High Throughput Low Latency Network Intrusion Detection on FPGAs: A Raw Packet Approach

Muhammad Ali Farooq
Arizona State University
Tempe, AZ, USA
mafarooq19@asu.edu

Abid Rafique
National University of Sciences and Technology
Islamabad, Pakistan
abid.rafique@seecs.edu.pk

Suhaib A. Fahmy
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
suhaib.fahmy@kaust.edu.sa

Aman Arora
Arizona State University
Tempe, AZ, USA
aman.kbm@asu.edu

***Abstract*—FPGA-accelerated Network Intrusion Detection Systems (NIDS) typically rely on pre-extracted features from network flows, adding complexity and latency to real-time detection. In contrast, this work proposes a novel methodology for direct classification of raw packets, bypassing the need for feature extraction. By leveraging a Look-Up Table (LUT)-based neural network, our approach achieves efficient real-time intrusion detection while maintaining high accuracy. We explore model architectures for binary, 6-class, and 15-class classification, incorporating two levels of sparsity for optimized resource utilization. Using the Edge-IIoT dataset, our models achieve over 99% classification accuracy while delivering up to 1162 million inferences per second on a Virtex Ultrascale+ FPGA. Compared to state-of-the-art raw packet-based NIDS, our approach improves throughput by up to 1000× while significantly reducing resource costs, making it well-suited for high-speed, resource-constrained environments.**

***Index Terms*—Network Intrusion Detection, FPGA, Neural Networks**

I. Introduction

The importance of robust cybersecurity measures continues to grow. As network traffic increases and cyber threats become more sophisticated, Network Intrusion Detection Systems (NIDS) have become essential in modern security infrastructures. Figure 1 illustrates a typical NIDS setup where the system is placed between the firewall and the connected devices to monitor and analyze incoming network traffic. By inspecting each packet, the NIDS detects malicious activity, ensuring protection of connected devices.

Traditional NIDS methods, such as signature-based detection [1] and statistical analysis [2], are being supplemented or replaced by machine learning (ML) techniques, which offer enhanced adaptability and the ability to detect novel threats. These advancements reflect a shift towards intelligent, real-time security measures. However, deploying ML-based NIDS in high-throughput environments poses significant challenges.

Motivation: FPGAs provide a suitable platform for the deployment of ML-based NIDS due to their capability in high-throughput packet processing and their ability to implement a wide range of ML models efficiently and with low-latency.

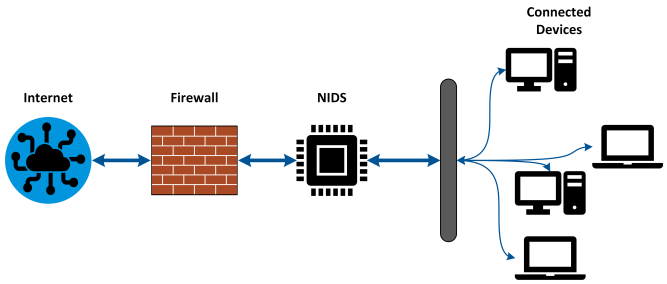


Fig. 1: Network Intrusion Detection System (NID)

However, the adoption of FPGA-accelerated NIDS in real-world scenarios faces challenges. One key issue is the reliance on pre-processed features from network flows. Many FPGA-accelerated NIDS systems rely on pre-calculated features, which adds complexity and latency for real-time detection from live network data since those features would have to be calculated in real-time.

The resource demands of complex ML architectures also pose obstacles. Implementing these on resource-constrained FPGA platforms is challenging due to their significant hardware needs. High-speed networks impose strict requirements on latency and throughput, which many ML-based NIDS solutions struggle to meet, especially for real-time protection.

Proposed Solution: To address these challenges, we propose a novel methodology for FPGA-accelerated NIDS that improves performance and efficiency. Our approach directly classifies raw packets, removing the need for complex feature extraction, and simplifying the pipeline for real-time NID. We further enhance this by optimizing model architectures for binary, 6-class, and 15-class classification tasks on the Edge-IIoT dataset, incorporating two levels of sparsity to improve resource utilization on FPGA platforms. Central to our solution is a Look-Up Table (LUT)-based neural network design, enabling efficient on-chip processing of network traffic for live packet classification with reduced overhead.

Key Contributions: Our contributions are:

- We propose an FPGA-accelerated NIDS that directly classifies raw packets, eliminating the need for feature extraction.
- We implement optimized model architectures for binary, 6-class, and 15-class classification with two levels of sparsity, leveraging the LogicNets [3] framework for efficient neural network implementation.
- Our implementation on the Virtex Ultrascale+ FPGA achieves high throughput using fewer than 10,000 LUTs, demonstrating suitability for resource-constrained environments.

