An FPGA-Based Weightless Neural Network Platform for Edge Intrusion Detection

ABSTRACT

The proliferation of internet-enabled edge devices in the last decade and the introduction of 5G mobile networks have expanded the demand for network functions deployed in resource-constrained computational environments. Traditionally, this functionality was deployed using dedicated ASICs. However, ASIC-based approaches have low flexibility: they offer limited support to adopt updated routing algorithms, fix security vulnerabilities, or adapt to a specific user or application. With the emergence of 6G, we expect this issue to be exacerbated since best practices for new features such as traffic prioritization are still unknown. Therefore, the embedded networking solutions of the future will likely include some combination of ASICs and FPGAs.

In this paper, we propose a novel FPGA-based solution for detecting anomalous or malicious network traffic on edge devices. While prior work in this domain is based on conventional deep neural networks (DNNs), we incorporate a weightless neural network (WNN), a table lookup-based model which learns sophisticated nonlinear behaviors. This allows us to achieve accuracy far superior to prior work at a very small fraction of the model footprint, enabling deployment on even the smallest FPGAs. We achieve a prediction accuracy of 98.9% on the UNSW-NB15 dataset with a total model parameter size of just 192 bytes, reducing error by 7.9x and model size by 262x vs. the prior work. Implemented on an FPGA, we demonstrate a 59x reduction in LUT usage with a 1.6x increase in throughput. Our model accuracy comes within 0.6% of the best-reported result in literature, a model many orders of magnitude larger, while actually achieving a superior false negative rate. Our results make it clear that WNNs are worth exploring in the emerging domain of edge routing solutions, and show that FPGAs are capable of providing the extreme throughput needed.

ACM Reference Format:

. 2022. An FPGA-Based Weightless Neural Network Platform for Edge Intrusion Detection. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

It is estimated that there are 14.6 billion Internet-of-Things (IoT) devices in the world today, and this number is expected to more than double by 2027 [17]. Although this proliferation has the potential to enrich the lives of consumers, it also has concerning implications due to IoT devices' notoriously poor security. For instance, in 2016, the Mirai malware was able to recruit hundreds of thousands of devices into vast botnets, launching distributed denial-of-service attacks with up to 1.1 Tbps of traffic [10]. The predicted sharp increase in the number of network-enabled edge devices threatens to significantly increase the attack surface.

Meanwhile, IoT devices have increasingly adopted fifth-generation (5G) mobile networking technologies such as NB-IoT and Cat-M [17], and sixth-generation solutions are in development. The introduction of Multi-access Edge Computing (MEC) [6] environments will allow us to deploy advanced network functions (NFs) such as IDS, IPS, and traffic prioritization in the vicinity of the mobile user using resource-constrained computational systems [9, 16]. This contrasts with present-day approaches, which leverage dedicated high-performance appliances with ASIC acceleration. We need novel mechanisms to fully realize the benefit of moving these NFs closer to the mobile user.

FPGAs are a promising technology that we can leverage to address some of these concerns. The FPGAs-based approach differs from traditional ASIC solutions due to its increased flexibility, which allows the mobile operator to update routing algorithms, fix security vulnerabilities, or adapt to a specific user or application. However, FPGAs are inferior to ASICs in terms of area, cost, and energy efficiency, and have therefore seen limited adoption in consumer devices beyond a few specialized accelerator cards [12]. A purely FPGA-based approach to mobile networking is likely infeasible; we envision future embedded networking solutions will use some combination of ASICs and FPGAs, leveraging the respective efficiency and flexibility of the two technologies. Crucially, FPGA-based accelerators must have as small of an area as possible in order to minimize broader system impact, while still providing sufficient throughput for high-speed networking.

In this paper, we propose and demonstrate a novel FPGA-based solution for detecting anomalous or malicious network traffic on edge devices. Unlike prior work in this domain, which was based on deep neural networks (DNNs), we take a fundamentally different approach based on weightless neural networks (WNNs). WNNs are composed of RAM nodes, neural units which perform computation using lookup tables (LUTs). Unlike the neurons of a DNN, RAM nodes can learn nonlinear functions of their inputs, allowing WNNs to learn sophisticated behaviors with very small model sizes. This enables the deployment of our model on even the smallest of FPGAs, allowing it to fit in resource-constrained shared infrastructure as the gNB or the future 6G Node-B [7]. We demonstrate binary classification on the UNSW-NB15 dataset with 98.5% accuracy with a model parameter size of just 192 bytes, coming within 1% of the accuracy of the most accurate results in the literature (EdgeDetect [15]) with more than 3000x reduction in parameter size. Compared to a recent FPGA-based model for this dataset proposed in LogicNets [18], our weightless model improves test accuracy from 91.3% to 98.9% on a balanced variant of the dataset, reduces

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

FPGA LUT usage by more than 98%, and demonstrates the potential to achieve superior throughput and energy efficiency.

