LogicNets vs. ULEEN : Comparing two novel high throughput edge ML inference techniques on FPGA

Shashank Nag[1§], Zachary Susskind[1§], Aman Arora[1], Alan T. L. Bacellar[2], Diego L. C. Dutra[2], Igor D. S. Miranda[3], Krishnan Kailas[6], Eugene B. John[7], Mauricio Breternitz Jr.[4], Priscila M. V. Lima[2], Felipe M. G. França[5,2], and Lizy K. John[1]

[1]Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA
Email: {shashanknag, zsusskind, aman.kbm}@utexas.edu, ljohn@ece.utexas.edu

[2]Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
Email: alanbacellar@poli.ufrj.br, ddutra@cos.ufrj.br, priscilamvl@cos.ufrj.br

[3]Federal University of Recôncavo da Bahia, Cruz das Almas, Brazil; Email: igordantas@ufrb.edu.br

[4]ISCTE - Instituto Universitario de Lisboa, Lisbon, Portugal; Email: Mauricio.Breternitz.Jr@iscte-iul.pt

[5]Instituto de Telecomunicações, Porto, Portugal; Email: felipe@ieee.org

[6]IBM T.J. Watson Research Center, USA; Email: kailas@us.ibm.com

[7]The University of Texas at San Antonio, San Antonio, TX, USA; Email: eugene.john@utsa.edu

***Abstract*—With the advent of Internet-of-Things (IoT) and edge computing devices, there has been an increased demand for low power and high-throughput machine learning inference on the edge. However, the trends of ever-increasing model sizes with numerous computations involved makes it increasingly difficult to deploy state-of-the-art models on edge computing devices. Of late, there has been a renewed interest in lookup table (LUT)-based ML models that replace typical weighted-addition operations in artificial neurons with lookup operations. These are well suited for edge FPGAs, both due to their underlying architecture, as well as their potential for low energy consumption. LogicNets and ULEEN are two such LUT-based model architectures, that have claimed to offer high throughput and low energy inferences. These two architectures are extensions of contrasting ideas of Deep Neural Networks and Weightless Neural Networks, and it is difficult to infer a suitable choice among these. In this paper, we compare these, and evaluate them on some high-throughput inference use cases. When evaluated on intrusion detection and physics-experiment classification tasks, our results suggest that ULEEN outperforms LogicNets on hardware and energy requirements making it well suited for edge deployment, albeit at a slight drop in accuracy for some datasets.**

***Index Terms*—Edge ML, High Throughput, Low Energy, Weightless Neural Networks, Deep Neural Networks, LogicNets, ULEEN, FPGA**

I. Introduction

Internet-of-Things (IoT) devices are widespread today, and will only become more ubiquitous in the foreseeable future [1]. A lot of these edge devices are required to perform tasks, such as segregation and fault detection, involving intelligent decisions by Machine Learning (ML) algorithms. Such applications require decisions to be made at a low-latency for their effectiveness, which are typically not achievable using cloud computing solutions. This has necessitated the need for high-throughput inferences of ML models at the edge [2].

Conventional ML models, including deep neural networks (DNNs), have large model sizes and involve a high inference latency. Moreover, they are compute intensive with many multiply-accumulate (MAC) operations, and aren't suitable to be deployed on hardware resource and energy constrained edge devices. As such, there is a growing need to redesign ML models and their hardware implementations specifically targeting such devices.

Hardware acceleration on edge devices is typically achieved using application-specific integrated circuits (ASICs). Though ASICs tend to be fast and energy-efficient with a small hardware area and low unit cost, they have a high non-recurring engineering cost. With changing algorithms and use cases in edge IoT applications, such as with intrusion detection algorithms, ASICs' limited flexibility once deployed proves to be a liability. Field-programmable gate arrays (FPGAs) are an alternative to ASICs for hardware acceleration. FPGAs are reconfigurable, which allows adaptability to varying user and application needs. Edge FPGAs like the Lattice iCE40 series provide a low-power low-cost alternative to ASICs, while offering the all the flexibility provided by FPGAs.

