ELEKTROTEHNIŠKI VESTNIK 86(1-2): 47–53, 2019
ORIGINAL SCIENTIFIC PAPER

Massively Parallel Combinational Binary Neural Networks for Edge Processing

Tadej Murovič[1], **Andrej Trost**[2]

[1]*ON Semiconductor Adria,*
Letališka 33, 1000 Ljubljana, Slovenia
[2]*Laboratory for Integrated Circuit Design, Faculty of Electrical Engineering, University of Ljubljana,*
Tržaška 25, 1000 Ljubljana, Slovenia

[1]*E-mail: Tadej.Murovic@onsemi.com*

Abstract. Binary Neural Networks (BNNs) have reached recognition performances close to those achieved by classic non-binary variants, enabling machine learning to be processed near-sensor or on the edge. The paper researches massively parallel combinational BNN logic and how it is to be used in real-world deployment situations which include training and constructing networks for imaging, cybersecurity and high-energy physics applications. For each field, a hardwired Verilog Hardware Description Language (HDL) code is built. It is synthesized for an FPGA system to create designs for a set of concrete edge processing problems. Synthesis results show that BNNs use minimal resources and achieve less than 30 ns inference delays, which is crucial for high-speed applications, less than 2 W power consumption and less than 60 k FPGA slices. It is shown that parallel BNNs enable efficient hardware machine learning performance for a variety of edge processing problems.

Keywords: BNN, Edge Processing, FPGA, Binary Neural Networks, Massively Parallel Neural Networks, HDL Design, Machine Learning

Masivno vzporedne kombinacijske binarne nevronske mreže za obdelavo na robu

Binarne nevronske mreže (BNM) dosegajo podobne zmogljivosti prepoznavanja kot klasične nebinarne različice in omogočajo izvedbo strojnega učenja v bližini senzorja ali na robu. Članek raziskuje masivno vzporedno kombinacijsko logiko BNM in njihovo uporabo v napravah s področja obdelave slik, kibernetske varnosti ter visokoenergijske fizike. Za vsako izmed področij uporabe smo izdelali model vezja v strojno-opisnem jeziku in ga sintetizirali v tehnologijo programirljivih vezij FPGA. Masovno vzporedna vezja binarnih mrež porabijo malo strojnih virov in energije (do 60k FPGA rezin in 2 W) in dosežejo sklep prej kot v 30 ns, kar je ključno za hitre aplikacije. V članku smo pokazali, da omogočajo masivno vzporedne BNM učinkovito strojno strojno učenje za vrsto nalog s področja računalništva na robu.

1 INTRODUCTION

Edge processing enables systems to effectively apply machine learning algorithms close to the sensor and only output processed data, drastically reducing data rates. With the recent research into real-time, latency-sensitive applications, such as Advanced Driver-Assistance Systems (ADAS) [1], High-Energy Physics [2] and Cyber-security [3], a need has come for low latency, always-on deep learning hardware systems. In addition to low latency, there is also the issue of power and size as applications are heavily affected by cost and power supply. State-of-the-art deployments must thus have a very low latency while fitting into a very small area and power constraints.

Binary Neural Networks (BNNs) have only recently been developed and defined by [4]. They are shown to reach a similar recognition accuracy as their floating-point versions while using only simple bit-wise operations and having a much smaller memory footprint. Their binary properties and simple operations make them prime candidates for embedded machine learning solutions on the edge. Research papers report efficient BNN deployment in Field Programmable Gate Array (FPGA) [5] and Application-Specific Integrated Circuit (ASIC) [6], [7] technologies. All of their BNN architectures use a similar structure configurable both in terms of latency and size producing considerable overhead, which is not desired in ultra-low power solutions. Progress toward that direction has been made by BinarEye [7] by developing an always-on ASIC neural processor for mobile platforms. Their solution achieves state-of-the-art performance in terms of size and power consumption while sacrificing configurability. Work by [6] goes

Received 21 January 2019
Accepted 18 February 2019

one step further by hard-wiring BNN interconnections making the network purely combinational. This way the best power and size performance for ultra-low power, always-on, latency critical systems is achieved.