II. Related Work

NID Systems (NIDS) have evolved from traditional signature-based [1] and statistical methods [2] to incorporate ML techniques [4]. However, many ML-based NIDS solutions are primarily designed for CPU [5] or GPU [6] platforms, which struggle to manage the high-speed traffic typical of modern networks.

To address these challenges, researchers have explored FPGA-based implementations of ML models for NID, which offer significant advantages in processing speed, parallelism, and resource efficiency. Various ML algorithms have been implemented on FPGAs for intrusion detection, including Principal Component Analysis (PCA) [7], Naive Bayes [8], and k-means clustering [9]. Nonetheless, neural networks have garnered attention recently due to their improved accuracy and adaptability for intrusion detection.

Several studies have investigated the application of neural networks in FPGA-based NIDS. Ngo et al. [10] implemented Artificial Neural Networks (ANNs) on FPGAs using datasets such as NSL-KDD and IoT-23. Similarly, Ioannou et al. [11] and Alrawashdeh et al. [12] accelerated neural networks on the NSL-KDD dataset, with the latter employing a Deep Belief Network (DBN). Murovič et al. [13] proposed a fully combinational Binary Neural Network (BNN), achieving significant improvements in speed and resource efficiency using the UNSW-NB15 [14] and NSL-KDD datasets. Vreča et al. [15] also implemented a BNN for the UNSW-NB15 dataset, further showcasing the viability of BNNs for FPGA-based NIDS. Another notable approach is LogicNets [3], which accelerates sparse deep neural networks (DNNs) using Look-Up Tables (LUTs) on FPGAs. This method demonstrated promising results on the UNSW-NB15 dataset, offering efficient hardware implementations for NIDS.

A significant limitation of existing implementations is their focus on model inference performance, often neglecting the effect of feature extraction on real-time detection. Real-time NIDS must manage packet parsing, feature extraction, and classification at high speeds while adhering to FPGA resource constraints.

Le Jeune et al. [16], [17] proposed an innovative method utilizing raw packet data from the UNSW-NB15 and CICIDS2017 [18] datasets. Their method grouped packet headers into "flow buckets" based on Flow Identifiers (FID), allowing for multi-packet analysis in binary Convolutional Neural Networks (CNNs) using the FINN framework. While this system enhances detection by incorporating flow-level context, it also introduces significant complexity. The process of sorting packets into flows and using costly CNN techniques adds memory overhead and complicates FPGA implementation, which already face challenges in balancing resource utilization, latency, and throughput. Furthermore, converting flow-based datasets into packet-based formats complicates training, and the datasets used only contain labeled TCP/UDP traffic, restricting the system's generalizability.

Our work eliminates the limitations in prior work by directly processing raw packet data without relying on complex techniques like Convolutional Neural Networks (CNNs) or flow-based sorting. This approach reduces the memory and computational overhead associated with sorting packets into flows and using resource-intensive ML architectures.

III. Proposed Solution

A. Single Packet Network Intrusion Detection

Feature extraction is a critical aspect of ML-based NID. Traditional approaches often rely on manually selected flow metadata, such as the number of transmitted bytes or flow duration. However, these methods require significant storage or packet parsing, which is inefficient for real-time, resource-constrained environments [19], [8], [15].

A more streamlined alternative is using raw traffic-based features directly from the binary packet stream, reducing preprocessing while maintaining competitive performance [20], [21], [22], [23]. This approach aligns with the end-to-end deep learning paradigm, where the initial layers of a deep neural network (DNN) can automatically learn higher-level features from raw data, enabling efficient real-time processing on hardware platforms like FPGAs.

Le Jeune et al. [16] explored raw packet-based intrusion detection by proposing a system that processes only the first few bytes of a packet. However, as discussed in Section II, this approach has several drawbacks that limit its suitability. Building on Le Jeune et al.'s work, we argue that the complexity introduced by flow buckets can be avoided in favor of a more efficient single-packet detection system, especially in resource-constrained environments. In our approach, similar to Le Jeune

