2023 International VLSI Symposium on Technology, Systems and Applications (VLSI-TSA/VLSI-DAT) | 979-8-3503-3416-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/VLSI-TSA/VLSI-DAT57221.2023.10133986

Area Efficient VLSI ASIC Implementation of Multilayer Perceptrons

Rajeev Joshi, Lakshmi Kavya Kalyanam, and Srinivas Katkoori
Department of Computer Science & Engineering
University of South Florida, Tampa, FL
Email: {rajeevjoshi, lakshmikavya, katkoori}@usf.edu

***Abstract*—We propose a simulated annealing (SA) based neural network (NN) optimization approach for designing an area and power efficient hardware model for multilayer perceptrons. Our approach targets optimizing the hidden layer neuron weights to integer values. For each SA move, the subset of the hidden weights is perturbed at random. The perturbation amount (p) is set proportional to the annealing temperature (T). All the newly generated weights that are proximate to integers are rounded to the nearest integer to produce an optimized model. To further minimize the hardware, register resizing and operator strength reduction are applied. The objective of preserving the model's accuracy is favored in our approach. Our proposed methodology is validated using five MLP benchmarks. We find the most optimal solution when the temperature reduction rate (α) is 0.99 and the number of iterations (N) at each temperature (T) is 10K. With an average model accuracy degradation of 2.74%, the average savings for the best case scenario for area, power, and FFs are 27.53%, 27.28%, and 27.71%, respectively.**

***Keywords*—** Metaheuristic optimization, Neural Networks, IEEE 754, FP32, IoT, Edge-AI, VLSI, ASICs

I. Introduction

We have witnessed significant advancements in neural network accuracy over the last decade for a wide range of applications, including speech recognition, image classification, and object detection [1]. This increased accuracy is attributed to the NN model's overparameterization. At the same time, the rise in popularity of resource-constrained devices such as IoT edge devices demands the development of an effective NN hardware model that is portable, lightweight, and require low latency and power [2], [3]. However, because of their complexity, high computational costs and latency, and high memory requirements, deploying these highly parameterized NN models on hardware platforms with limited resources is a significant problem. [4]. Therefore, for applications in environments with resource constraints, researchers are becoming more and more interested in compressing NN model size while retaining model efficiency. This has emerged as one of the most popular research topics at the moment.

We propose an SA-based NN optimization method for MLP for designing efficient hardware models for resource-constrained hardware platforms. Our SA-based approach aims at optimizing the hidden layer neuron weights to the nearest integer values. This is accomplished by randomly perturbing the subset of the hidden layer neuron weights in each SA move. The newly generated neuron weights that are proximate to an integer are rounded to the nearest integer value. For each SA move, our method also seeks to maintain the accuracy of the model as an objective function. After generating the new weights, the weights with the value 0 are pruned away, and the weights with the value 1 and multiple of 2 are optimized by an operator strength reduction approach using bit shifting. The resizing of the registers is also performed to store all the new weight values that are rounded to integers. This further reduces the memory requirement to store the model weights. Our proposed SA-based MLP optimization method is evaluated based on an estimate of the hardware resources used by a single unit of an IEEE-754 single precision FP32 multiplier and adder. We perform a gate-level implementation of a single unit of single precision IEEE-754 FP32 multiplier and adder using Standard VT (SVT) cell libraries with the Synopsys Design Compiler in a 32 nm CMOS technology node using the SAED 32 nm Process Design Kit to validate our optimized NN inference hardware architecture. The total areas of a unit multiplier and adder are 5,697.11 μm^2 and 881.85 μm^2, respectively. Similarly, a unit multiplier and adder have average power dissipations of 122.65 μW and 24.77 μW, respectively, at a supply voltage of 1.05 V. For five MLP benchmarks, we performed 12 different experiments varying α for each N at each change in T. We find the best case solution when α is 0.99 at each T and N is 10K. The average savings for the best case solution for area, power, and FFs are 27.53%, 27.28%, and 27.71%, respectively, with an average loss of 2.74% in model accuracy when compared to the regular NN models.

The rest of the paper is organized as follows: Section II presents related works. Section III describes the proposed SA-based MLP optimization methodology for designing efficient hardware . Section IV reports experimental results and comparison. Section V draws conclusion.

