion Prevention Systems

Cuong Pham-Quoc[1,2(✉)] and Tran Ngoc Thinh[1,2]

[1] Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet Street, District 10, Ho Chi Minh City, Viet Nam
cuongpham@hcmut.edu.vn

[2] Vietnam National University - Ho Chi Minh City (VNU-HCM), Thu Duc, Ho Chi Minh City, Vietnam

Abstract. With the rapid increase of network-based services during and after the COVID-19 pandemic, preventing network attacks is an essential demand. This paper proposes an efficient FPGA-based framework for developing high-performance intrusion prevention systems. The framework allows both signature/pattern-based and anomaly/AI-based prevention techniques to be deployed. We implement the framework with two efficient FPGA-based network processing platforms, NetFPGA-10G and NetFPGA-SUME, for evaluating the proposed framework. Three signature-based approaches, SYN flood defender, hop count filtering, and port ingress/egress filtering, are applied to the framework with the NetFPGA board. Meanwhile, five different neural network models are deployed with both platforms. Three well-known datasets, NSL-KDD, UNSW-NB15, and CIC-IDS2017, are used for training and testing these models. Experimental results show that we achieve a throughput of up to 39.48 Gbps with the signature-based techniques. Our framework can obtain a throughput of up to 34.74 Gbps for the neural network models. Our network intrusion prevention systems can offer up to 99.48% accuracy with only a 0.05% false positive rate and a 0.99% false negative rate in the best case.

Keywords: FPGA · Network intrusion prevention · Anomaly-based detection · signature-based detection

1 Introduction

The development of internet devices has been exponential in recent years, with a range of smart gadgets and wearable technologies flooding the market. According to a report by Statista [30], the number of internet-connected devices is expected to reach 75 billion by 2025, most of which are smartphones, tablets, and other portable devices. The proliferation of such devices has increased demand for high-speed internet connectivity, resulting in the rapid deployment of 5G networks worldwide.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14109, pp. 83–98, 2023.
https://doi.org/10.1007/978-3-031-37120-2_6

As technology and network architectures continue to develop rapidly, cybersecurity has become a crucial concern for organizations such as businesses, banks, and military networks. For example, a report by Microsoft reveals that, on average, 1,435 attacks per day are taken into account at the organization in 2022 [32]. Furthermore, the security today report shows that in 2022, compared to 2021, in total 150% DDoS attacks are increased [28]. These intrusion threats can have severe consequences, including financial losses, reputational damage, and legal liabilities. Therefore, developing an efficient framework for creating high-performance network intrusion prevention systems (NIPS) to countermeasure these attacks is essential.

In modern NIPS, two primary approaches have been mainly utilized: pattern-based (signature-based) [20] and anomaly-based [11]. The former method compares network packets with predefined patterns/signatures to detect harmful packets. The latter technique uses AI models to classify packet behaviors and recognize illegitimate flows. While pattern-based NIPS methods are known for their high accuracy rates and throughput, they suffer from inflexibility, the overhead for processing rules, and the need for data storage to store attacking rules [3]. This issue becomes even more problematic when the number of stored patterns is significant and frequently updated. Meanwhile, deep learning has made impressive strides in recent years, and the anomaly-based approach, which relies on deep neural networks (DNN), offers several advantages over the signature-based method. For example, it can efficiently prevent unknown attack types and does not require extensive storage resources [12].

In this work, by taking advantages of the Field Programmable Gate Array (FPGA) technology [8] such as reconfigurability and high parallelism, by a massive amount of resources into account, we design an FPGA-based efficient framework to deploy different IPS techniques from both the above approaches to build high-performance network intrusion prevention systems. The framework allows network administrators to configure the NIPS with hardware computing cores with signature-based or neural network models. The number of cores and types (signature-based or neural networks) depends on the hardware resources available and the system's sensitivity. We implement our prototype system with a NetFPGA 10G board (equipped with a Virtex 5 FPGA device) and a NetFPGA SUME board (equipped with a Virtex 7 FPGA device). Various techniques, including signature-based hardware cores and AI-based models, have been deployed into the prototype system to evaluate the throughput and accuracy.

The main contributions of our paper can be summarized in three folds.

1. We propose an efficient FPGA-based framework for deploying high performance intrusion prevention systems with signature- and anomaly-based approaches.
2. We present our FPGA-based system implementation for developing different intrusion prevention techniques.
3. We show our experiments with various prevention approaches using the system. Our experimental results can be used for future studies as a reference.

The rest of the paper is organized as follows. Section 2 presents the background of intrusion prevention systems and related work in the literature. We introduce our proposed FPGA-enabled framework in Sect. 3. Section 4 shows the FPGA-based system implementation so that various intrusion prevention techniques can be deployed. Experiments with some approaches, both signature-based and anomaly-based, are discussed in Sect. 5. Finally, Sect. 6 concludes our paper.

2 Background and Related Work

This section presents the background of network intrusion detection/prevention systems (NIDS/NIPS). We also summarize related work of FPGA-based NIDS/NIPS in the literature.

2.1 Background

In this section, we present at first an overview of network intrusion detection/prevention systems, followed by the datasets most used for training and testing these systems.