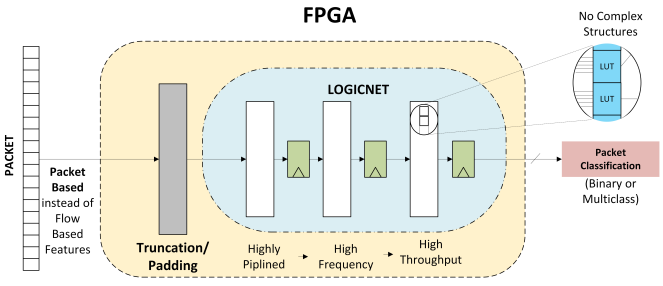


Fig. 2: System Diagram of Proposed Solution

TABLE I: Edge-IIoT Dataset Traffic Distribution

IoT Traffic	Class	Records
Normal	Normal	11,223,940
Attack	Backdoor	24,862
	DDoS_HTTP	229,022
	DDoS_ICMP	2,914,354
	DDoS_TCP	2,020,120
	DDoS_UDP	3,201,626
	Fingerprinting	1,001
	MITM	1,229
	Password	1,053,385
	Port_Scanning	22,564
	Ransomware	10,925
	SQL_Injection	51,203
	Uploading	37,634
	Vulnerability_Scanner	145,869
	XSS	15,915
Total		20,952,648

et al. [17], packets are represented by their first 64 bytes, which is sufficient to capture essential header information for TCP/UDP traffic, as most critical metadata is contained within the first few bytes. Shorter packets are padded to 64 bytes, ensuring consistency in data processing.

Figure 2 presents a high-level system diagram of our approach. Raw packets are fed into a Look-Up Table (LUT)-based neural network, like LogicNets [3], after truncation/padding. The output of the model is a classification of the packet into a class, identifying whether the packet is normal traffic or a type of intrusion. This design choice reduces complexity and hardware overhead while still maintaining high classification accuracy.

B. Edge-IIoT Dataset

Most datasets used in NIDS research are flow-based, which introduces further complexity while performing training in packet-based contexts. The Edge-IIoT dataset [24] offers a more suitable alternative by providing packet-based data for IoT and IIoT environments. The Edge-IIoT dataset captures data from a wide range of IoT devices, which are frequent targets of cyber threats, and provides a comprehensive simulation of real-world network environments. Moreover, unlike the popular UNSW-NB15 and CICIDS-2017 datasets, the Edge-IIoT dataset is not limited to TCP/UDP traffic, thereby providing more comprehensive training data.

Table I summarizes the distribution of all traffic types in the dataset, providing details on the class of traffic, such as Normal and various types of Attacks.

To be consistent with the paper introducing the Edge-IIoT dataset [24], we develop models for binary, 6-class, and 15-class classification, as discussed further in the paper.

C. LogicNets Framework

The LogicNets framework [3], developed by Umuroglu et al. at Xilinx Research Labs, is designed for extreme-throughput machine learning applications on resource-constrained FPGAs. The core concept involves mapping individual neuron functions into Look-Up Tables (LUTs), enabling efficient FPGA implementations. LogicNets employs quantized artificial neurons, representing each neuron as a truth table. These truth tables, termed Hardware Building Blocks (HBBs), form the framework's fundamental components. Neuron Equivalents (NEQs) (neurons with quantized fixed inputs and outputs) are trained in conventional deep neural networks and subsequently mapped to HBBs for FPGA deployment. The framework emphasizes maintaining a low fan-in (number of inputs to a neuron) to avoid exponential growth in LUT cost. The LogicNets design flow comprises three key steps: (Defining and training NEQs, (2) Converting NEQs into a netlist of HBBs, and (3) Synthesizing the netlist into an FPGA bitfile. This approach ensures neural networks are co-designed with FPGA hardware, optimizing for resource efficiency and high throughput. LogicNets is particularly suitable for high-throughput FPGA-based applications like Network Intrusion Detection Systems (NIDS).

LogicNets supports sparsity, and allows users to change the number of inputs to each, which consequently changes the number of connections between layers, based on hyperparameters. There are two parameters that control the input size of each NEQ. γ defines the number of inputs for each NEQ while β defines the width of each input. Each NEQ has a single output. For the first and hidden layers, the output width is defined by the input width of the next layer. The output width of NEQs in the output layer matches their input width. Increased sparsity, achieved by lower values of γ or β, allows the creation of resource-efficient implementations.