The remainder of this paper is laid out as follows: In Section 2, we provide some additional background on weightless neural networks, the UNSW-NB15 dataset, and the prior work. In Section 3 we provide an overview of our experimental methodology and FPGA implementation. In Section 4, we compare our model against prior work and demonstrate its viability in a resource-constrained environment. Lastly, in Section 5, we discuss future work and conclude.

Contributions

2 BACKGROUND

2.1 Weightless Neural Networks

The distinguishing feature of WNNs is that they primarily use table lookups to perform computation, as opposed to the multiply-accumulate operations typical of DNNs or the XNOR-and-popcount operations used in binary NNs. The RAM nodes of a WNN are traditionally n-input, 2^n-entry lookup tables with binary inputs and outputs, and are therefore capable of representing any of the 2^{2^n} possible Boolean functions of their inputs. One challenge imposed by this structure is that, as the size of a RAM node grows exponentially with its number of inputs, it is infeasible to provide all inputs to each node in non-trivial models. While the WNN we use in this paper is in the family of models descended from the WiSARD classifier, there are other approaches to WNNs as well, though they frequently have complications such as randomized behavior which make them difficult to implement in practice; refer to [11] for a more comprehensive overview.

WiSARD (Wilkie, Stoneham and Aleksander's Recognition Device) [1] is an early WNN from which many subsequent models are derived, in much the same way as the Perceptron laid the groundwork for modern DNNs. As shown in Figure 1, WiSARD is composed of submodels known as *discriminators*, which are each specialized to learn a single output class. Inputs are partitioned between the RAM nodes of a discriminator using a mapping function, which is typically shared between discriminators. *(1a)* During inference, RAM nodes emit the table entries corresponding to their input patterns, and a popcount is performed for the outputs of the nodes in each discriminator to determine a response score. *(1b)* The class corresponding to the discriminator with the strongest response is taken to be the network's prediction.

A crucial hyperparameter in the specification of a WiSARD model is the number of inputs to each RAM node, n. Small values of n prevent the model from learning complex input behaviors, which usually results in unsatisfactory performance. On the other hand, an excessively large value of n may cause overfitting, as the model memorizes spurious patterns associated with particular inputs. Model size also grows exponentially with n, posing an additional constraint.

Many improvements to the baseline WiSARD architecture have been proposed, of which two are relevant to our discussion. In Bloom WiSARD [5], it was observed that the RAM nodes of a WiSARD model with a large choice of n are highly sparse, and can therefore be replaced with hash-based data structures such as Bloom filters with a minimal loss of accuracy.

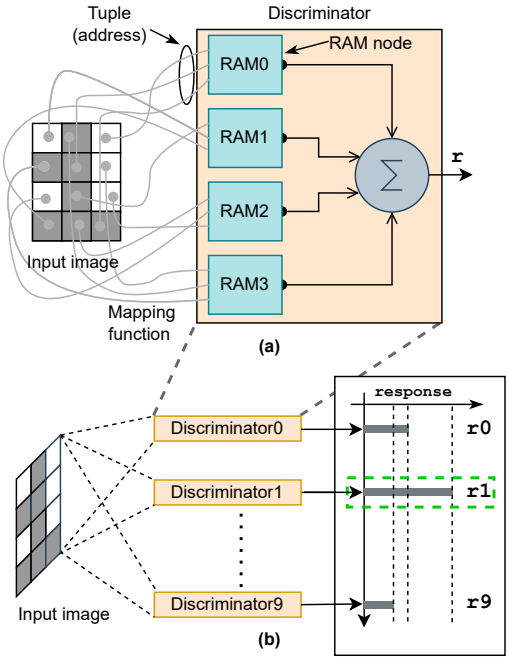

Figure 1: WiSARD, an early WNN for classification tasks. WiSARD trains discriminators for each output class, which are in turn composed of RAM nodes.

