On the Complexity Reduction of Dense Layers from $O(N^2)$ to $O(NlogN)$ with Cyclic Sparsely Connected Layers

Morteza Hosseini[†], Mark Horton[†], Hiren Paneliya[†], Uttej Kallakuri[†], Houman Homayoun[§], and Tinoosh Mohsenin[†]
[†]Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, MD, USA
[§]Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA, USA

ABSTRACT

In deep neural networks (DNNs), model size is an important factor affecting performance, energy efficiency and scalability. Recent works on weight pruning have shown significant reduction in model size at the expense of irregularity in the DNN architecture, which necessitates additional indexing memory to address non-zero weights, thereby increasing chip size, energy consumption and delays. In this paper, we propose cyclic sparsely connected (CSC) layers, with a memory/computation complexity of $O(NlogN)$, that can be used as an overlay for fully connected (FC) layers whose number of parameters, $O(N^2)$, can dominate the parameters of the entire DNN model. The CSC layers are composed of a few sequential layers, referred to as support layers, which result in full connectivity between the Inputs and Outputs of each CSC layer. We introduce an algorithm to train models with FC layers replaced with CSC layers in a bottom-up approach by incrementally increasing the CSC layers characteristics such as connectivity and number of synapses, to achieve the desired accuracy given a compression rate. One advantage of the CSC layers is that there will be no requirement for indexing the non-zero weights. Our experimental results using AlexNet on ImageNet and LeNet300100 on MNIST indicate that by substituting FC layers with CSC layers, we can achieve 10× to 46× compression within a margin of 2% accuracy loss, which is comparable to non-structural pruning methods. A scalable parallel hardware architecture to implement CSC layers, and an equivalent scalable parallel architecture to efficiently implement non-structurally pruned FC layers are designed and fully placed and routed on Artix-7 FPGA and ASIC 65nm CMOS technology for LeNet300100 model. The results indicate that the proposed CSC hardware outperforms the conventional non-structurally pruned architecture with an equal compression rate by ~2× in power, energy, area and resource utilization when running at the same frequency.

1 INTRODUCTION

Sparsifying DNNs is a subject of interest because sparsity in DNNs not only compresses the model, but also allows for computation reduction by skipping unnecessary computation. Pruning methods such as L2 regularization [6] can zero-out a large portion of the weights while maintaining accuracy, and can significantly compress the model. However, this compression is achieved at the cost of irregularity in the patterns of the non-zero weights. The irregularity results in a necessity for index saving, i.e. instead of saving every weight, saving the non-zero elements in one memory space, and their indexes (addresses or pointers) in another. In addition to the requirement for the extra index memory, another drawback is the need for two inter-dependent and energy-hungry memory-read cycles to fetch every weight. The two read-memory cycles can also deteriorate the bandwidth if the pruned model is stored on a DRAM.

Structural sparsity, or coarse-grained pruning, on the other hand, is also a current area of research. This has recently been applied to convolution (CONV) layers and fully connected (FC) layers [2, 4, 12, 18]. Structural sparsity does not require indexing, avoids the two read-memory cycles, and can employ simple circuitry or existing hardware resources to locate non-zero elements. This paper is complementary to methods in [12, 18] that consider coarse-grained pruning for only CONV layers. FC layers are the subject of this paper, as they dominate the model size of DNNs such as VGG, AlexNet, LeNet and nearly every trainable matrix in other domains such as recurrent neural networks (RNNs) and Long Short-Term Memory (LSTMs). We make the following contributions in this paper:

- Propose a substitute for the FC layer referred as the CSC layer which, due to its cyclic connectivity, inherently incorporates index generation and implicitly performs zero-skipping.
- Propose an algorithm to train models with CSC layers using a bottom-up approach by incrementally increasing the parameters such as connectivity and number of synapses, to achieve a desired accuracy.
- Train AlexNet and LeNet300100 with CSC layers instead of FC layers, with total compression rates of 10× and 46× respectively and within a margin of 2% accuracy loss.
- Propose two scalable hardware designs to implement CSC layers as well as non-structurally pruned FC layers for comparison, both adapted to LeNet300100 model with equal compression rate, and fully placed-and-routed on Artix-7 FPGA and on ASIC in 65nm CMOS technology.

2 RELATED WORK

A number of recent works have investigated fine-grained [5, 6] and coarse-grained [12, 18] pruning on DNNs. In [12], vector-level and kernel-level pruning on the filters of CONV layers is shown to save the memory references by ~2× as a result of minimizing indexing, as well as model compression on par with fine-grained methods. However, authors in [12] use only fine-grained pruning for FC layers. Similarly, authors in [18] employ structural sparsity methods, such as filter-wise, channel-wise, and shape-wise pruning, and incorporate stochastic gradient descent (SGD) with adaptive alternation direction method of multipliers (ADMM). In doing so,