Network Intrusion Detection/prevention Systems (NIDS/NIPS): are used to detect/react to potential threats on a network by analyzing network traffic and packet behaviors [15]. These systems can respond to threats by either alerting administrators (detection) or blocking network access (prevention). There are two main types of detection/prevention: signature-based and anomaly-based. Signature-based NIDS uses previous knowledge or abnormal behaviors to detect/prevent threats (so-called knowledge-based approaches). In contrast, anomaly-based NIDS uses normal activities as a reference point for identifying/reacting to abnormal behaviors. Recently, many studies have applied machine learning/deep learning for anomaly detection. This can help NIDS adapt quickly to changing environments and identify anomalous intrusions by training machine learning models on normal network activities. However, anomaly-based detection may result in more false positives, generating many alarms in the monitoring system. Nevertheless, this issue is not a concern for protected systems as these alerts do not indicate actual attacks. In contrast, although signature-based techniques suffer from overhead for data storing and low ability to adapt to new attacking types, they offer high throughput and accuracy compared to the anomaly-based approach.

Dataset: Both the approaches for NIDS/NIPS require datasets for building or training and testing to validate the scanning cores. As reported in [2], three most frequently used datasets in the literature are KDD Cup99 & NSL-KDD (60%), UNSW-NB15 (18%), CIC-IDS2017 (12%)in which:

- KDD Cup99 [10] is one of the most well-known and widely used datasets for NIDS/NIPS. The dataset contains about 5M+ and 2M+ records for training and testing purposes, respectively. Each record consists of 41 features labeled as usual or attack. Four attacking types, DoS (Denial of Services), Probe, R2L (Remote to Local), and U2R (User to Root), can be detected with this dataset. Consequently, NSL-KDD [31] is an enhanced version of the KDD Cup99 dataset to overcome the integral issues.
- UNSW-NB15 [22] contains 2M+ records with 49 features per record. The dataset covers nine attacking types: Worms, Shellcode, Reconnaissance, Port Scans, Generic, Backdoor, DoS, Exploits, and Fuzzers.
- CIC-IDS2017 [1] includes normal flows and attacks collected from the real world with common attacking types such as DDoS, Botnet, Brute Force Attack, etc.

In this work, we use three latter datasets for testing our prototype because NSL-KDD already covers KDD Cup99. Along with these two popular datasets, we also use our in-house building dataset collected from our organization network to test the signature-based scanning cores.

2.2 Related Work

In recent years, many studies have focused on building NIDS/NIPS on FPGAs to exploit the parallelism and reconfigurability of this technology. However, they all propose implementing a standalone, signature-based, or anomaly-based approach. Compared to these proposals in the literature, our framework supports both signature- and anomaly-based strategies to be implemented on the FPGA devices. Furthermore, thanks to the reconfigurability, our NIPS system can be switched among different techniques/models based on the two primary approaches. The following summary review FPGA-based NIDS/NIPS published in the literature during the past five years.

FPGA-Based Signature-Based: The work in [35] proposed IDS/IPS Pigasus in which FPGA performs most execution of NIDS to protect a network of 100 Gbps. An FPGA and rule-based NIDS was introduced in [19] with 256 entries. The packet processing latency of this work is only $0.032\,\mu s$. The RegEx matching and NFA (Non-deterministic Finite Automata) were used for building a NIDS [7] and its updated version [6] to offer a Snort rule FPGA-based NIDS with 100 Gbps and 400 Gbps, respectively. Research in [33] used UltraRAMs and rolling hash functions for implementing an FPGA-based intrusion detection with a significant performance improvement. FPGA/Bloom filter-empowered IDS was introduced in [27] with many of Bloom filter's disadvantages removed. The Cuckoo filters and FPGA were used for developing an IDS with improved performance was discussed in [16].

FPGA-Based Anomaly-Based: The k_modes, a modified version of k_mean, was proposed in [21] using the NSL-KDD. Research in [14] used KNN (K-Nearest Neighbours) with a custom dataset to design an FPGA-based NIDS that offers 95% accuracy with 3.913 ms latency. Accepting the low accuracy with BNN (Binary Neural Network), researchers in [23] presented a NIDS system of 90.74% accuracy with only 19.6 ns latency. The same authors continued improving their BNN-based NIDS and reported in [24] with 38.1% higher accuracy than their previous work. Study in [18] implemented an FPGA-based NIDS with runtime updating coefficients. The implementation with Zynq FPGA offered an accuracy of 80.52% with 9.02 ms latency.

3 The FPGA-Enabled Framework

Figure 1 presents our proposed efficient framework for building high-performance network intrusion prevention systems on FPGA. As shown in the figure, the proposed framework consists of three different layers, *Application & Administrator*, *Communication layer*, and *FPGA-based NIPS*, hosted by both software (CPU or GPU) and hardware (FPGA). The software part mainly manages the entire system by generating anomaly-based models or building signature-based hardware cores to configure the FPGA-based NIPS on-the-fly. Meanwhile, the hardware part performs the main functionalities of the NIPS by executing the neuron network models and hardware computing cores for anomaly-based and signature-based prevention, respectively.

