High Throughput Neural Network for Network Intrusion Detection on FPGAs: An Algorithm-Architecture Interaction

Muhammad Ali Farooq[1,2][a], Syed Muhammad Fasih Ul Hassan[1], Muhammad Umer Farooq[3] and Abid Rafique[1,2]

[1]*RapidSilicon, Islamabad, Pakistan*

[2]*School of Engineering, Hong Kong University of Science and Technology, Hong Kong SAR*

[3]*School of Electrical Engineering and Computer Sciences, NUST, Islamabad, Pakistan*

Keywords: FPGA, Shallow Neural Networks, Machine Learning, Loss Function, Network Intrusion Detection.

Abstract: With the increasing digitization of human activities, the risk of cyberattacks has increased. The resulting potential for extensive harm underscores the need for robust detection mechanisms. Neural network-based solutions deployed on FPGAs provide robust and fast solutions to this challenge by scrutinizing network traffic patterns to identify malicious behaviours. This paper introduces a novel loss function tailored for use on the UNSW-NB15 dataset. This loss function allows a small, binarized neural network deployed on FPGAs to function at high speed with competitive accuracy. This paper further introduces a model trained using this method which has a maximum operating frequency of 1.028 GHz and LUT and flip-flop usage of 135 and 148 respectively, with an accuracy of 90.91% and an F1 score of 91.81%. The high operating frequency and low LUT footprint provide avenues for further research, even though the accuracy and F1 score are not groundbreaking.

1 INTRODUCTION

The rapid growth of the digital world provides increased opportunities for attackers to prey upon individual users and critical infrastructure (Ardagna et al., 2022). Furthermore, the rapid increase in network traffic has led to the use of high-speed network infrastructure. Multiple avenues of attack detection are being developed to provide security to the increasingly high-speed network infrastructure. One such avenue is that of a network intrusion detection system (NIDS) which detects attacks in local traffic. Currently, NIDS focus on attack detection through pattern matching and statistical analysis. However, such methods are slow and computationally intensive. As such, current research focuses on the development of machine learning (ML) based systems to achieve the same effect. These include convolutional neural networks (CNNs) (Azizjon et al., 2020; Jeune et al., 2022; Wang et al., 2017), recurrent neural networks (RNN's) (Yin et al., 2017), and support-vector machines (SVNs) (Yang et al., 2021) for anomaly detection and attack classification.

A common challenge across these methods is dealing with imbalanced datasets where network attacks are infrequent, leading to reduced detection accuracy. Strategies to address this issue include oversampling (Zheng et al., 2015), undersampling (Tahir et al., 2012), Synthetic Minority Oversampling Technique (SMOTE)(Wang and Huang, 2018), the use of generative adversarial networks (GANs) to generate additional minority samples (Andresini et al., 2021), and Difficult Set Sampling Technique (DSSTE) to both reduce the data points of the majority class and increase the number of minority samples (Liu et al., 2021). Recent advancements propose loss functions like the attack-sharing loss to handle class imbalance effectively, especially in the realm of network intrusion detection (Dong et al., 2021; Ehmer et al., 2022).

The UNSW-NB15 is a popular choice for hardware-deployed binary neural networks, and it is discussed in depth in 4. An important point to note is that it has a class imbalance, with more total attack class samples as compared to normal samples in the training data.

Another point to note is that most of the currently explored strategies are prohibitively slow, with higher

[a] https://orcid.org/0009-0009-8138-8118

Farooq, M., Fasih Ul Hassan, S., Farooq, M. and Rafique, A.
High Throughput Neural Network for Network Intrusion Detection on FPGAs: An Algorithm-Architecture Interaction.
DOI: 10.5220/0012367800003648
In *Proceedings of the 10th International Conference on Information Systems Security and Privacy (ICISSP 2024)*, pages 423-429
ISBN: 978-989-758-683-5; ISSN: 2184-4356

Table 1: Previous Work on BNN Based Network Intrusion Detection Systems Trained on UNSW-NB15.

