Dynamic Voltage and Frequency Scaling to Improve Energy-Efficiency of Hardware Accelerators

Siqin Liu and Avinash Karanth
School of Electrical Engineering and Computer Science
Ohio University Athens, OH, 45701
Email:ls847719@ohio.edu and karanth@ohio.edu

***Abstract*—Neural networks (NNs) have been used in a wide variety of artificial intelligence (AI) applications, including speech recognition, image recognition, automatic robotics, and games. State-of-the-art NNs provide high prediction accuracy at the expense of massive computation that involves large model parameters which consume substantial energy. Though sparse NNs have emerged to reduce the computation and storage overhead, existing specialized DNN accelerators cannot maximize the energy savings when exploiting both dynamic and static sparsity, especially for irregular NNs. In this paper, we propose a dynamic voltage and frequency scaling (DVFS) based hardware accelerator that effectively exploits the dynamic and static sparsity of NNs with dynamic voltage/frequency (V/F) scaling and power gating techniques to reduce both static and dynamic power. To explore the efficiency of DVFS implementation at different granularities, we evaluate both coarse-grained and fine-grained DVFS implementation with different design trade-offs. Further, our proposed DVFS model predicts the dynamic computation workloads as well as V/F pairs to be supplied to processing elements (PEs) in the hardware intelligently through pre-trained weight vectors. The machine learning based prediction algorithm is deployed to improve the DVFS mode selection accuracy. Our simulation results on AlextNet, VGG16, and ResNet50 show that we can achieve an average dynamic energy savings of 59-66% and an average static power reduction of 69-80% compared to the baseline.**

***Index Terms*—Dynamic Frequency and Voltage Scaling (DVFS), Hardware Accelerator, Machine Learning, Neural Networks.**

I. Introduction

Many modern artificial intelligence (AI) applications such as speech recognition, computer vision, autonomous cars, disease detection, complex games, and many more employ Deep Neural Networks (DNNs) [1]. DNNs can achieve high precision in accuracy in many of these areas, even beyond human intelligence. However, this outstanding performance is achieved by imposing massive computation and storage requirements which poses a significant challenge while running on conventional CPU or GPU platforms. To address this problem, recent studies have resorted to building specialized hardware accelerators to augment the computation requirements of DNNs (Eg. ShiDianNao [2], DaDiaNao [3], Eyeriss [4] and others [5], [6]). Most prior designs (i) evaluate the computation complexity and data movement patterns in the representative layers, such as convolution or fully connected layers, (ii) propose hardware accelerators with processing element (PE) arrays for parallel computing, and (iii) exploit data reuse opportunities to reduce energy cost for data movement.

As DNN model sizes continue to increase for higher accuracy, researchers explore reducing the number of weights by pruning the networks without much loss in accuracy. However, there exist trade-offs between computational efficiency and pruning of NN models. Pruning the network at a pixel-level granularity of synapse would break the data matrix regularity with some of the values pruned to be zero in a non-deterministic pattern. The irregularity caused by fine-grain pruning hampers sparsity processing dedicated accelerators such as SCNN [7], Cnvlutin [8], Cambricon-X [9] and Sigma [10]. Static synapse sparsity refers to the removed or zero-valued weights after the pruning. Dynamic neuron sparsity arises from the non-linearity activation function. Addressing both sparsity forms in the accelerator design is challenging. To circumvent the irregularity, researchers study structured pruning techniques with coarse-grain granularity, claiming that coarse-grained pruning can achieve comparable sparsity ratio as unstructured pruning given no loss of accuracy. However, the regular pruning patterns for DNNs in a wide variety of applications are not guaranteed to be identical [11]. The decision to operate either at a coarse-grain or a fine-grain pruning is a trade-off between the system complexity and maximum energy savings that can be achieved.

There has been a significant amount of research where Dynamic Voltage and Frequency Scaling (DVFS) is applied to various on-chip components including the processor, caches, memory and Network-on-Chips (NoCs). The supply voltage is decreased at low workload and any marginal loss in performance is tolerated in order to save dynamic energy. At medium to high workload, a loss in performance would lead to saturation and increased latency, and therefore, the supply voltage is proportionally increased. Prior work [12] has also shown that machine learning techniques can be applied to select the optimal voltage level through proactive predictions of future workload. On the other hand, static power is further targeted through power-gating, a technique that switches off the supply voltage to unused or lightly used on-chip components to reduce leakage current.

In this paper, we propose a dynamic voltage and frequency scaling (DVFS) based hardware accelerator that effectively exploits the dynamic and static sparsity of neural networks (NN) with dynamic voltage/frequency (V/F) scaling and power-

gating techniques to reduce both static and dynamic power. Instead of processing sparse data in a regular manner with dedicated hardware, we track the runtime data pattern of both weights and activations during the NNs implementation and predict the workload assigned for each PE for the next epoch. Based on prediction results, DVFS technique is applied to the PE array to exploit the sparsity feature in order to save energy by dynamically scaling the voltages and frequencies. Our scheme can support both dense and sparse neural networks due to the runtime data adaptability rather than a specific pattern of the synaptic weights. Furthermore, based on the spatial feature of DNN accelerator, we explore fine-grain and coarse-grain DVFS schemes. We define our fine-grained DVFS scheme to be implemented for each processing element, while coarse-grain is for a row/column of PEs in the array. The machine learning based prediction algorithm is deployed to improve the DVFS mode selection accuracy. Our simulation results on AlextNet, VGG16, and ResNet50 show that we can achieve an average dynamic energy savings of 59-66% and an average static power reduction of 69-80% compared to the baseline. In Section III, comparison and trade-offs between both schemes are discussed. The major contributions of this work are as follow:

- **Applying DVFS to Hardware Accelerator:** We evaluated in detail how convolution neural networks (CNNs) can be mapped to accelerators. Using this analysis, we applied power-gating to PEs during periods of low computation to save static power and dynamically scale voltage and frequency (V/F) during periods of medium to high computation to reduce dynamic energy consumption.
- **Machine Learning based Prediction:** We propose a machine learning based algorithm to predict the DVFS states using a few neural network features to maximize energy efficiency and minimize the latency penalty. Offline training and feature sharing enable minimal overhead and improved performance.
- **Exploration of Applying DVFS at different granularities:** The proposed DVFS-based accelerator design is implemented as coarse-grained and fine-grained schemes to evaluate the trade-offs in energy savings and the area overhead. Fine-grained scheme manages each PE in independent domain, achieves higher V/F state prediction accuracy and energy efficiency, but incurs larger area overhead, while coarse-grained scheme provides an alternative solution with relatively lower energy saving, but better scalability.

II. Proposed Architecture

A. Accelerator Hardware and Micro-architecture

Architecture Overview: The proposed architecture is shown in Figure 1. The architecture consists of an on-chip global buffer (GB), data dispatcher (DD), DVFS predictor (DP), dynamic voltage and frequency generator (DVFG) and 16 $\times$ 16 PE array connected with a mesh-based Network-on-Chip (NoC). All neurons and synapses are fetched from

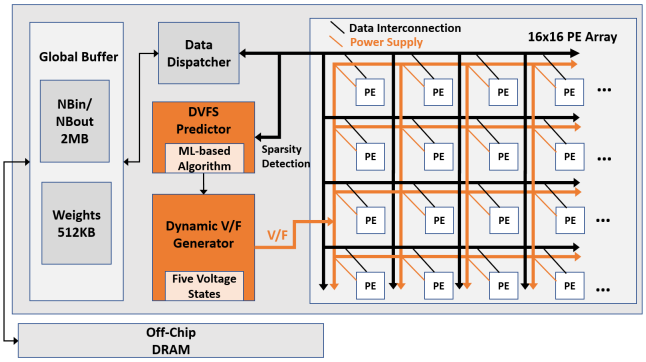

Fig. 1. Proposed architecture that handles both dense and sparse NNs. The proposed architecture consists of on-chip global buffer (GB), data dispatcher (DD), DVFS predictor (DP), dynamic voltage and frequency generator (DVFG) and the 16 $\times$ 16 PE array.

off-chip DRAM and all overhead associated with DRAM in terms of energy and latency are included in the evaluation.

DP and DVFG are the key components that implement DVFS functionality. The predictor, DP, receives data to be transmitted to each PE via the data dispatcher (DD) module. Runtime data are collected and analyzed periodically for a specific time window, i.e., an epoch. During each epoch, the predictor counts up the number of non-zero valued data, including weights and input maps. Based on the collected amount of sparse workloads, the DVFG resets the voltage and frequency of the PE for the next epoch. The workloads of MAC operations with zero-valued operands are calculated as follows:

$$SparseWorkload = (F^2 - Sw)O^2 - F(F - Sa) \quad (1)$$

where F $\times$ F is the size of weights, O $\times$ O is the size of output activations, Sw is the total number of zero-valued weights, and Sa is the total number of zero-valued input maps. The workload percentage of the PE for one epoch is the ratio of computed sparse workload to the theoretical amount of MAC operations with no zero-valued operand.

DVFG provides the voltage and frequency to each PE as well as the interconnection network. To support enough granularity of voltage scaling and effective energy saving contrast, the DVFG is designed to offer five voltage and frequency (V/F) states. Four V/F states are selected from 1.2 V as the highest working state to 0.8 V as the lowest. Power-gating is regarded as the fifth V/F state, which consumes no dynamic and static energy when the computation load predicted for the next epoch is below a certain threshold.

PE Micro-architecture: For the PE micro-architecture, the machine learning based proactive DVFS is implemented by a local sparsity detector (SD), cooperating with the global dynamic voltage/frequency generator (DVFG). Besides sparsity units, the microarchitecture consists of three register files, which stores the sparse weights, input maps, or activations after the ReLu function of previous layers and the partial sums

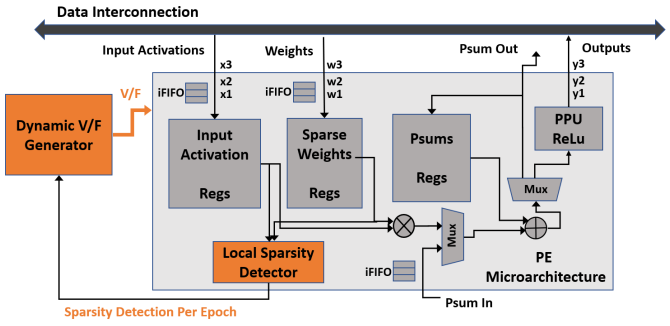

Fig. 2. Proposed PE micro-architecture that consists of the Sparsity Detector (SD), Dynamic Voltage/Frequency Generator (DVFG), register file, an arithmetic logic unit (ALU), which performs the multiplication and accumulation functionality, and a post-processing unit, which applies the activation function (non-linearity layer) on the output neuron.