Inputs to WiSARD are typically expressed using a single bit per feature by comparing features against their mean values in the training data. Thermometer encoding is a form of multi-bit unary encoding where features are instead compared against a set of increasing thresholds. Thermometer encodings have been demonstrated to significantly increase the accuracy of WNNs, at a cost to model size [8].

2.2 UNSW-NB15

The UNSW-NB15 dataset [13] is a network intrusion dataset which can be used for binary or multi-class classification tasks. Each sample consists of 47 features representing an aspect of network activity, as well as either a label of normal network activity or an attack category. Most works using this dataset, including this paper, focus on the binary classification problem, where the goal is to identify whether or not a sample represents an attack, rather than which category an attack belongs to.

While USNW-NB15 is not specifically targeted for IoT threats, other common network intrusion datasets, such as KDD98, KDDCUP99, and NSLKDD, are based on data that is now more than 20 years old, and are therefore no longer necessarily representative of the modern threat landscape.

2.3 Prior Work

3 METHODOLOGY

As discussed previously, our objective is to deploy a WNN-based model for the UNSW-NB15 dataset on an FPGA-based inference accelerator. In this section, we will discuss how we prepared the

dataset, created and trained the model in software, and implemented and evaluated the hardware model.

3.1 Dataset Preparation

Before we use the UNSW-NB15 dataset to train a model, we perform some basic preprocessing steps. Samples in the dataset contains several non-numeric features, including source IP addresses and transaction protocols. Prior work [14, 18] experimented with one-hot encoding these features, but we found that this provided no benefit to accuracy, while increasing model size. Therefore, we remove these features entirely.

A small number (~0.01%) of samples are malformed in some way and are culled. A further 19.4% of samples are duplicates, and are removed to avoid data leakage (the appearance of the same samples in the training and test data). After this step, we are left with 205k samples, of which 4.3% belong to the "attack" category. We perform a 9:1 train/test split on this data.

Training with severely imbalanced datasets is known to be a difficult problem [19]. To rectify the ~23:1 imbalance in the training data, we randomly oversample the minority ("attack") class by selecting entries multiple times until the dataset is at a 1:1 ratio.

3.2 Software Model

In Section 2.1, we introduced the WiSARD WNN architecture and discussed some subsequent improvements to this baseline. Bloom filters and thermometer encoding are complementary techniques, and we incorporate both into our model. To determine the thresholds for the thermometer encoding, we determine the means and standard deviations of the features in the training data and pick thresholds which divide the Gaussian into equally probable intervals. We ensure that thresholds are at least as large as the second-smallest vale of the feature, and not larger than the largest value; this prevents them from going out of range for e.g. binary-valued features.

One significant deviation of our model from prior work lies in our training approach. WiSARD and derived models are typically trained with a one-shot approach, where RAM nodes become sensitive to patterns seen during training (either once or some threshold number of times [3]). While this technique is computationally efficient, it is unable to incorporate feedback. We instead use a backpropagation-based technique inspired by the process used for training binary neural networks [4]. At training time, RAM node (Bloom filter) entries are treated as floating point numbers between -1.0 and 1.0. During the forward pass, these values are binarized using the unit step function. However, during backpropagation, the unit step is ignored and gradients pass by it unchanged, a process known as the *straight-through estimator* [21]. After training is complete, floating-point entries can be permanently binarized, so there is no overhead to inference.

Once the model is trained, we perform pruning by eliminating the RAM nodes which contribute least to overall model accuracy, effectively replacing them with a constant 0 or 1.

We implemented this model using custom extensions to the PyTorch machine learning library, which allowed us to leverage GPU acceleration for training.

3.3 FPGA Implementation

Our accelerator model is written using the Mako [2] template library to generate SystemVerilog. This allows us to automatically convert a pretrained model into an RTL implementation with a user-specified bus width.

We consider two contexts for the deployment of our solution: as part of a larger FPGA-based solution, or independently on a very small FPGA. For the former context, we use the Xilinx xcvu9p-flgb2104-2-i FPGA, which was also used for LogicNets. We perform synthesis in out-of-context mode with a 160b input width, which is equal to the encoded size of a single sample. For the small FPGA, we use the Xilinx xc7s6csga225-2. Synthesis for this board is *not* performed in out-of-context mode. We consider both an 80b input width, which allows an input to be read in two cycles, and a 64b input width, which is a more typical interface width for system-on-chip solutions. All synthesis runs are performed using Vivado with the `Flow_PerfOptimized_high` strategy, which instructs synthesis to prioritize timing over power and area.