Model	Acc.(%)	Latency(ns)	No. of LUTs	f_{max}(MHz)
MPCBNN (Murovič and Trost, 2019)	90.74	19.6	51353	-
MPBNN (Murovič and Trost, 2021)	92.04	19	26879	-
BNN (Vreča et al., 2021)	82.1	91	26556	142.85
NID-S (Umuroglu et al., 2020)	83.88	3.70	3586	811
NID-M (Umuroglu et al., 2020)	91.3	10.5	15949	471
NID-S (Umoroglu et al., 2023)	90.5	3.96	650	758.15
NID-M (Umoroglu et al., 2023)	92.6	3.57	1649	839.63
NID-L (Umoroglu et al., 2023)	92.9	10.0	8106	498.26

accuracies in intrusion detection corresponding with slower systems. One approach to solving the speed challenge is to deploy and accelerate NIDS on field programmable gate arrays (FPGAs). While many acceleration frameworks are in development (such as Vitis AI by Xilinx (2023), hls4ml by Duarte et al. (2018), and FINN by Umuroglu et al. (2017) and Blott et al. (2018)), only LogicNets (Umuroglu et al., 2020) is designed to cater towards the acceleration of models in scenarios where speed is of the highest priority.

In this paper, we introduce a loss function based on a novel regularisation term. We further train a binarized neural network on the UNSW-NB15 dataset using the LogicNets framework and this new loss function. Synthesis is performed for FPGA deployment and the results are discussed and compared with other state-of-the-art solutions.

The rest of the paper is organised as follows. In Section 2 we introduce related work, after which we introduce the LogicNets framework in Section 3. After detailing the specifics of the UNSW-NB15 in Section 4, we present our experiment in Section 5 and the results and appropriate discussion in Section 6. We conclude our paper in Section 7 with a summary of the results and possible directions for future work.

2 RELATED WORK

While network intrusion detection systems are not a new concept, there has been a large research effort to use machine learning to detect attacks (Buczak and Guven, 2016). Most of these efforts are designed for deployment on CPUs or GPUs. Such approaches can not handle the high speeds of current network traffic, which can exceed 100 GBPS.

FPGAs provide an alternative platform for the deployment of such models since they allow for hardware implementation. Among FPGA-based implementations of ML-based NIDSs, neural-network-based architectures are popular. Ngo et al. explored various iterations of a neural network deployed on an FPGA for the NSL-KDD dataset and IoT-23 dataset (Ngo et al., 2019, 2021). Murovič et al. proposed and iterated upon a fully combinational Binary Neural Network (FCBNN) that is evaluated on the UNSW-NB15 dataset (Murovič and Trost, 2019, 2020, 2021). Similarly, Umuroglu et al. proposed LogicNets as a technique to deploy BNNs on FPGAs, and demonstrated its potential by accelerating a NIDS for UNSW-NB15 and further iterating on the design (Umuroglu et al., 2020; Umoroglu et al., 2023). Vreča et al. (2021) also developed a BNN for the UNSW-NB15 dataset.

We observe that BNNs are the fastest hardware architectures when considering latency or throughput, and have the smallest LUT footprints. These neural networks are very small and feature a reduced complexity. More complex architectures tend to introduce a significant cost in terms of hardware resources and speed. We compare some state-of-the-art models from past work in Table 1.

3 LOGICNETS FRAMEWORK

LogicNets is an approach developed by Umuroglu et al. (2020) that specializes in crafting and deploying sparse, quantized neural networks using hardware building blocks, delivering impressive levels of speed and efficiency on FPGAs. At its core, LogicNets is based on the concept of equating artificial neurons to truth tables with quantized inputs and outputs. Take, for example, an artificial neuron with C_{in} inputs, each spanning β-bits, and producing a single β-bit output. Regardless of the neuron's internal intricacies, its function can always be represented by an X-input, Y-output truth table, achieved by exhaustively enumerating all possible 2^X inputs and recording their respective outputs.