LogicNets [3] is one such recent work that addresses this issue, and seeks to codesign DNN topologies and their FPGA implementations for extreme-throughput applications. Based on the observation that modern FPGAs are largely composed of lookup tables (LUTs), LogicNets seeks to convert each neuron in a trained DNN into an equivalent LUT for an optimal hardware mapping. LogicNets proposes a design flow for such an implementation, and demonstrates high throughput performance with competitive accuracy for Network Intrusion Detection [4] and Jet Substructure Classification (JSC) [5]

This research was supported in part by Semiconductor Research Corporation (SRC) Task 3148.001, National Science Foundation (NSF) Grant #2326894, NVIDIA Applied Research Accelerator Program Grant, Fundação para a Ciência e a Tecnologia, I.P. (FCT) [ISTAR Projects: UIDB/04466/2020 and UIDP/04466/2020], and by FCT/MCTES through national funds and, when applicable, co-funded by EU funds under the projects UIDB 50008/2020 and Route 25 (C645463824-00000063). Any opinions, findings, conclusions, or recommendations are those of the authors and not of the funding agencies.

[§]Equal contributions

tasks, which demand such low-latency inferences. Its novel approach of targeting LUT implementations for FPGAs has garnered it recent interest.

ULEEN - Ultra Low Energy Edge Networks [6] is another such model and FPGA-based architecture proposed recently for extreme-edge and latency-critical inferences, inspired by the idea of Weightless Neural Networks (WNNs). ULEEN incorporates a novel multi-pass training strategy to update the entries in a single-layer of Look-Up-Tables or RAM nodes, to form a discriminator-based classification model that learns patterns in the input sequence. ULEEN reports lower latency, improved energy-efficiency, and smaller area-delay product against iso-accuracy FINN Binary Neural Network (BNN) [7] models.

Though both these works end up seeking LUT-based implementations on FPGAs, they draw inspirations from two very different ideas. While LogicNets extends the idea of pruning and quantizations of DNNs to make neurons representable by LUTs, ULEEN designs neural networks with LUTs at the heart of them. Though both demonstrate extreme throughput performances on edge devices, interestingly there has been no prior work that compares these two, to the best of our knowledge. With both works using very different design flow and training techniques, and evaluated on different datasets, there is not much insight into how these compare against each other. In this paper, we seek to address this gap in the literature, and evaluate both the works on common parameters. Our specific contributions in this paper are as follows:

- This paper is the first to systematically compare LogicNets and ULEEN, two of the recent works in the area of LUT-based neural networks.
- We evaluate ULEEN and compare it against LogicNets on two Network Intrusion Detection datasets, UNSW-NB15 and BoT-IoT, and the Jet Substructure Classification (JSC) and Higgs particle detection datasets. We propose a search strategy for ULEEN and provide insights into picking an optimal model configuration for newer datasets, which is otherwise absent in literature.
- We compare the performance of the FPGA implementations of ULEEN and LogicNets on hardware parameters. Notably, we perform an analysis of power consumption, which has not been reported for LogicNets in the original work [3].

The rest of the paper is organized as follows : in Section II, we provide a background of what the two ideas stemmed from, and more details of it. We discuss the methodology used for comparisons in Section III and present the findings from the experiments in Section IV. Section V concludes the paper.

II. Background and Related Work

A. Quantized and Pruned DNNs

Several pruning and quantization techniques for Deep Neural Networks have been explored in the past with the aim of co-designing efficient hardware implementations [8], [9]. Pruning of networks intends to sparsify the connections between neurons across layers, in contrast to a fully connected neural network. Since each neuron now receives fewer inputs, this reduces the computational costs, at the expense of minimal drop in accuracy. Quantization on the other hand restricts weights and/or activations in neural networks to lower precision quantised values, and is useful in reducing the hardware costs for computations during inference [10].