In the paper we present trained and synthesized combinational circuits of BNNs on the edge via FPGA technology. FPGAs are seeing more usage lately because, in contrast to CPUs and GPUs, their completely reconfigurable digital circuitry enables them to create more efficient designs for specific applications. While our designs are also directly applicable to ASIC solutions, one apparent issue is that combinational BNN designs are not optimal as ideally one would like to change the weights and thresholds of the network while the chip is already deployed. Previous work with BNNs on ASIC [7], [8] has its network parameters stored in dedicated memory blocks. Doing a large number of memory accesses is far from optimal in terms of inference latency, i.e. the time needed for a neural network logic to produce a valid output and power consumption. Additionally, memories usually occupy the largest percentage of the final design size. FPGAs thus represent the most efficient way of achieving combinational parallelism on the edge, while maintaining full circuit reconfigurability.

In the paper, three different application areas requiring ultra low-latency, size and power consumption solutions are researched:

- *Speed, Power and Cost-Critical Imaging Solutions.* Visual systems are arguably the most important and widely used sensor systems. Cameras and their related machine learning algorithms are currently flag-shipped by convolutional neural networks in security, mobile and automotive markets. Systems are getting faster while requiring better machine vision performance and lower cost.
- *Flow-based IP packet classification hardware.* Modern IP networks need to classify possible malicious traffic. Because of the 100 Gb/s speed, this must be done with the lowest possible latency.
- *Particle Collision Classification for High Energy Physics.* Particle accelerators collide particles to break them into fundamental building blocks of nature. These collisions are numerous and not all can be processed. Therefore, a latency-critical hardware is needed close to collision sensors to trigger data capture if the event represents something worth exploring.

For each application a representative original or modified dataset to construct and train a small BNN is used. Its weights and thresholds are then used to create a fully combinational hardwired Verilog Register Transfer Level (RTL) code, which is synthesized for an FPGA system. While previous works only show that BNNs are effective on classic machine learning datasets and that there is a possibility to create very efficient parallel logic, we are the first to create multiple designs for a set of real-world applications.

In Section 2, the theory behind floating-point, quantized and binary neural networks and their massively parallel models is briefly explained. In Section 3, our tools, datasets and network construction flow for each application field are introduced. BNN circuitry synthesis results are shown and commented on in Section 4. In Section 5, our conclusions are drawn.

2 BACKGROUND

Artificial neural networks are mathematical/computational models vaguely based on the workings of biological neural networks. They are comprised of a set of interconnected simple processing elements called artificial neurons whose weighted connections and activation thresholds describe the behavior of the network. The sum of multiplications of neuron inputs and their corresponding weights is added to the neuronal bias or threshold. The resulting value is then processed through a non-linear activation element which sets the output of the neuron. This element usually represents a simple mathematical function, i.e. a step/sign, linear function or sigmoid function.

Computer vision and pattern recognition systems use artificial neural networks for sample pre-processing, feature extraction, regression, clustering, detection and recognition/classification. There are many different variations and topologies of neural networks, such as multilayer perceptrons, recurring, Hamming and recently convolutional networks. One of the simplest classification networks are multilayer perceptrons which are also used in this paper.

Multilayer perceptrons have fully connected layers, meaning every neuron at an individual layer is connected to every neuron in the next layer but never to neurons in the same, previous or next-to-next layers. It is proven that one can create linear boundaries in real-valued space with a single-layer perceptron network, piecewise linear boundaries with two-layer networks and arbitrary complex boundaries with three or more layered networks. Three-layer perceptrons can thus approximate Bayesian decision functions with the least-square error.

Neural networks can be trained to produce a certain output based on its inputs with backpropagation. Neuronal weights and biases are first randomly generated and then trained using our training set of labeled samples/features. For every input, we know what the output should be, from which we can calculate the error the network has generated. With backpropagation the network parameters are corrected using the derivatives of the error function throughout the network starting from the outputs via stochastic gradient descent. The derivatives are predominately described with the selected activation function, which is usually easily differentiable. We can

calculate the effect each weight and bias will have on the final output error, which we then change by its gradient multiplied by some learning factor so that the error will decrease. The error function gradients for every parameter are derived using the derivation chain rule, which means that we can easily propagate the error gradient through the network. By iterating and correcting our network parameters for the whole training set multiple times, we build the desired feature space boundaries for effective classification. Backpropagation always finds the local minimum in the cost/error function, which might not be optimal. This can be avoided by training the network multiple times using different starting parameters. Moreover, trained networks might be over-fitted meaning that they perform poorly on the testing set as they have not learned anything fundamental or general about the problem and have only created a one to one mapping of the training inputs to the desired class. This can be mitigated using fewer neurons or layers, a larger and/or more diverse training set, using regularization terms with the cost function (try to "push" all weights to be around 0, so not to explode in value) and batch normalization (each layer's outputs are normalized).