In the LogicNets framework, the Verilog implementation of these $X : Y$ truth tables are referred to as Hardware Building Blocks (HBBs), while trained artificial neurons that can be converted into HBBs are termed Neuron Equivalents (NEQs). The beauty of NEQs lies in their flexibility, allowing the addition

of components to simplify the DNN training process. NEQs and HBBs are the cornerstones of LogicNets in PyTorch and Verilog, respectively.

The design flow begins with identifying $X : Y$ values that yield HBBs with reasonable LUT cost and defining corresponding NEQs in PyTorch that comply with sparsity and activation quantization constraints. These NEQs can map to specific FPGA configurations or generic X:Y truth tables for synthesis. Using these NEQs, a deep neural network topology is constructed, followed by training in PyTorch, employing standard DNN optimization techniques. Post-training, the network transforms into a Verilog netlist of HBB instances and their sparse connections, allowing for further optimization. Ultimately, this approach focuses on single-FPGA implementations for high-throughput applications and concludes the process with the generation of an FPGA bit file through synthesis and place-and-route algorithms.

4 DATASET

The dataset used for training is UNSW-NB15, an updated and enhanced version of the former KDD Cup dataset. The KDD Cup dataset, which has become obsolete, had numerous anomalies.

The UNSW-NB15 is relatively smaller in size compared to other datasets, it stands out due to reduced redundancy, providing sufficient data for training a reasonably accurate model. This dataset comprises ten distinct target classes, including one denoting normal activity or benign behaviour, and nine representing various forms of attacks.

The dataset consists of a total of 45 features, each of significant importance in accurately classifying the aforementioned targets. Out of this dataset, 175,341 samples are used for training and 82,332 for testing. The data distribution of the training and testing dataset are displayed in Table 2.

We utilized the binarized version of UNSW-NB15. In this version, all original features, including data types like strings, categorical values, and floating-point values have been systematically transformed into a binary bit string consisting of 593 bits. Each value within these features is discretized into either '0' or '1' and stored as a uint8 value. These uint8 values are conveniently represented as numpy arrays and are distributed separately for both the training and test datasets, maintaining the same partitioning as the original dataset. The conclusive binary value within each sample serves as an indicative representation of the expected output. This binarized dataset was used by both Murovič and Trost (2019) and Umuroglu et al. (2020), and can be found online (Umuroglu, 2021). Note the class imbalance between the normal data packets and total attack packets in the training and test distributions.

Table 2: Data Set Record Distribution By Yang et al. (2019).

Category	**Training Dataset**	**Testing Dataset**
Normal	56,000	37,000
Generic	40,000	18,871
Exploits	33,393	11,132
Fuzzers	18,184	6062
DoS	12,264	4089
Reconnaissance	10,491	3496
Analysis	2000	677
Backdoor	1746	583
Shellcode	1133	378
Worms	130	44
Total	175,341	82,332

Our primary goal was to develop an accurate model for binary classification on this dataset. To address the class imbalance issue and avoid losing the variety of attacks in the dataset, we opted for a loss-function-based approach.

5 PROPOSED SOLUTION

5.1 Loss Function

To address the pronounced data imbalance within our dataset, we must first examine the high contrast between the quantities of normal and attack samples presented in Table 2. Specifically, we observe a total of $56,000$ normal samples in contrast to a significantly higher count of $119,341$ attack samples. This notable disparity underscores the presence of a class imbalance, which warrants our consideration regarding its impact on our model's performance.

It is imperative to recognize that such a substantial class imbalance could impart a substantial bias to our model's predictions. This inherent skew in the data distribution could result in a propensity for the model to favour predictions in favour of the attack class, due to the disproportionately higher number of training samples allocated to this category, as compared to the normal class. Consequently, this class imbalance may lead the model to produce erroneous predictions, particularly in the form of false positives within the attack class.