Power and area numbers for our design were derived from post-implementation Vivado reports. We assume the default (15%) toggle rate. While LogicNets provides area, clock, frequency, and throughput values, it does not discuss power or energy. Therefore, we approximated these values using Xilinx Power Estimator (XPE) [20], a spreadsheet-based analytical model. Edge-Detect does not propose an FPGA or ASIC implementation or provide any performance figures, so we use it for model comparison only.

4 RESULTS

4.1 Model Selection

We identified Bloom filters with 10 inputs and 64 entries as striking a good balance between model size and accuracy when used as RAM nodes. The number of hash functions k associated with a Bloom filter impacts its false positive rate and therefore classification accuracy. Associating more hash functions also increases the number of hash computations which need to be performed at runtime. For this particular model, we did not see any benefit to using more than $k = 1$ hash functions per Bloom filter, but this is not true of Bloom filter-based WNNs in general. Input features were encoded using a 4-bit unary thermometer encoding.

We found that we could eliminate 30% of RAM nodes via pruning without any significant decrease in model accuracy. This process reduced the size of the model from 272 to 192 bytes. After pruning, two of the 42 numeric features in UNSW-NB15 were entirely unused by the model (`sloss` and `ct_ftp_cmd`). Eliminating these features reduced the encoded size of an input to 160b.

4.2 Model Evaluation

Our weightless model achieves an accuracy of 98.5% on the unbalanced UNSW-NB15 test set, or 98.9% on the 2:1 normal to "attack" split used in LogicNets. Comparative results, including false positive and negative rates where available, are shown in Table 1. The WNN is substantially more accurate than LogicNets, the prior FPGA-based model, reducing test error by a factor of 7.9x. While Edge-Detect's state-of-the-art accuracy is significantly better than

our own, as Figure 2 shows, this comes at an increase in parameter size of more than four orders of magnitude.

Table 1: Accuracies for LogicNets, Edge-Detect, and our weightless model on the UNSWNB-15 dataset with a 2:1 normal to "attack" split. FPR and FNR for Edge-Detect are calculated from published accuracy, precision, and recall values.

Model Name	Accuracy	False Pos. Rate	False Neg. Rate
LogicNets [18]	91.3%	N/D	N/D
Edge-Detect [15]	*99.5%*	*0.57%*	0.25%
WNN	**98.9%**	**1.57%**	***0.10%***

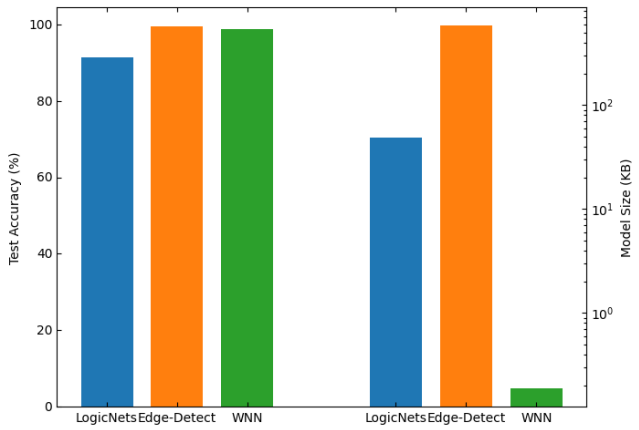

Figure 2: Comparison of test accuracies and model parameter sizes for LogicNets, Edge-Detect, and our weightless model.

Figure 3 shows the receiver operating characteristic (ROC) curve of the weightless model. The ROC is a common figure of merit for binary classifiers, obtained by biasing the classification and recording the resulting true and false positive rates. We accomplish this by adding a constant value to the output of the discriminator corresponding to the "attack" class. The area under the ROC is 0.994, compared to 0.5 for a random classifier or 1.0 for a hypothetical perfect classifier.