B. LogicNets

LogicNets [3] extends the idea of sparse and quantized neural networks to the extreme case, where the neurons can be mapped to small LUTs that form the building blocks of FPGAs. Through sparsification, each neuron's fan-in can be restricted to a few inputs from the previous layer, and through quantization, the bit-width of the output of a neuron is restricted. Consequently, the input and output bits from a neuron can be made so small, that they can be represented by an equivalent X-input:Y-output LUT - where X is the number of bits concatenated together from all fan-ins to the neuron, and Y is the number of output bits from the quantized neuron. The LUT encapsulates the output from the neuron for all the 2^X possible input combinations. *Fig 1* represents a LogicNets model during inference.

The design and implementation flow for LogicNets involves the following steps. First, a set of X:Y configurations of LUTs are identified that can be implemented with a reasonable amount of hardware units (LUTs) on the target FPGA (typically a 6:1 or 5:2 LUT), referred to as Hardware Building Blocks (HBBs). For these, corresponding Neuron Equivalents (NEQs) are defined, which have the specified number of bits for fan-in and output. Effectively, this imposes constraints on the sparsity and output quantization of the neurons. Subsequently, a Neural Network with these constrained NEQs as building blocks is defined in PyTorch and trained. The trained network is then converted into the netlist of HBBs, and synthesised on a target FPGA.

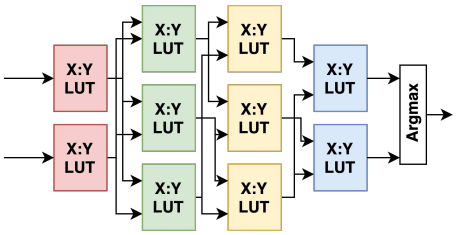

Fig. 1: LogicNets model inference. The input bits are processed through a network of LUTs, that are representative of neurons in a DNN. The final layer has LUTs corresponding to each of the classes in the dataset, and Argmax predicts the class.

C. Weightless Neural Networks (WNNs)

Unlike conventional Deep Neural Networks (DNNs), in which computation is dominated by multiply-accumulate operations, WNNs primarily use table lookups to perform computation. The individual neurons, or *RAM nodes*, of a WNN are conventionally look-up tables representing Boolean functions of binary inputs and outputs. WNNs typically contain many

small RAM nodes which are each only sensitive to a small subset of model inputs.

WiSARD [11] is an early WNN for classification tasks which serves as the baseline for much subsequent work. As shown in *Fig 2*, WiSARD is composed of submodels called *discriminators*, with one discriminator per output class. Inputs are partitioned between the RAM nodes of a discriminator using a mapping function, which is typically shared between discriminators. During inference, RAM nodes are indexed using the addresses formed by concatenating their inputs. Next, each discriminator performs a popcount of the outputs of its constituent RAM nodes to produce a *response* score (*Fig 2a*). Lastly, the class corresponding to the discriminator with the largest response is chosen as the network's prediction (*Fig 2b*).

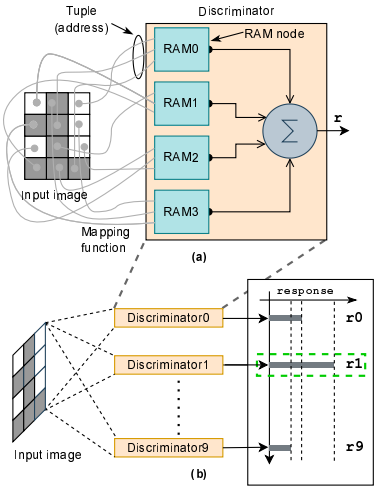

Fig. 2: WiSARD, a basic WNN for classification tasks. WiSARD trains discriminators for each output class, which are in turn composed of LUT-based RAM nodes. Figure adapted from [12].

Subsequent work has greatly improved the accuracy and memory efficiency of WiSARD. In Bloom WiSARD [13], the authors implemented RAM nodes using Bloom filters instead of simple LUTs, greatly reducing model sizes with minimal loss to even improvements in accuracy.[1] In [14], the authors proposed a multi-bit unary "thermometer" encoding of the inputs, in which each input is compared against a series of linearly increasing thresholds.