To mitigate this issue, our work explores the utilization of modified loss functions as a strategic approach. The focus of our study became the work of Dong et al. Dong et al. (2021) and Ehmer et al. Ehmer

et al. (2022). The method suggested by Ehmer et al., shown in Equation 1 is as follows:

$$\text{Loss} = J_{CE} - \frac{1}{N}\left[\sum_{i=1}^{N}\left(\alpha \cdot I(y^{(i)},1)\log(p_1^{(i)}) + \sum_{j=2}^{c} s_j \cdot I(y^{(i)},j)\log(1-p_1^{(i)})\right)\right] \tag{1}$$

Where:

$$I(a,b) = \begin{cases} 1 & \text{if a = b} \\ 0 & \text{Otherwise} \end{cases}$$

$$s_j = \beta \cdot \left(1 - \frac{n_j}{N_{mc}}\right)$$

And:

- J_{CE} is the binary cross-entropy loss,
- N is the number of samples in the batch,
- $p_1^{(i)}$ is the predicted probability of the majority class for the i-th sample,
- $y^{(i)}$ is the prediction for the i-th sample,
- α is a scaling factor,
- β is a scaling factor,
- n_j is the number of samples of the j-th minority class in the batch,
- N_{mc} is the total number of minority samples in the batch

This method does not adapt to binary classification, because the term for the scaling factor for the minority class simplifies to 0.

The method suggested by Dong et al. is as follows:

$$\text{Loss} = J_{CE} - \frac{1}{N}\left[\sum_{i=1}^{N}\lambda\left(I(y^{(i)},1)\log(p_1^{(i)}) + \sum_{j=2}^{c} I(y^{(i)},j)\log(1-p_1^{(i)})\right)\right] \tag{2}$$

Where λ is a scaling factor. This method failed to yield acceptable results using our small quantized neural network architecture.
Drawing inspiration from the works of Dong et al. and Ehmer et al., we created a loss function which worked well for our small architecture. The goal in mind was to penalize unconfident predictions for the minority class. The loss function we created is detailed in Equation 3. In line with the specifics of the UNSW-NB15 dataset discussed in Section 4, 0 is the label for the minority class and 1 is the label for the majority class. During our experiments, we set $\lambda = 0.5$.

$$\text{Loss} = J_{CE} - \frac{1}{N}\left[\sum_{i=1}^{N}\left(\lambda \cdot I(y^{(i)},1)\log(p_1^{(i)}) + I(y^{(i)},0)\log(1-\log(1-p_1^{(i)}))\right)\right] \tag{3}$$

Where:

$$I(a,b) = \begin{cases} 1 & \text{if a = b} \\ 0 & \text{Otherwise} \end{cases}$$

- J_{CE} is the binary cross-entropy loss, computed using nn.BCEWithLogitsLoss(),
- N is the number of samples in the dataset,
- $p_1^{(i)}$ is the predicted probability of the majority class for the i-th sample,
- $y_{(i)}$ is the binary prediction for the i-th sample (1 for attack, 0 for benign)

5.2 Our Network Topology

The artificial neural network employs a structured three-layer configuration, with a single hidden layer. Keeping in line with the strategy maintained by Umoroglu et al. Umuroglu et al. (2020), we exclude the softmax layer at the output to tailor each layer to specific computational needs. The foundational layer consists of 49 neurons, each with 7 input channels, akin to synapses in biological neural networks. These neurons output 2-bit data for nuanced responses. In the second layer, featuring 7 neurons, the input bit width expands to 7 for more complex processing, while maintaining a 2-bit output. The top layer, a single neuron, aggregates data from 7 inputs and also provides a 2-bit output.

Figure 1 illustrates the overview of the artificial neural network architecture used and also an exploded view of the perceptron.

