Memory and Energy Efficient Method Toward Sparse Neural Network Using LFSR indexing

Foroozan Karimzadeh* and Arijit Raychowdhury*
*School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332
Email: fkarimzadeh6@gatech.edu, arijit.raychowdhury@ece.gatech.edu

2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC) | 978-1-7281-5409-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/VLSI-SOC46417.2020.9344090

***Abstract*—Deep Neural Networks (DNNs) require enormous computational power and storage memory. This impose a critical challenge to their efficient deployment on resource-constrained computing platforms such as edge devices. In this paper, we present a novel pruning algorithm and its hardware implementation to reduce the required memory-footprint and power usage of DNNs to enable them to be deployable on edge and mobile devices. we demonstrated a hardware-friendly pruning method where the locations of non-zero weights are derived from a Linear Feedback Shift Registers (LFSRs) in real-time. The results show a total power and area savings up to 49.97% and 50.20% for VGG-16 network on down-sampled ImageNet, respectively.**

I. Introduction

Ever increasing network of Internet of Thing (IoT) and edge devices requires the computation to be done on or close the source of the data. Edge computing has several benefits such as decreased latency and required energy, improved response times and better bandwidth availability. Accurate and fast computing on the edge plays an important role on different resource constrained applications such as autonomous driving and health-care monitoring. DNN has shown a great performance improvement especially in applications such as computer vision and healthcare data analysis [1]. However, their large and over-parameterized networks unable them to be used on resource-constrained edge devices.

To overcome this problem, various network compression techniques have been proposed to sparsify large DNNs, and enable them to fit in on-chip SRAM. However, sparse network add irregularity to the network which impose another level of complexity. Platforms such as GPU and DNN accelerators cannot take advantage of these sparse networks due to the lack of structure [2]. To overcome this problem, we propose a novel methodology to sparse large DNNs to reduce the memory footprint and energy usage while preserving the accuracy. We used an on-die linear feedback shift register (LFSR), to generate a pseudo random sequence (PRS). The generated PRS is utilized to prune the network. In addition, during inference we use the LFSR with the same seed to generate the indices in real-time to perform multiplication between the sparse weight matrix and input/activation vector. The advantage of our proposed hardware-friendly method is that we no longer need to store the sparse weight addresses – thereby reducing the memory foot-print significantly.

In this section, we first describe the pruning algorithm to sparse the network during the training step. Then the proposed hardware architecture for inference will be explained.

978-1-7281-5409-1/20/$31.00 © 2020 IEEE

A. Training pipeline of the proposed method

The algorithm consists of four steps (figure 1): (1) generating pseudo-random sequences (PRS) using two LFSRs (2) training the network with regularization (3) Pruning the weights (4) retraining the sparse network [3].

We first generate a PRS using two LFSRs, one for row indices indicating the address of the element in the input vector and another one for column indices which encode the address of the output vector. LFSR, consists of an array of flip-flops with an initial state called input seed (s), followed by linear feedback performed by several exclusive-or (*XOR*) gates (c_i). The output of LFSRs can be mathematically described through n^{th} order characteristic polynomials as follow:

$$x^n + c_{n-1}x^{n-1} + ... + c_1x + 1 \quad (1)$$

We then use these generated addresses for the weights that we want to keep during training and regularize the rest of the weight values to make them zero. As an example, a fully connected (*FC*) layer of DNN performs the following function:

$$a = ReLU(Z) \quad \text{where} \quad Z = W^T x + b \quad (2)$$

Where W is a weight matrix, x is an input vector, b is a vector of bias values. For simplification, b can be merged with W by appending an additional column to the end of matrix W. We used L2 regularization which a regularizer component is added to the cost (J) , as shown in Eq.3.

$$J(W^{[l]}, b^{[l]}) = \frac{1}{m}\sum_{c=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2m}\sum_{l=1}^{L} ||W_{ij}^{[l]}||_F^2 \quad (3)$$

Where *i* and *j* are correspond to the remaining indices (i.e. not chosen from LFSR 1 and LFSR 2) for rows and columns, respectively. L is the layer's number in the network. λ is the regularization parameter and can be tuned. Larger λ will more penalized the weights values and make them closer to zero.

After training step, in order to make sure that the regularized weights are exactly zero, we use a mask to make them exactly equal to zero. Finally, we retrain the network to better compensate the deleted weights and connections.

For inference hardware implementation, The generated LFSR #1 indices is used for the input to be multiplied to

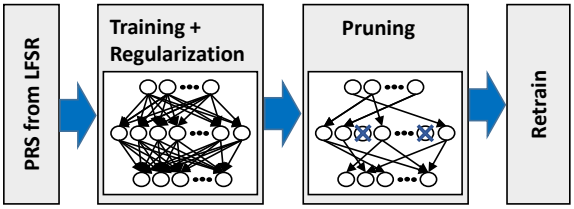

Fig. 1: Training pipeline for the proposed method.

the corresponding weight in sparse matrix weight (figure 2). LFSR generate the PRS with values between 1 to $2^N - 1$. To avoid redundant clock cycles when the generated number is greater that the number of neurons and keep the values between number of neurons, the generated PRS is multiplied to the length of neurons and the most significant bits (MSBs) are selected. Finally, the result is stored in the output buffer with the address generated by LFSR #2.

II. Results

The results are demonstrated on three networks: Lenet300-100, LeNet-5 and VGG-16 using MNIST, CIFAR-10 and down-sampled ImageNet data-sets. Training is done using Tensorflow platform on Nvidia GTX 1080 Ti GPUs.

