Invited Paper: Resource Sharing in Feed Forward Neural Networks for Energy Efficiency

Abdullah M. Zyarah, Dhireesha Kudithipudi
NanoComputing Research Laboratory, Rochester Institute of Technology, Rochester, NY 14623

***Abstract*—Exploiting resource reusability and low precision in neural networks is a promising approach to achieve energy efficient computational platforms. This research presents two generalizable approaches to reuse resources in feed-forward neural networks and demonstrated on extreme learning machines. In the first approach, coalescing, a single stack of neuronal units perform both feature extraction and classification tasks through shared resources. In the second approach, folding, the neurons in a high-dimension feedforward layer are folded to execute multiple-tasks. The folding technique can also be combined with low precision modules. The proposed design techniques are validated for a classification task on binary (Australian credit and Diabetes corpus) and multi-class (MNIST) dataset. The total power consumption is measured to be 3.65 mW on TSMC 65nm technology node, while yielding an accuracy of 91.7% for MNIST.**

***Keywords*—Extreme Learning Machine; Folding; Coalescing.**

I. Introduction

Advancements in computational platforms have catapulted deployment of large scale neural networks. Conventional realizations of these neural networks focused on performance efficiency. Recent research is focused on holistic designs that prioritizes energy efficiency. Few key approaches to reduce power at architecture level are stochastic computation, low precision, resource sharing, and use of emerging technologies like memristors [1]–[3].

In this research, we study the power and area minimization in digital neuromorphic architectures with feed forward neural networks. Extreme learning machine (ELM) [4] is shown as a case study. ELM is a feed-forward neural network that offers high training speed and simple learning compared to other multi-layer perceptron networks with backpropagation learning [5]. This can be attributed to training only the output neuron weights while the hidden layer neuron weights remain unchanged after random initialization.

Several researchers have optimized the ELM in hardware targeting network performance, power dissipation, scalability, etc. An early digital implementation of ELM, proposed by Decherchi et al. [6] in 2012, performed training in software and only the inference was mapped onto an FPGA. In 2016, Frances-Villora et al. proposed a hardware implementation of a real-time ELM [7] with on-chip training. This work has presented an in-depth analysis of ELM in hardware and discusses the effect of data representation on network performance. Christiani et al. [8], in 2016, adopted the stochastic No-prop algorithm to implement a variant of ELM. Despite the fact that stochastic computing reduces hardware complexity, it suffers from low throughput. In 2016, a binary mixed signal ELM design is presented in [9], in which the hidden layer is an analog block and the output layer is a digital block.

This paper explores the resource reusability and low precision neuromorphic hardware to minimize the power and area in an a feedforward neural network (Ex: ELM). Centrally, we address two main questions through this research. Can neurons in the hidden and output layers share resources to taper the overall area and power dissipation? Can a low precision data representation (number of bits) provide additional degree of freedom in reducing the power with minimal effect on network performance? To answer these questions, three topologies are proposed and validated with binary and multi-class benchmarks, with a detailed analysis on throughput, power, and area.

II. ELM Algorithm

An ELM has three layers comprising of input layer, hidden layer, and output layer. Each of these layers is fully connected with the successive layer. Similar to any generic network, the input data is presented to the network, which is relayed further to the hidden layer neurons. In the hidden layer, the input features are stochastically chosen by mapping the input onto an N-dimensional space, where the most important and relevant features are extracted [10], [11]. The output layer neurons perform classification or regression. A distinct feature of ELM is that the training is accomplished by tuning only the output layer weights, while the hidden layer neuron weights are initialized randomly and remain unchanged. This is advantageous for fast training and also reduces the complexity of hardware circuitry.

A representative ELM network is shown in Fig. 1. At run time, each example in the input data set, L, is presented to the network as a pair. Each pair contains an input feature vector X_p and its associated class label t_p, where $X_p \in R^n$, $\forall p = 1, 2, ..., L$. Using (1), the network feed-forward output can be computed, where t_i^* represents the predicted output of the i^{th} output unit, $\forall i = 1, 2, ...M$. M is the total number of output layer neurons, NH denotes the number of hidden neurons, while f and z are the activation functions for hidden and output neurons, respectively.

$$t_i^* = z_i\Big(\sum_{j=1}^{NH} \beta_j f_j(X, b)\Big) \tag{1}$$

$$\beta = H^{-1}T \tag{2}$$

This material is based on research sponsored by AirForce Research Laboratory under agreement number FA8750-16-1-0108. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.
The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AirForce Research Laboratory or the U.S. Government.