5.3 Training

We trained the model for 100 epochs with a learning rate 10^{-1} and batch size of 1023. The training was performed on Intel(R) Core(TM) i7-10750H with 16 GB of RAM in Ubuntu 20.04. Results are discussed in Section 6.

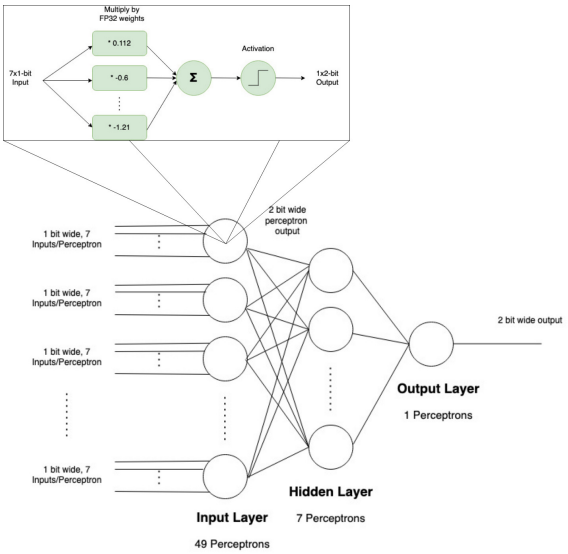

Figure 1: Network Topology.

5.4 Synthesis

Out-of-context synthesis was performed for an AMD Xilinx Alveo U280 (part number xcu280-fsvh2892-2L-e) on an Intel(R) Core(TM) i7-10750H with 16 GB of RAM in Ubuntu 20.04 using Vivado 2019.2. The results of the synthesis are discussed in Section 6.

6 RESULTS

We have rigorously evaluated our system to assess its suitability for network intrusion detection, focusing on both computational resource usage and classification accuracy.

6.1 System Performance Metrics

To begin, we examine the system's performance in terms of its maximum operating frequency and resource utilization. These metrics serve as essential indicators of the system's efficiency and practicality for real-world deployment. Our system achieved an impressive maximum operating frequency of 1.028 GHz, demonstrating its ability to process data rapidly and efficiently. Additionally, the resource utilization metrics reveal that the design is resource-efficient, with a consumption of 135 Look-Up Tables (LUTs) and 148 Flip-Flops. A comparison of our synthesis results with past works is displayed graphically in Figure 2.

6.2 Classification Performance Evaluation

Our evaluation extends to the classification performance of the system, a critical aspect for applications such as pattern recognition and anomaly detection. We employed a binary confusion matrix to rigorously analyze the system's ability to correctly classify instances.

As shown in Table 3, our proposed neural network provides competitive accuracy with a near negligible LUT footprint at a maximum operating frequency which crosses 1 GHz.

From the confusion matrix in Figure 3, we derived the following essential performance metrics:

- True Positive (TP): The number of correct positive classifications made by the system
- False Positive (FP): The number of incorrect positive classifications made by the system
- False Negative (FN): The number of incorrect negative classifications made by the system
- True Negative (TN): The number of correct negative classifications made by the system

6.3 Accuracy and F1 Score

With these performance metrics in hand, we assess the overall effectiveness of our classification system. Our results indicate an impressive accuracy of 90.91%, signifying the system's ability to make accurate predictions. Additionally, we achieved an F1 score of 91.82%, highlighting the system's capability

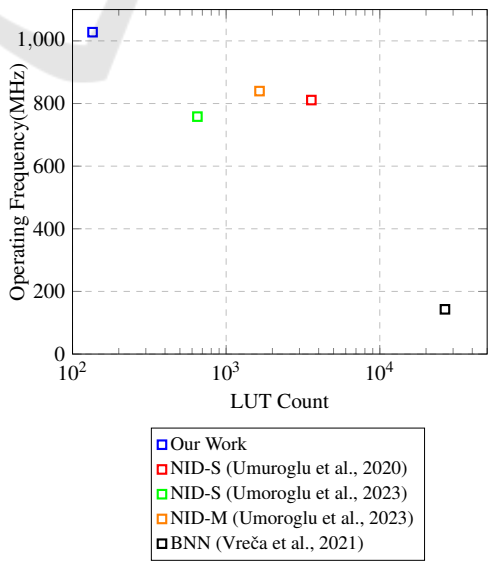

Figure 2: LUT Count vs. Operating Frequency.