To define a LogicNet, the following hyperparameters are necessary:

- Number and size of each layer: e.g., [45, 5] defines a model with 3 layers (45 NEQs, 5 NEQs, and output layer)
- Sparsity definitions:
  - β_i, γ_i: settings for the first layer
  - β, γ: settings for hidden layers
  - β_o, γ_o: settings for the last layer

IV. Methodology

A. Formatting the Edge-IIoT Dataset

We preprocess the Edge-IIoT dataset to enable direct packet-based classification. This step is necessary due to the dataset's original format, which contains whole packets. Our preprocessing steps ensure compatibility with our model while preserving the raw packet structure for effective intrusion detection. The formatting steps are:

1) Truncating packets in the .pcap files to 64 bytes using Wireshark [25]
2) Extracting byte-level information from the truncated packets and converting to plaintext
3) Cleaning the text files and converting the data from hexadecimal to binary
4) Padding packets smaller than 64 bytes for uniformity
5) Adding labels to indicate the traffic class
6) Removing duplicate packets to ensure data integrity

TABLE II: Formatted Dataset Split and Total Counts

Class	Dataset Split		
	Train	Validation	Test
Normal	6,506,277	2,695,457	92,948
Backdoor	16,870	6,989	241
DDoS_HTTP	159,556	66,102	2,280
DDoS_ICMP	2,039,769	845,047	29,141
DDoS_TCP	1,413,288	585,505	20,191
DDoS_UDP	2,250,668	932,419	32,153
Fingerprinting	730	302	12
MITM	203	84	3
Password	736,450	305,100	10,522
Port_Scanning	14,439	5,982	207
Ransomware	6,871	2,846	99
SQL_Injection	35,829	14,843	513
Uploading	26,308	10,899	376
Vulnerability_Scanner	185,847	76,994	2,656
XSS	10,848	4,494	156
Total	13,403,953	5,553,063	191,498

7) Randomly split the dataset into train, validation, and test sets, as shown in Table II.

For binary classification, we simply collapse all the attack classes into a single label, thus transforming the dataset presented in Table II to a binary dataset. For 6-class classification, we collapse the labels as follows:

- **DDoS**: DDoS_HTTP, DDoS_ICMP, DDoS_TCP, DDoS_UDP
- **Scanning**: Fingerprinting, Port_Scanning, Vulnerability_Scanner
- **Injection**: XSS, SQL_Injection, Uploading
- **Malware**: Backdoor, Password, Ransomware
- **Normal**: Normal
- **MITM**: MITM

The data remains unchanged for the 15-class classification. This approach causes issues since there are not enough data samples for each class for multiclass classification. As such, random oversampling is used to increase the training data.

B. Neural Architecture Search (NAS)

To find the optimal LogicNets model for each form of classification problem, we perform manual NAS over a variety of models and two sparsity settings. The model architectures are presented in Table III. We consider small, regular, medium, and large models, by gradually increasing the number of NEQs and layers. Each model is assigned a name for easy reference, as shown in the table. We consider a regular (non-sparse) and sparse model for each model architecture. Our sparse models differ from their non-sparse counterparts in the γ value, but not in the β value. The sparsity settings used are presented in Table IV.

The results for the binary, 6-class and 15-class NAS are presented in Figures 3a, 3b and 3c respectively.

Figure 4 shows a zoomed-in view of the charts in Figure 3 to show detailed trends in the region of interest. It is clear that the accuracy increases at first as models get larger and then saturates.

TABLE III: Model Architectures Explored During NAS

Model	Binary	6-Class	15-Class
c-s2	25,1	30,6	80,15
c-s1	100,25,1	100,25,6	100,25,15
c	128,32,1	128,32,6	128,32,15
c-m1	128,60,15,1	128,60,30,6	128,60,30,15
c-m2	200,86,15,1	300,86,30,6	300,86,30,15
c-big	256,86,15,1	512,86,15,6	593,100,33,15

For 6-class and 15-class models, accuracy even declines after a certain size, making further increase in model size unnecessary. For binary models, the accuracy gain beyond a certain point is minimal, making further increases in model size inefficient.

Based on the performance results presented in Figure 3 and Figure 4, the c-m2 model was selected for all categories due to its superior performance across both regular and sparse settings. The performance of the regular models is primarily used to select the architecture, but the performance of the sparse models is also taken into consideration.

For Binary classification, each model is trained for 2 epochs with a learning rate of 10^{-3} and a batch size of 256 on the training data and evaluated on both the validation and test sets. For multiclass classification, each model is trained for 5 epochs with a learning rate of 10^{-3} and a batch size of 256 on the training data and evaluated on both the validation and test sets.

C. Synthesis and Implementation

Once the final models are determined through NAS, the layer specifications are input into LogicNets, resulting in Register Transfer Level (RTL) output. Out-of-context synthesis is performed using Vivado 2021.1 for the Virtex Ultrascale+ card, with a timing goal of $P = 1ns$. The post-implementation results are then utilized for further analysis.

V. Results

A. Machine Learning Metrics

Table V presents the results of the six models derived from NAS, including accuracy, F-1 score, and detection score for both validation and test sets.