II. Related works

In this section, we briefly review the related works from the aspect of designing an efficient hardware NN models. Several works have been done in the literature for compressing the size of NN models. Lee et al. [5] proposed an unstructured pruning method that prunes the insignificant weights once when a NN is initialized before training. Yu et al. [6] presented a general framework based on structured pruning method for compressing the NN based on neuron significance levels. Reinforcement learning is employed by Wang et al. [7] to propose a hardware-aware automated quantization framework that automatically determines a quantization policy to compress the size of the NN model. A one-shot pruning-quantization approach is presented

in [8] that reduces the NN size by analytically evaluating the compression allocation of each layer with pre-trained weight parameters without the requirement for manual tuning. The co-design of the NN architecture and hardware utilizing automated machine learning (AutoML) and neural architecture search (NAS), as proposed in [9]–[12] has also been investigated as a method for optimizing the NN model for resource-constrained environments.

III. The Proposed Method

We propose an SA-based optimization technique for the MLP model to design efficient hardware architectures. The proposed NN model optimization approach is based on calibrating the hidden layer neuron weights to the closest integer in order to compress the size of the NN model. The subset of hidden layer neuron weights are randomly perturbed as part of the optimization process, with the amount of perturbation (p) being proportional to the annealing temperature (T). At each T, the perturbation of hidden layer neuron weights are performed for N number of iterations. The newly generated weights that are proximate to integers are rounded as a part of the optimization process. We prune the integer weights with a value of 0, and the weights with a value of 1 and multiple of 2 are reduced using operator strength reduction such as bit shifting operations. In addition, all of the integer weights are adjusted by resizing the registers to reduce the number of bits to store the weight values. During the SA move, the accuracy of the model is favored while generating new weights. This helps reduce the hardware to design an efficient hardware architecture for MLP.

Figure 1 shows an overview of SA-based optimization process. A single precision IEEE-754 FP32 pre-trained MLP model and a training dataset (D) are used as inputs to our proposed method. We initialize our customized SA algorithm by generating an initial solution at random. We also specify starting initial temperature T_{init} and the number of iterations N. The T_{init} is kept at 100, and the N varies among 100, 1K, and 10K. We perturb random subset of hidden layer neuron weights W_p at each temperature, T. The amount of perturbation p is kept proportional to the T. The p is set at 10% for this experiment. Finally, we set the temperature reduction function, α. The value of α varies among 0.8, 0.9, 0.95, and 0.99 with variation in N for different experiments to seek the optimal solution.

We train the MLP model using a customized SA algorithm. The newly generated weights (W') that are proximate to the integer are rounded during the optimization process. We assess each SA move's prediction performance based on the optimized model's accuracy. If the performance criteria are met, we accept the new solution along with W' otherwise, we calculate the acceptance probability ($P_{acceptance}$). For the evaluation, we also generate a random number, R, where $R \in [0, 1]$. If ($R < P_{acceptance}$), we accept the new solution and W'. If not, we discard the new weights and run the SA again until the model converges to an optimal solution. Following acceptance of the new solution and W', the SA determines whether or not the maximum number of iterations (N_{max}) has been reached. If not, it loops back to the iterative process. If N_{max} is reached, it checks for the final temperature (T_{final}).

