FPGA deployment of neuroevolved Binary Neural Networks

Raul Valencia
School of Computer Science,
University of Auckland
Auckland, New Zealand
rval735@aucklanduni.ac.nz

Chiu-Wing Sham
School of Computer Science,
University of Auckland
Auckland, New Zealand
b.sham@auckland.ac.nz

***Abstract*—Deep Learning has reached a prominent area of research thanks to advances in semiconductor technologies, with Deep Neural Network (DNN) as the pivot of change. It is capable of solving complex multi-dimensional problems. This paper focuses on one particular example, the Binary Neural Network (BNN): it uses fixed-length bits in its connections and logic functions to perform excitation operations, reducing memory requirements. The conventional use of Field Programmable Gate Arrays (FPGAs) dictates inference only of deep learning networks. This publication demonstrates how the algorithm Binary Spectrum-diverse Unified Neuroevolution Architecture (BiSUNA) can perform training and inference on FPGA by dismissing gradient descent, solving reinforcement learning and reaching maximum parallelism and energy efficiency, up to 16% faster compared to a CPU. Source code can be found in github.com/rval735/bisunaU50**

***Index Terms*—BiSUNA, Binary Neural Networks, OpenCL, Reinforcement Learning, Neuroevolution.**

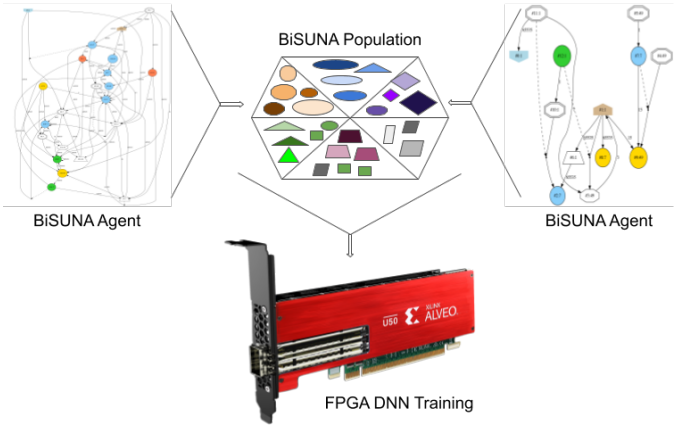

Fig. 1. A general overview of BiSUNA agents and the population deployed on an FPGA

I. Introduction

Training of Deep Neural Network (DNN) has taken multiple approaches to make them faster, more accurate or power efficient; one technique used to simplify models is the utilization of quantized values for weights, activations, inputs or outputs [1]. Traditional DNN (ex. AlexNet, Resnet, GoogLeNet) employs the principles of an algorithm named "Backpropagation" [2]: it operates derivatives to calculate a set of floating-point values that produces the least error, trained in two stages: forward and backward steps. When training finishes, a single fixed topology DNN is employed to resolve supervised learning problems.

However, as those models grow, also the memory and computation requirements. Althought there is a significant advancement on VLSI technology [3]–[5], only a partial solution is applied to reduce their impact, i.e. the "Quantization" of continuous values [6], [7]. This solution is deploying the trained models onto FPGAs but even some commercial entities offer tools to reduce the precision and execute inference only [8], [9]. However, given research performed for this publication, FPGAs are not widely used for training in comparison to CPUs or GPUs.

The last quantization frontier is a binary state (0 or 1), creating what is known as Binary Neural Network (BNN). BNN reduces the memory footprint by taking weights and biases values to the lowest denominator, which uses mainly bitwise operations and promotes model compression, further explored by [10]. On the other hand, BNN lack behind its continuous counterparts in accuracy when tested on Image Classification tests [11]. Most research around BNN has focused on the translation of backpropagation, an algorithm designed for floating-point values, to engage with discrete sequences; such transformation exacerbates instabilities at training time [12].