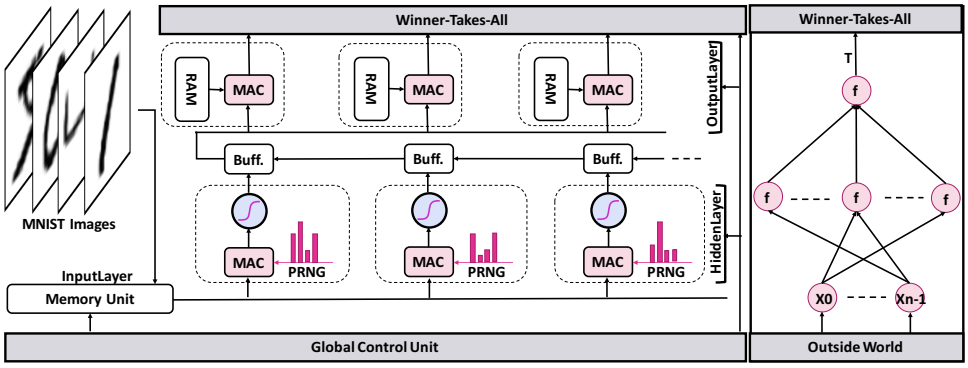

Fig. 1. An ELM network architecture, with three main layers: input, hidden, and output. The input layer buffers the input data stream into a block RAM, while the hidden and output layers perform the feature extraction and network classification, respectively. Global control unit controls the dataflow and synchronization of the entire network.

$$\beta = (H^T H)^{-1} H^T T \tag{3}$$

By adopting the normal equation, the output layer weight matrix β results from multiplying the Moore-Penrose generalized inverse of the hidden layer output matrix H by the training class label matrix T (T is zeros matrix with 1 indexed by class label), as in (2). The inverse of the matrix H can be found using the orthogonal projection, where $H^{-1} = (H^T H)^{-1} H^T$ [12]. As a result the output weight matrix, β, can be computed from (3).

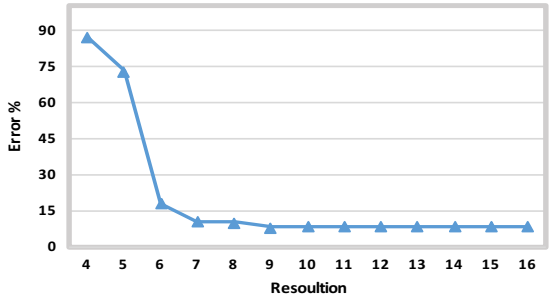

Fig. 2. The miss classification rate in MNIST dataset when changing the data resolution (data representation in number of bits) in the output layer of ELM.

III. Baseline ELM Architecture

A baseline ELM architecture is shown in Fig. 1, which consists of three pipeline stages mapping to the three ELM layers. The input features are fetched in a sequential manner and stored into block RAM memory of the input pipeline stage. The output of this memory reaches all the neurons in the hidden layer. Therefore, whenever a feature is read from the memory, it gets multiplied by its corresponding weight value and stored in an accumulator. To maximize the benefit of not tuning the hidden neuron weights, a 14-bit Pseudo random number generator (PRNG) is used for weight generation rather than storing weights in block RAMs [13]. The hidden neurons perform parallel processing of features from input layers and generate the final output simultaneously. The output of the hidden layer is stored in a buffer and used by the output layer to generate the final network output that would be fetched to winner-takes-all unit to determine the winning class.