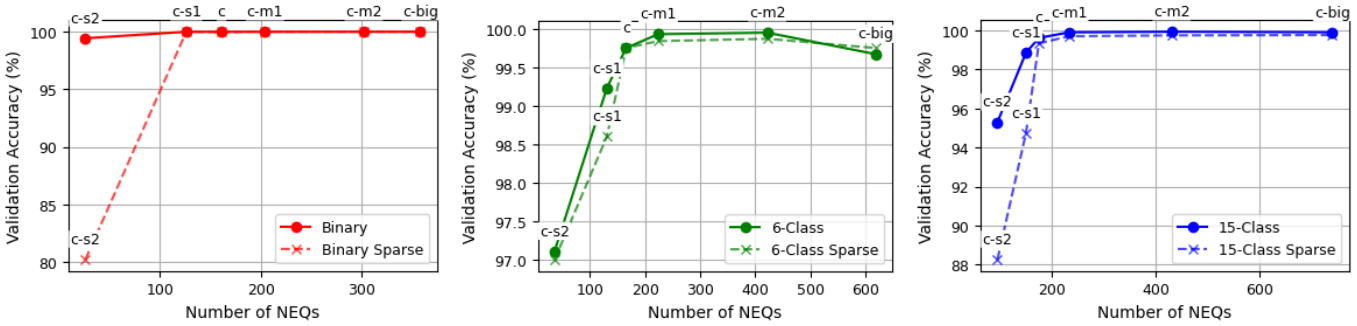
Classification accuracy is defined as:

$$Acc = \frac{tp + tn}{tp + fp + tn + fn}$$

where tp, fp, tn, and fn are true positives, false positives, true negatives, and false negatives, respectively. All models achieve

TABLE IV: Sparsity Settings During NAS
(β and γ are explained in Section III-C)

	β_i	γ_i	β	γ	β_o	γ_o
Regular	1	6	2	6	2	7
Sparse	1	4	2	4	2	7



(a) Binary (b) 6-Class (c) 15-Class

Fig. 3: Validation Accuracy for Neural Architecture Search

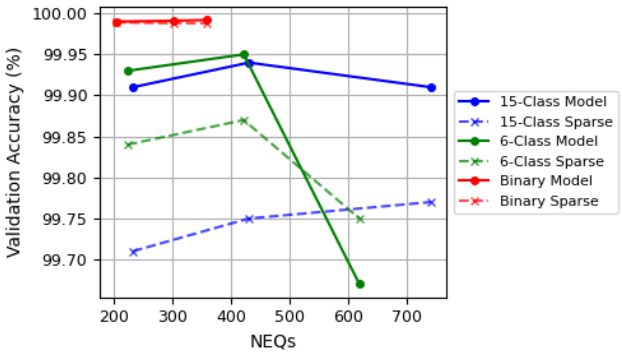


Fig. 4: Comparison of the Accuracy of the Models in Fig. 3

TABLE V: ML Results for Selected Models
(All Values are Percentages)

	Validation			Test		
	Acc	**F-1**	**DS**	**Acc**	**F-1**	**DS**
binary	99.991	99.991	99.991	99.994	99.994	99.994
binary-sparse	99.988	99.988	99.988	99.992	99.992	99.992
6-class	99.95	99.95	99.989	99.95	99.95	99.995
6-class-sparse	99.87	99.88	99.975	99.87	99.88	99.982
15-class	99.94	99.94	99.983	99.95	99.95	99.989
15-class-sparse	99.75	99.78	99.963	99.76	99.79	99.964

accuracy above 99.9% in both validation and test sets, indicating excellent performance with minimal misclassification. Sparse models maintain high accuracy despite reduced feature representations, demonstrating robust architecture.

The F-1 score is given by:

$$F_1 = \frac{tp}{tp + 0.5 \cdot (fp + fn)}$$

The F-1 score, ranging from 0 to 100%, reflects model precision and recall, with a higher score indicating better performance. The weighted F-1 score is used for multiclass models to address class imbalance. As shown in the table, F-1 scores align closely with accuracy, hovering around 99.9%.

The Detection Score (DS) metric (introduced by Le Jeune et al. [26]), based on the F-1 score for binary classification, adapts to multiclass scenarios by treating any attack classification as a true positive (tp). This indicates how effectively a Network Intrusion Detection System (NIDS) detects attacks. Both binary and multiclass models show DS values above 99.9%, highlighting strong detection capabilities with minimal false positives or negatives. Sparse models exhibit slight performance drops but remain highly effective.