3.1 Application and Administrator Layer

As mentioned above, the primary purpose of this layer, deployed at the software level (CPU or GPU), is to manage the entire system. The layer contains two main components, the visualization report and the FPGA-based computing cores for both neuron networks and signature-based cores. Through the visualization reports, system administrators can recognize attack types and sizes that the system has been preventing currently and in the past. Based on statistical data, administrators can determine which types of prevention (for example, DDoS or virus/malware) should be deployed.

Along with these reports, the layer takes certified datasets into account the generate FPGA-based computing cores for prevention attacks. Furthermore, the layer can generate trained AI neuron network models with different toolchains (such as Tensorflow or Kaffe) for anomaly-based prevention. The trained neuron networks will then be manually implemented by hardware description languages (such as Verilog, VHDL, or SystemVerilog) or high-level synthesis tools (such as Vitis from Xilinx or Stratus HLS from Cadence). Finally, the computing cores are synthesized for programming the FPGA board with the support of the design tools provided by target FPGA device vendors. The output of this layer will be used to configure the system through the communication layer.

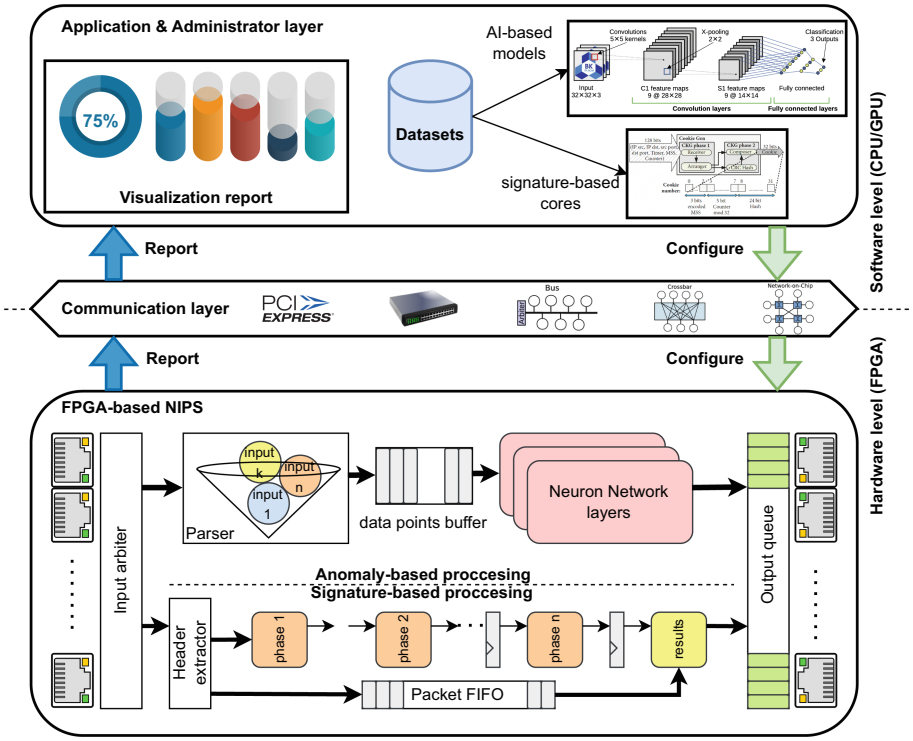


Fig. 1. The FPGA-enabled framework for high-performance NIPS

3.2 Communication Layer

The communication layer is the interface between software and hardware. This layer transfers configuration data (parameters of AI neuron network models and programming files for reconfiguring the FPGA boards). Based on the deployment platforms, the communication layer can be PCI-express, giga-speed switch, high-performance bus, crossbar, or network-on-chip. This layer is mainly responsible for communication overhead that affects the system's downtime (handling for reconfiguration). Therefore, high bandwidth communication infrastructure should be used for the system.

3.3 FPGA-Based NIPS

The FPGA-based NIPS layer is the central part and one of the main contributions of our work. FPGA-enabled NIPS hosts various FPGA-based processing cores to implement the anomaly-based or signature-based prevention approaches[1]. Our NIPS includes high-speed input/output network ports for

[1] We call them scanning cores when we do not distinguish them.

incoming network packets receiving and legitimated packets sending. Input packets from ports are pre-processed by the *Input arbiter*. Depending on predefined behaviors, the Input arbiter selects packets coming from a particular port to send to processing cores (by anomaly/signature-based processing) or forward directly to outputs without scanning. This behavior can be reconfigured by administrators because there may exist some ports connected to truth sources.

In contrast to the input, the output queue stores packets waiting for decisions of the scanning cores. Based on classifications from deployed scanning cores, packets in this queue will be forwarded to a desired output port if they are legitimated or removed from the queue to protect the internal systems if they are classified as attacks. The heart of the system is the scanning cores which can be deployed with the anomaly-based approach or a particular technique using the signature-based method to classify packets.

Anomaly-Based: As introduced above, the anomaly-based approach uses neural network models to classify packets as normal or abnormal. Network packets, at first, are analyzed to extract feature vectors for neural networks by the *Parser* module. Next, these input features for AI models are stored in the *data points buffer* to support high-speed incoming network flows. Finally, layers of neurons collect data from the buffer to classify network packets as normal or abnormal. The datasets and models determine the number of layers and neurons per layer.