IV. Proposed Topologies

A high-level representation of the baseline topology is shown in Fig. 3(a). The neurons of each layer are physically modelled in hardware, without any resource sharing. The classification rate, CR, can be computed as in (4). The CR is defined by three time elements including transfer the features to the network (T_{fetch}), perform the feature extraction ($T_{FeaturesExtract.}$), and classification time ($T_{Class.}$). Each time element is defined in Table I.

$$CR = \frac{1}{T_{fetch} + T_{FeaturesExtract.} + T_{Class.}} \tag{4}$$

1) Coalescing: In this topology, the hidden and output layers are meshed together to share the neuron resources, as demonstrated in Fig. 3(b). For example, by coalescing M neurons in the output layer with N neurons in the hidden layer, the total number of neurons will reduce to N instead of N+M. This topology requires identical nonlinear functions for neurons in both the layers and similar physical models. The classification rate for this topology, according to (4) and Table I, is slightly less compared to the baseline due to buffering and additional synchronization cycles.

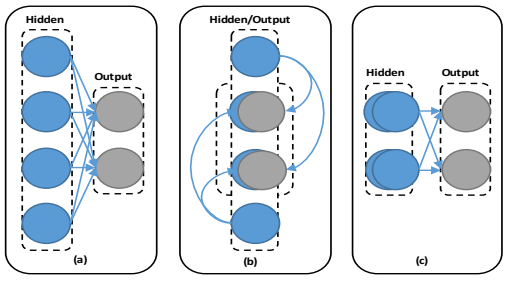

Fig. 3. Proposed Topologies: (a) Baseline, (b) Coalescing, and (c) Folding.

2) Folding: In this topology, Fig. 3(c), the layer with large number of neurons is folded k times by time-sharing resources. Folding a layer leads to extra latency (see (4) and Table I) in the network and throughput degradation. Thus, the value of k is application dependent.

TABLE I. The time elements of CR is defined in this table. The final expression of the time equation should be multiplied by the clk_{period}

Topology	T_{fetch}	$T_{FeaturesExtract.}$	$T_{Class.}$
Baseline	features, n [a]	features,n + 1	HiddenNeurons, NH [b]
Coalescing	features, n	features,n + c[c]	HiddenNeurons, NH
Folding	features, n	k × n	HiddenNeurons, NH

[a] n = Number of input features
[b] NH = Number of hidden neurons
[c] c = Constant varying between 1-3 for layers time synchronization

3) Low Precision Folding: This topology is an extension of the folding topology combined with low precision. In all previous topologies a 14-bit resolution is utilized. In case of LP folding, 14-bit resolution is used in hidden layer, and 9-bit for output layer. 9-bit is chosen via observing the impact of resolution changing in the output layer on the overall network performance. As shown in Fig. 2, 16-bit and 9-bit almost results the in same performance.

V. Results and Analysis

TABLE II. Summary of classification accuracy for binary and multi-class datasets using ELM

DataSet	Dimen.	ELM [14][a] N = 1000	VLSI ELM [9] N = 120	This work N = 80
Diabetes	8x1	77.95%	77.09%	76.25%
Australian Credit	14x1	87.89%	87.89%	88.86%
MNIST [b]	145x10	-	-	91.2%

[a] Software implementation of ELM.
[b] MNIST images are processed with HOG descriptor prior to fetch it to ELM as explained in [15].

A. Classification Accuracy

In order to validate the proposed hardware for classification task, three datasets are identified. Two of them are binary classification benchmarks from UCI library, Diabetes and Australian Credit [16]. The third dataset is MNIST [17], a hand-written digit standard dataset with 10 classes. The MNIST dataset is validated on the full digital architecture implementation of ELM and the proposed topologies. The Diabetes and Australian Credit are validated on the Matlab emulation of ELM architecture. An ELM architecture proposed earlier by our group is chosen as a baseline [15]. Table II depicts the classification accuracy (average of 10 runs) achieved in this work and compares it to [14] and [9]. It can be noticed that with fewer number of hidden neurons, comparable performance is achieved. Accuracy variation within 1% can be attributed to the random initialization of the hidden layer.