Network size can change during training by using either pruning techniques or constructive techniques. Pruning techniques iteratively remove network interconnections and neurons which have a negligible effect on the performance. Constructive techniques work the other way with a small number of neurons, which are then iteratively added to improve the performance on both the training and testing set.

Performance of neural networks is usually measured with $RMSE$ - (root mean squared error) for regression problems and for classification problems with $accuracy$ (number of correct classifications vs all observations), $recall$ (number of positive correct classifications vs all observations), $precision$ (number of positive correct classifications vs all positive observations) and $F1-score$ $(2*(Recall*Precision)/(Recall+Precision))$ [9].

2.1 Quantized Neural Networks

Previously described neural networks use full precision parameters and activations. This can be overwhelming for real-time deployment and small memory applications. Moreover, fixed-point arithmetic requires fewer resources than floating-point arithmetic when designing FPGA or ASIC solutions. Researchers have recently developed neural networks where all parameters and activations are presented in an N-bit fixed-point representation. This creates networks that are much smaller, faster, have a lesser memory footprint and are simpler, which enables them to be effectively used in real-time applications and be introduced to hardware implementations. These networks are trained via backpropagation either directly using fixed-point values or are trained using floating-point representation, which is then quantized. Their performance in recognition tasks has been shown to reach the accuracy of their floating-point cousins, while their fixed-point design enables them to be straightforwardly implemented in FPGA and ASIC technologies.

2.2 Binary Neural Networks

BNNs represent the extreme case of quantized neural networks, where all (full binarization) or some (partial binarization) weights and activations are constrained to values of +1 and -1. These signed values can be easily changed to 1 and 0 for a true digital representation. By binarizing network parameters, we can achieve smaller size and memory footprints of neural network implementations while reducing the complexity of calculations. Previous works show that BNNs reach similar recognition and detection performances as floating-point or fixed-point networks [4], [5]. In this paper, we focus on fully binary neural networks as opposed to only partially binary, which still require some floating or fixed-point operations and are thus not optimal for tiny power/size/latency hardware deployment.

The most prominent improvement of binary networks can be seen in their neural activation models. Non-binary networks use a large number of full precision multipliers and adder in addition to complex logic for neuron activation. A usual inference operation uses a weight multiplier for every neuron input. These values are then added together and used with a certain activation function, which is usually ReLU (clipped linear function). Figure 1 shows the most general neuron activation model.

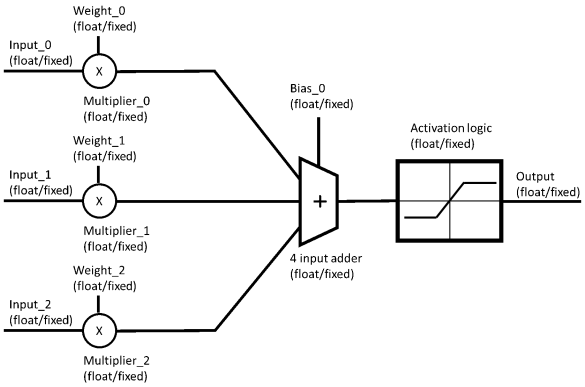

Figure 1. Typical floating/fixed-point neuron activation model. Every input is multiplied with its corresponding weight, these results and bias are summed and the sum is processed by some activation function.

Binary activation models for BNNs are, on the other hand, constructed of simple XNOR gates which represent binary multiplication and population count modules,

as the usual activation function for BNNs is the sign function. By counting weighted input bits and comparing that number to a certain threshold one can efficiently implement sign activation. Moreover, by changing the threshold, we can also include the bias and batch normalization terms [5]. Figure 2 shows the typical BNN neural activation model.