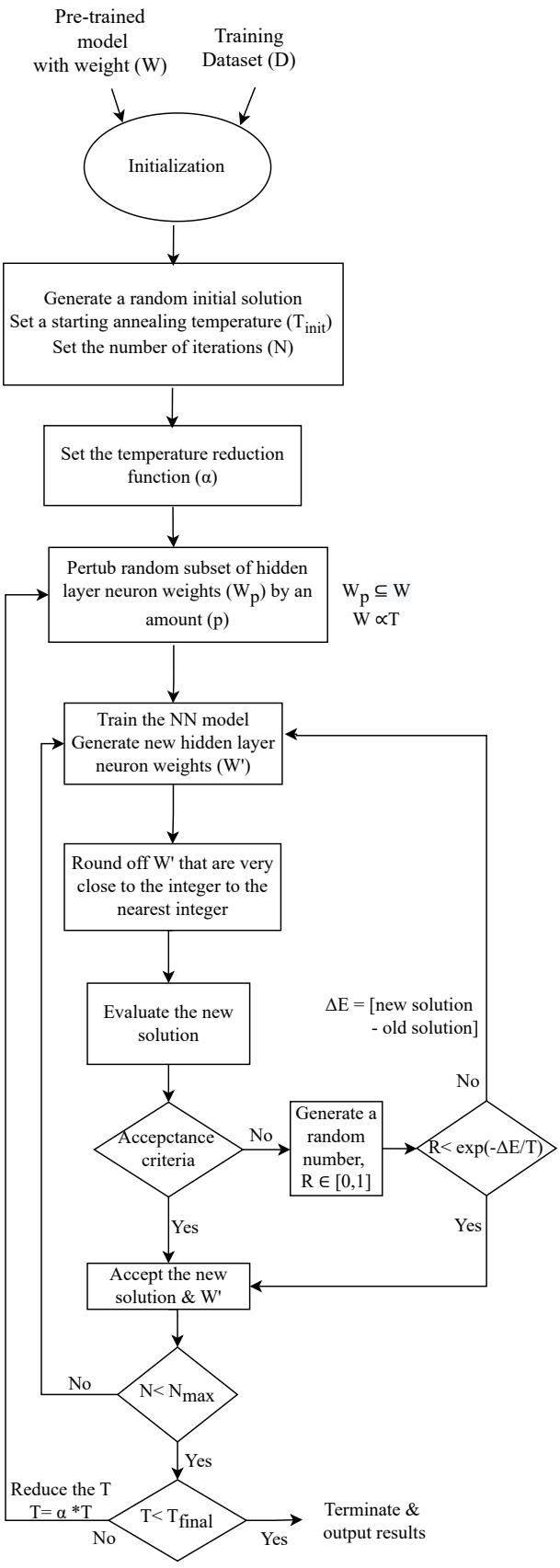

Fig. 1: An overview of our proposed SA-based optimization method

If T_{final} is not reached, the temperature is reduced using an equation, $T = \alpha * T$. The T decays with α. And, it loops back to the iterative process to train the NN model. Otherwise, the SA optimization process is terminated with an optimized MLP model as output. In our experiments, we find that all the integer W' are with in the value ranging between 0-10. Hence, we consider 4-bit registers for storing the integer W'. This further helps to reduce the hardware components contributing to the optimization of the MLP model. When the SA algorithm starts at a high temperature, the bad moves are accepted with a higher probability, and on each sweep it then gradually cools down until it reaches the lowest set temperature. SA moves that do not optimize the objective function are less likely to be accepted at lower temperatures.

IV. Experiments

In this section, we evaluate our proposed method using five well-known MLP classification datasets namely Iris [13], Heart Disease [14] , Breast Cancer Wisconsin (Diagnostic) [15], Credit Card Fraud Detection [16], and Fetal Health Classification [17], respectively. The MLP model configuration along with the number of parameters used for each of the datasets are shown in Table I. The pre-trained MLP model is generated using Tensorflow. The training and testing datasets are randomly divided in an 80:20 ratio.

TABLE I: MLP model configurations.

Dataset	MLP Configurations	# of Parameters
Iris	4-4-3	35
Heart Disease	13-10-2	162
Breast Cancer Wisconsin (Diagnostic)	30-10-2	332
Credit Card Fraud Detection	29-15-2	482
Fetal Health	21-21-3	528

The evaluation of our SA-optimized MLP hardware model is based on the estimation of the hardware resources utilized by a single unit of an IEEE-754 single precision FP32 multiplier and adder. To validate our optimized NN inference hardware architecture, we perform a gate-level implementation of a single unit of single precision IEEE-754 FP32 multiplier and adder using Standard VT (SVT) cell libraries with the Synopsys Design Compiler in a 32 nm CMOS technology node using the SAED 32 nm Process Design Kit. A unit multiplier and adder have total areas of 5,697.11 μm^2 and 881.85 μm^2, respectively. Similarly, the average power dissipation of a unit multiplier and adder is 122.65 μW and 24.77 μW, respectively, at a supply of 1.05 V.

We perform extensive experiments to evaluate the efficacy of our proposed methodology by comparing it with the regular MLP models. We perform a total of 12 experiments varying the α along with different values of N for each of the datasets to determine the area, power, and number of flipflops (FFs) used in the SA optimized MLP model. The p value is held constant at 10% for all the experiments. We compare these attributes to those of the regular MLP model to verify the effectiveness of our proposed method. And, we also evaluate the accuracy of each optimized MLP model. The bar graph in

TABLE II: Accuracy comparison between the regular and SA optimized models.