Signature-Based: Different from the anomaly-based, the signature-based approach implements particular prevention techniques to investigate network packets, such as Hop-count filtering or SYN Flood Defender. Therefore, the architecture of the signature-based scanning cores relies on the selected methods. In general, we need a *Header extractor* to collect features from the header field of packets and send them to computing blocks of the cores. The blocks are usually designed and implemented in the pipeline to improve performance.

To cope with high-speed network packet arriving, signature-based scanning cores are equipped with *packet FIFOs* waiting for decisions. First, a legitimate packet is transferred to the *Output queue* to forward to target ports. Otherwise, it will be removed from the FIFO to protect the local network resources.

4 The FPGA System Implementation

In this section, we present our prototype implementation with two target platforms, NetFPGA-10G and NetFPGA-SUME.

4.1 Signature-Based Techniques

For our implementation to test the proposed framework, we use three prevention techniques, including SYN defender [13], hop-count filtering [34], and port

ingress/egress filtering [29]. Figure 2 depicts the implementation architectures for the three signature-based prevention techniques.

As shown in the figure, a table of known data is embedded into each architecture. Then, based on analysis information of incoming packets, the *results* block checks the table to determine whether the packet should be dropped due to its harmfulness or forwarded to destinations. We implement these architectures by Verilog-HDL for building on the two above-mentioned FPGA target platforms.

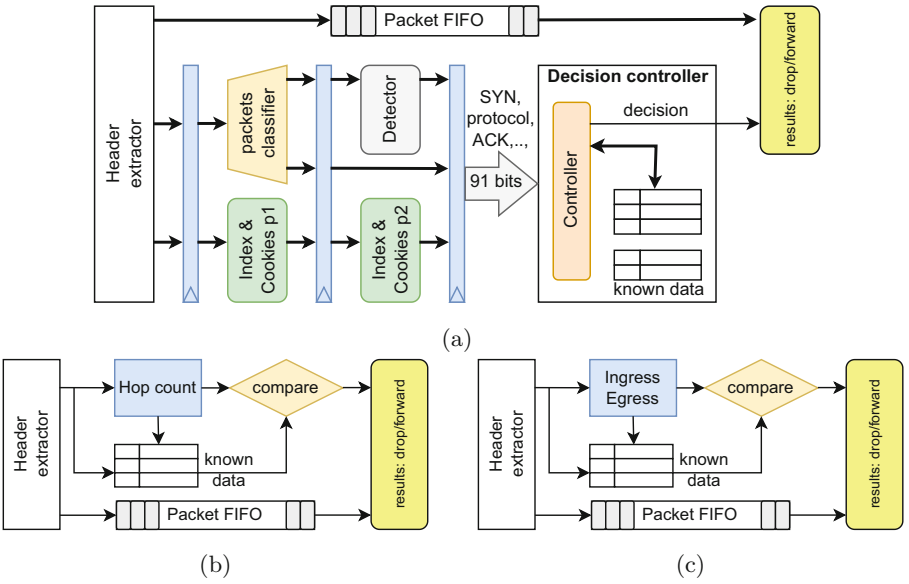


Fig. 2. The implementation architectures for three signature-based prevention techniques: (a) SYN flood, (b) hop-count, (c) port ingress/egress filter.

4.2 Neural Networks

Figure 3 presents our implementation architecture to build an FPGA-based neural network layer. A chain of multiplication and addition blocks are used for calculating a neuron's operations ($y = \sum_{i=1}^{n}(x_i \times w_i) + b$), where x, w, and b are feature vector, weight vector, and bias, respectively. In this work, we apply Nordstroem and Svensson's activation function [26] (depicted in Eq. 1) due to its hardware efficiency.

$$f(x) = \frac{x + |x| + 1}{2(|x| + 1)} \tag{1}$$

Like the signature-based techniques above, we use Verilog-HDL to describe an FPGA-based IP core for neural network layers. Using the *parameter* directive, we can effectively change the number of elements in input vectors and neurons

per layer before synthesizing and programming the platforms. Depending on deployment models, neural network layer cores are connected in cascade form.

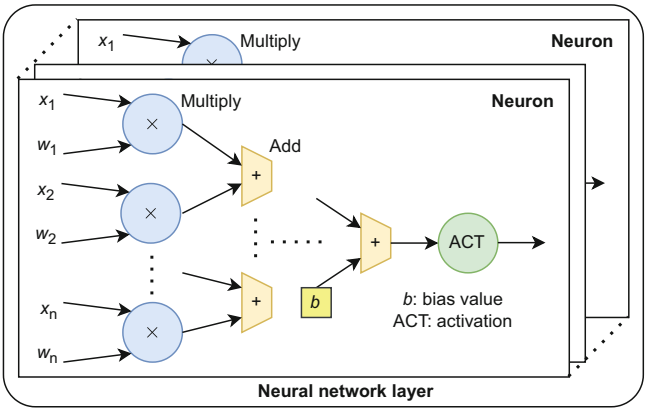


Fig. 3. The implementation architecture a for neural network layer

4.3 Target Platforms

In this work, we use two FPGA platforms dedicated to network processing for building our prototype implementation. The first is the NetFPGA-10G [25] which includes an xc5vtx240t Xilinx FPGA device (149,760 LUTs, 96 DSP48s, 11,664 Kbit block RAM, and 27 MBytes external SDRAM) and 4 SFP+ connections supporting 10 Gbps per port. For our prototype version, we build the three above-mentioned signature-based prevention techniques (SYN flood, hop-count filtering, and port ingress/egress filtering) and three neural network models (10×1, $10 \times 2 \times 1$, and $10 \times 4 \times 1$). The three neural network models are trained and tested with the NSL-KDD dataset.