DAC '19, June 2–6, 2019, Las Vegas, NV, USA
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6725-7/19/06…$15.00
https://doi.org/10.1145/3316781.3317873

they achieve an increased accuracy of AlexNet with a ~3× parameter compression of CONV layers. Authors in [2] propose to train the FC layers such that the resulted weight matrix is a circulant matrix, upon which utilizing the fast Fourier transform (FFT) can significantly reduce computation complexity. By doing so, the memory footprint is decreased from O(N^2) to O(N) and the computational complexity is decreased from O(N^2) to O($NlogN$). CIRCNN [4] is a principled approach to represent weights using block-circulant matrices that borrows the method in [2] to generalize the FFT-based technique to CONV layers as well. Both [2] and [4] require utilizing the FFT and Inverse FFT in their computations. [14] proposes to convert dense weight matrices of FC layers to the Tensor Trained (TT) format [15] and, by employing TT format instead of FC layers, achieving a compression of ~7× on VGG models. [3] proposed PermDNN which is an approach to customizing sparse weight matrices with a diagonalization scheme. PermDNN targets FC layers in DNNs and LSTMs. It can structurally compress the FC layers of AlexNet by ~9× and ~18.1× with respectively full-precision and 16-bit weights.

3 PROPOSED METHOD

Throughout this paper we use italic lower case letter x to represent generator polynomials (e.g. $p(x) = 1 + x + x^2$), italic capital letters (e.g. N) for integer values, bold lower case letters (e.g. $\mathbf{x}$) for vectors, bold upper case letters (e.g. $\mathbf{W}$) for matrices, and parenthesis for the elements of a vector (e.g. $\mathbf{y}(i)$) or a matrix (e.g. $\mathbf{W}(i, j)$).

3.1 Problem Definition

Inspired by the butterfly diagrams such as in the radix-2 FFT, in this section, we seek sparsely connected (SC) layers composed of a unidirectional sequence of layers: an Input layer, $L - 1$ layers in between referred to as *support* layers, and an Output layer. We denote the first (Input) and the last (Output) layers by capitalizing their first letters. All the consecutive layers in the diagram are connected to each other via edges (synapses). We call this diagram *homogeneous* if the size of every layer is N nodes, and if its intermediate connectivity is such that the fan-out of every node in every layer, excluding the Output layer, as well as the fan-in of every layer, excluding the Input layer, is exactly F. Therefore, the total number of edges, E, in this diagram is equal to:

$$E = NFL \tag{1}$$

For every layer in the diagram we define an adjacency matrix, $\mathbf{A}$, whose length and width is equal to the input and output sizes of the layer respectively (N in this statement) and whose elements $\mathbf{A}(i, j)$ indicate the number of edges that connect the input node i to the output node j. Therefore, NF out of N^2 elements of the adjacency matrix of every support layer in our problem statement are 1s and the rest are 0s. We then define a merit of connectivity, C, and constrain the diagram to provide C and only C paths between every pair of nodes chosen arbitrarily from the diagram Input and Output layers. For this homogeneous diagram, one can show:

$$F^L = NC \tag{2}$$

The objective of homogeneity is to provide a basis where every arbitrary node in the diagram is equally exploited, every arbitrary pair of nodes from Input and Output layers are as equally connected, the flux through the support layers is fairly equal, and the rank of the transforming matrix has the potential to remain full. This foundation will be proposed as an overlay and replacement for FC layers, and is defined such that a FC layer of size N-by-N is concluded as a special case of which ($F = N$, $L = 1$, and $C = 1$). Combining the equations (1) and (2), a logarithmic relationship between the number of edges and the size of the layers is deduced:

$$E = NFlog_F(NC) \tag{3}$$

Equations (3) is the pivot of this work. E is the number of edges that holds the non-zero elements of the L layers and governs the number of Multiply-Accumulate (MAC) operations in the diagram. As an example, given $F = 2$ and $C = 1$, equations (2) and (3) infer $L = log_2 N$ and $E = 2Nlog_2 N$ which is the case in radix-2 butterfly diagrams. Figs. 1-A, 1-B and 1-C respectively depict a FC layer and two examples of SC diagrams.

3.1.1 *Input/Output Adjustment.*

In order to replace a FC of Input size N_I and Output size N_O with a SC that has a different value for N, we tile and truncate only the Input and Output layers of the SC to match them to those of the FC layer. To adjust the Input layer to an input vector of size N_I we remove rows from the bottom of the adjacency matrix if $N_I < N$ (truncation), or we recursively copy from the first consecutive rows of the adjacency matrix and add them to its bottom if $N_I > N$ (tiling) until its number of rows is equal to N_I. For the Output layer to be adjusted to an output of size N_O, we manipulate the columns of the last adjacency matrix similarly. By doing so, the connectivity between the adjusted Input and Output layers still remains C, but the fan-in of the layer following the adjusted Input and the fan-out of the layer preceding the adjusted Output layers deviate from F. For a SC with parameters N, F, L, C, and adjusted Input and Output sizes of respectively N_I and N_O the number of edges are as:

$$E_{adj} = NF(L - 2) + N_I F + N_O F \tag{4}$$

3.1.2 *Compression Rate.*

Since the adjusted SC diagram with E_{adj} synapses is going to substitute a FC layer with N_I Input nodes, N_O Output nodes, and $N_I N_O$ synapses, the compression rate, denoted by γ, is equal to:

$$\gamma = \frac{N_I N_O}{E_{adj}} \tag{5}$$

This quantity should clearly be less than 1. It can also be considered as the average number of synapses from the SC that contribute to forming every synapse from the equivalent FC. If the SC is homogeneous, each of its synapses equally contributes to the formation of every synapse from the equivalent FC layer. For a FC, $\gamma = 1$, indicating no compression and that all connections are independent. For a homogeneous SC, $\gamma = \frac{N}{Flog_F(NC)}$. For SC with adjusted I/O, the largest value for γ is given by the smallest E_{adj}, which is derived for $L = 2$ and $F = 1$, that results in $\gamma_{most} = (N_I^{-1} + N_O^{-1})^{-1}$ and is equivalant to a rank one FC layer.