We compared our results with a state-of-the-art pruning algorithm proposed in [2] where the pruning is done based on a threshold during training. For the baseline algorithm, the sparse weight matrix is compressed and saved in memory in three vectors: the non-zero weights' value (S), non-zero weights' location (I) and a pointer, pointing to each column of the weight matrix (P).The accuracy versus sparsity rates of our method is compared with the baseline (figure 3). The results show that our method can sparse the network more than 80% while preserving the accuracy even in complex network and datasets such as VGG-16 on down-sampled ImageNet.

To evaluate our method during inference, we have synthesized baseline and our method with 65nm CMOS technology to measure hardware metrics. The bit-width precision is considered to be 8 bits for weights and activations and 4, 8 bits for indices. The pre-layout analysis demonstrates up to 64% reduction in required memory footprint compared to the baseline pruning technique (Figure 4). Moreover, the power and area of the overall system (memory, multiplier, accumulator and input/output buffers) are measured for VGG-16 network (figure 5). We observe a maximum of 49.97% power savings and 50.20% area savings across varying sparsity versus baseline designs.

References

[1] R. Boostani, F. Karimzadeh, and M. Nami, "A comparative review on sleep stage classification methods in patients and healthy individuals," *Computer methods and programs in biomedicine*, vol. 140, pp. 77–91, 2017.

[2] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

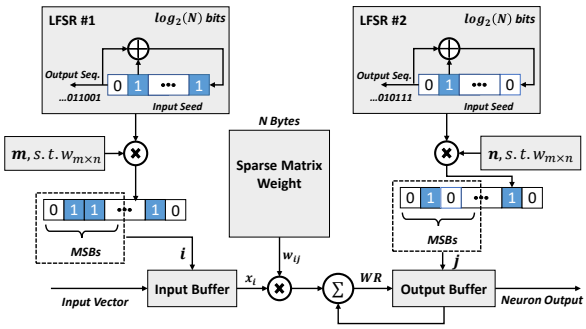

Fig. 2: The proposed method high-level architecture.

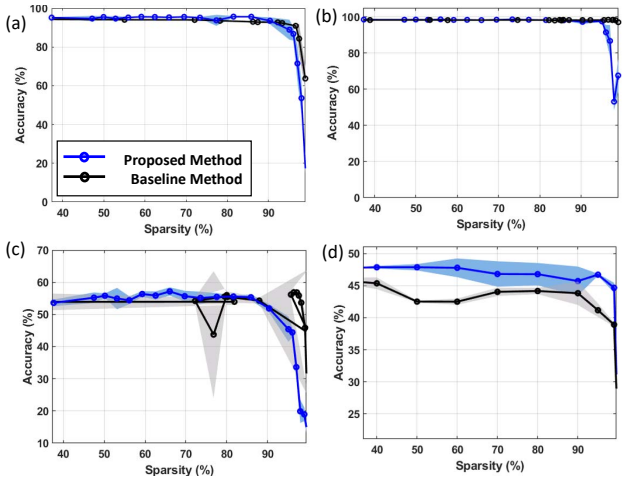

Fig. 3: Accuracy (%) versus different sparsity rates. (a) LeNet300-100 on MNIST, (b) LeNet5 on MNIST, (c) LeNet5 on Cifar10, (d) VGG-16 on down-sampled ImageNet.

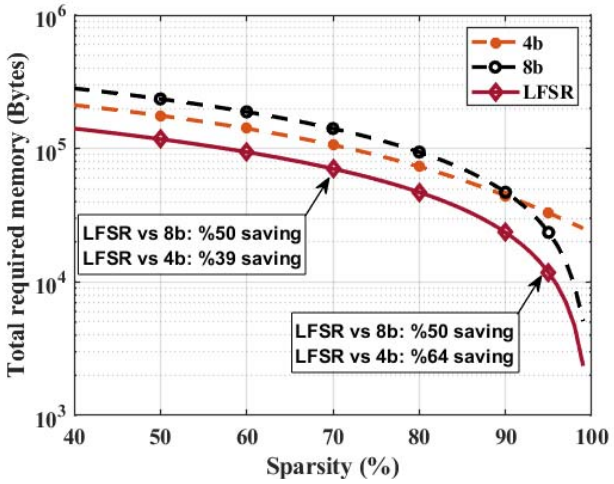

Fig. 4: Total required memory for our method and the baseline method with 4 and 8 bit precision at different levels of sparsity.

[3] F. Karimzadeh, N. Cao, B. Crafton, J. Romberg, and A. Raychowdhury, "Hardware-aware pruning of dnns using lfsr-generated pseudo-random indices," *arXiv preprint arXiv:1911.04468*, 2019.

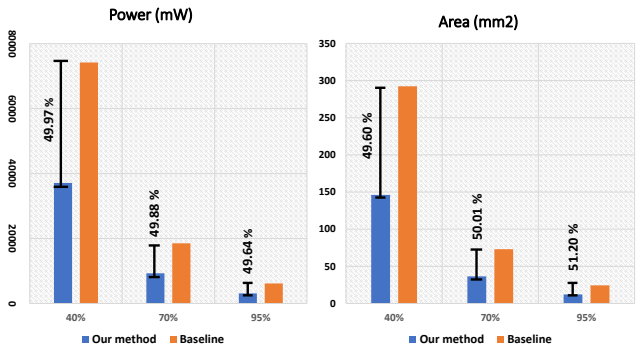

Fig. 5: Measured Power (mW) and area (mm^2) of the overall system for our proposed and baseline method on VGG-16.