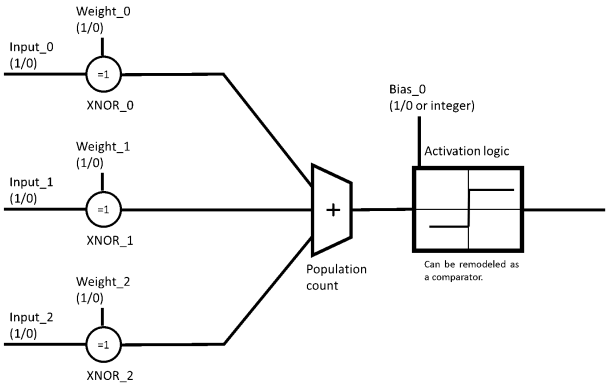

Figure 2. Binary neuron. Inputs are bit-wise XNOR'ed with their corresponding binary weights. The results are summed and compared with a certain threshold. Based on this comparison, the activation is purely binary.

The issue still remains that in fully-connected neural networks, every input neuron is connected to every output neuron in each layer of the network. It is important to note that convolutional layers for CNNs are actually special cases of fully-connected layers and vice versa. To achieve low latency and small logic size, the overhead of memory access and multiplexing logic can be drastically reduced by making the architecture fully parallel with hard-wired weights and thresholds [6].

3 Parallel Binary Neural Network Deployment

3.1 Training and RTL development

Our selected datasets are binarized and divided into training and testing subsets. These data formats are then used as inputs to the main Python script, which trains our binary MLP networks for binary classification problems. Training is done with backpropagation using the straight-through estimator which assumes that the gradient of a step function can be approximated with the value of 1 when in the neighbourhood of the step. This estimator must be used as otherwise the sign activation function does not allow gradient-decent optimization. Networks are trained using real-valued representation which is then binarized for inference testing. Learning algorithms are based on [4], where an in-depth explanation of binary neural network training and their performance is given. The software implementations of these algorithms are taken from GitHub repositories of [10]. These rely on different machine learning libraries, which are standard in research and industry neural network development: Theano, pylearn2 and Lasagne. Training is repeated multiple times with different network sizes and parameters (Figures 3, 4 and 5). The best performing network in relation to its size is then estimated and taken to be implemented in FPGA. The resulting network is afterward formatted into hardware appropriate form. The procedure is similar to the one in [5], where all thresholds, batch norms and weights are transformed into positive binary (weights) and integer (thresholds) values.

Finaly, we create our combinational BNN layer RTL models, which take the form of Verilog files. These are then exported to Vivado 2017.4 Synthesis Tool where the final synthesis and FPGA implementation for Xilinx's Zed Board is made. Our binarization scripts for each dataset as well as our final per layer Verilog files for each application are available online [11].

3.2 Cybersecurity

Packet classification is an essential procedure in modern IP networks, especially in the framework of cybersecurity or intrusion detection systems (IDS). Machine learning algorithms are used in such systems to detect and block malicious attacks and suspicious incoming data. Flow-based IDS are required to use only IP packet header data (source address, destination address, time-to-live, protocol etc.) and network flow information (speed of packets passing, density of packets, etc.). This way the size of data to process is drastically reduced because there is no need to access the actual user data inside a packet. Equally as important is the fact that not monitoring packet data maintains user anonymity, which is nowadays of an increasing concern [12]. As our networks are ever growing in size and speed there has never been more need for ultra-fast and efficient IDS packet classification algorithms. While most of them still use CPU-based classification, which is flexible as the algorithm code can be quickly updated in case new and unknown attacks become obvious, they are inherently sequential in nature and can struggle to reach the latency and throughput demands of modern networks, where data speeds can go up to 100 Gb/s. In this scenario, FPGAs and GPUs represent the move towards hardware solutions, which can dramatically improve packet processing speeds while maintaining specific algorithm configurability. Here we emphasize the use of FPGAs as in contrast to GPUs their tabula-rasa circuit concept enables them to be optimally massively parallel for any kind of machine learning algorithm and reach the desired performance. Configurability is maintained as circuits can be changed dynamically off-site and from inside the protected network [13].

Recently, artificial neural networks have been proven to efficiently and accurately classify incoming packets,

which makes them prime candidates for implementation in hardware-based network machine learning classifiers. In this case, parallel binary neural networks can play a major role in developing efficient intrusion detection systems [3].