Recent WNN architectures, including LogicWiSARD [12] and BTHOWeN [15], have incorporated improved single-pass training techniques [16], model compression, and nonlinear thermometer encodings to produce efficient and accurate hardware accelerators, outperforming quantized and binary multilayer perceptrons.

D. ULEEN

ULEEN [6] is a FPGA based inference accelerator design for a WNN, as an extension of the multi-pass training technique proposed in [17]. Early WNNs were typically trained with a one-pass learning rule, where each sample was presented to the discriminator corresponding to its label. While subsequent works [16], [15] explored computationally efficient techniques, they did not incorporate feedback, and discriminators could not learn inhibitory patterns. The multi-pass WNN learning rule in ULEEN incorporates gradient-based feedback. This technique is also similar to learning rules used for training binary neural networks [18]. While the training is performed with floating-point values in RAM nodes, once training is complete, these values can be binarized by taking only their signs (treating negative values as 0 and positive values as 1). Therefore, while it is impractical to support training on the edge using this technique, it does not introduce any overhead to edge inference. Hence, ULEEN is developed to be used as an inference accelerator architecture. ULEEN also incorporates uniform pruning techniques [17] to identify and eliminate the RAM nodes which contribute the least to accuracy. This reduces both the memory footprint of the model and the amount of accelerator area needed for hash computation. *Fig 3* shows the view of a ULEEN model during inference. An ensemble of discriminator-based models is used to predict the scores of each class. The discriminator with the highest score across the submodels in the ensemble adjudicates the predicted class. Unlike LogicNets, ULEEN only has a single-layer of LUTs, that were updated when a similar pattern was observed during training.

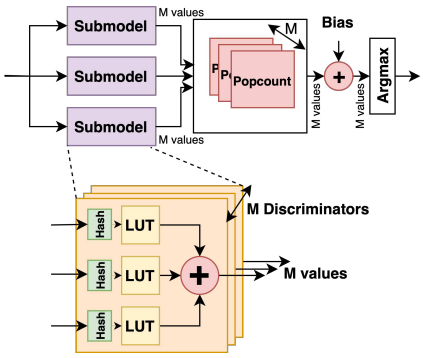

Fig. 3: ULEEN model inference. The input bits pass through an ensemble of submodels, each of which contain M discriminators corresponding to the M-classes, with a single layer of LUTs. Figure adapted from [6].

III. Methodology

A. Dataset Selection and Preprocessing

In order to compare LogicNets and ULEEN, we consider datasets in areas having high throughput applications and evaluate both on the same set. We note that while the design space available for exploration for ULEEN is defined and parameterized, that for LogicNets is quite vast. As such, we primarily focus on evaluating both the works on the datasets evaluated in the LogicNets [3] work - namely UNSW-NB15 and JSC, and try to find optimal configurations of ULEEN for these. We also evaluate these on BoT-IoT and Higgs datasets, two other closely-related dataset.

[1] In bloom filters, inputs to the filter are hashed through multiple functions, and the AND of the accessed LUT entries is computed.

1) UNSW-NB15 Dataset: The UNSW-NB15 dataset [4] is a common dataset used in the Network Intrusion Detection domain, where high throughput detection of malicious packets is critical. The dataset comprises of 49 input features, with each entry labelled as "normal" or "attack" - thus forming a binary classification problem. In order to evaluate it on ULEEN, we apply some basic preprocessing and balancing steps to it, sticking closely to the approach used in LogicNets. Samples in the dataset contain non-numeric features such as transaction protocol names. While some prior works including LogicNets and [19] experimented with one-hot encodings for these features, we found that this provided no benefit to the accuracy of ULEEN and significantly increased the model size. Therefore, we remove these features entirely. A small number ($\sim$0.01%) of samples are malformed (e.g., they contain non-numeric values for numeric features), and are culled. A further 19.4% of samples appear to be duplicates, so we remove all but one copy to avoid data leakage (the appearance of the same sample in the training and test data). After these steps, we are left with 205k samples, of which just 4.3% belong to the "attack" category. We perform a 9:1 train/test split on this data. To avoid training with highly imbalanced data, which is known to be a difficult problem [20], we balance the training data by randomly oversampling the minority "attack" class to a 1:1 ratio with the majority "normal" class. In the test data, we instead balance to a 2:1 normal to attack ratio. This is the same ratio used by LogicNets, and allows us to make a direct accuracy comparison. Note that training and test data are balanced separately to ensure test samples are not replicated in the training data.