The second platform for testing the proposed framework is NetFPGA-SUME [9], an updated version of NetFPGA-10G [36]. The platform is equipped with the Xilinx Virtex-7 xc7vx690T-2FFG1761C (433,200 LUTs, 3600 DSPs, 52,920 Kbit block RAM, and three x36 72Mbits QDR II SRAM) and 4 SFP+ interface supporting 10 Gbps per port like the NetFPGA-10G. Due to the massive available resources compared to the NetFPGA-10G platform, we build a neural network model of $40 \times 31 \times 1$ trained and tested by both UNSW-NB15 and CIC-IDS2017 datasets. In addition, we also apply the Autoencoder model with neural networks of $122 \times 64 \times 122$ trained and tested with the NSL-KDD dataset for this platform.

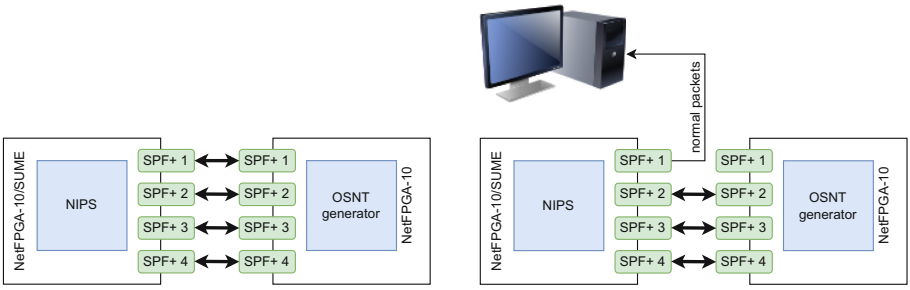
5 Experiments

In this section, we present our experiments to validate and estimate the acceleration ability of the above system. At first, the experimental setup is introduced.

Then, we summarize the results with our experiments to illustrate the goals of our work.

5.1 Experimental Setup

To verify the two platforms implemented with our NIPS, we build the following testing models: (i) we use two NetFPGA boards, for our system and the OSNT (Open Source Network Tester) generator [5] tool as depicted in Fig. 4a to estimate throughput. The board with the OSTN tool generates and sends packets at the maximum bandwidth to our NIPS. (ii) we use two NetFPGA boards and a traditional computer with a gigabit network card to test our neural networks' accuracy and signature-based techniques. Figure 4b shows this testing model.



(a) (b)

Fig. 4. Testbed used for our NIPS: (a) throughput and latency evaluation; (b) accuracy evaluation

5.2 Synthesis Results

Our proposed framework with the aforementioned prevention approaches for the two target FPGA platforms is synthesized with Xilinx Vivado 2022.1 [4]. Table 1 shows the hardware resources usage for our framework with the two target platforms.

The table shows that the system implemented on the NetFPGA-10G platforms with the three signature-based approaches, SYN defender, hop-count, and port ingress/egress filtering, requires almost all on-chip memory (blockRAM) for storing known patterns. Due to the substantial amount of resources used, the working frequency of the system with the three prevention techniques is only 89.254 MHz, the worst compared to other experiments. When neural networks are used on the NetFPGA-board, the system uses less BlockRAM than before. As shown in the table, the number of LUTs, FFs, and DSP48 increases according to the neural network size.

Table 1. Synthesis results for our prototype implementation with both target platforms

Resources	NetFPGA-10G				NetFPGA-SUME	
	Signature	$10 \times 0 \times 1$	$10 \times 2 \times 1$	$10 \times 4 \times 1$	Autoencoder	$40 \times 32 \times 1$
FF	107,275	77,869	95,261	122,085	131,341	131,592
	71.63%	52.00%	63.61%	81.52%	15.61%	15.19%
LUT	96,767	65,621	78,883	99,890	90,869	92,661
	64.61%	43.82%	52.67%	66.70%	20.97%	21.39%
DSP48	0	11	25	49	245	132
	0.00%	11.46%	26.04%	51.04%	6.81%	3.67%
BlockRAM	323	139	139	140	285.5	214.5
	99.69%	42.90%	42.90%	43.21%	19.43%	14.59%
Freq. (MHz)	89.254	102.218	100.644	100.532	206.7	204.2
Power (W)	11.977	10.143	10.364	10.996	11.27	10.32

Regarding the framework implemented on the NetFPGA-SUME, the table shows that our neural networks use a few resources. Due to the many DSP48 embedded, many DSP48 elements are exploited for building neurons. Regarding working frequency, the system can work with more than 200 MHz for both neural network models. The table also shows that the system with neural network models on NetFPGA-SUME requires similar power consumption to the NetFPGA-10G.