B. Throughput

Fig. 4 illustrates the throughput of various topologies. As expected, the baseline has the highest throughput. This is attributed to resource reusability in the three topologies which leads to 2.16x performance degradation (when k = 4). The throughput can be measured using (5), where $\#MACs$ refers to the number of multiply-accumulation operations that take place starting from feeding an inference sample to the network until the final network output is generated. The hardware can run up to 184.18 MHz. However, as the throughput and total power of the system are highly affected by the frequency, 100 MHz is chosen.

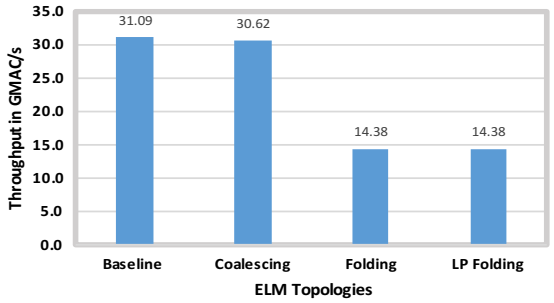

Fig. 4. The throughput of different ELM architecture topologies represented in GMAC/s.

$$Throughput = \#MACs \times CR \tag{5}$$

C. Device Resource Utilization

In order to observe the impact of resource reusability and low precision, all the proposed topologies are synthesized on FPGA platform Zynq-7000 xc7z030fbv676-3 and characterized by the number of LUTs, FFs, and DSPs units as shown in Fig. 5. For a network size of 145x80x10, LP Folding requires fewest number of resources (DSPs, LUTs, FFs, and memory blocks (not included in Fig. 5)) as it uses less number of neurons and less resolution.

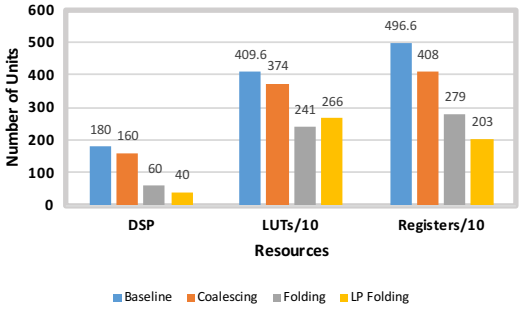

Fig. 5. Resource utilization for the three topologies in terms of LUTs, FFs, and DSPs.

D. Area and Power Consumption

Since we identify that LP Folding offers reasonable performance and reduced resource utilization, this topology was synthesized in Synopsys for TSMC 65nm technology node and verified with 2000 feature vectors that are propagated through the network running at 100 MHz. The area and total power of the LP Folding are 0.10 mm^2 and 3.65 mW, respectively. In an attempt to make a fair comparison with the previous ELM implementations presented in Table III, the measurement gap in dynamic power between FPGA and ASIC proposed by Ian Kuon [18] is adopted. This measurement estimates the dynamic power measured on FPGA as approximately 12 times

TABLE III. Comparison of proposed ELM architecture with hardware implementations on different ASIC and FPGA platforms. Data is obtained from [6], [7], [9].

Algorithm	**Real-Time ELM [7]**	**ELM [6]**	**ELM [6]**	**VLSI ELM [9]**	**This work**
Task	Regression	Classification	Classification	Classification	Classification
Device	FPGA-XC6VCX75TL-1L	FPGA-XC6VLX75T	CPLD-5M1270Z	ASIC	ASIC
Operating Frequency	86 MHz	127 MHz	30 MHz	31.6 KHz	100 MHz
Resolution input-output	12	16	16	10	14-9
Resolution internal	14-40	16	16	Analog-14	14-9
Power consumed	1.625 W	744 mW	90 mW	0.188 mW	3.62 mW
Power analysis tools	Xilinx Xpower 13	-	-	SPICE	Synopsys PrimePower
Network Accuracy	MAE $\leq$ 0.01	97.3%	97.3%	84.43%	91.2%
DataSet	Boston Houses Prices	MNIST	MNIST	Adult	MNIST
Network Size	13x40x1	81x18x10	81x18x10	128x100x1	144x80x12
Throughput	-	-	-	404.5 MMAC/s	13.57 GMAC/s
Technology node	-	-	-	35 nm	65 nm
Learning	On-Chip	Off-Chip	Off-Chip	Off-Chip	Off-Chip