3.2 Solution to The Problem Statement

There is no unique solution for structures which meet the above problem statement. The butterfly diagrams used in the FFT give one set of solutions. In every radix-2 butterfly diagram $F = 2$, $C = 1$, and $L = log_2 N$. We show that circulant matrices generated from cyclic codes give another set of solutions for the adjacency matrices of SC layers which satisfy equation (2). From here on, we call these diagrams cyclic sparsely connected (CSC) layers. Suppose the

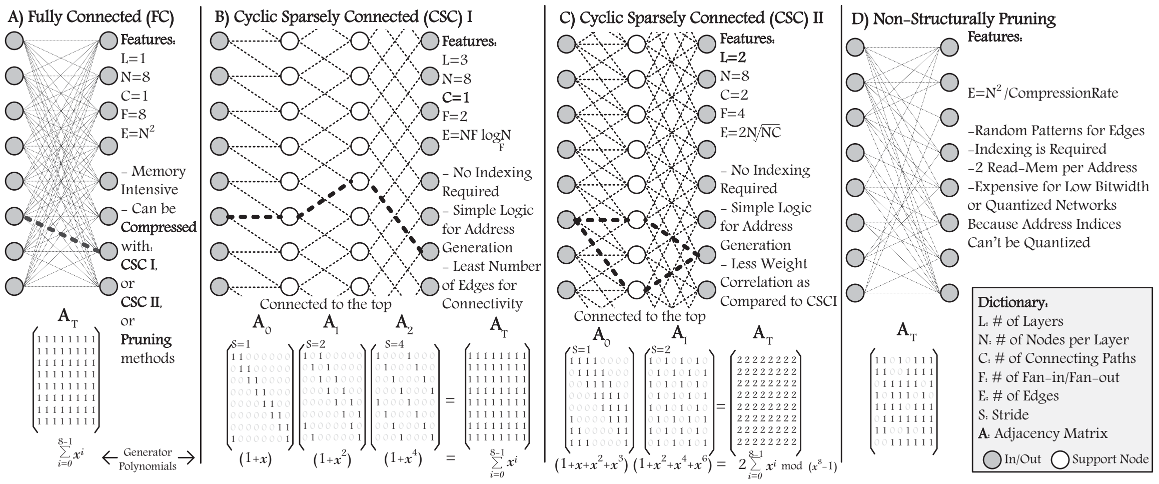


Figure 1: Example of a FC layer and 3 candidates to replace it. In each sub-figure, the adjacency matrices represent the connections, and are generated by generator polynomials. A) A non-pruned FC layer. B) The proposed cyclic sparsely connected layers type I, with a connectivity of one. C) The proposed cyclic sparsely connected layers type II, with two sub-layers. D) A representation of the pruned FC layer with irregular connectivity

adjacency matrix $\mathbf{A}_i$ ($0 \leq i \leq L-1$) of every support layer i in our CSC layers has a generator polynomial $p_i(x)$ that has F terms and can generate a cyclic adjacency matrix of block length N. It can be shown that the product of the L adjacency matrices attributed to the L layers is a matrix, which we call $\mathbf{A}_T$. $\mathbf{A}_T(i,j)$ represents the number of paths between the Input node i and Output node j of the CSC layers. In the problem statement, the connectivity should be equal to C for all arbitrary pairs of Input and Output nodes, thus all the elements in the resulting product matrix $\mathbf{A}_T$ are equal to C. In other words, $\mathbf{A}_T = C\mathbf{J}$ where $\mathbf{J}$ is an all-one matrix with size N-by-N. The all-C matrix $\mathbf{A}_T$ can be attributed to another generator polynomial $p_T(x) = C\sum_{i=0}^{N-1} x^i$. The generator polynomial (different from the characteristic polynomial of a matrix) constructs a cyclic matrix as follows: the first row of the matrix corresponds to the coefficients of the polynomial–represented as big-endian in this paper–and then, every next row is a cyclic right shift of its previous row. The cyclic adjacency matrices in this paper are not ideals as in ideal cyclic codes, and might have non-distinctive rows. We provide two different factorization sets of $p_T(x)$.