5.3 Throughput Results

The signature-based prevention techniques scan both header and payload of packets for detecting attacking packets. Therefore the size of packets will affect the throughput of the system. Hence, for evaluating the throughput of the framework with these approaches, we classify testing packets into six groups based on their sizes, including 64-byte, 128-byte, 256-byte, 512-byte, 1024-byte, and 1500-byte packets. Figure 5 depicts the processing throughput in terms of Gbps of the three signature-based prevention techniques.

Figure 5 shows that with large packet sizes, the system with these prevention techniques offers higher throughput due to the payload. When packets' sizes are 64 B, the throughput is decreased because many packets' headers need to be verified. For all packets' sizes, throughput values with the hop count and port ingress/egress filtering are the same because they share the same structure.

The neural networks, as mentioned above, convert packets to feature vectors to process neurons. Therefore, we do not need to classify packets according to their sizes as the signature-based approaches. We use two platforms, NetFPGA-10G and NetFPGA-SUME, for deploying different neural network models. Due to more resources, the NetFPGA-SUME can host larger models, as mentioned in the previous sections. To estimate the system throughput, we

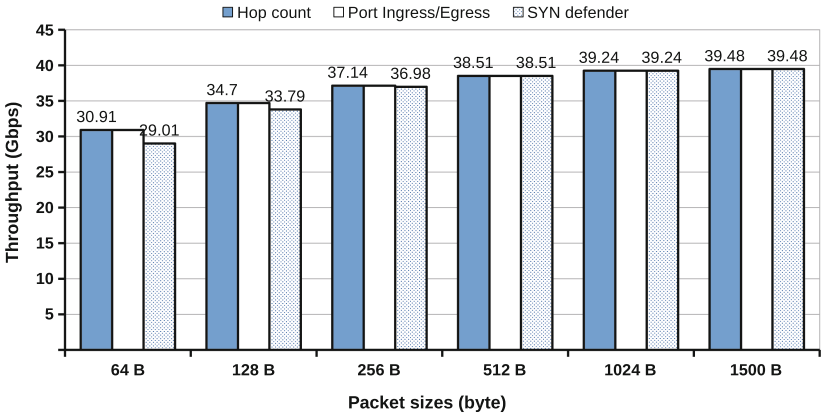


Fig. 5. Processing throughput of signature-based techniques: hop count, port ingress/egress, and SYN defender

leverage the three most frequently used datasets NSL-KDD, UNSW-NB15, and CIC-IDS2017. Figure 6 depicts the throughput of neural network models on the two platforms with different datasets. In this figure, for each model, the above chart bar shows the processing throughput while the below one depicts the maximum packet sending speed that the OSNT tool can send to the testing platform.

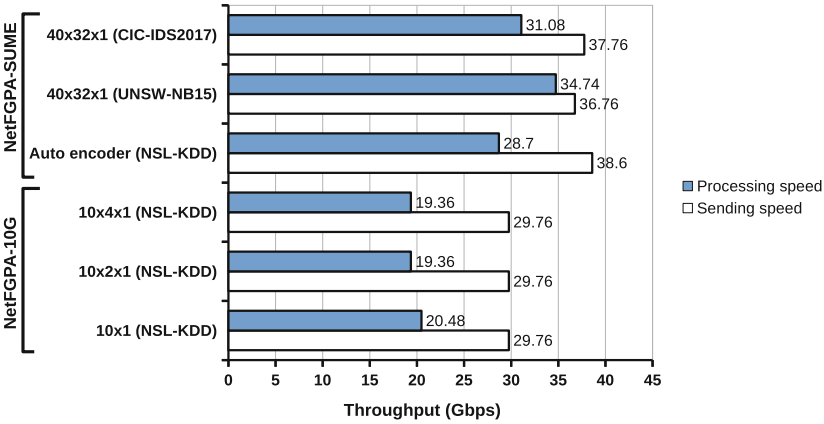


Fig. 6. Processing throughput neural networks with different models and datasets

Figure 6 shows that the neural network models for anomaly-based network intrusion prevention with NetFPGA-10G offer lower throughput than NetFPGA-SUME due to the low working frequency (∼100 MHz compared to ∼200 MHz). Regarding the NetFPGA-10G platform, three models provide almost similar

throughput values with a slightly higher of the simplest model (10×1 - no hidden layer).

Consider the NetFPGA-SUME platform, two neural network models, Autoencoder and $40 \times 32 \times 1$, are built. We test the first model with NSL-KDD and the second with CIC-IDS2017 and UNSW-NB15 datasets. Figure 6 shows that the $40 \times 32 \times 1$ model outperforms the Autoencoder model because it is more straightforward than Autoencoder.