(Psums); an arithmetic logic unit (ALU), which performs the multiplication and accumulation; and a post-processing unit, which applies the activation function (non-linearity ReLu) on the output neuron. The SD monitors both input activation and weight registers to ensure that the sparse activations from the rectified linear activation unit and the sparse weights from the compressed neural networks can be both exploited. The SD accumulates the number of non-zero valued weights and input activations each cycle and outputs the value of the sparse computation workload for the current epoch using Equation 1.

B. DVFS Models

In this subsection, we describe the selection of DVFS models with respect to the observed workload. Each DVFS model consists of one inactive state (power-gated) and four active states. In an inactive state, the voltage supply to the specific PE and its outgoing interconnection is reduced to 0 V with no clock applied to the PE. V/F pair for one PE may switch in every epoch, which is at least 200 ns duration. This interval constraint ensures sufficient time for low-dropout (LDO) voltage regulators to generate different voltages. According to [13], the worst case of power-gating delay is 8.8 ns, and voltage switching delay is 6.9 ns. As the power-gated duration is substantially greater than the wake-up delay, we do not require a wake-up state between inactive and active states. PE in an active state can operate in any one of the four different voltage levels. The V/F pairs used in the DVFS models are {0.8V/2.75ns, 1.0V/2.25ns, 1.1V/2ns, 1.2V/1.8ns} which are numbered as V/F modes 2-5 with power-gated state as mode 1. These voltage levels are commonly configured in accelerators and selected based on the principle that when they operate in different modes, the voltage and frequency are proportionally decreased/increased. We set up the thresholds in Table I based on the runtime data distribution when training the datasets. We used different percentages of workload to determine the bins for the five V/F configurations. When larger workloads, e.g. >50%, V/F pair 1.2V/1.8ns with the highest voltage and frequency is selected

TABLE I
WORKLOAD DISTRIBUTION AMONG DIFFERENT MODELS AND DVFS MODES.

DVFS Models	0 V	0.8V/ 2.75ns	1.0V/ 2.25ns	1.1V/ 2ns	1.2V/ 1.8ns
Power-Gating	<20%	20-30%	30-40%	40-50%	>50%
Power-Saving	<10%	10-70%	70-80%	80-90%	>90%
Balanced	<10%	10-30%	30-55%	55-80%	>80%
Performance	<10%	10-20%	20-30%	30-40%	>40%

to provide the shortest propagation latency while incurring the highest power consumption.

To explore the optimal configuration of the DVFS model, we present four different DVFS models, i.e., Power-Gating, Power-Saving, Balanced and Performance models as shown in Table I. Power-gating model maximizes static power reduction by assigning more PEs to power-gating mode (<20%) than any other models. Power-saving model maximizes dynamic power reduction and assigns the highest portion of workloads (10-70%) to the lowest V/F mode (0.8V/2.75ns). Performance model on the contrary, operates the PEs at the highest voltage whenever the workload percentage is higher than 40% to enable the highest computation performance. Balanced model is based on a moderate strategy that evenly allocates the workloads to all V/F modes.

C. Dataflow and Walkthrough Example

Output stationary (OS) dataflow best suits the proposed architecture among other alternate dataflows such as weight stationary, row stationary or column stationary [14], because psums are accumulated inside the PE, avoiding the requirement of synchronization between other PEs at different V/F modes. However, weights and input maps are required to be buffered inside each PE when operating at different voltages and frequencies. Especially when power-gated, PE needs to buffer all the incoming weights and propagate input activations after waking up to active voltage stage in the next epoch. A further explanation is presented in the following walk-through example.

To illustrate with an example, Figure 3 shows how neurons compute and propagate within PE arrays under different epochs. To represent generality, we consider a small design with a 3×3 PE array, a convolutional layer with 3×3 kernel size, and a small patch of input activation map with 5×5 size, and 1×1 step size with no padding. We depict the flow for three cycles in two epochs. For the first epoch, all PEs are initially configured in regular voltage mode 1.0V, while in epoch #1, each PE is voltage scaled by the proposed DVFS system. For simplicity, the timing chart shows the weights, input activations and voltages for four PEs namely $PE_{0,0}$ $PE_{0,1}$ $PE_{1,0}$ and $PE_{1,1}$.

Cycle #0, epoch #0: All PEs fetch the first input activation of the current convolution window from the global buffer as outlined in red (x_{00},x_{01},x_{10} and x_{11}), while weight w_{00} is broadcast to all PEs according to the output stationary dataflow. Each PE performs the multiplication and accumula-