Table 3: Comparison of Our Work With Select Previous Works.

Model	Acc. (%)	Latency	LUT	f (MHz)
MPBNN (Murovič and Trost, 2021)	92.04	19 ns	26879	-
NID-S (Umuroglu et al., 2020)	83.88	3.70 ns	3586	811
NID-S (Umoroglu et al., 2023)	90.5	3.96 ns	650	758.15
NID-M (Umoroglu et al., 2023)	92.6	3.57 ns	1649	839.63
Our Work	90.91	2.92 ns	135	1027.75

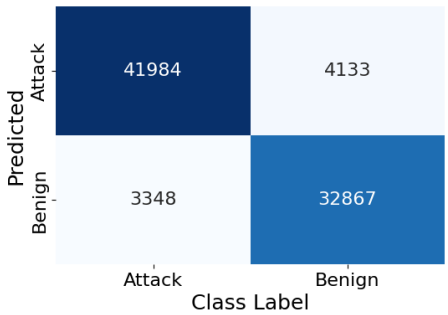

Figure 3: Confusion Matrix.

to balance precision and recall effectively. These outcomes underscore the system's robustness and its potential to excel in a wide range of classification tasks, making it a valuable asset for applications that require reliable decision-making.

To evaluate the classification performance, we calculate the accuracy and F1 score using the following formulas:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{4}$$

$$F1 = \frac{2 \cdot TP}{2 \cdot TP+FP+FN} \tag{5}$$

So, using the values enumerated in Fig.3, the calculated values are as follows:

- Accuracy: 0.909137 (or 90.91%)
- F1 Score: 0.918195 (or 91.82%)

7 CONCLUSION AND FUTURE WORK

We firmly advocate the need for further efforts in devising innovative architectures and exploring logic optimization techniques, particularly for parallel binary neural networks, which are essential in the high-speed domain of network intrusion detection. Additionally, we stress the importance of studying the interaction between algorithmic changes and the LogicNets framework, as it can significantly impact model performance and logic reduction. This comprehensive analysis will contribute to the continued advancement of hardware-based machine learning in the field of network intrusion detection.

ACKNOWLEDGEMENTS

We are thankful for the support of Dr. Mazhar Ali, at the University of Central Florida, and the guidance of Dr. Hammond Pearce, at the University of New South Wales.

REFERENCES

Andresini, G., Appice, A., De Rose, L., and Malerba, D. (2021). Gan augmentation to deal with imbalance in imaging-based intrusion detection. *Future Generation Computer Systems*, 123:108–127.

Ardagna, C., Corbiaux, S., Impe, K. V., and Sfakianakis, A. (2022). ENISA Threat Landscape 2022. Technical report, European Union Agency for Cybersecurity ENISA.

Azizjon, M., Jumabek, A., and Kim, W. (2020). 1d cnn based network intrusion detection with normalization on imbalanced data. In *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE.

Blott, M., Preußer, T. B., Fraser, N. J., Gambardella, G., O'brien, K., Umuroglu, Y., Leeser, M., and Vissers, K. (2018). Finn- r: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Transactions on Reconfigurable Technology and Systems*, 11(3):1–23.

Buczak, A. L. and Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys I& Tutorials*, 18(2):1153–1176.

Dong, B., Hui, W., Varde, A. S., Li, D., Samanthula, B. K., Sun, W., and Zhao, L. (2021). Cyber intrusion detection by using deep neural networks with attack-sharing loss.