3.2.1 *CSCI: Connectivity Equal to One.*

If $C = 1$ and F, L and N are such that $N = F^L$, then $p_T(x) = \sum_{i=0}^{N-1} x^i$ can be factorized as:

$$\begin{cases} \sum_{i=0}^{N-1} x^i &= \prod_{i=0}^{L-1} p_i(x) \\ p_i(x) &= \sum_{j=0}^{F-1} x^{S_i . j} \qquad , S_i = F^i \end{cases} \tag{6}$$

By assigning $p_i(x)$ as the generator polynomial of support layer i, the CSC layers are completely described. S_i is the stride value and indicates the distance between elements of value 1s in the first row of the adjacency matrix of layer i. Fig 1-B shows a CSCI diagram with 3 layers, and generator polynomial $p_i(x) = \sum_{j=0}^{2-1} x^{2^i j}$ for the layer i. For $F = 2$ and $C = 1$, the number of edges as well as the number MAC operations are equal to $2N\log_2 N$, which makes the computation complexity $O(N\log_2 N)$. We evaluate LeNet300100 on MNIST by replacing FC layers with CSCI.

3.2.2 *CSCII: Layers Equal to Two.*

For homogeneous diagrams, the compression rate γ is smaller for less number of layers, and the contributing synapses are less correlated to build up the equivalent FC layer. If $L = 2$ and F, C and N are such that $NC = F^2$, then $p_T(x) = \sum_{i=0}^{N-1} x^i$ can be factorized as:

$$\begin{cases} C\sum_{i=0}^{N-1} x^i &= p_0(x).p_1(x) \qquad mod(x^N - 1) \\ p_0(x) &= \sum_{i=0}^{F-1} x^{S_0 . i} \qquad , S_0 = 1 \\ p_1(x) &= \sum_{i=0}^{F-1} x^{S_1 . i} \qquad , S_1 = \frac{F}{C} \end{cases} \tag{7}$$

By assigning $p_0(x)$ and $p_1(x)$ respectively to the first and second layers, the CSC layers are completely described. For $L = 2$, $\gamma = \frac{N}{2\sqrt{NC}}$ and $E = 2N\sqrt{NC}$. For $C > \frac{N}{4}$ the γ tends to less than 1 that results in redundancy. For $L = 2$ and $C = 1$ the number of MAC operations are equal to $2N\sqrt{N}$ that makes the computation complexity $O(N\sqrt{N})$. Fig 1-C shows a CSCII diagram with $C = 2$, and generator polynomial $p_i(x) = \sum_{j=0}^{4-1} x^{2^i j}$ for layer i. We evaluate AlexNet on ImageNet with FC layers replaced with CSCII layers.

3.3 Algorithm: substituting FC with CSC

We propose substituting each FC layer that has learnable weights such as DNNs and RNNs with CSC layers with or without linear activation at each support layer and with adjusted Input and Output sizes to conform to those of the FC layer.

Algorithm 1 shows a bottom-up procedure of replacing FC with CSC layers. In summary, the model with a FC layer is trained first and a reference accuracy λ_{FC} is obtained. Then, the FC is replaced with adjusted CSC layers, starting from the most compressed CSC and increasing in size up to that of the original FC layer. A stopping criteria ϵ which is the tolerable accuracy loss is defined (2% in this paper). In each experiment, if λ_{FC} is ϵ greater than the λ_{CSC}, the procedure is terminated, and the CSC model is accepted. As depicted in Fig. 2, we use algorithm 1 to train LeNet300100 and AlexNet with FC layers respectively replaced with CSCI and CSCII layers, with 50 epochs and employing SGD optimizer. For AlexNet

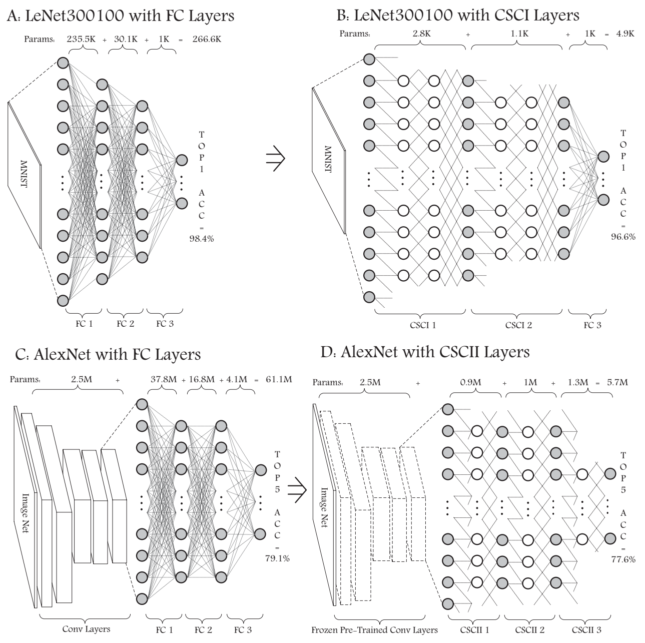

A: LeNet300100 with FC Layers
Params: 235.5K + 30.1K + 1K = 266.6K
MNIST
TOP1 ACC = 98.4%
FC 1
FC 2
FC 3
B: LeNet300100 with CSCI Layers
Params: 2.8K + 1.1K + 1K = 4.9K
MNIST
TOP1 ACC = 96.6%
CSCI 1
CSCI 2
FC 3
C: AlexNet with FC Layers
Params: 2.5M + 37.8M + 16.8M + 4.1M = 61.1M
Image Net
TOP5 ACC = 79.1%
Conv Layers
FC 1
FC 2
FC 3
D: AlexNet with CSCII Layers
Params: 2.5M + 0.9M + 1M + 1.3M = 5.7M
Image Net
TOP5 ACC = 77.6%
Frozen Pre-Trained Conv Layers
CSCII 1
CSCII 2
CSCII 3

Figure 2: Configurations of LeNet300100 and AlexNet with either FC layers or CSC layers. For AlexNet, we use pre-trained weights for the CONV layers and have them frozen when training on CSC configs

the pre-trained weights obtained from the baseline model are frozen for the CONV layers and only the dense layers are re-trained.