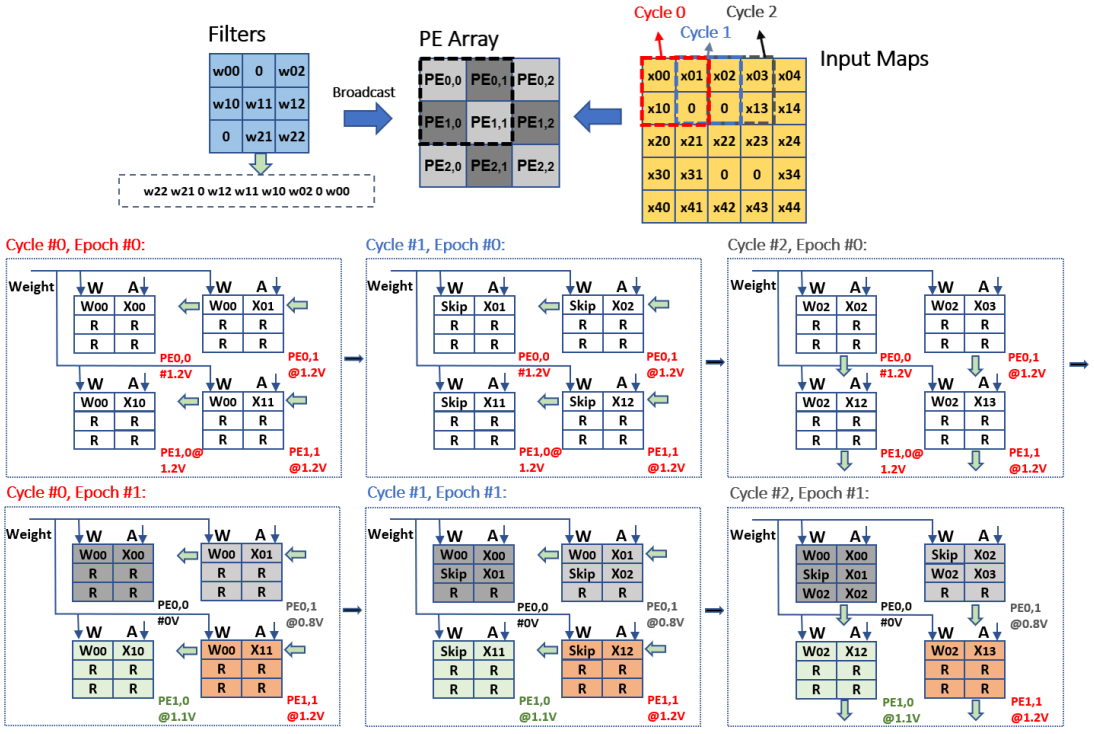

Fig. 3. Walkthrough example of mapping the convolutional layer (convolutional window: 3 × 3; step size: 1 × 1) and an PE array implementation (with 2 × 2 PEs).

tion (MAC) operation and stores the partial sum results in its local register file for further reduction. In the meanwhile, each PE transmits the current fetched activation to its neighboring PE for further reuse. The extra buffer space is reserved for DVFS modes.

Cycle #1, epoch #0: Convolution sliding window moves one step to the right and a new column of input maps (x_{02},x_{12}) are fed into the PE array. The rest of the data in the window move to the right and reused by the PEs. The next weight w_{01} is broadcast to all PEs, however, only a flag bit needs to be transmitted to save energy and buffer area due to the zero value. The MAC computation is completely skipped by the hardware to further save power.

Cycle #2, epoch #0: Similar to cycle #0, another pixel of weight w_{02} is broadcast to all PEs and a new column of input maps (x_{03},x_{13}) are fetched with other inputs reused among PEs. To maximally exploit data reuse, we switch to propagate the input maps vertically for the next cycle and the new inputs are only needed by the top row of PEs. The trade-off is the complexity of the on-chip network supporting both horizontal and vertical communication.

Cycle #0, epoch #1: $PE_{1,0}$ and $PE_{1,1}$ are operating at the higher voltage levels such as 1.1V and 1.2V respectively. $PE_{0,1}$ switches to lower voltage level 0.8V. $PE_{0,0}$ is power-gated due to few computations to save power, while the buffer inside $PE_{0,0}$ is active to store the weights that are broadcast and input maps that propagate to the power-gated PE.

Cycle #1, epoch #1: $PE_{0,0}$ and $PE_{1,1}$ function similarly as in epoch #0 as the frequency difference between 1.1V mode and 1.2V mode is not sufficient to incur a delay in cycles. $PE_{0,1}$ lags behind compared to prior epoch. The current weight w_{01} and input activation x_{02} need to be buffered until the prior weight (w_{00}) and input activation (x_{01}) are computed. $PE_{0,0}$ is in power-gated mode and the dedicated buffer is used to store the incoming weight and input.

Cycle #2, epoch #1: Buffer for $PE_{0,0}$ continues to store the data and propagate the input activations until waking up into active voltage mode for the next epoch. These light workloads are consequently merged into the next epoch, switching the PE into 1.2V mode for the highest throughput. The epoch period determines the size of the required buffer, which is a trade-off between hardware cost and prediction accuracy. Experiments are conducted to show the design space in the next section.

In fine-grained DVFS scheme, each PE is working in independent power domain, whereas the data dispatcher keeps sending the data from the global buffer at a fixed frequency to maintain the dataflow stream. Dedicated buffer is assigned to each PE for storing the weights and input activations so that each PE can execute the computation at its own frequency without dropping any input data. The buffer size is determined by the rate of data consumption and differs according to the V/F mode predicted. With a large epoch duration, the input weights and activations would accumulate in low-frequency PEs, incurring huge buffer overhead. This becomes worse in power-gating state as the incoming data is not consumed in the local PE and all the data has to be buffered for the next working epoch. On the other hand, if the epoch size is too small, the DVFS predictor has to frequently generate

the V/F state of the PE and the dynamic V/F generator frequently updates the V/F state as well, which leads to extensive dynamic energy cost. In Section III, we performed detailed experiments to study the design trade-off between the buffer size, the time interval of an epoch and the hardware overhead.