B. FPGA Implementation Results

The LUT (Look-Up Table) count and FF (Flip-Flop) count indicate the resource utilization for implementing the models on an FPGA. No DSPs and BRAMs are used in our implementations. Latency and Maximum Frequency are derived from the Worst Negative Slack (WNS) reported by Vivado after implementation. The minimum clock period is calculated as:

$$P_{\text{min}} = 1 - \text{WNS ns}$$

The latency of the model is given by P_{min} multiplied by the number of layers, as the models are pipelined with registers between layers. The maximum frequency achievable is determined by:

$$F_{\text{max}} = \frac{1000}{1 - \text{WNS}} \text{ MHz}$$

Since the model is pipelined, throughput directly corresponds to the frequency; for instance, a frequency of 1 MHz equates to a throughput of 1 million packets per second, since we only ingest the first 64 bytes of packets in a single cycle. These metrics indicate the model's operational efficiency and suitability for real-time applications.

The implementation results for the trained c-m2 models are shown in Table VI. All models utilize fewer than 10,000 LUTs and exhibit latencies below 7 ns. Notably, sparse models achieve throughputs exceeding 1100 million packets per second while demonstrating significantly lower resource usage compared to their regular counterparts.

C. Comparison with State-of-the-Art NIDS

Our evaluation demonstrates significant improvements in both hardware efficiency and detection accuracy, particularly when comparing our binary and binary-sparse models against prior works in FPGA-based Network Intrusion Detection

TABLE VI: FPGA Implementation Results of Selected Models (Throughput Measured in 10^6 Inferences per Second)

	LUT	FF	Latency (ns)	Throughput	f_{max} (MHz)
binary	3337	1496	5.984	668.45	668.45
binary-sparse	500	337	3.496	1144.16	1144.16
6-class	7033	2319	6.136	651.89	651.89
6-class-sparse	1652	736	3.44	1162.79	1162.79
15-class	9033	2455	5.863	681.66	681.66
15-class sparse	2764	861	3.528	1133.79	1133.79

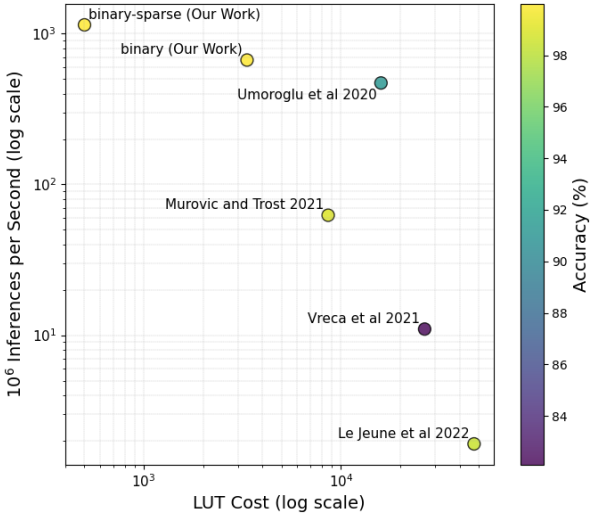


Fig. 5: Comparison of Our Binary Classification Model with Previous Work on Binary Classification ML NIDS

Systems (NIDS). Figure 5 highlights the trade-offs between LUT cost, inference speed, and accuracy in various binary classification models, providing a clear comparison of our approach with the state of the art.

The results, as visualized in the plot, indicate that our binary-sparse model outperforms others, achieving over 1000 million inferences per second with minimal LUT cost. Notably, our work surpasses previous implementations such as those by Umuroglu et al. [3], Vreča et al. [15], and Le Jeune et al. [17], in terms of both inference speed and accuracy. While BNN-based methods like Murovič et al. [13] achieve notable throughput, they still lag in resource efficiency compared to our binary-sparse approach.

Table VII further underscores this performance. While models like Vreča et al. [15] and Le Jeune et al. [17] have relatively high LUT costs and lower throughput, our proposed solution maintains a much lower LUT utilization while simultaneously delivering higher accuracy and throughput. The table further underscores this performance by illustrating key metrics across different models:

(1) *LUT Utilization:* Our binary-sparse model achieves the lowest LUT utilization at only 500 LUTs, significantly lower than other models such as Le Jeune et al., which uses over 47,000 LUTs.

(2) *Throughput:* Our models achieve superior throughput, with the binary-sparse model reaching over 1144 million inferences per second, far exceeding others like Vreča et al., which manages only about 11 million.

(3) *Accuracy:* Both our models maintain high accuracy levels above 99%, outperforming others like Umuroglu et al., which achieves around 91%.

These results demonstrate that our approach not only enhances hardware efficiency but also maintains or improves detection accuracy compared to existing FPGA-based NIDS solutions.