The UNSWNB15 dataset [13], [14] was created by the Australian Center of Cyber Security by simulating modern intrusion attacks. It has 9 types of attacks with 49 generated flow features. For our training, all features are binarized to get a purely binary vector as input. Original features have different formats ranging from integers, floating numbers to strings. Integers, which for example represent a packet lifetime, are binarized with as many bits as to include the maximum value. Another case is with features formatted as strings (protocols), which are binarized by simply counting the number of all different strings for each feature and coding them in the appropriate number of bits. Floating-point numbers are reformatted into fixed-point representation. In the end, transforming all features into bits ended with 593-wide binary vectors. All vectors are labeled as bad (0) or normal (1). For training and testing, we used 25767 vectors (IP packets) randomly selected from the dataset. The data is also randomly divided into thirds, where the two-thirds are used for training and the remaining third for testing. The ratio of normal to bad vectors is 2:1.

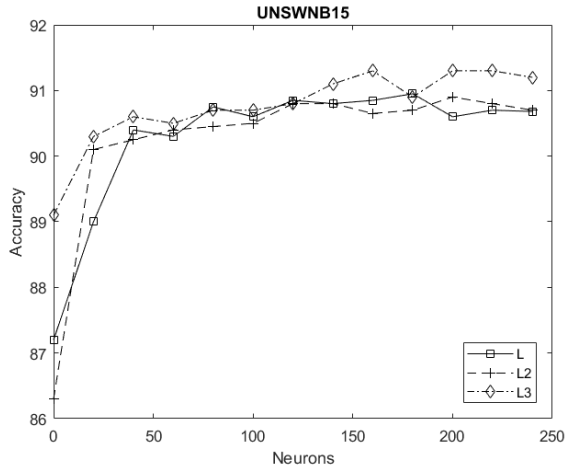

Figure 3. Packet classification accuracy with 1/2/3 layer BNN vs layer neuron number, where each layer has the same number of neurons. With 1 layer and 100 neurons we can achieve an accuracy of over 90%, while still having a relatively small circuit.

Figure 3 shows packet classification accuracy by our BNN. We can observe that increasing the number of layers does not have much influence on the final performance. Moreover, the curves seem to converge at an accuracy of around 90-91%. With that in mind, we opted to use 1 layer with 100 neurons, as that gives us the best accuracy, while keeping the networks size at a minimum. The final accuracy of our implementable BNN is 90.74%.

3.3 Exotic Particle Search in High-Energy Physics

Particle accelerators are used to speed up the building blocks of matter (usually protons) to high energy densities and then to collide them together in hopes of detecting new exotic particles that represent more fundamental constitutes of nature. After the particles are "crashed", their resulting particle shower is captured by different sensors around the point of impact. Sensors detect their kinematic properties from which the nature of the particles can be inferred. This way, the scientist at CERN discovered the Higgs boson in 2013, solidifying the accuracy of the standard model of particle physics [15].

The search for new particles is based on statistical models and large amount of data, where it is practically impossible to sift through all the data from all collisions, most of which hold no relevant information. Accelerators thus use special trigger hardware to classify collisions as background or signal, the latter of which represents candidates for interesting events. Devices must achieve a large number of classifications per unit of time with as little latency as possible because events occur at a rate of 10^{11} per hour at CERN's Large Hadron Collider, which leads to sub-microsecond inference requirements. Recently, development has been made using FPGAs with quantized neural networks for triggers which achieve good accuracy and latency performance on some datasets [2]. The need for latency critical machine-learning systems and FPGA technology hardware options naturally leads us to explore hardware BNNs for collision trigger systems.

For our experiments, we use data produced using Monte Carlo simulations of collisions gathered in the SUSY (Supersymmetry) dataset [15], [16]. In addition to low-level kinematic features, there are also derived features discovered by physicists to help discriminate between events. Collisions are labeled as belonging to a signal (1) or background (0). For training and testing, we use 50000 data vectors. The data is also randomly divided into thirds, where two-thirds are used for training and the remaining third for testing.

Figure 4 shows that by using 2 layers the accuracy increases to over 70% with the least amount of neurons (75 neurons per layer are selected). The final accuracy of our implementable BNN is 72.18%. The performance is drastically worse than in the other examples in this work. This stems from the fact that exotic particle event classification from the SUSY dataset is an extremely difficult problem, where even the state-of-the-art benchmark research achieves an accuracy around 80% with much larger and more complex non-binary neural networks [15].