In the coarse-grained DVFS scheme, instead of controlling the voltage and frequency for each PE, a row of PEs in the array are grouped together to share one voltage domain. The workload for a row of PEs are predicted together by the DVFS predictor and the dynamic V/F generator assigns the same voltage and frequency to the entire row of PEs. Since the PEs are synchronized in one row, the buffer size is only determined by the different data consumption rates between the PEs from different rows, thus buffer overhead is smaller than that of the previously discussed fine-grained DVFS scheme. Furthermore, as shown in the walk-through example, weights are broadcast to the PE array. Data reuse can be deployed by sharing the buffer within one row of PEs. However, the input activations are unicast to PEs and should be buffered separately, which constitutes the major overhead of the shared buffer design in coarse-grained DVFS scheme.

D. Machine Learning-based DVFS States Selection

We discovered through simulation that the statistical workload assigned for each PE diverges significantly with various mapping algorithms and the input images that need to be classified. Manually approximating the function of runtime workload for each PE is impractical. To address this challenge, we deploy machine learning to learn the feature from the runtime data distribution of all the PEs through our experiments. After training through the collected datasets, the machine learning model enables us to predict the workload of current PE for the next epoch based on the learned features.

Figure 4 shows the proposed machine learning based DVFS mode prediction. A compact neural network is used to approximate the relationship between feature sets and the labelled V/F modes for the current epoch. The input layer carries the feature sets. The last is the output layer with a Softmax layer to do the classification, which predicts the V/F mode. The dataset for training the network is collected when implementing benchmark CNNs. We simulate AlextNet, VGG16 and ResNet50 and map them onto the 16×16 PE array. The feature sets are carefully crafted and tuned with machine learning algorithm such that prediction accuracy is maximized while overhead is kept to a minimum. The DVFS models are trained offline using Algorithm 1. The trained parameters are applied to the DVFS predictor in hardware to implement voltage and frequency prediction as Algorithm 2.

Algorithm 1 Training of the neural network for DVFS state prediction

Total number of processing element N
One-hot encoding of weight and activation matrix w, a
Aggregate number of non-zero valued weights and activations W, A
Number of MAC operations M
Dataset Collection:
for each $PE_i, 0 \leq i < N$ **do**
  for each epoch t, $0 \leq t < T$ **do**
    $W_t = \sum_i w_i, if(w_i \neq 0)$
    $A_t = \sum_j a_j, if(a_i \neq 0)$
    $M_t = W_t \times O^2 - F \times (F - X_t)$, F, O denotes the size of weight and output maps as Equation 1
  end for
  $W = \sum_t^T W_t$
  $A = \sum_t^T A_t$
end for

Training:
Initialize parameter vector $v = v_{l1}, v_{l2}, v_{l3}$ at random
repeat
  Create input $X = \{W, X, W_t, A_t\}$.
  Compute the input of the Softmax layer: $f(X) = relu(v_{l2} \times relu(v_{l1} \times X))$
  Compute the cross entropy: $y = crossentropy((v_{l3} \times f(X)), M_t)$
  Do backpropagation
  Update v_{l1}, v_{l2}, v_{l3}
until Convergence, $y \leq 0.01$
return v as the parameter vector for current voltage state epoch.

Algorithm 2 Implementation of DVFS prediction with pre-trained parameters

Input:
Number of Non-zero valued weights and activations ratio in current epoch W_t, A_t
Aggregate number of non-zero valued weights and activation W, A
Current DVFS state S
Output:
Voltage and Frequency state for the next epoch.
Initialize start voltage 1.2V and state $S_0 = 4$,
Load parameter vector v
for each epoch **do**
  Collect feature sets $X = \{W, X, W_t, A_t\}$ from Data Dispatcher and Sparsity Detector
  $f(X) = relu(v_{l2} \times relu(v_{l1} \times X))$
  Softmax Classification as predicted workload $t = argmax(w_{l3} \times f(X))$
  Switch voltage $= \begin{cases} power-gated, & if(t \leq 10\%) \\ 0.8V, & if(10\% \leq t < 20\%) \\ 1.0V, & if(20\% \leq t < 30\%) \\ 1.1V, & if(30\% \leq t < 40\%) \\ 1.2V, & if(t \geq 40\%) \end{cases}$
  Update state S_{t+1}
end for

III. Performance and Evaluation

A. Simulation Setup

We evaluate the our DVFS based accelerator on three representative CNNs, i.e. AlexNet [15], VGG16 [16] and ResNet50 [17] as benchmarks. All the networks are trained and evaluated on the same dataset of CIFAR-10 [18]. The energy, latency and execution time of implementing these tasks are collected and analyzed to compare the performance of the proposed DVFS models with different design strategies. The intermediate activations and trained weights as well as pruned ones are monitored every window for each PE and the workload percentages are collected and transformed into datasets for our ML-based algorithm for predicting the DVFS mode. After training and optimization, the ML-based algorithm is applied to the CIFAR-10 test dataset to validate the DVFS prediction accuracy and estimate the energy performance.