This project introduces a brand new approach to train Binary Neural Network using neuroevolution to perform the space search, with the addition of FPGA acceleration as a viable alternative to CPU/GPU training, previously barely explored.

II. Contribution

To gain all the BNN features, extend its capabilities to Reinforcement Learning (RL) and avoid drawbacks from a quantized backpropagation, this project takes a groundbreaking approach:

- Weights and activations use bitsets.
- Only logic operations (AND, XOR, OR, XNOR) are applied, nullifying the need for Arithmetic Logic Unit (ALU).
- Neuroevolution is employed to drive the space search, replacing a two-stage backpropagation.

- It generates a dynamic network topology to adapt and optimize for the problem at hand.
- The FPGA performs training and inference of BiSUNA neural networks.

These fundamental changes simplify hardware requirements to execute the algorithm. Given FPGA's reconfigurability, this algorithm acquiesces a pipelined architecture that effectively adapts to any RL problem to be engaged. At the same time, BNN models remain power/memory efficient. The name of this algorithm is Binary Spectrum-diverse Unified Neural Architecture (BiSUNA) [13].

Thanks to the BiSUNA framework, it is possible to solve RL challenges by creating different BNN topologies with multiple outcomes. The fundamental milestone is how FPGAs are used to train neuroevolved BNN competitively in terms of time when measured against a single thread CPU execution.

With the battery of tests performed, the FPGA 8 Compute Unit (CU) bitstream was executed with comparable time to a single thread CPU runtime, despite data movement over a PCI-Express channel.

III. Binary Spectrum-diverse Unified Neural Architecture

Authors of [14] demonstrated that reducing weight precision for floating-point connections decreases the memory needed to transfer neural models. BiSUNA moved that idea further by taking advantage of diverse weight sizes along an evolving topology that adapts to RL problems, with the addition of having binary sets instead of floating-point connections and neurons [13].

Two more publications [15] & [16] explored further how BiSUNA is a state-of-the-art TWEANN[1] algorithm with distinctive characteristics that strengthen its suitability for discontinuous-nonlinear systems. Those attributes are: Inputs and Outputs are bitsets; all neuron operations are logic functions; it implements population metaheuristics to find solutions; lastly, it trains without backpropagation.

BiSUNA uses evolutionary techniques to optimize neurons, connections and its topology structure. Such an arrangement enables the automatic survey of diverse constructions: some are fast, others are memory or execution efficient; variety promotes exploration and exploitation of different perspectives for the problem to be solved.

This publication capitalizes on a particular FPGA device, the Alveo U50 board. It has fast HBM2 memory, massive LUT resources and a simple integrated software development environment [18].

IV. FPGA training

Limited research has taken place to take advantage of FPGA flexibility at the time of training deep neural networks [19], a place where GPGPU dominates the market [20]. On the other hand, multiple examples have demonstrated FPGA's advantage when performing inference of DNN models [21].

[1]Topology and Weight Evolving Artificial Neural Network [17].

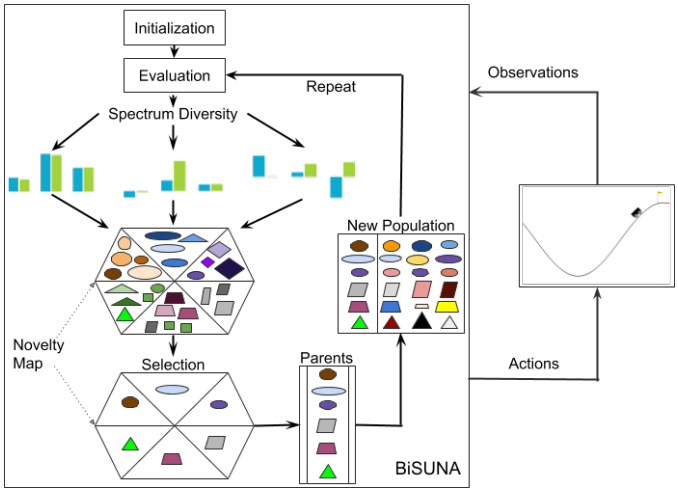

Fig. 2. BiSUNA evolutionary overview and environment interaction.