2) BoT-IoT: BoT-IoT [21] is another Network Intrusion Detection dataset, that consists of 73,370,344 "attack" samples and 9531 "normal" samples extracted sequentially from network traffic. The dataset authors also provide a preprocessed subset of the data. However, this subset undersamples both classes equally, meaning it contains just 477 samples from the minority class. We instead preprocess the entire dataset using the technique described by the authors, generating new features based on a sliding window of 100 samples. We then select all "normal" samples and 2.2 million "attack" samples and perform a 9:1 train/test split. After this undersampling, BoT-IoT is still far more imbalanced than UNSW-NB15 (22:1 vs. 234:1). Therefore, we use ADASYN [22] to generate synthetic "normal" training samples. We balance the test data to a 1:1 ratio using random oversampling, and there is no synthetic data in the test set.

3) Jet Substructure Classification (JSC): The Jet Substructure Classification dataset [5] used in LogicNets is a low-latency inference requirement task, representative of workloads typically found in large physics experiments including those in CERN ATLAS and CMS. We use the same preprocessing and representation used in LogicNets and [5], to form a 16-feature 5-class classification task.

4) Higgs: The Higgs dataset [23] is another dataset in the realm of high-energy particle physics experiments. The dataset consists of features measured by particle detectors, and the accelerator is tasked with detecting if it is a particle (signal) or not (background). It is formulated as a 28-feature binary classification task.

B. Model Configuration Search

As noted in Section III-A, the design space available for exploration of LogicNets is quite vast, and there are no insights provided in the original work on how the configurations are found. For UNSW-NB15 and JSC, we use their most accurate reported configurations for comparison, considering them to be optimal. We train a LogicNets model similar to the one for UNSW-NB15 on the BoT-IoT dataset, and perform a parameter search to find a good model configuration for Higgs. While these may not necessarily be optimal configurations, it serves as an additional context for generalization.

For ULEEN, we note that we have the following flexibility: (i) picking the sub-models in the ensemble, (ii) configuring the number of hash functions, (iii) configuring the input bits mapped to each LUT, and (iv) configuring the size of each LUT. We seek to first experiment with randomly sampled set of configurations of sub-models and observe their hardware implementation's trends. Based on the inferences obtained from this, and some insights provided in [6], we identify constraints on the configurable parameters to make the sub-models feasible to be implemented on hardware. We then perform a grid search on the parameters within these constraints, and evaluate the performance of these sub-models. Susequently, a set of well-performing submodels that complement each other are picked to form the ensemble of models for ULEEN. We also prune the model using a conservative pruning ratio, that doesn't degrade the model accuracy significantly.

C. FPGA Implementation and Comparison Metrics

We implement ULEEN and LogicNets on the same FPGA, and compare them against various metrics of relevance. As with any ML-inference task, accuracy is one of the primary concerns. However, for edge applications, resource utilization, inference latency, throughput, and power consumption become equally critical - all of which depend on the model's deployment on a target FPGA.

We use the Python scripts provided by the authors of ULEEN to generate the SystemVerilog files for the accelerator designed for the finalized model configuration. We deploy this on the Xilinx Virtex UltraScale+ xcvu9p-flgb2104-2-i FPGA, the same used by the authors of LogicNets to enable a direct and fair comparison. Synthesis runs are performed using Vivado 2019.2 with the `Flow_PerfOptimized_high` strategy (which instructs synthesis to prioritize timing over power and area) in out-of-context mode, considering an inference sample-wide data bus as used in LogicNets[3]. We derive power and resource utilization numbers for ULEEN from Vivado implementation reports. We assume the default (12.5%) toggle rate. We generate the RTL for LogicNets and run it through Vivado's synthesis routines to estimate resource utilization and power numbers.