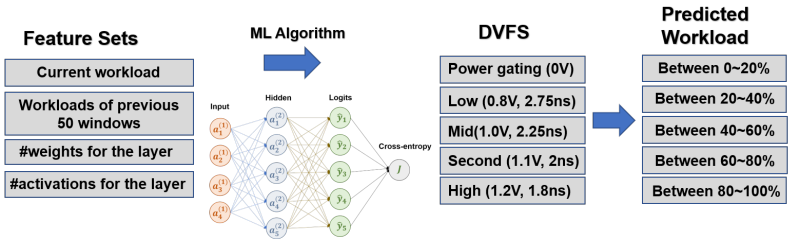

Fig. 4. Machine Learning based DVFS states prediction algorithm.

TABLE II
STATIC/DYNAMIC POWER, AREA AND TIMING OF HARDWARE MODULES UNDER THE 1.2V/1.8NS DVFS MODE.

Module	Area (um^2)	Static (uW)	Dynamic (mW)	Latency (ns)
Register	4096	13.57	1.72	0.94
Adder	145	0.63	0.08	1.29
Multiplier	2214	2.75	0.35	6.22
ReLu Activation	36.6	0.08	0.01	0.14
Sparsity Detector	38,792	7.5	0.95	2.32
Dynamic V/F Generator	15,053	9.95	1.26	1.53
DVF Predictor	57,612	38.51	5.38	10.35
Data Dispatcher	1,255	4.22	0.77	0.92
Weights Memory	357,400	53.56	10.32	33.25
Activation Memory	72,349	10.71	2.06	5.34

We use Design Compile Ultra from Synopsys [19] to compile and synthesis the RTL design to obtain the power consumption and latency of the hardware components. FreePDK45 [20], a 45nm technology node design kit, is used as the target library. To simulate the hardware metrics under the predefined V/F modes, we manually change the voltage level and frequency inside the library setup file. In Table II, area, static power, dynamic power and latency of each submodule of the accelerator are shown under 1.2V/1.8ns configuration for 45nm technology node. For other V/F pairs, the area cost remains the same. Dsent [21] is used to model the accelerator and the communication cost to obtain the dynamic and static energy. Baseline is defined as the hardware accelerator with no sparsity unit (highlighted by yellow in Figure 1). The V/F pair for baseline is fixed at 1.2V/1.8ns.

B. Simulation Results

To explain the motivation of deploying DVFS techniques to address the sparsity issues, we randomly sample successive epoch windows and capture the workload distribution in each epoch in Figure 5 when implementing benchmark neural networks on CIFAR-10 dataset. The expected V/F mode to be supplied to PEs for each epoch varies according to the workload-voltage setting in DVFS models. Figure 5 shows the experiment results and the corresponding V/F modes in DVFS Performance and Power-saving models. We conduct experiments on the benchmarks with the four DVFS models (Power-gating, Power-saving, Balanced and Performance) to explore the optimal energy efficient model and the trade-off between throughput reduction and energy saving. In the highlighted epoch in Figure 5, 56% of the workload is assigned to 1.2V voltage level in Performance model (second plot), whereas in Power-saving model (third plot), 0.8V is expected for this workload.

In Figure 6, we demonstrate the distribution of computational workload for all layers of ALextNet, VGG16 and ResNet50 respectively with the four DVFS models. Figure 6 shows the results where the workload is distributed among the five V/F modes. When a PE is assigned to a certain V/F setting, it will operate at this specific V/F for the entire epoch duration until the DP predicts a different V/F mode for the PE in the next epoch.

With the distribution of workloads for each DVFS model, we further display the breakdown of energy cost as shown in Figure 7. The value of energy cost is presented in percentage form for each DVFS model to demonstrate the importance of contributing to the total energy cost. As power is completely gated and no clock is supplied in power-gating state, the energy cost is zero and not seen in the figure. The baseline model exploits no sparsity in CNNs and operates always at the highest voltage level with the fastest frequency. It fetches one weight and input map pixel and does the MAC (multiplication and accumulation) operation each clock cycle no matter whether the weight or input map pixel is zero or not. In all networks, power-saving model achieves the lowest energy cost, because a large portion of sparse workload are processed at the lowest voltage level as shown in the bar. Contrary to the expectation, Power-gating model performs the worst. The main reason is that constant switching from power-gated to active mode leads to huge energy cost, which could be optimized in the future. Figure 8 further displays the energy cost divided into dynamic and static energy normalized to that of the baseline. Power-saving model shows the lowest dynamic energy cost in all the three neural networks with 69% to 80% average static energy savings, because the dynamic energy cost dominates the total energy cost. However, if one aims to reduce static power consumption, Balanced mode or Performance mode demonstrates better performance depends on which neural network is applied. More than 83% static power cost is reduced in Balanced model on AlexNet, making it the optimal choice if reducing static power is the highest priority. .