As an alternative optimization technique, evolutionary algorithms [22] operate similar bio-inspired operations continuously applied to multiple individuals at the same time, called "agents"; querying each takes most of the computation time with particular features:

- Agents' calculations are embarrassingly parallelizable for well-designed applications.
- Sharing updates and communication has low overhead, with examples of linear growth as more processors become available.
- Stored data required for computations can be efficiently cached, given the lack of backpropagation.
- Evolution has been proven by nature to perform an open exploratory behavior of complex environments.

Therefore, by iteratively querying multiple agents in parallel using dedicated CUs, it is possible to achieve more performance at scale. The BiSUNA framework employs neuroevolution with only binary operations/values in the creation of non-sequential neural networks in charge of solving RL problems.

All these characteristics fit FPGA's flexibility: unconventional data paths are comfortably implemented, specifically with hardware architectures optimized for logic functions and popcount operations. FPGAs typically consume less power compared to CPUs and GPUs [23].

V. Functional diagram

BiSUNA uses a fixed number of "agents" to explore any RL environment where every entity is a possible solution to the problem. Therefore, every agent must perform three basic operations: Process Input, Process Primers/Control, Process Remaining/Output, all of them refer to neurons in the mesh. Also, it is possible to visualize how each agent operates in global memory in Fig. 5.

Using the pipelined design pattern offered the right balance between resource utilization and development simplicity, similar to [24]. At every clock cycle, an agent moves its execution

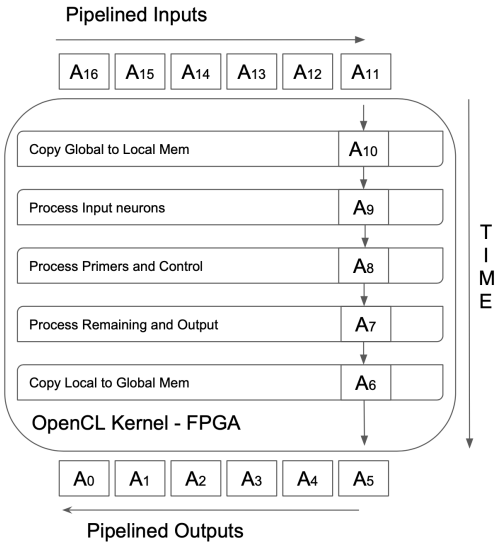

Fig. 3. Functional diagram of one Compute Unit, buffer catching and communication inside the OpenCL Kernel

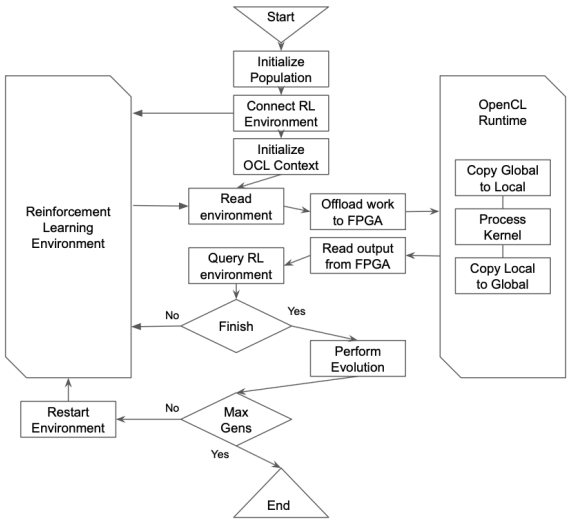

Fig. 4. Algorithm flow communication between the OpenCL runtime and the CPU.