5.4 Accuracy Results

To compare the accuracy of neuron network models, we use the evaluation metrics as shown in Eq. 2, including *Accuracy*, *False Negative Rate*, and *False Positive Rate*, in which TP (true positive), TN (true negative), FP (false positive), and FN (false negative) are defined in [17].

$$\begin{aligned} \text{Accuracy} &= \frac{\text{True positive} + \text{True negative}}{\text{Total testing packets}} \\ \text{False positive rate (FPR)} &= \frac{\text{False positive}}{\text{Total negative in dataset}} \\ \text{False negative rate (FNR)} &= \frac{\text{False negative}}{\text{Total positive in dataset}} \end{aligned} \tag{2}$$

Figure 7 illustrates the three metrics for evaluating the deployed models with three datasets. The figure shows that we achieve the $10 \times 4 \times 1$ with NSL-KDD and $40 \times 32 \times 1$ with CIC-IDS2017 achieve better accuracy, FNR, and FPR values than the others. Although the same neural network model is used, the $40 \times 32 \times 1$ model with UNSW-NB15 provides low accuracy and high FPR because the dataset includes 68 features after refined, while the model accepts only feature vectors with 40 elements.

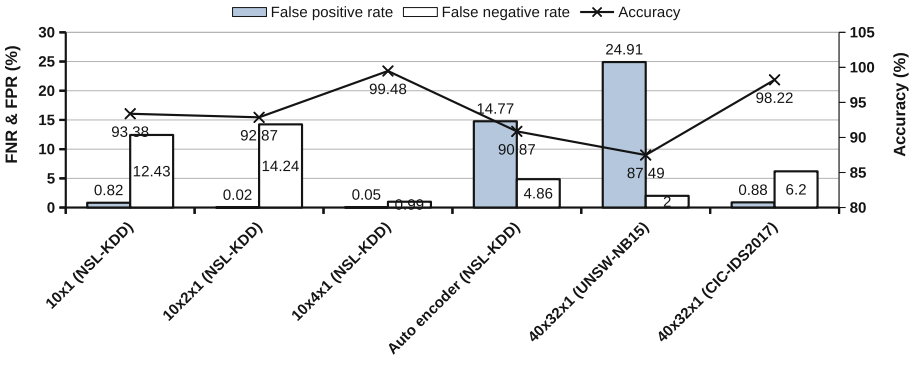


Fig. 7. Accuracy, False Positive Rate (FPR), and False Negative Rate (FNR) of the neural network models

6 Conclusion

As network-based services have rapidly increased during and after the COVID-19 pandemic, preventing network attacks has become essential. This study proposes a high-performance intrusion prevention system framework that utilizes FPGA technology. The framework allows for deploying signature/pattern-based and anomaly/AI-based prevention techniques. Two efficient FPGA-based network processing platforms, NetFPGA-10G and NetFPGA-SUME, are implemented to evaluate the proposed framework. The framework is tested using three well-known datasets, NSL-KDD, UNSW-NB15, and CIC-IDS2017, with three signature-based approaches and five different neural network models. The results show that with signature-based techniques, the framework achieves a throughput of up to 39.48 Gbps, while with neural network models, the throughput can reach up to 34.74 Gbps. Furthermore, the network intrusion prevention system developed using this framework can achieve up to 99.48% accuracy with only a 0.05% false positive rate and a 0.99% false negative rate in the best case.

Acknowledgement. We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

References

1. Abdulhammed, R., Musafer, H., Alessa, A., Faezipour, M., Abuzneid, A.: Features dimensionality reduction approaches for machine learning based network intrusion detection. Electronics **8**(3), 322 (2019)
2. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., Ahmad, F.: Network intrusion detection system: a systematic study of machine learning and deep learning approaches. Trans. Emerg. Telecommun. Technol. (ETT) **32**(1), e4150 (2021). https://doi.org/10.1002/ett.4150. https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4150
3. Ahmed, M., Naser Mahmood, A., Hu, J.: A survey of network anomaly detection techniques. J. Netw. Comput. Appl. **60**, 19–31 (2016). https://doi.org/10.1016/j.jnca.2015.11.016. https://www.sciencedirect.com/science/article/pii/S1084804515002891
4. AMD Xilinx: Vivado overview (2023). https://www.xilinx.com/products/design-tools/vivado.html. Accessed 01 Apr 2023
5. Antichi, G.: OSNT - the open source network tester (2023). https://osnt.org. Accessed 01 Apr 2023
6. Ceška, M., et al.: Deep packet inspection in FPGAs via approximate nondeterministic automata. In: 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 109–117. IEEE (2019)
7. Češka, M., Havlena, V., Holík, L., Lengál, O., Vojnar, T.: Approximate reduction of finite automata for high-speed network intrusion detection. Int. J. Softw. Tools Technol. Transfer **22**(5), 523–539 (2020)
8. Cong, J., et al.: FPGA HLS today: successes, challenges, and opportunities. ACM Trans. Reconfigurable Technol. Syst. **15**(4) (2022). https://doi.org/10.1145/3530775