After training the CSC layers embedded in the DNN, every weight matrix $\mathbf{W}_i$ from layer i has non-zero weights located at corresponding non-zero elements of its adjacency matrix $\mathbf{A}_i$. The weighted CSC layers, which have substituted one FC layer, transform the Input to the Output linearly, given linear activation for support layers, and can be composed into an equivalent FC layer with $\mathbf{W}_T = \prod_{i=0}^{L-1} \mathbf{W}_i$. It's axiomatic that an arbitrary FC layer with a weight matrix $\mathbf{W}_T$ can't be losslessly decomposed into CSC layers that have fewer synapses in total. However, since training doesn't usually produce a concrete solution for $\mathbf{W}_T$, we use the back-propagation algorithm to learn good weights for the CSC layers. Now, if the $\mathbf{W}$ of every support layer with parameters N, F, S, and size of N-by-N is redefined and reformed such that it contains only the non-zero entries (i.e. only the F non-zero terms corresponding to those of the matrix $\mathbf{A}$), then the new compact matrix, denoted by $\hat{\mathbf{W}}$, has size N-by-F. When multiplied with a vector $\mathbf{x}$ of size N, an output vector $\mathbf{y}$ of size N is produced, whose elements $\mathbf{y}(i)$ can be calculated by:

$$\mathbf{y}(i) = \sum_{j=0}^{F-1} \hat{\mathbf{W}}(i,j).\mathbf{x}((i + jS) \bmod N) \tag{8}$$

We refer to the equation (8) as the circulant product between the cyclic matrix $\mathbf{W}$ (with parameter S as the stride and the compact form of $\hat{\mathbf{W}}$) and the vector $\mathbf{x}$. This equation inherently incorporates index generation and implicitly performs zero-skipping. If the Input or the Output layers are adjusted to N_I or N_O which are not equal to N, then the truncation/tiling conditions of the corresponding adjusted matrix $\mathbf{W}$ as explained in the section 3.1.1 are required.

Algorithm 1 Training NN with CSC instead of FC layers

```
1:  Inputs:
2:  FC layer with parameters $N_I$, $N_O$, and a validation accuracy $\lambda_{FC}$ on a given dataset. A criteria $\epsilon$ to terminate the procedure. A function $q(\gamma)$ to propose $N, C, F, L$ given a desired compression rate. Any standard back-propagation training algorithm $train(model)$. And a decrementing function $\gamma \longleftarrow decrement(\gamma)$.
3:  Outputs:
4:  CSC with parameters $N, C, F, L$, and validation accuracy $\lambda_{CSC}$
5:  procedure BOTTOM-UP-TRAINING(model)
6:      $\gamma \longleftarrow (N_I^{-1} + N_O^{-1})^{-1}$          ▷ The largest compression, $\gamma_{most}$
7:      while $\lambda_{FC} - \lambda_{CSC} > \epsilon$ do
8:          $N, C, F, L \longleftarrow q(\gamma)$          ▷ Propose a CSC$|N_I, N_O, \gamma$
9:          $\lambda_{CSC} \longleftarrow train(model|N, C, F, L)$
10:         $\gamma \longleftarrow decrement(\gamma)$          ▷ Bounded by $\gamma_{least} = 1$
11:     end while
12:     return $N, C, F, L, \lambda_{CSC}$
13: end procedure
```

Table 1: Compressing LeNet300100 by means of replacing FC with CSCI layers compared to related pruning methods

Layer	Params. (Baseline)	Pruning [6]	Pruning [5]	**CSCI (This work)**
Index Memory Required?		yes	yes	**no**
FC1	236K	8%	1.8%	**1.5%**
FC2	30K	9%	1.8%	**4.4%**
FC3	1K	26%	5.5%	**100%**
Total	267K	8% (12×)	1.8% (56×)	**2.2% (46×)**
Top-1 Acc.	98.4%	98.4%	98.0	**97.2%**

Table 2: Compressing AlexNet by means of replacing FC with CSCII layers compared to related pruning methods

Layer	Params. (Baseline)	Pruning [6]	Pruning [12]	**CSCII (This work)**
Index Memory Required?		yes	yes	**no**
All CONVs	2.5M	36%	25%	**100% (NA)**
FC1	38M	9%	7%	**2%**
FC2	17M	9%	7%	**6%**
FC3	4M	25%	18%	**33%**
Total	61M	11% (9×)	8% (12×)	**10% (10×)**
Top-5 Acc.	79.1%	80.2%	80.4%	**77.6%**

4 TRAINING EXPERIMENTS

We used PyTorch framework to train LeNet300100 and AlexNet with CSC layers using the procedure described in Algorithm 1. Fig. 3 shows the impact of replacing FC layers with CSCI and the bottom-up method of increasing the layers and weights for LeNet300100. Fig. 4 shows the impact of the bottom-up method on the top-5 accuracy of AlexNet throughout training epochs. Tables 1 and 2 show the compression and the accuracy of the selected CSC configurations for the two DNNs in this work and compare them with related non-structural pruning methods from the literature. It is noteworthy that in all non-structural pruning methods there exists another implicit memory space for storing non-zero weight indexes. Thus, if every non-zero weight requires an index with the same number of bits, then the actual compression in non-structural pruning methods is approximately half the reported rate.