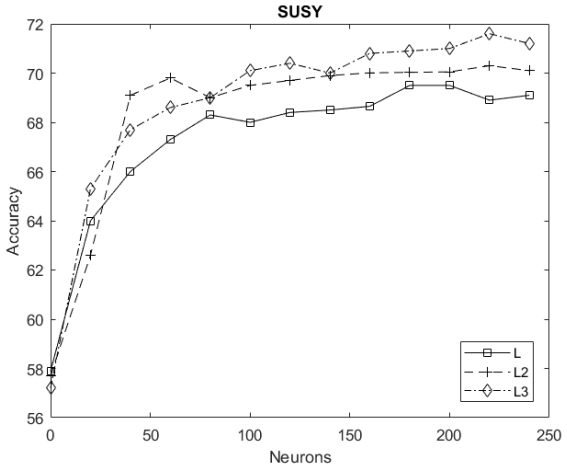

Figure 4. Particle classification BNN accuracy of around 70% is achieved with 2 layers and 75 neurons in each layer.

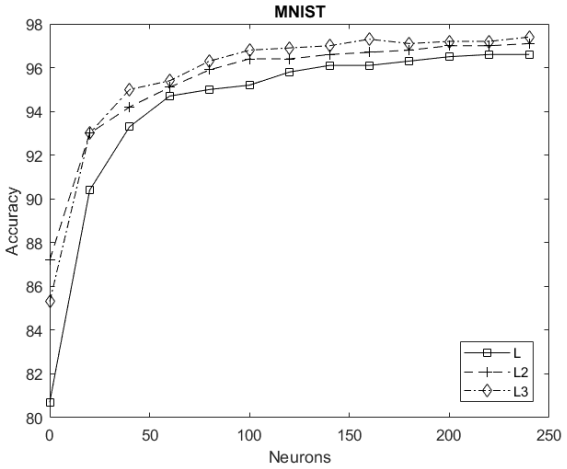

Figure 5. BNN for binary digit classification. Additional layers do not increase much the accuracy and we can use only one layer and 150 neurons to achieve over 96% accuracy.

3.4 Imaging

Machine vision is one of the most widely researched machine learning areas. Recently, autonomous driving/ADAS and security markets have been trying to achieve cheap and simple solutions to image classification. BNNs naturally represent a new step in developing super-high frame-rate, low-cost machine vision solutions.

The MNIST dataset [17] is considered the exemplary problem of machine learning. Most neural networks are firstly tested on this dataset to see if they even work. The dataset has 60000 images of handwritten digits from 0 to 9 and their labels. Recently, it has been shown that BNNs achieve results similar to floating-point neural networks, but with a fraction of their weight and calculation heaviness [4], [7], [5]. These networks are still relatively large and complex in respect to size, latency and power requirements of actual real-world deployment. For example, the network for MNIST from the seminal work [4] has 3 layers and 4 k neurons each. Therefore we argue that currently our expectations for achieving state-of-the-art accuracy in embedded systems on the edge on difficult benchmark machine learning datasets is too high and we should relax the goals of our machine vision chips to achieve the required performance. We relabel the MNIST dataset to transform the digit multiclass classification problem to binary classification in the sense that our BNNs must only detect if the digit in the number is above or below a certain threshold (in our case 4).

In Figure 5 we can see that the accuracy of the binary digit classification is mostly dependent on layer size and not their number, therefore we decide to use 150 neurons in 1 layer. The final accuracy of our implementable BNN is 96.13%. This accuracy is comparable to a more difficult MNIST problem (classify each digit) and a much larger neural network.

4 Synthesis Results

After, constructing and training our networks we built Verilog models of fully parallel/combinational BNN circuits. These are then synthesized for the Xilinx Zynq 7000 Zedboard FPGA and their resource usage is presented in Table 1.

Field	Inputs	Network	Delay [ns]	Power [W]	LUT
IDS	593	1x100	19.624	1.568	51353
HEP	301	2x75	24.692	0.68	19140
IM	400	1x150	21.432	1.474	44670

Table 1. Resource usage and performance of our binary MLP BNNs shown per application field. All inputs are one bit, as is the true/false binary output. IDS - Intrusion-Detection System; HEP - High-Energy Physics; IM - Imaging

Synthesis tools create an optimized circuitry from FPGA slices that functionally performs exactly as coded in its input files. Our cases have the added benefit of using zero registers (unless we want to buffer input or output data coming in or from our network respectively). Final synthesis results show that our parallel circuits achieve an inference latency of below 30 ns for all applications which is sufficient to be comfortably used in every deployment scenario presented in this paper. Additionally, power consumption falls below that of FPGA examples of quantized [2] and binary [5] neural network examples found in other works. The number of LUTs needed in any given design is below that of FPGAs in the lower price range (sub 60 k LUTs). This enables parallel BNNs to be readily deployed by most researchers and to be ideal for cost and performance driven market.