From the ROC, we see that by biasing the prediction towards the "attack" class, we can effectively eliminate false negatives while only increasing the false positive rate to ~2%. However, we would suggest caution in interpreting this result too broadly - the UNSW-NB15 dataset generates these samples using synthetic data, and certainly can not be expected to cover all possible cases. The very low false negative rate of our design - 0.10% - is attractive in an edge context where positive detections can be sent to a larger, less efficient model or a central server for verification.

4.3 FPGA Implementation Results

Results for the FPGA implementations of the WNN model, along with results for LogicNets, are show in table 2. Compared to LogicNets on the larger (xcvu9p-flgb2104-2-i) FPGA, our model improves throughput by 1.6x, reduces dynamic energy per inference by 9.0x,

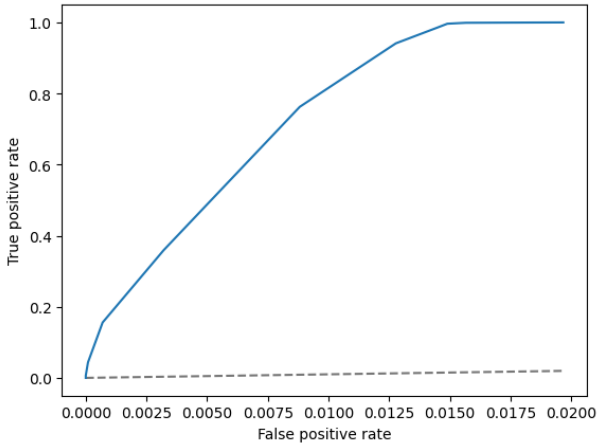

Figure 3: Receiver operating characteristic curve for the WNN model; dashed line is ROC curve for a random classifier

and decreases LUT and FF usage by 59x and 2.4x respectively. While the energy value for LogicNets (which was generated using XPE) is less accurate than the values for our models, it still suffices to show a clear advantage for the WNNs.

The dramatic decrease in area can be attributed to two factors. First, LogicNets uses a much larger encoding of the UNSW-NB15 dataset, require 593 bits per input versus the 160-bit encoding used in our model. Additionally, as mentioned previously, unlike DNNs, the individual RAM nodes within a WNN can learn non-linear functions of their inputs. This provides a significant efficiency advantage to WNNs, as DNNs need multiple layers to capture the same behavior.

The implementations of the weightless model on the smaller FPGA require multiple cycles to read in a single sample due to bus width limitations. The deserialization logic required for these implementations also introduces significant LUT and energy overheads. The smaller FPGA is also built on a much larger process node (28nm vs. 14/16nm), introducing additional energy overhead.

Overall, our weightless model introduces very little area overhead as part of a larger FPGA-based design, while providing a throughput of up to 740 million samples per second, sufficient for even demanding networking applications. As a stand-alone model, it can easily fit on a small, low-cost commercial FPGA while still providing considerable throughput, and can easily be updated or replaced with a new model as the threat landscape changes.

5 CONCLUSION

Code

REFERENCES

[1] I. Aleksander, W.V. Thomas, and P.A. Bowden. 1984. WISARD·a radical step forward in image recognition. *Sensor Review* 4, 3 (1984), 120–124. https://doi.org/10.1108/eb007637

[2] Michael Bayer. 2021. Mako Templates for Python. https://www.makotemplates.org/

[3] Danilo Carvalho, Hugo Carneiro, Felipe França, and Priscila Lima. 2013. B-bleaching : Agile Overtraining Avoidance in the WiSARD Weightless Neural Classifier. In *ESANN*.

Table 2: FPGA implementation details of the prior work, LogicNets, and the WNN discussed in this paper. Energy values for LogicNets are estimated using XPE.

Model Name	FPGA Part	Clock Freq. (MHz)	Bus Width	Initiation Interval	Throughput (MS/s)	Dynamic Energy (pJ/Sample)	LUTs	FFs
LogicNets	xcvu9p-flgb2104-2-i (Large; 14/16nm)	471	593b	1	471	654	15,949	1,274
WNN	xcvu9p-flgb2104-2-i (Large; 14/16nm)	740	160b	1	740	73	269	538
WNN	xc7s6csga225-2 (Small; 28nm)	300	64b	3	100	340	354	379
WNN	xc7s6csga225-2 (Small; 28nm)	300	80b	2	150	266	299	418

[4] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. arXiv:1602.02830 [cs.LG]

[5] Leandro Santiago de Araújo, Letícia Dias Verona, Fábio Medeiros Rangel, Fabrício Firmino de Faria, Daniel Sadoc Menasché, Wouter Caarls, Maurício Breternitz, Sandip Kundu, Priscila Machado Vieira Lima, and Felipe Maia Galvão França. 2019. Memory Efficient Weightless Neural Network using Bloom Filter. In *ESANN*.