9. Digilent an NI companay: NetFPGA-SUME virtex-7 FPGA development board (2023). https://digilent.com/shop/netfpga-sume-virtex-7-fpga-development-board/. Accessed 01 Apr 2023
10. Dua, D., Graff, C.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml. Accessed 01 Apr 2023
11. Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E.: Anomaly-based network intrusion detection: techniques, systems and challenges. Comput. Secur. **28**(1–2), 18–28 (2009)
12. García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E.: Anomaly-based network intrusion detection: techniques, systems and challenges. Comput. Secur. **28**(1), 18–28 (2009). https://doi.org/10.1016/j.cose.2008.08.003. https://www.sciencedirect.com/science/article/pii/S0167404808000692
13. Ghanti, S., Naik, G.M.: Defense techniques of SYN flood attack characterization and comparisons. Int. J. Netw. Secur. **20**, 721–729 (2018)
14. Gordon, H., Park, C., Tushir, B., Liu, Y., Dezfouli, B.: An efficient SDN architecture for smart home security accelerated by FPGA. In: 2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), pp. 1–3. IEEE (2021)
15. Hindy, H., et al.: A taxonomy of network threats and the effect of current datasets on intrusion detection systems. IEEE Access **8**, 104650–104675 (2020). https://doi.org/10.1109/access.2020.3000179
16. Ho, T., Cho, S.J., Oh, S.R.: Parallel multiple pattern matching schemes based on cuckoo filter for deep packet inspection on graphics processing units. IET Inf. Secur. **12**(4), 381–388 (2018)
17. Hossin, M., Sulaiman, M.N.: A review on evaluation metrics for data classification evaluations. Int. J. Data Mining Knowl. Manag. Process **5**(2), 1 (2015)
18. Ioannou, L., Fahmy, S.A.: Network intrusion detection using neural networks on FPGA SOCS. In: 2019 29th International Conference on Field Programmable Logic and Applications (FPL), pp. 232–238. IEEE (2019)
19. Kang, J., Kim, T., Park, J.: FPGA-based real-time abnormal packet detector for critical industrial network. In: 2019 IEEE Symposium on Computers and Communications (ISCC), pp. 1199–1203. IEEE (2019)
20. Liao, H.J., Lin, C.H.R., Lin, Y.C., Tung, K.Y.: Intrusion detection system: a comprehensive review. J. Netw. Comput. Appl. **36**(1), 16–24 (2013)
21. Maciel, L.A., Souza, M.A., de Freitas, H.C.: Reconfigurable FPGA-based k-means/k-modes architecture for network intrusion detection. IEEE Trans. Circuits Syst. II Express Briefs **67**(8), 1459–1463 (2019)
22. Moustafa, N., Slay, J.: The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf. Secur. J. Glob. Perspect. **25**(1–3), 18–31 (2016). https://doi.org/10.1080/19393555.2015.1125974
23. Murovic, T., Trost, A.: Massively parallel combinational binary neural networks for edge processing. Elektrotehniski Vestnik **86**(1/2), 47–53 (2019)
24. Murovič, T., Trost, A.: Resource-optimized combinational binary neural network circuits. Microelectron. J. **97**, 104724 (2020)
25. NetFPGA: NetFPGA 10G information (2023). https://netfpga.org/NetFPGA-10G.html
26. Nordström, T., Svensson, B.: Using and designing massively parallel computers for artificial neural networks. J. Parallel Distrib. Comput. **14**(3), 260–285 (1992). https://doi.org/10.1016/0743-7315(92)90068-X. https://www.sciencedirect.com/science/article/pii/074373159290068X

27. Sateesan, A., Vliegen, J., Daemen, J., Mentens, N.: Novel bloom filter algorithms and architectures for ultra-high-speed network security applications. In: 2020 23rd Euromicro Conference on Digital System Design (DSD), pp. 262–269. IEEE (2020)
28. Security today: Malicious DDoS attacks rise 150% in 2022 according to new report (2023). https://securitytoday.com/articles/2023/02/17/malicious-ddos-attacks-rise-150-in-2022-according-to-new-report.aspx
29. Senie, D., Ferguson, P.: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827 (2000). https://doi.org/10.17487/RFC2827. https://www.rfc-editor.org/info/rfc2827
30. Statista Research Department: Internet of things - number of connected devices worldwide 2015–2025 (2016). https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/. Accessed 01 Apr 2023
31. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6 (2009). https://doi.org/10.1109/CISDA.2009.5356528
32. Team, A.N.S.: 2022 in review: DDoS attack trends and insights (2023). https://www.microsoft.com/en-us/security/blog/2023/02/21/2022-in-review-ddos-attack-trends-and-insights/. Accessed 01 Apr 2023
33. Wada, T., Matsumura, N., Nakano, K., Ito, Y.: Efficient byte stream pattern test using bloom filter with rolling hash functions on the FPGA. In: 2018 Sixth International Symposium on Computing and Networking (CANDAR), pp. 66–75. IEEE (2018)
34. Wang, H., Jin, C., Shin, K.G.: Defense against spoofed IP traffic using hop-count filtering. IEEE/ACM Trans. Networking **15**(1), 40–53 (2007). https://doi.org/10.1109/TNET.2006.890133
35. Zhao, Z., Sadok, H., Atre, N., Hoe, J.C., Sekar, V., Sherry, J.: Achieving 100Gbps intrusion prevention on a single server. In: Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation, pp. 1083–1100 (2020)
36. Zilberman, N., Audzevich, Y., Covington, G.A., Moore, A.W.: NetFPGA SUME: Toward 100 Gbps as research commodity. IEEE Micro **34**(5), 32–41 (2014). https://doi.org/10.1109/MM.2014.61