(we understand that current FPGA boards have more power efficiency, and this is a slightly pessimistic approach) on average more than that on ASIC for the same design. According to the power analysis of the real-time ELM implementation proposed in [7], the dynamic power is equivalent to 22.91 mW on ASIC. Although the real-time ELM network has a smaller network size compared to LP Folded ELM, it still consumes more power. This is because the real-time ELM implementation uses more memory units and represents data with higher number of bits. For the proposed design in [6], the authors did not report the dynamic and the static power independently. Also, they did not discuss the specific power measurement approaches which make it hard to compare with the aforementioned design. In case of VLSI ELM [9], the network exhibits very low power consumption but low throughput compared to LP Folding. Running the LP Folding for the same throughput in [9] results in a power consumption of 200 µW.

VI. Conclusions

This paper investigates the resource reusability and low precision to optimize feedforward neural networks for energy constrained platforms. The results indicate that the resources used by ELM can be reduced significantly via time-sharing resources. Although this approach minimizes the network throughput, ELM still offers reasonable throughput and classification accuracy. The proposed design is validated with binary datasets (classification accuracy: Australian Credit= 88.86%, Diabetes= 76.25%) and multi-class dataset (classification accuracy: MNIST= 91.2%).

References

[1] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision." in *ICML*, 2015, pp. 1737–1746.

[2] I. E. Ebong and P. Mazumder, "Cmos and memristor-based neural network design for position detection," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 2050–2060, 2012.

[3] E. M. Ortigosa, A. Cañas, E. Ros, P. M. Ortigosa, S. Mota, and J. Díaz, "Hardware description of multi-layer perceptrons with different abstraction levels," *Microprocessors and Microsystems*, vol. 30, no. 7, pp. 435–444, 2006.

[4] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[5] R. Hecht-Nielsen *et al.*, "Theory of the backpropagation neural network." *Neural Networks*, vol. 1, no. Supplement-1, pp. 445–448, 1988.

[6] S. Decherchi, P. Gastaldo, A. Leoncini, and R. Zunino, "Efficient digital implementation of extreme learning machines for classification," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 8, pp. 496–500, 2012.

[7] J. V. Frances-Villora, A. Rosado-Muñoz, J. M. Martínez-Villena, M. Bataller-Mompean, J. F. Guerrero, and M. Wegrzyn, "Hardware implementation of real-time extreme learning machine in fpga: Analysis of precision, resource occupation and performance," *Computers & Electrical Engineering*, 2016.

[8] D. Christiani, C. Merkel, and D. Kudithipudi, "Invited: Towards a scalable neuromorphic hardware for classification and prediction with stochastic no-prop algorithms," in *Quality Electronic Design (ISQED), 2016 17th International Symposium on*. IEEE, 2016, pp. 124–128.

[9] E. Yao and A. Basu, "VLSI extreme learning machine: A design space exploration," *CoRR*, vol. abs/1605.00740, 2016. [Online]. Available: http://arxiv.org/abs/1605.00740

[10] J. E. Auerbach, C. Fernando, and D. Floreano, "Online extreme evolutionary learning machines," in *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, no. EPFL-CONF-200273. The MIT Press, 2014, pp. 465–472.

[11] R. K. Roul, A. Nanda, V. Patel, and S. K. Sahay, "Extreme learning machines in the field of text classification," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on*. IEEE, 2015, pp. 1–7.

[12] J. Tang, C. Deng, G.-B. Huang, and J. Hou, "A fast learning algorithm for multi-layer extreme learning machine," in *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 175–178.

[13] A. M. Zyarah, A. Ramesh, C. Merkel, and D. Kudithipudi, "Optimized hardware framework of mlp with random hidden layers," in *SPIE Defense+ Security*. International Society for Optics and Photonics, 2016, pp. 985 007–985 007.

[14] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 513–529, 2012.

[15] A. M. Zyarah and D. Kudithipudi, "Extreme learning machine as a generalizable classification engine," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017.

[16] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 26, no. 2, pp. 203–215, 2007.