[6] European Telecommunications Standards Institute (ETSI). 2019. *Multi-access Edge Computing (MEC); Study on MEC support for alternative virtualization technologies.* Technical Report. https://www.etsi.org/deliver/etsi_gr/MEC/001_099/027/02.01.01_60/gr_MEC027v020101p.pdf

[7] Mohammad Asif Habibi, Bin Han, Meysam Nasimi, Nandish P. Kuruvatti, Amina Fellan, and Hans D. Schotten. 2021. *Towards a Fully Virtualized, Cloudified, and Slicing-Aware RAN for 6G Mobile Networks.* Springer International Publishing, Cham, 327–358. https://doi.org/10.1007/978-3-030-72777-2_15

[8] Andressa Kappaun, Karine Camargo, Fabio Rangel, Fabrício Firmino, Priscila Machado Vieira Lima, and Jonice Oliveira. 2016. Evaluating Binary Encoding Techniques for WiSARD. In *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*. 103–108. https://doi.org/10.1109/BRACIS.2016.029

[9] Yacine Khettab, Miloud Bagaa, Diego Leonel Cadette Dutra, Tarik Taleb, and Nassima Toumi. 2018. Virtual security as a service for 5G verticals. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. 1–6. https://doi.org/10.1109/WCNC.2018.8377298

[10] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. DDoS in the IoT: Mirai and other botnets. *Computer* 50 (01 2017), 80–84. https://doi.org/10.1109/MC.2017.201

[11] Teresa Ludermir, Andre de Carvalho, Antônio Braga, and M.C.P. Souto. 1999. Weightless neural models: A review of current and past works. *Neural Computing Surveys* 2 (01 1999), 41–61.

[12] Michael Mattioli. 2022. FPGAs in Client Compute Hardware: Despite Certain Challenges, FPGAs Provide Security and Performance Benefits over ASICs. *Queue* 19, 6 (dec 2022), 66–88. https://doi.org/10.1145/3512327

[13] Nour Moustafa and Jill Slay. 2015. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*. 1–6. https://doi.org/10.1109/MilCIS.2015.7348942

[14] Tadej Murovič and Andrej Trost. 2019. Massively parallel combinational binary neural networks for edge processing. *Elektrotehniski Vestnik/Electrotechnical Review* 86 (01 2019), 47–53.

[15] Praneet Singh, Jishnu Jaykumar P, Akhil Pankaj, and Reshmi Mitra. 2021. Edge-Detect: Edge-Centric Network Intrusion Detection using Deep Neural Network. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. 1–6. https://doi.org/10.1109/CCNC49032.2021.9369469

[16] Chiara Suraci, Giuseppe Araniti, Andrea Abrardo, Giuseppe Bianchi, and Antonio Iera. 2021. A stakeholder-oriented security analysis in virtualized 5G cellular networks. *Computer Networks* 184 (2021), 107604. https://doi.org/10.1016/j.comnet.2020.107604

[17] Telefonaktiebolaget LM Ericsson. 2022. IoT connections outlook. *Ericsson Mobility Report* (06 2022). https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook

[18] Yaman Umuroglu, Yash Akhauri, Nicholas J. Fraser, and Michaela Blott. 2020. LogicNets: Co-Designed Neural Networks and Circuits for Extreme-Throughput Applications. *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)* (2020), 291–297.

[19] Shoujin Wang, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng, and Paul Kennedy. 2016. Training deep neural networks on imbalanced data sets. *IEEE*, 4368–4374. https://doi.org/10.1109/IJCNN.2016.7727770

[20] Xilinx. 2021. Xilinx Power Estimator (XPE). https://www.xilinx.com/products/technology/power/xpe.html

[21] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley J. Osher, Yingyong Qi, and Jack Xin. 2019. Understanding Straight-Through Estimator in Training Activation Quantized Neural Nets. *CoRR* abs/1903.05662 (2019). arXiv:1903.05662 http://arxiv.org/abs/1903.05662