VI. Discussion

The balance between accuracy and hardware resource efficiency is critical for real-time NIDS applications, especially in environments with constrained resources, such as Industrial IoT (IIoT). By reducing complexity, using LUT-based networks, and leveraging sparse neural networks, our method enhances detection performance without the overhead typically associated with more complex architectures [17], [11], [15].

However, it is crucial to recognize the limitations of single-packet intrusion detection systems, despite their impressive speed and efficiency. While these systems excel in rapid detection by analyzing individual packets, they cannot be expected to function independently as a complete defense mechanism. Single-packet analysis may lack the context needed to identify more sophisticated, multistep attacks or those spread across multiple packets.

Therefore, future work should focus on developing a hybrid approach that integrates single-packet intrusion detection with flow rule-based systems. Flow rule-based systems [27], [28], which examine sequences of packets to detect patterns indicative of malicious activity, provide the necessary context and depth of analysis. A combined implementation would offer the speed of single-packet detection with the comprehensive view of flow-based methods, creating a more robust and versatile intrusion detection framework. This integration would enable real-time detection while addressing more complex attack patterns, ensuring enhanced security for IIoT and other network environments.

VII. Conclusion

This work presents a novel approach to FPGA-based Network Intrusion Detection Systems (NIDS) that achieves exceptional performance in terms of both accuracy and hardware efficiency. The results demonstrate significant improvements in throughput, achieving up to a 1000× increase compared to state-of-the-art NIDS using raw packet features, while simultaneously reducing resource utilization. All models require fewer than 10,000 LUTs, ensuring suitability for resource-constrained FPGA platforms. Our models achieve over 99% accuracy across all classification tasks. Future work should focus on integrating this method with flow-based systems to create a more comprehensive security solution for IIoT and other network environments.

TABLE VII: Comparison of Selected Binary Classification Models with Prior Work (OFE means On-Chip Feature Extraction. Throughput (Tput) is in 10^6 Inferences per Second. A value of "–" means the metric is not specified in the paper)

Ref.	OFE	Dataset	FPGA	Acc. (%)	LUTs	FFs	T'put	f_{max} (MHz)	Latency (ns)
[3]	No	UNSW-NB15	Virtex Ultrascale+	91.3	15949	1274	471	471	10.5
[15]	No	UNSW-NB15	Zynq-7000	82.1	26556	2228	10.98	142.8	91
[17]	Yes	CICIDS2017	ZCU104	98.4	47297	–	1.901	336.13	6680
[13]	No	NSL-KDD	Kintex Ultrascale+	98.96	8606	0	62.5	–	19
binary (ours)	Yes	EDGE-IIOT	Virtex Ultrascale+	99.994	3337	1496	668.45	668.45	5.984
binary-sparse (ours)	Yes	EDGE-IIOT	Virtex Ultrascale+	99.992	500	337	1144.16	1144.16	3.496

REFERENCES

[1] C. Kruegel and T. Toth, "Using decision trees to improve signature-based intrusion detection," *Recent Advances in Intrusion Detection*, p. 173–191, 2003.

[2] E. Kabir, J. Hu, H. Wang, and G. Zhuo, "A Novel Statistical Technique for Intrusion Detection Systems," *Future Generation Computer Systems*, vol. 79, p. 303–318, Feb. 2018. [Online]. Available: http://dx.doi.org/10.1016/j.future.2017.01.029

[3] Y. Umuroglu, Y. Akhauri, N. J. Fraser, and M. Blott, "LogicNets: Co-Designed Neural Networks and Circuits for Extreme-Throughput Applications," in *International Conference on Field-Programmable Logic and Applications (FPL)*, 2020, pp. 291–297.

[4] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, and A. Wahab, "A Review of Intrusion Detection Systems Using Machine and Deep Learning in Internet of Things: Challenges, Solutions, and Future Directions," *Electronics*, vol. 9, no. 7, p. 1177, 2020.

[5] G. De Carvalho Bertoli, L. A. Pereira Júnior, O. Saotome, A. L. Dos Santos, F. A. N. Verri, C. A. C. Marcondes, S. Barbieri, M. S. Rodrigues, and J. M. Parente De Oliveira, "An End-to-End Framework for Machine Learning-Based Network Intrusion Detection System," *IEEE Access*, vol. 9, pp. 106 790–106 805, 2021.

[6] G. Karatas, O. Demir, and O. K. Sahingoz, "A Deep Learning Based Intrusion Detection System on GPUs," in *2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2019, pp. 1–6.