Figure 9 displays the results of the circuit delay and EDP to demonstrate the trade-offs between latency penalty and energy reduction. The value of delay and energy are both

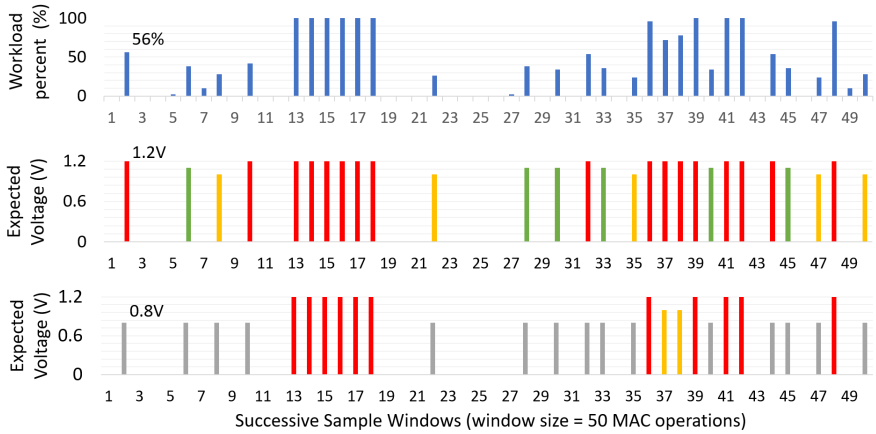

Fig. 5. Example workload percentages (first plot) when implementing AlexNet on CIFAR-10 and the expected voltage levels by DVFS Performance model (second plot) and Power-Saving model (third plot).

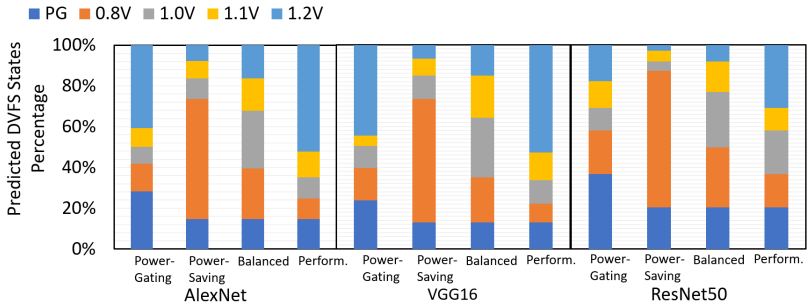

Fig. 6. Workloads breakdown of each DVFS voltage state (Power gated, 0.8V, 1.0V, 1.1V and 1.2V) for different proposed models (Power-Gating, Power-Savings, Balanced and Performance) on different neural networks (AlexNet, VGG16 and ResNet50) compared with the baseline.

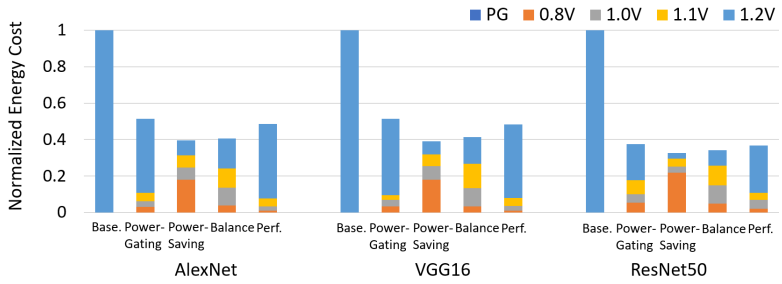

Fig. 7. Energy breakdown of the five DVFS voltage states (Power gated, 0.8V, 1.0V, 1.1V and 1.2V) for the four proposed DVFS models (Power-Gating, Power-Savings, Balanced and Performance model) on different neural networks (AlexNet, VGG16 and ResNet50), compared with the baseline.

normalized to those of the baseline. The execution delay varies under different voltage settings. We average the total delay based on the amount of workload under each voltage mode for each DVFS model. For the Power-saving model with the highest energy efficiency, the EDP is also the best in VGG16 and ResNet50 and almost the same as in Balanced model in AlexNet. The throughput reduction of 11-14% in Power-saving model is the traded-off to achieve the 60-67% savings in total energy.

One deciding factor to explore for the design space is the epoch size, which has a direct impact on the prediction accuracy and the buffer size requirement that dominates the area overhead of the DVFS scheme. We simulate the prediction accuracy for all benchmarks with different epoch sizes for fine-grained DVFS scheme and the results are shown in Figure 10. The prediction accuracy increases initially with larger epoch size, because coarse granularity of prediction provides the algorithm with more input data. However, increasing epoch size over 50 cycles does not incur further improvement in prediction accuracy. This is due to the inherent randomness of data distribution inside the neural networks. Inconsistency and signal noise in input images also contribute to the upper bound of prediction accuracy. Polynomial-16 achieves higher overall accuracy with more complex feature pre-processing.

The energy saving of the DVFS scheme comes at a critical overhead of on-chip buffer area. Figure 11 shows that on ResNet50, epoch size with 100 cycles achieves best energy efficiency for fine-grained DVFS scheme. Larger epoch size

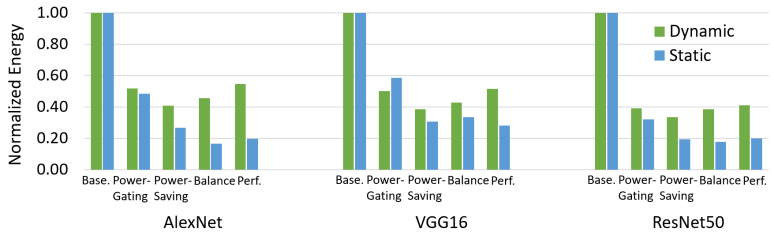

Fig. 8. Normalized dynamic and static energy cost for the baseline and the four DVFS models (Power-Gating, Power-Savings, Balanced and Performance model) on AlexNet, VGG16 and ResNet50, compared with the baseline.