Duarte, J., Han, S., Harris, P., Jindariani, S., Kreinar, E., Kreis, B., Ngadiuba, J., Pierini, M., Rivera, R., Tran, N., and Wu, Z. (2018). Fast inference of deep neural networks in fpgas for particle physics. *Journal of Instrumentation*, 13(07):P07027–P07027.

Ehmer, J., Granado, B., Denoulet, J., Savaria, Y., and David, J.-P. (2022). Low complexity shallow neural network with improved false negative rate for cyber intrusion detection systems. In *2022 20th IEEE Interregional NEWCAS Conference (NEWCAS)*. IEEE.

Jeune, L. L., Goedeme, T., and Mentens, N. (2022). Feature dimensionality in cnn acceleration for high-throughput network intrusion detection. In *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE.

Liu, L., Wang, P., Lin, J., and Liu, L. (2021). Intrusion detection of imbalanced network traffic based on machine learning and deep learning. *IEEE Access*, 9:7550–7563.

Murovič, T. and Trost, A. (2019). Massively parallel combinational binary neural networks for edge processing. *Elektrotehniski Vestnik*, 86(1/2):47–53.

Murovič, T. and Trost, A. (2020). Resource-optimized combinational binary neural network circuits. *Microelectron. J.*, 97(C).

Murovič, T. and Trost, A. (2021). Genetically optimized massively parallel binary neural networks for intrusion detection systems. *Computer Communications*, 179:1–10.

Ngo, D.-M., Temko, A., Murphy, C. C., and Popovici, E. (2021). Fpga hardware acceleration framework for anomaly-based intrusion detection system in iot. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE.

Ngo, D.-M., Tran-Thanh, B., Dang, T., Tran, T., Thinh, T. N., and Pham-Quoc, C. (2019). *High-Throughput Machine Learning Approaches for Network Attacks Detection on FPGA*, page 47–60. Springer International Publishing.

Tahir, M. A., Kittler, J., and Yan, F. (2012). Inverse random under sampling for class imbalance problem and its application to multi-label classification. *Pattern Recognition*, 45(10):3738–3750.

Umoroglu, Y., Fraser, N. J., and Akhauri, Y. (2023). Xilinx/Logicnets. [online] Available: https://github.com/Xilinx/logicnets.

Umuroglu, Y. (2021). The unsw-nb15 dataset with binarized features.

Umuroglu, Y., Akhauri, Y., Fraser, N. J., and Blott, M. (2020). Logicnets: Co-designed neural networks and circuits for extreme-throughput applications. In *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE.

Umuroglu, Y., Fraser, N. J., Gambardella, G., Blott, M., Leong, P., Jahre, M., and Vissers, K. (2017). Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '17. ACM.

Vreča, J., Ivanov, I., Papa, G., and Biasizzo, A. (2021). Detecting network intrusion using binarized neural networks. In *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*. IEEE.

Wang, H. and Huang, H. (2018). Lad-smote: A new oversampling method based on locally adaptive distance. In *2018 Ninth International Conference on Intelligent Control and Information Processing (ICICIP)*. IEEE.

Wang, W., Zhu, M., Zeng, X., Ye, X., and Sheng, Y. (2017). Malware traffic classification using convolutional neural network for representation learning. In *2017 International Conference on Information Networking (ICOIN)*. IEEE.

Xilinx (2023). Vitis AI. [online] Available: https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html.

Yang, K., Kpotufe, S., and Feamster, N. (2021). An efficient one-class svm for anomaly detection in the internet of things. *ArXiv*, abs/2104.11146.

Yang, Y., Zheng, K., Wu, C., and Yang, Y. (2019). Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. *Sensors*, 19(11):2528.

Yin, C., Zhu, Y., Fei, J., and He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5:21954–21961.

Zheng, Z., Cai, Y., and Li, Y. (2015). Oversampling method for imbalanced classification. *Comput. Informatics*, 34:1017–1037.