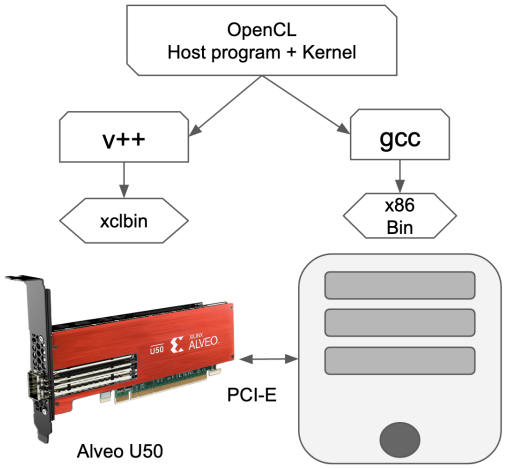

Fig. 5. High-level CPU - FPGA communication bridge of OpenCL host and device binaries

from one stage to the next, given that the Initiation Interval of most loops had to be at most 1 [25].

Zooming out from kernel processing, Fig. 4 shows the FPGA functional steps: Initialization is related to population and configuration, properties tightly related. It allocates enough memory to keep Neurons and Connections across growth in evolution stages. Next, the function "Connect RL environment" verifies links to internal or external actuators arrangements.

The next important part is the OpenCL context because it takes charge of device initialization, memory allocation and ongoing communication. Every time a new interaction from the RL environment, the OCL runtime moves data between the host and the device seamlessly. Finally, more conditional confirms that either a trial, a generation or execution has finished.

Moving one abstraction layer up, 5 shows the communication process that BiSUNA uses to offload work to the FPGA. First, the CPU enables an environment controller that is in charge of communicating with the RL puzzle, keeping track of the number of iterations, adaptation and agent organization. The data flow manager codifies the population and identifies which inputs/outputs correspond to each individual; everything using the PCI-E port, once again with Global memory represented as DRAM in this diagram.

VI. BiSUNA ADVANTAGE

BiSUNA is an open-source framework with a CPU implementation [26]. Given the non-linear characteristics, the subsequent step was the exploitation of commodity hardware to fit a particular purpose. After a profound analysis of current technologies, OpenCL was a clear choice given its hardware-agnostic philosophy, C/C++ compatibility and wide user baseline adoption [27]. Also, this selection enabled a targeted platform development, where FPGA offers a unique set of features that cover multiple areas critical for the usability of the algorithm: raw compute power, power efficiency and functional adaptation.

1) **Programmable Compute Power**: A good FPGA design can reach state-of-the-art execution exceeding GPU deployments, [28] in particular: Stratix 10 INT6 performance is more than 50% better than the Titan X theoretical peak INT8, with performance-per-watt 2x higher. It is important to note that large on-chip cache memory reduces bottlenecks/round trips to external memory, reducing energy costs.
2) **Design flexibility**: FPGA supports any range of data types precision; deployment of INT4, INT8, INT16, FTP16 or custom data lengths are key features for discretized DNN applications.
3) **Energy Efficiency**: Once more, [28] showed that Arria 10 FPGA reduces almost 10x power consumption

against a Titan X GPU. One factor influencing the power-hungry architecture found in GPUs originates at additional complexity required to facilitate software programmability (ex. CUDA). On the other hand, FPGA's reconfigurability, an agnostic software development stack offered by OpenCL (5), enables developers much higher efficiency for any number of applications using C as its base language.

4) **Functional Adaptation**: Multiple industries with application-critical functions take advantage of FPGAs, where safety is a key deciding factor; cases like factory automation, autonomous aviation and defense [29]. From its inception, the FPGA design has met special requirements that typical consumer electronics do not need. Given that future BiSUNA deployments could involve environments out-of-reach for humans, it is essential to design products that can adapt to extreme conditions.

VII. Experiments

Table I displays the hardware and software stack needed to deploy BiSUNA for the series of experiments performed for this paper. Taking the code from the BiSUNA repository [30], fulfilling all the necessary dependencies listed above, it is possible to replicate all claims found here.