Dataset	Accuracy Regular model (%)	Accuracy SA optimized model (%)	Difference in accuracy (%)
Iris	96.67	94.41	2.26
Heart Disease	98.83	95.96	2.87
Breast Cancer Wisconsin (Diagnostic)	98.70	95.66	3.04
Credit Card Fraud Detection	96.82	93.93	2.89
Fetal Health	96.06	93.42	2.64
Average			2.74

Figure 2, 3, and 4 shows the percentage saving of our optimized MLP hardware architecture as compare to the regular MLP hardware architecture. From the experimental observations, we find that increasing the value of α improves the optimization performance of our proposed method. This, in turn, generates a more optimized MLP model. This results in improvements in hardware architecture in terms of area, power, and FFs, as observed in Figure 2, 3, and 4. For all datasets, the best case savings are observed when α = 0.99 and N = 10K with the average savings in area, power, and FFs being 27.53%, 27.28%, and 27.71%, respectively. Although the training time for the SA algorithm increases with increases in α and N, it still generates an optimal MLP model for hardware. The efficiency of each optimized MLP model is also analyzed compared to the regular model. Table II shows the accuracy comparison between the regular and SA optimized models. The average variation in accuracy of the best case solution in the optimized model as compared to the regular model for all the datasets is found to be 2.74%.

The extensive experimental results on five classification datasets show that the proposed SA-based MLP optimization method might be suitable for designing area and power efficient MLP hardware architectures. These lightweight, compressed architectures can be easily deployed on resource-constrained hardware platforms such as edge-AI IoT devices.

V. Conclusion

In conclusion, we propose an SA-based optimization method for MLP to design efficient NN hardware for resource-constrained hardware platforms. In our methodology, we finetune the hidden layer parameters by perturbing the subset of hidden layer neuron weights. The perturbation amount (p) is proportional to the annealing temperature, T. The new hidden layer weights that are proximate to integers are rounded, and operator strength reduction techniques are used to reduce the number of hidden layer parameters. Resizing of registers is also performed to reduce the amount of memory needed to store all the integer weights in the optimized model. We find the best case solution when α = 0.99 with N = 10K based on experimental observations on five classification datasets. Seeing the promising results, in the future, we are interested in extending this work to create an optimized model for a more complex NN model without sacrificing accuracy for resource-constrained hardware platforms.

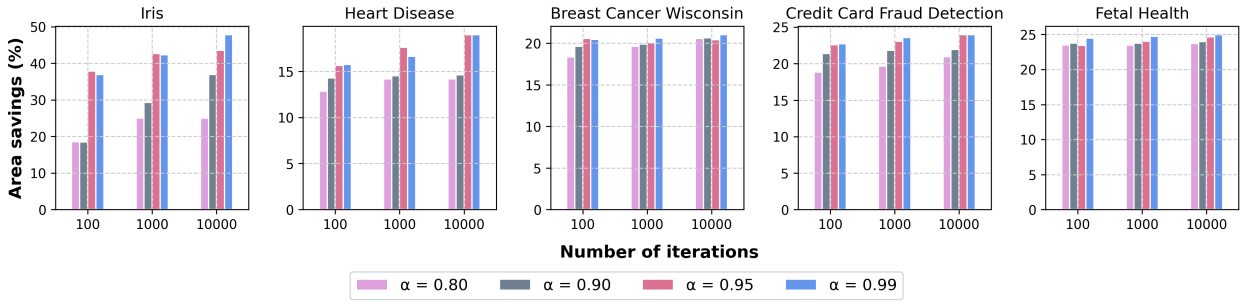

Fig. 2: Comparison of Area savings (%) vs. N, varying the α between regular and SA-optimized model for different datasets.