IV. Experiments & Results

A. Model Selection

We varied inputs, entries, and hash functions per Bloom filter and the number of thermometer encoding bits per input feature to identify efficient and accurate sub-models for ULEEN. We observed that while the hash functions in Bloom filter do not impact the model parameter size, they do require a substantial amount of hardware area. Therefore, we use as few unique hash functions as possible without significantly harming accuracy. We also observe that as we decrease the number of input bits mapped to each LUT, the hardware area increases multiplicatively. Consequently, we restrict ourselves to a larger number of input bits when exploring larger LUTs. Increasing the number of thermometer encoding bits eventually yields diminishing returns, though the point where this happened differed between the datasets. Within these constraints, we perform a grid search on the parameters. We prioritize picking fewer and more accurate sub-models for hardware efficiency. The final ULEEN model configurations are shown in Table I. We prune the ULEEN models by prioritizing high accuracy over an absolute minimum model size, as explored in prior work [17].

TABLE I: Optimal model configurations for ULEEN

Dataset	Bloom Filter			Thermometer
	Inputs	Entries	Hashfns.	Encoding Bits
UNSW-NB15	10	64	1	4
BoT-IoT	12	128	1	8
JSC (ensemble)	8	128	1	32
	10	256	1	
	12	512	1	
	14	512	1	
	16	1024	1	
	20	1024	1	
Higgs (ensemble)	6	16	1	12
	7	32	1	
	8	64	1	

B. Model Evaluation

Model parameter sizes and accuracies are shown in *Fig 4* and Table II respectively. LogicNets is initially trained as a sparse neural network, and then converted to a LUT-based model after training. This conversion greatly increases the model parameter size, but is readily optimized by synthesis tools to give a much smaller final hardware area than the parameter count would suggest. We provide both figures (before and after the LUT conversion) in *Fig 4* for clarity. ULEEN achieves excellent accuracy with very small model sizes on both the Network Intrusion Detection datasets. As shown in *Fig5*, ULEEN performs significantly better when evaluated on the ROC curve. On the JSC dataset, we observe that ULEEN performs comparably to LogicNets in terms of the accuracy. In this case, while ULEEN has a higher model size compared to the pre-LUT conversion stage of LogicNets, it is lower than that of post-LUT conversion. On the Higgs dataset as well, we note that ULEEN performs comparably to LogicNets at a lower model size.

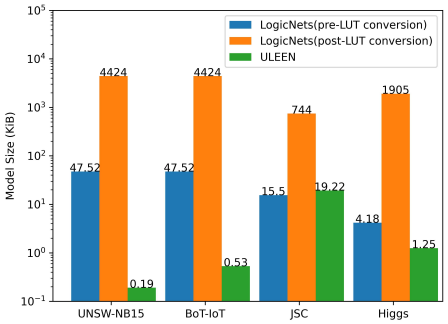

Fig. 4: Model parameter sizes for ULEEN and LogicNets. We report two figures for LogicNets: the training-time sparse model, and the much larger post-training LUT-based model.

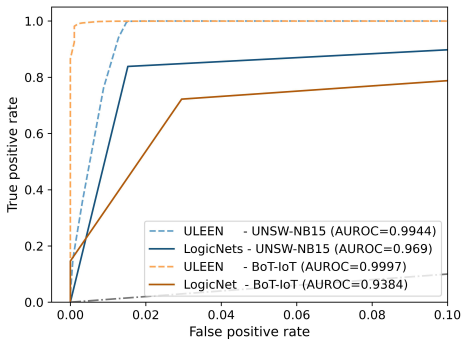

Fig. 5: ROC curve for ULEEN and LogicNets on the UNSW-NB15 and BoT-IoT intrusion detection tasks.