5 HARDWARE IMPLEMENTATION

Two hardware designs that implement multilayer perceptrons (MLPs) with either CSC layers or pruned FC layers are introduced in this section. For simplicity, only the blocks that perform matrix-vector

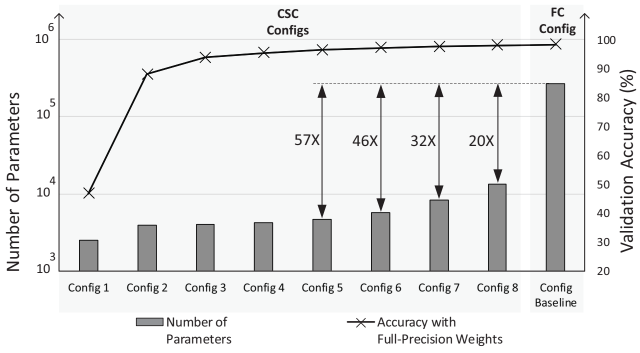
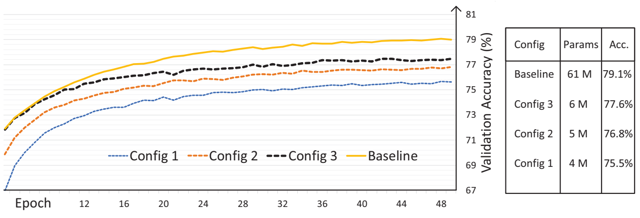


Figure 3: Impact of the bottom-up method of replacing FC with CSC layers on the accuracy of LeNet300100 trained with full-precision weights. In each config, FC1 and FC2 are replaced with two CSCI diagrams, CSCI1 and CSCI2, and the last layer remains as FC. For Config 1, CSC layers are one-node (rank 1) support layers. For Config i ($i > 1$) CSCI1 has parameters $C_1 = 1$, $F_1 = 2$, $L_1 = i + 1$, $N_1 = 2^{i+1}$, and CSCI2 has parameters $C_2 = 1$, $F_2 = 2$, $L_2 = i$, $N_2 = 2^i$. In total, each Config i ($i > 1$) has $E_T = 2^i(6i - 8) + 3968$ trainable parameters



Config	Params	Acc.
Baseline	61 M	79.1%
Config 3	6 M	77.6%
Config 2	5 M	76.8%
Config 1	4 M	75.5%

Figure 4: Impact of the bottom-up method of replacing FC with CSC layers on the accuracy of AlexNet vs epoch, trained with full-precision weights. In all configs, FC1, FC2 and FC3 are replaced with CSCII diagrams, CSCII1, CSCII2, and CSCII3, respectively with support layers of size 4096, 4096 and 1024. In Config 1, $C_1 = 1$, $C_2 = 1$, $C_3 = 4$. In Config 2, $C_1 = 1$, $C_2 = 4$, $C_3 = 16$. In Config 3, $C_1 = 1$, $C_2 = 4$ and $C_3 = 64$.

multiplications are shown in the designs, and the ReLU units and other control logic are omitted. We will use the LeNet300100, which is a 3-layer MLP, to test both compression methods.

Fig. 5-top depicts the block diagram of the CSC matrix-vector multiplication processor which implements the equation (8) in hardware. The schematic is adapted to accommodate the LeNet300100 with CSCI layers that has a model size of ~6KB (Config 6) and uses one-byte precision for weights and activations. The schematic has three memory blocks: Weight, Data and Output. The non-zero weights are stored in row-major order in the Weight memory. The read values of the Weight and Data memory units are multiplied and accumulated once per clock cycle per PE in the MAC units. The Address Generator Unit generates the indexes for the weight matrix and the input data using two counters based on the equation (8), and is updated with the parameters and the starting address (Layer Offset) of the layer-under-process. The State Machine updates the Address Generator with the layer's parameters, controls the conditions of tiled/truncated layers, and switches the task between the Data and Output memory units alternately.