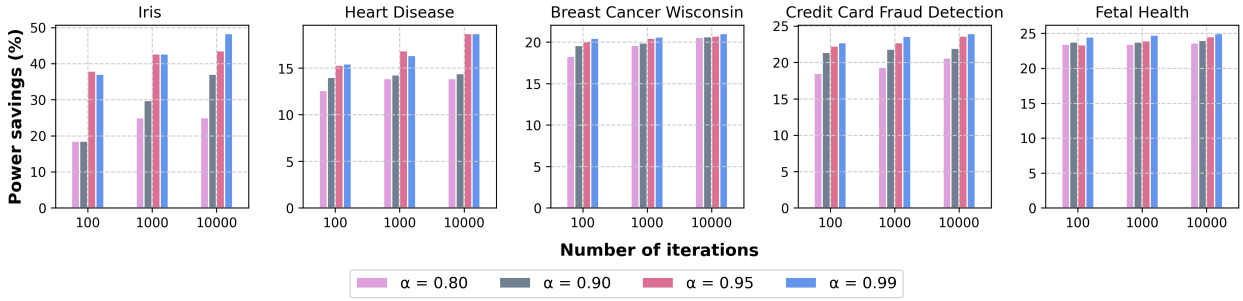

Fig. 3: Comparison of Power savings (%) vs. N, varying the α between regular and SA-optimized model for different datasets.

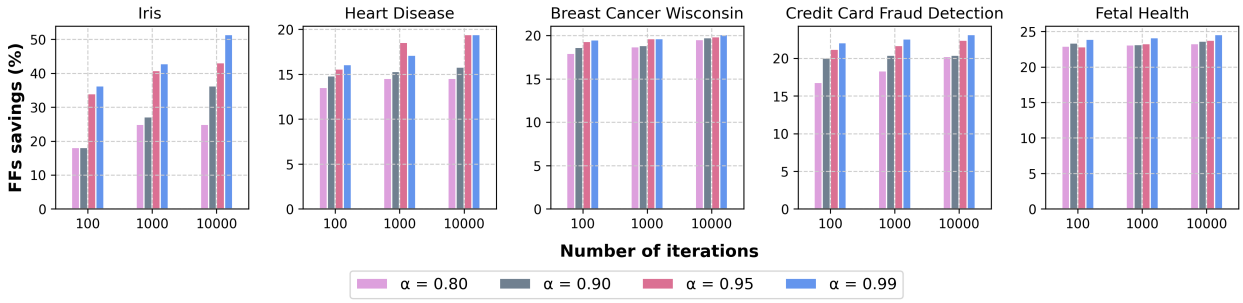

Fig. 4: Comparison of FFs savings (%) vs. N, varying the α between regular and SA-optimized model for different datasets.

REFERENCES

[1] L. K. Kalyanam, V. L. Ramnath, S. Katkoori, and H. Zheng. A distributed framework for real time object detection at low frame rates with iot edge nodes. In *2020 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS)*, pages 285–290. IEEE, 2020.

[2] R. Joshi, M. A. Zaman, and S. Katkoori. Novel bit-sliced near-memory computing based VLSI architecture for fast Sobel edge detection in IoT edge devices. In *2020 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS)*, pages 291–296. IEEE, 2020.

[3] R. Joshi, M. A. Zaman, and S. Katkoori. Fast Sobel Edge Detection for IoT Edge Devices. *SN Computer Science*, 3(4):1–13, 2022.

[4] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.

[5] N. Lee, T. Ajanthan, and P. H. Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018.

[6] R. Yu, A. Li, C. Chen, J. Lai, V. I. Morariu, X. Han, M. Gao, C. Lin, and L. S. Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9194–9203, 2018.

[7] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019.

[8] P. Hu, X. Peng, H. Zhu, M. M. S. Aly, and J. Lin. Opq: Compressing deep neural networks with one-shot pruning-quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7780–7788, 2021.

[9] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[10] H. Cai, L. Zhu, and S. Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[11] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.

[12] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[13] R. A. Fisher. Uci machine learning repository: Iris data set, 2011.

[14] R. Detrano. Uci machine learning repository: Heart disease data set, 2019.

[15] W. Wolberg, W. Street, and O. Mangasarian. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1995.

[16] A. Dal Pozzolo. Adaptive machine learning for credit card fraud detection. 2015.

[17] D. Ayres-de Campos, J. Bernardes, A. Garrido, J. Marques-de Sa, and L. Pereira-Leite. Sisporto 2.0: a program for automated analysis of cardiotocograms. *Journal of Maternal-Fetal Medicine*, 9(5):311–318, 2000.