C. FPGA Implementation Results

We compare FPGA implementations of LogicNets and ULEEN for the datasets, shown in Table II. This comparison is performed on the 14/16nm xcvu9p-flgb2104-2-i FPGA, where the models are envisioned as being one IP block in a larger design. This enables the use of a very wide input bus width and an aggressive pipeline initiation interval of one inference sample per cycle. We note that ULEEN surpasses LogicNets for UNSW-NB15 and BoT-IoT datasets in terms of accuracy, throughput, area, and energy efficiency. On the Jet Substructure Classification (JSC) and Higgs particle detection tasks, LogicNets achieves a slightly higher accuracy than ULEEN, but ULEEN has a much smaller hardware and energy cost. For all the datasets, ULEEN achieves a much higher clock frequency, explained by its deeply pipelined design. Consequently, there is a 1.4×-1.9× improvement in throughput across the datasets. On most cases, ULEEN reduces the dynamic energy per inference[2] by a significant factor, and decreases LUT usage by 2.44×-59×. However, ULEEN utilizes more FFs in certain cases, owing to its deep pipeline.

The hardware savings of ULEEN are more pronounced when compared in terms of the area (LUT count)-delay(1/throughput) and energy-delay products. The dramatic decrease in area between LogicNets and ULEEN can be mainly attributed to the fact that **unlike DNNs, the individual RAM nodes within a WNN can learn non-linear functions**

[2]We use dynamic rather than total energy since the static power of unused blocks in FPGAs could not be isolated.

TABLE II: Comparison on Model Accuracy & Implementation Parameters

Dataset	Model Name	Test Accuracy	Clock (MHz)	Bus Width	Init. Intv.	Xput (MS/s)	Dyn.Energy (pJ/Inference)	LUTs	FFs	Area × Delay	Energy × Delay
UNSW-NB15	LogicNets NID-M	91.3%	471	593b	1	471	654	15,949	1,274	33.86	1.39
	ULEEN	**98.9%**	**740**	160b	1	**740**	**73**	**269**	**538**	**0.36**	**0.10**
BoT-IoT	LogicNets	88.6%	471	68b	1	471	654	15,949	1,274	33.86	1.39
	ULEEN	**99.4%**	**920**	272b	1	**920**	**156**	**530**	**1,079**	**0.58**	**0.17**
JSC	LogicNets JSC-L	**71.8%**	427	48b	1	427	6222	37,931	**810**	88.83	14.57
	ULEEN	71.3%	**773**	512b	1	**773**	**1222**	**4,774**	5,541	**6.18**	**1.58**
Higgs	LogicNets	**64.8%**	509	112b	1	509	**736**	6,380	**348**	12.53	1.45
	ULEEN	64.2%	**737**	336b	1	**737**	759	**2,612**	4,156	**3.55**	**1.03**

of their inputs. This provides a significant efficiency advantage to WNNs, as DNNs need multiple layers to capture the same behavior. Overall, we note that ULEEN has the potential to greatly reduce FPGA resource utilization while providing a high throughput, making it well poised for edge applications.

V. Conclusion

Machine Learning inference at the edge is a critical problem, that needs to be addressed by specialized algorithm-hardware co-design schemes. LogicNets and ULEEN are two recent FPGA-based designs developed in this direction, stemmed from different research directions. We comprehensively compare these techniques on various model and hardware parameters, and evaluate these on common datasets. While LogicNets has a slight accuracy advantage in two of the datasets, ULEEN outperforms LogicNets under similar resource environments and is ideal for high-throughput edge inference applications, achieving about $1.4\times$-$1.9\times$ throughput as compared to LogicNets on the evaluated datasets.

References

[1] Telefonaktiebolaget LM Ericsson, "IoT connections outlook," *Ericsson Mobility Report*, 06 2022. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/iot-connections-outlook

[2] N. N. Alajlan and D. M. Ibrahim, "Tinyml: Enabling of inference deep learning models on ultra-low-power iot edge devices for ai applications," *Micromachines*, vol. 13, no. 6, 2022. [Online]. Available: https://www.mdpi.com/2072-666X/13/6/851

[3] Y. Umuroglu, Y. Akhauri, N. J. Fraser, and M. Blott, "Logicnets: Co-designed neural networks and circuits for extreme-throughput applications," *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 291–297, 2020.