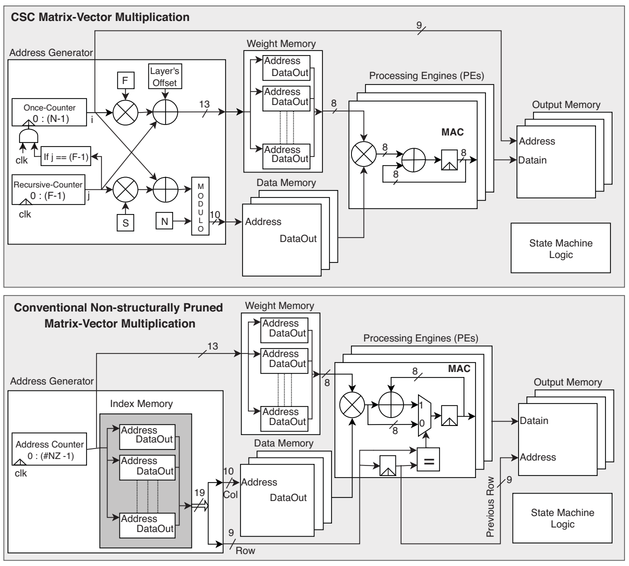


Figure 5: Schematic for the CSC architecture (top) and conventional non-structurally pruned architecture processor (bottom), which requires the highlighted index memory. Both adapted to accommodate compressed LeNet300100.

Fig. 5-bottom implements the same logic as in the Fig. 5-top, but incorporates Index memory and logic to handle the indexing scheme in non-structurally pruned MLPs. If LeNet300100 is compressed with a non-structural pruning method producing ~6K non-zero parameters, then it will require a 19 bit index per non-zero weight, 10 bits of which are column-pointers and 9 bits of row-pointers to the non-zero weights of the pruned **W**, thus requiring an extra ~6K×19-bit index memory. The column-pointers address Data memory, while the row-pointers address Output memory. The row-pointers are compared with their previous read value to determine the next MAC operation, then the accumulation is stored at the Output memory. The State Machine controls processing of each layer of the MLP. Similar to the CSC hardware design, this design also performs 1 MAC per clock cycle per PE.

The two hardware designs can be configured to employ an arbitrary number of PEs to speed up the computation and to improve energy efficiency. For each PE, the Data and Output memory units along with the logic are replicated while the Weight memory (and Index memory in the conventional architecture) is equally divided among the PEs. Output memory synchronization is performed after computing each layer to unify the divided output data in the Output memory and prepare them as the input data for the next layer.

5.1 FPGA Implementation Results

The two hardware designs for LeNet300100 are implemented using Verilog HDL, with 1, 4 and 8 PEs. Each configuration is synthesized and placed-and-routed using Xilinx Vivado tools on the low-power Xilinx Artix-7-15T FPGA which has sufficient on-chip Block RAMs (BRAMs) to store the model weights as well as the input and intermediate data. Table 3 summarizes the implementation results for 1, 4 and 8PEs at clock frequency of 150 MHz and indicates that the dynamic power as well as the BRAM usage of the CSC processor

Table 3: Artix-7 FPGA implementation results of the CSC and conventional non-structurally pruned networks for LeNet300100 at clock frequency of 150 MHz.

Case Studies	CSC (This work)			Non-Structurally Pruned		
# of PEs	1	4	8	1	4	8
BRAM	3	6	12	7	12	20
# of slices	133	443	874	146	416	812
Latency (us)	50.7	14.7	8.7	50.7	14.7	8.7
Throughput(Kilo label/s)	19.7	68.2	114.4	19.7	68.2	114.4
Dynamic Power (mW)	8	27	52	24	57	98
Total Power (mW)	79	98	123	94	127	168
Classification Energy (uJ)	4.00	1.44	1.08	4.76	1.86	1.47

Table 4: Post place-and-route results of the processors with 1 PE for CSC and conventional non-structurally pruned architectures for LeNet300100 on ASIC CMOS 65nm, 1.1v at 1 GHz. The "Total" column includes the logic power as well.

Case Studies	CSC (This work)					Non-Structurally Pruned				
	Weight Memory	Index Memory	Data Memory	Output Memory	**Total**	Weight Memory	Index Memory	Data Memory	Output Memory	Total
Height (mm)	0.36	0	0.10	0.71	**0.36**	0.36	0.36	0.10	0.71	0.36
Width (mm)	0.19	0	0.12	0.12	**0.31**	0.19	0.37	0.12	0.12	0.68
Area (mm2)	0.07	0	0.01	0.08	**0.10**	0.07	0.13	0.01	0.08	0.24
Utilization (%)	-	-	-	-	**97.2**	-	-	-	-	95.6
Power (mW)	1.1	0	0.34	0.17	**2.15**	1.1	1.68	0.35	0.17	4.07

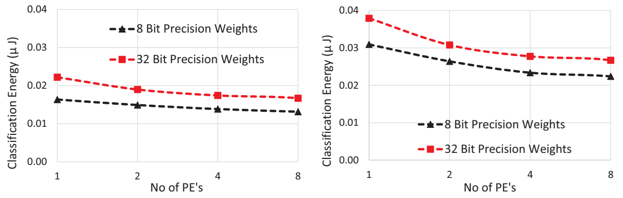


Figure 6: Impact of number of PEs for implemented in ASIC 65nm, 1.1v at 1 GHz on energy for CSC (left) and conventional non-structurally pruned (right) for LeNet300100

on the FPGA is ∼2× less than the conventional non-structurally pruned processor architectures.