5 Conclusion

We identify and present application fields for hardware machine learning algorithms on the edge and show how to design massively parallel combinational binary neural networks on FPGAs for binary classification problems starting from specific datasets. Network sizes and their classification performance are presented for cybersecurity, high-energy physics and imaging application fields. The design synthesis results show that parallel BNNs enable efficient hardware machine learning deployment on the edge while consuming minimal resources in terms of size, cost and power and achieve sub 30 ns inference delays, which is crucial for high-speed applications. The presented problems can be deployed on a low-end FPGA, enabling powerful machine learning classification systems to be widely used by researchers and industry professionals alike. We do argue however that further work is required to invent new architectures or logic optimization techniques for parallel binary neural networks to achieve an even better performance for near-sensor/edge applications.

References

[1] S. Virmani and S. Gite, "Performance of convolutional neural network and recurrent neural network for anticipation of driver's conduct," in *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–8, July 2017.

[2] 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, 2018.

[3] L. V. Efferen and A. M. T. Ali-Eldin, "A multi-layer perceptron approach for flow-based anomaly detection," in *2017 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–6, May 2017.

[4] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), pp. 4107–4115, Curran Associates, Inc., 2016.

[5] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. H. W. Leong, M. Jahre, and K. A. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *FPGA*, 2017.

[6] M. Rusci, L. Cavigelli, and L. Benini, "Design automation for binarized neural networks: A quantum leap opportunity?," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2018.

[7] B. Moons, D. Bankman, L. Yang, B. Murmann, and M. Verhelst, "Binareye: An always-on energy-accuracy-scalable binary cnn processor with all memory on chip in 28nm cmos," in *2018 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–4, April 2018.

[8] A. Ardakani, C. Condo, and W. J. Gross, "A convolutional accelerator for neural networks with binary weights," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, May 2018.

[9] N. Pavešić, *Razpoznavanje vzorcev: uvod v analizo in razumevanje vidnih in slušnih signalov*. Fakulteta za elektrotehniko, 2000.

[10] M. Courbariaux, "Binary net." https://github.com/MatthieuCourbariaux/BinaryNet, 2016.

[11] "Bnn deployment." Deployment [Available Upon Acceptance], 2019.

[12] G. Karatas and O. K. Sahingoz, "Neural network based intrusion detection systems with different training functions," in *2018 6th International Symposium on Digital Forensic and Security (ISDFS)*, pp. 1–6, March 2018.

[13] W. Jiang and V. K. Prasanna, "A fpga-based parallel architecture for scalable high-speed packet classification," in *2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors*, pp. 24–31, July 2009.

[14] "UNSW-NB15 Dataset," 2015 (accessed September 10, 2018). https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/.

[15] P. Baldi, P. Sadowski, and D. O. Whiteson, "Searching for exotic particles in high-energy physics with deep learning.," *Nature communications*, vol. 5, p. 4308, 2014.

[16] "SUSY Dataset," 2014 (accessed September 10, 2018). https://archive.ics.uci.edu/ml/datasets/SUSY.

[17] "MNIST Dataset," 2004 (accessed September 10, 2018). http://yann.lecun.com/exdb/mnist/.

Tadej Murovič recieved his Master's degree from the Faculty of Electrical Engineering, University of Ljubljana in 2017. His research includes the development and implementation of signal processing algorithms and hardware applications of machine learning. Presently he works as an Algorithm Design Engineer for ON Semiconductor's Intelligent Sensor Group where he develops hardware CMOS solutions for radar and imaging sensor technologies inside the automotive market.

Andrej Trost received his Ph.D. degree from the Faculty of Electrical Engineering, University of Ljubljana in 2000. Currently he works at the same faculty as an associate professor teaching high-level design techniques on several graduate and post-graduate study levels. His research interests include the FPGA technology and digital systems design for academic and industrial applications.