[4] N. Moustafa and J. Slay, "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)*, 2015, pp. 1–6.

[5] J. Duarte, S. Han, P. Harris, S. Jindariani, E. Kreinar, B. Kreis, J. Ngadiuba, M. Pierini, R. Rivera, N. Tran, and Z. Wu, "Fast inference of deep neural networks in fpgas for particle physics," *Journal of Instrumentation*, vol. 13, no. 07, p. P07027–P07027, Jul. 2018. [Online]. Available: http://dx.doi.org/10.1088/1748-0221/13/07/P07027

[6] Z. Susskind, A. Arora, I. D. S. Miranda, A. T. L. Bacellar, L. A. Q. Villon, R. F. Katopodis, L. S. de Araújo, D. L. C. Dutra, P. M. V. Lima, F. M. G. França, M. Breternitz Jr., and L. K. John, "Uleen: A novel architecture for ultra-low-energy edge neural networks," *ACM Trans. Archit. Code Optim.*, vol. 20, no. 4, dec 2023. [Online]. Available: https://doi.org/10.1145/3629522

[7] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. ACM, Feb. 2017. [Online]. Available: http://dx.doi.org/10.1145/3020078.3021744

[8] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," 2021.

[9] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2016.

[10] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Post-training 4-bit quantization of convolution networks for rapid-deployment," 2019.

[11] I. Aleksander, W. Thomas, and P. Bowden, "WISARD·a radical step forward in image recognition," *Sensor Review*, vol. 4, no. 3, pp. 120–124, 1984. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/eb007637/full/html

[12] I. D. Miranda, A. Arora, Z. Susskind, L. A. Villon, R. F. Katopodis, D. L. Dutra, L. S. De Araújo, P. M. Lima, F. M. França, L. K. John, and M. Breternitz, "LogicWiSARD: Memoryless synthesis of weightless neural networks," in *2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2022, pp. 19–26.

[13] L. Santiago, L. Verona, F. Rangel, F. Firmino, D. S. Menasché, W. Caarls, M. Breternitz Jr, S. Kundu, P. M. Lima, and F. M. França, "Weightless neural networks as memory segmented bloom filters," *Neurocomputing*, vol. 416, pp. 292–304, 2020.

[14] H. Carneiro, F. França, and P. Lima, "Multilingual part-of-speech tagging with weightless neural networks," *Neural Networks*, vol. 66, 03 2015.

[15] Z. Susskind, A. Arora, I. D. Miranda, L. A. Villon, R. F. Katopodis, L. S. De Araújo, D. L. Dutra, P. M. Lima, F. M. França, M. Breternitz, and L. K. John, "Weightless neural networks for efficient edge inference," in *31st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2022.

[16] D. Carvalho, H. Carneiro, F. França, and P. Lima, "B-bleaching : Agile overtraining avoidance in the wisard weightless neural classifier," in *ESANN*, 04 2013.

[17] Z. Susskind, A. T. Bacellar, A. Arora, L. A. Villon, R. Mendanha, L. S. De Araújo, D. L. Dutra, P. M. Lima, F. M. França, I. D. Miranda, M. Breternitz, and L. K. John, "Pruning weightless neural networks," in *ESANN 2022 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2022, pp. 37–42.

[18] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016.

[19] T. Murovič and A. Trost, "Massively parallel combinational binary neural networks for edge processing," *Elektrotehniski Vestnik/Electrotechnical Review*, vol. 86, pp. 47–53, 01 2019.

[20] S. Wang, W. Liu, J. Wu, L. Cao, Q. Meng, and P. Kennedy, "Training deep neural networks on imbalanced data sets," 07 2016, pp. 4368–4374.

[21] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," 2018. [Online]. Available: https://arxiv.org/abs/1811.00701

[22] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 1322–1328.

[23] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning," *Nature Communications*, vol. 5, no. 1, Jul. 2014. [Online]. Available: http://dx.doi.org/10.1038/ncomms5308