TABLE I
HARDWARE AND SOFTWARE USED TO DEPLOY THE BISUNA ALGORITHM

Hardware	Software
Xilinx Alveo U50	Ubuntu 18.04
Intel i5 2.5GHz 2400S	Xilinx Vitis 2020.1.1
16 GB RAM	Xilinx Runtime 2020.1.1
1TB	Alveo U50 XDMA 201920.3
PCI Express 3.0	
225W Power Supply	

Multiple executions of the RL problem "MountainCar" [31] across platforms helps to reduce variability in the results. Despite MountainCar use of continuous values, BiSUNA can adapt to use discrete and floating-point neuron types.

Repository [30] contains the raw output results and screenshots of tools. It shows how acceleration numbers used programmable elements of FPGAs and traditional CPU executions.

One more feature this framework provides is the possibility to compile the executable with continuous neurons, also reviewed in the next section. It is challenging to compare objectively different architectures given the exponential number of tests a person could employ; consequently, the main focus was execution time.

What is even more encouraging, this project could resort to virtual cloud machines, like Nimbix or AWS F1 instances. Future projects would embrace the deployment of intelligent low power devices that are capable of learning from adverse environments.

This publication has explored the following metrics to confirm FPGA development reduces subjectiveness; from top to bottom are listed the types of characteristic analyzed along with a quick explanation of its meaning:

- **Processor**: The architecture used to execute code, options are CPU (Intel i5) or FPGA (Alveo U50).
- **Compute Unit**: How many kernel instances or threads were concurrently executing at the same time.
- **Exe Type**: This signals the underlying operations used, either floating-point or binary.
- **Generations**: Number of iterations each population had to evolve and reach a solution. In other words, a full cycle from Fig 1.
- **Best score**: The highest reward obtained at the end of a generation.
- **Min total time**: The minimum value to execute a generation over five iterations. It includes OpenCL set up for the FPGA.
- **Mean total time**: The mean value to finish all generations over five iterations. It includes OpenCL set up for the FPGA.
- **TDP**: Thermal Design Power estimates nominal power consumption of each device. Both devices considered the spec sheet for the maximum power consumption of the model under test.
- **Frequency**: Reported circuit operational frequency, CPU used the box specs, whereas the FPGA used the memory Vitis Report "KERNEL_CLK".
- **Time / TDP**: Ratio of the processing capacity against power consumption; in other words, it represents how fast the architecture can operate in comparison to the energy consumption it draws. Lower is better, signaling that it can undertake more agents wasting fewer resources.
- **Time / Freq**: BiSUNA Processing ratio in proportion to the reported frequency; in other words, the capacity to operate agents concerning the functioning rate. The lower the number, the more agents can be analyzed by the processing element.

A. Results

Table II summarizes all results regarding the execution between CPU and FPGA hardware. The reconfigurable hardware is capable of accelerating BiSUNA workloads by creating custom-design CU and reduce processing time. Despite OpenCL runtime time overhead, passing data to the device, waiting for it to be processed and retrieving it back, the outcome is even more dazzling.

The last two ratios included show a respectable improvement for 8CU when compared to the CPU execution. The Alveo U50 hastens BiSUNA operations considerably, with gains of 15x(binary) and 28x(continuous) for Time/TDP neuron processing.

In other words, executing the BiSUNA kernel on the Alveo U50 is 15x more power-efficient than performing the same task on the Intel i5 2.5GHz 2400S. Raw data used here can be accessed in the same repository [30], as the open-source implementation to confirm these claims.

TABLE II
SUMMARY OF EXECUTION BETWEEN CPU AND FPGA OPENCL KERNELS THAT SOLVES THE MOUNTAINCAR REINFORCEMENT LEARNING ENVIRONMENT.