5.2 ASIC Implementation Results

Both designs were synthesized with different number of PEs using RC Compiler, and were post place-and-routed with the 65nm TSMC CMOS standard cell library using Encounter SOC from Cadence. Fig. 7 shows the ASIC layouts for the single PE CSC and conventional non-structurally pruned architectures. Table 4 contains more details including the chip power, area and utilization for the two single PE designs. Fig. 6-(left) and Fig. 6-(right) show the classification energy curves vs the number of PEs for the CSC processor and the conventional non-structurally pruned architecture processor for LeNet300100 with 8-bit and 32-bit precision weights. The two curves indicate that each configuration of the CSC architecture consumes ∼2× less energy than its corresponding conventional non-structurally pruned architecture.

6 CONCLUSION

Inspired by the radix-2 FFT butterfly diagrams, we introduced cyclic sparsely connected (CSC) layers, a new architecture that is composed of a few sequential layers whose memory usage and MAC computation is $O(NlogN)$ and whose structural connectivity can be adjusted to form a suitable replacement for FC layers, which are intrinsically memory intensive in DNNs. We also propose a bottom-up method of replacing FC layers with CSC layers. Due to its structural nature, the CSC layer eliminates the need for indexing

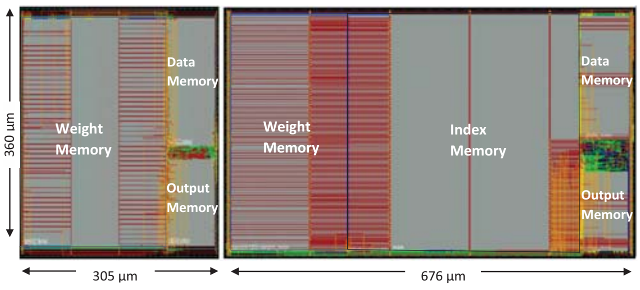


Figure 7: The post-layout views of CSC (left) and conventional non-structurally pruned (right) architectures (1PE) for LeNet300100, in 65nm TSMC CMOS tech. The key difference is the large index memory required in the latter design.

the non-zero weights. Our experimental results show that by replacing FC with CSC layers in LeNet300100 and AlexNet, the models can be compressed on par with non-structural pruning methods for full-precision weights. For comparison, we implemented a scalable parallel hardware architecture for the CSC and conventional non-structurally pruned layers both on Artix-7 FPGA and ASIC 65nm CMOS. Our results show that CSC design outperforms the conventional by ∼2× both in energy and area.

7 ACKNOWLEDGMENT

This research was supported by the NSF CAREER award #1652703.

REFERENCES

[1] Tahmid Abtahi et al. 2018. Accelerating convolutional neural network with fft on embedded hardware. *IEEE Transactions on VLSI Systems* (2018).
[2] Yu Cheng et al. 2015. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedingof IEEE ICV.*
[3] Chunhua Deng et al. 2018. PermDNN: Efficient Compressed DNN Architecture with Permuted Diagonal Matrices. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO).* IEEE.
[4] Caiwen Ding et al. 2017. C ir cnn: accelerating and compressing deep neural networks using block-circulant weight matrices. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture.* ACM.
[5] Yiwen Guo et al. 2016. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems.*
[6] Song Han et al. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems.* 1135–1143.
[7] M. Hosseini et al. 2018. MPT: Multiple Parallel Tempering for High-Throughput MCMC Samplers. In *IEEE International System-on-Chip Conference (SOCC).* IEEE.
[8] M. Hosseini et al. 2019. Minimizing Classification Energy of Binarized Neural Network Inference for Wearable Devices. *arXiv preprint arXiv:1903.11381* (2019).
[9] Ali Jafari et al. 2018. BiNMAC: Binarized neural network manycore accelerator. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI.* ACM.
[10] Ali Jafari et al. 2018. Sensornet: A scalable and low-power deep convolutional neural network for multimodal data classification. *IEEE Transactions on Circuits and Systems I: Regular Papers* (2018).
[11] HM Makrani et al. 2019. XPPE: cross-platform performance estimation of hardware accelerators using machine learning. In *Proceedings of the 24th ASP DAC.*
[12] Huizi Mao et al. 2017. Exploring the regularity of sparse structure in convolutional neural networks. *arXiv preprint arXiv:1705.08922* (2017).
[13] K. Neshatpour et al. 2018. ICNN: An iterative implementation of convolutional neural networks to enable energy and computational complexity aware dynamic approximation. In *DATE'18 Conference Exhibition.* IEEE.
[14] Alexander Novikov et al. 2015. Tensorizing neural networks. In *Advances in Neural Information Processing Systems.*
[15] Ivan V Oseledets. 2011. Tensor-train decomposition. *SIAM Journal on Scientific Computing* (2011).
[16] H. Sayadi et al. 2019. 2SMaRT: A Two-Stage Machine Learning-Based Approach for Run-Time Specialized Hardware-Assisted Malware Detection. In *DATE'19 Conference Exhibition.*
[17] C. Shea et al. 2018. SCALENet: a SCalable Low power AccELerator for real-time embedded deep neural Networks. In *ACM Proceedings of the 28th Edition of the Great Lakes Symposium on VLSI (GLSVLSI).* ACM.
[18] Tianyun Zhang et al. 2018. Adam-admm: A unified, systematic framework of structured weight pruning for dnns. *arXiv preprint arXiv:1807.11091* (2018).