[7] A. Das, D. Nguyen, J. Zambreno, G. Memik, and A. Choudhary, "An FPGA-Based Network Intrusion Detection Architecture," *Trans. Info. For. Sec.*, vol. 3, no. 1, p. 118–132, mar 2008. [Online]. Available: https://doi.org/10.1109/TIFS.2007.916288

[8] Z. Todorov, D. Efnusheva, and T. Nikolić, "FPGA Implementation of Computer Network Security Protection with Machine Learning," in *2021 IEEE 32nd International Conference on Microelectronics (MIEL)*, 2021, pp. 263–266.

[9] L. Andrade Maciel, M. Alcântara Souza, and H. Cota de Freitas, "Reconfigurable FPGA-Based K-Means/K-Modes Architecture for Network Intrusion Detection," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 8, pp. 1459–1463, 2020.

[10] D.-M. Ngo, B. Tran-Thanh, T. Dang, T. Tran, T. N. Thinh, and C. Pham-Quoc, "High-Throughput Machine Learning Approaches for Network Attacks Detection on FPGA," *Context-Aware Systems and Applications, and Nature of Computation and Communication*, pp. 47–60, 2019.

[11] L. Ioannou and S. A. Fahmy, "Network Intrusion Detection Using Neural Networks on FPGA SoCs," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2019, pp. 232–238.

[12] K. Alrawashdeh and C. Purdy, "Reducing Calculation Requirements in FPGA Implementation of Deep Learning Algorithms for Online Anomaly Intrusion Detection," in *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, 2017, pp. 57–62.

[13] T. Murovič and A. Trost, "Genetically Optimized Massively Parallel Binary Neural Networks for Intrusion Detection Systems," *Computer Communications*, vol. 179, pp. 1–10, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366421002693

[14] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[15] J. Vreča, I. Ivanov, G. Papa, and A. Biasizzo, "Detecting Network Intrusion Using Binarized Neural Networks," in *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, 2021, pp. 622–627.

[16] L. Le Jeune, T. Goedemé, and N. Mentens, "Towards Real-Time Deep Learning-Based Network Intrusion Detection on FPGA," *Applied Cryptography and Network Security Workshops: ACNS 2021 Satellite Workshops, AIBlock, AIHWS, AIoTS, CIMSS, Cloud S&P, SCI, SecMT, and SiMLA, Kamakura, Japan, June 21–24, 2021, Proceedings*, p. 133–150, 2021.

[17] L. L. Jeune, T. Goedemé, and N. Mentens, "Feature Dimensionality in CNN Acceleration for High-Throughput Network Intrusion Detection," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, 2022, pp. 366–374.

[18] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, INSTICC. SciTePress, 2018, pp. 108–116.

[19] T. Murovič and A. Trost, "Resource-Optimized Combinational Binary Neural Network Circuits," *Microelectronics Journal*, vol. 97, p. 104724, 2020.

[20] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[21] L. Han, Y. Sheng, and X. Zeng, "A Packet-Length-Adjustable Attention Model Based on Bytes Embedding Using Flow-WGAN for Smart Cybersecurity," *IEEE Access*, vol. 7, pp. 82 913–82 926, 2019.

[22] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, "Network Intrusion Detection: Based on Deep Hierarchical Network and Original Flow Data," *IEEE Access*, vol. 7, pp. 37 004–37 016, 2019.

[23] Y. Zhang, X. Chen, D. Guo, M. Song, Y. Teng, and X. Wang, "PCCN: Parallel Cross Convolutional Neural Network for Abnormal Network Traffic Flows Detection in Multi-Class Imbalanced Network Traffic Flows," *IEEE Access*, vol. 7, pp. 119 904–119 916, 2019.

[24] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning," *IEEE Access*, vol. 10, pp. 40 281–40 306, 2022.

[25] Wireshark Foundation, "Wireshark," network protocol analyzer. [Online]. Available: https://www.wireshark.org/

[26] L. Le Jeune, T. Goedemé, and N. Mentens, "Machine Learning for Misuse-Based Network Intrusion Detection: Overview, Unified Evaluation and Feature Choice Comparison Framework," *IEEE Access*, vol. 9, pp. 63 995–64 015, 2021.

[27] F. Erlacher and F. Dressler, "On High-Speed Flow-Based Intrusion Detection Using Snort-Compatible Signatures," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 495–506, 2022.

[28] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An Integrated Rule-Based Intrusion Detection System: Analysis on UNSW-NB15 Data Set and the Real-Time Online Dataset," *Cluster Computing*, vol. 23, no. 2, p. 1397–1418, Oct. 2019. [Online]. Available: http://dx.doi.org/10.1007/s10586-019-03008-x