Processor	CPU	CPU	FPGA	FPGA	FPGA	FPGA	FPGA	FPGA	FPGA	FPGA
Compute Unit	1	1	1	1	2	2	4	4	8	8
Exe Type	Float	Binary	Float	Binary	Float	Binary	Float	Binary	Float	Binary
Generations	100	100	100	100	100	100	100	100	100	100
Best score	-106	-123	-118	-122	-111	-123	-125	-124	-121	-121
Min Total Time	236	163	941	571	491	306	282	193	178	153
Mean Total time	260.8	184.2	1101.2	586.4	559.8	334	331.4	225.6	215.4	178.8
Max TDP (watts)	65	65	75	75	75	75	75	75	75	75
Frequency (MHz)	2500	2500	500	500	500	500	500	500	500	500
Time / TDP	4.012	2.834	14.683	7.819	7.464	4.453	4.419	3.008	2.872	2.384
Time / Freq	0.104	0.074	2.202	1.173	1.120	0.668	0.663	0.451	0.431	0.358

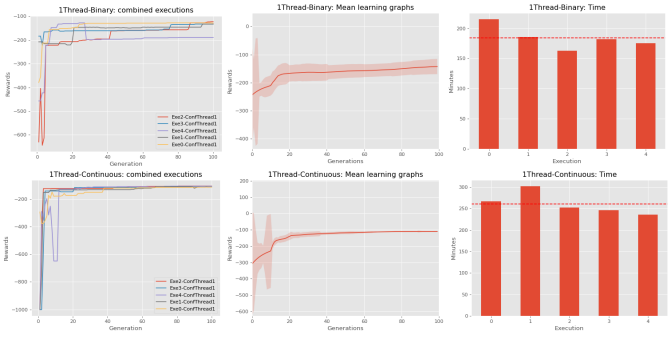

Fig. 6. BiSUNA agents CPU execution with 1 thread.

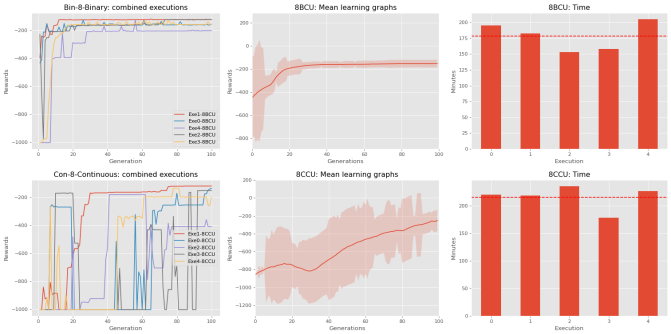

Fig. 7. BiSUNA agents FPGA execution with 8 CU.

By Assembling OpenCL kernels specialized for FPGAs, it is possible to retrieve key parameters of hardware design. Fig 9 provides a snapshot of the resource utilization when executing BiSUNA networks: 15.7K for a single CU with binary neurons, compared against 36.3K for its continuous counterpart.

Another BNN advantage is the reduced utilization of Digital Signal Processors (DSP) within the FPGA, only 4. On the other hand, floating-point neurons require 110 units. Finally, it takes 100 generations an average of 178.8 seconds to finish its execution with binary neurons, 215 with the continuous version, or even 184.2 for a single thread CPU to execute.

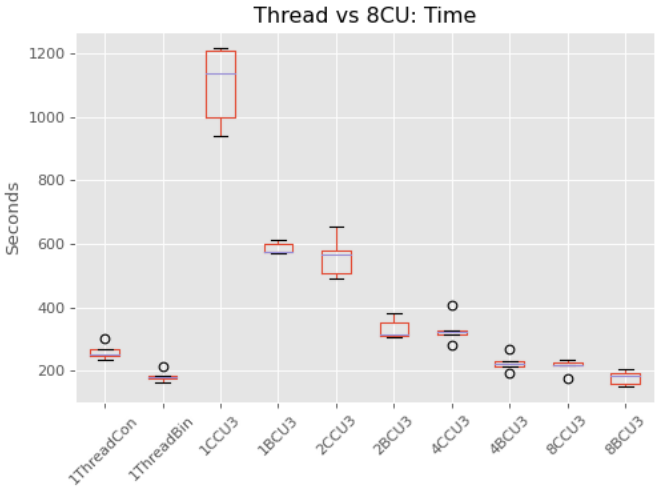

Fig. 8. Time summary to resolve MountainCar with different execution types.

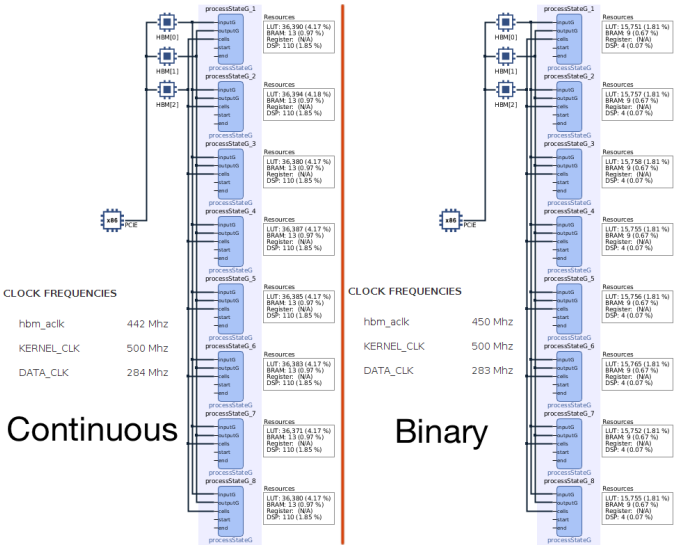

Fig. 9. Resource utilization of Continuous and Binary neurons.

VIII. Conclusion

The research community benefits directly from this work, given that it adds a new tool to train BNN using evolutionary principles, an innovative alternative that sidesteps completely gradient descent.

As a result of BiSUNA's proficiency to conditionally compile between discrete/floating-point systems, this work can be generalized to more environments, setting a precedence of BNN application to RL.

Experiments showed how binary neurons consume fewer FPGA resources in comparison with its continuous counterpart. Future research will optimize its architecture to the Alveo U50 board and extend its hardware support to other platforms.

As a benefit of how BiSUNA is architected, simple compile flags enable a switch from continuous to binary values. This project demonstrates a new research path where discrete values resolve RL problems previously not explored, given that traditional DNN (ex. QLearning [32]) relies on floating components to reach a solution.

Simplifying hardware requirements to train BNN contributes towards the creation of more efficient networks and circuits, facilitating FPGA adaptive compute acceleration inference and training.

The near future will convey more intelligent computation to any device, expanding their capabilities, increasing technology expertise to the next level, the Intelligent Internet of Things (IIoT).

References

[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, vol. 18, no. 187, pp. 1–30, 2018.

[2] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015.

[3] J. Lu, W.-K. Chow, and C.-W. Sham, "A new clock network synthesizer for modern vlsi designs," *Integration*, vol. 45, no. 2, pp. 121 – 131, 2012.

[4] J. L. ; and Chiu-Wing Sham, "LMgr: A low-M emory global router with dynamic topology update and bending-aware optimum path search," in *International Symposium on Quality Electronic Design (ISQED)*, March 2013, pp. 231–238.

[5] Chiu-Wing Sham, Wai-Chiu Wong, and E. R. Y. Young, "Congestion estimation with buffer planning in floorplan design," in *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*, March 2002, pp. 696–701.

[6] C.-Y. Lo, F. C. M. Lau, and Chiu-Wing Sham, "Fixed-Point Implementation of Convolutional Neural Networks for Image Classification," in *2018 International Conference on Advanced Technologies for Communications (ATC)*, Oct 2018, pp. 105–109.

[7] C. Y. Lo and C.-W. Sham, "Energy efficient fixed-point inference system of convolutional neural network," in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2020, pp. 403–406.

[8] Xilinx. (2020) Vitis ai.

[9] Intel. Fpgas for artificial intelligence (ai).

[10] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, 2019.

[11] Y. Hu, J. Zhai, D. Li, Y. Gong, Y. Zhu, W. Liu, L. Su, and J. Jin, "Bitflow: Exploiting vector parallelism for binary neural networks on cpu," in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2018, pp. 244–253.

[12] W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?" in *AAAI*, 2017.

[13] R. Valencia, C. Sham, and O. Sinnen, "Using neuroevolved binary neural networks to solve reinforcement learning environments," in *2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Nov 2019, pp. 301–304.

[14] X. Sun, J. Choi, C.-Y. Chen, N. Wang, S. Venkataramani, V. V. Srinivasan, X. Cui, W. Zhang, and K. Gopalakrishnan, "Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 4900–4909.

[15] R. Valencia, C. W. Sham, and O. Sinnen, "Evolved binary neural networks through harnessing fpga capabilities," in *2019 International Conference on Field-Programmable Technology (ICFPT)*, Dec 2019, pp. 395–398.

[16] R. H. V. Tenorio, C. W. Sham, and D. V. Vargas, "Preliminary study of applied binary neural networks for neural cryptography," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 291–292.

[17] M. Y. Ibrahim, R. Sridhar, T. V. Geetha, and D. S. S, "Advances in neuroevolution through augmenting topologies – a case study," in *2019 11th International Conference on Advanced Computing (ICoAC)*, Dec 2019, pp. 111–116.

[18] Xilinx. (2020) Alveo u50.

[19] W. Zhao, H. Fu, W. Luk, T. Yu, S. Wang, B. Feng, Y. Ma, and G. Yang, "F-cnn: An fpga-based framework for training convolutional neural networks," in *2016 IEEE 27th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2016, pp. 107–114.

[20] J. Kobielus. (2019) Gpus continue to dominate the ai accelerator market for now.

[21] A. Shawahna, S. M. Sait, and A. El-Maleh, "Fpga-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.

[22] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning," *ArXiv e-prints*, Mar. 2017.

[23] C. Grozea, Z. Bankovic, and P. Laskov, *FPGA vs. Multi-core CPUs vs. GPUs: Hands-On Experience with a Sorting Application*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 105–117.

[24] B. Shackleford, G. Snider, R. J. Carter, E. Okushi, M. Yasuda, K. Seo, and H. Yasuura, "A high-performance, pipelined, fpga-based genetic algorithm machine," *Genetic Programming and Evolvable Machines*, vol. 2, no. 1, pp. 33–60, Mar 2001.

[25] Xilinx. (2020) Loop pipelining and loop unrolling.

[26] R. Valencia. (2019, Sep) Binary spectrum-diverse unified neuroevolution architecture. https://github.com/rval735/BiSUNA

[27] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. Vrudhula, J.-s. Seo, and Y. Cao, "Throughput-optimized opencl-based fpga accelerator for large-scale convolutional neural networks," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '16. New York, NY, USA: ACM, 2016, pp. 16–25. http://doi.acm.org/10.1145/2847263.2847276

[28] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. Ong Gee Hock, Y. T. Liew, K. Srivatsan, D. Moss, S. Subhaschandra, and G. Boudoukh, "Can fpgas beat gpus in accelerating next-generation deep neural networks?" in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: ACM, 2017, pp. 5–14.

[29] J. Lant, J. Navaridas, A. Attwood, M. Lujan, and J. Goodacre, "Enabling standalone fpga computing," in *2019 IEEE Symposium on High-Performance Interconnects (HOTI)*, Aug 2019, pp. 23–26.

[30] R. Valencia. (2020) Bisuna - alveo u50.

[31] A. Moore, "Efficient memory-based learning for robot control," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, March 1991.

[32] T. G. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition," *J. Artif. Int. Res.*, vol. 13, no. 1, pp. 227–303, Nov. 2000.