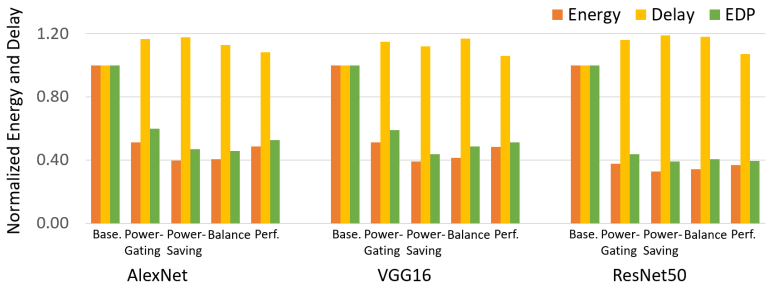

Fig. 9. Normalized energy, delay and EDP for the baseline and the four DVFS models (Power-Gating, Power-Savings, Balanced and Performance)

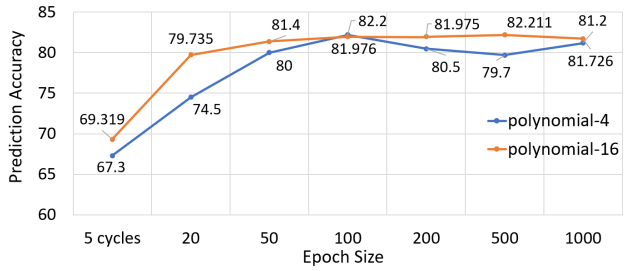

Fig. 10. Prediction accuracy with ranging epoch sizes for AlexNet, VGG16 and ResNet50.

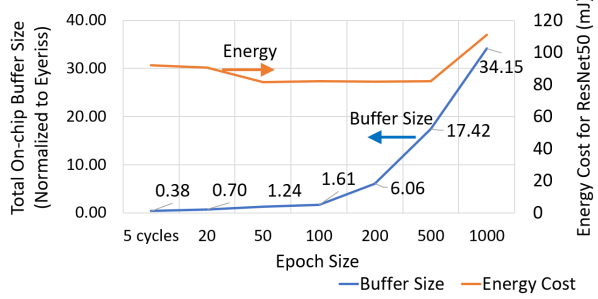

Fig. 11. Buffer size requirement with ranging epoch sizes and energy cost on ResNet50.

does not benefit prediction accuracy; in fact, with larger buffer size only energy consumption increases. Smaller epoch size induces more energy due to lower prediction accuracy. The buffer size requirement escalates with the number of cycles of an epoch, because more weights and input activations need to be buffered for asynchronous PEs under different voltage domains. In Figure 11, the value of the buffer size is normalized to that of Eyeriss [14], which contains 196 PEs and 96KB total on-chip SRAM and scaled linearly to 256 PEs for fair comparison. Epoch size with 50 cycles provides the best performance with 24% on-chip memory overhead.

Figure 12 compares the buffer size between fine-grained and coarse-grained DVFS schemes. Coarse-grained scheme proves to be an effective alternate approach with moderate buffer overheads and better scalability. At an epoch size of 200 cycles, the buffer size is reduced by more than 6 times compared to the fine-grained scheme. Although the coarse-grain scheme combines 16 PEs (one row) in one voltage domain and the weights only need to be stored once for the entire row, the improvement can not reach 16 times because input activations are stored separately for each PE in the buffer. Figure 13 shows the energy cost, delay and EDP for both schemes. For example, in AlexNet, the energy for the coarse-grained scheme is 2.25 times more than that of fine-grained scheme. The buffer saving in coarse-grained scheme is marginal in terms of energy cost, because in fine-grained scheme, each PE is controlled independently and V/F can be adjusted to the optimal level. However, the coarse-grained scheme performs better in terms of delay. For AlexNet, coarse-grained scheme achieves 18.2% delay reduction when compared to the fine-grained scheme. The combined workload distribution for the entire row of PEs reduces the probability of power-gated states, thus making it run faster than fine-grained scheme.

In Table III, we compare the proposed DVFS-based accelerator design with Eyeriss V2 [4] and Cambricon-S [22], two state-of-art sparse accelerators for DNN. For fairness, we attempt iso-resource comparison, where the peak throughput constraint is fixed to 256 GOP/s. In order to get accurate area and energy values, we re-implement eyeriss V2 and Cambricon-S in the same FreePDK45 45nm CMOS library.

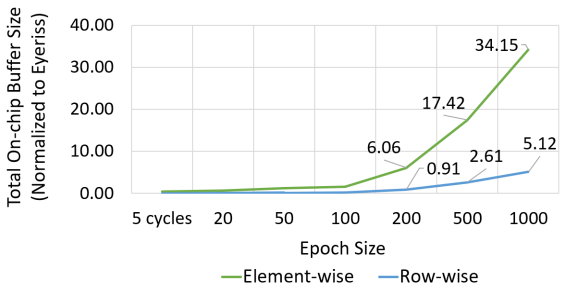

Fig. 12. Buffer size requirement with ranging epoch sizes between the fine-grained and coarse-grained DVFS Schemes.

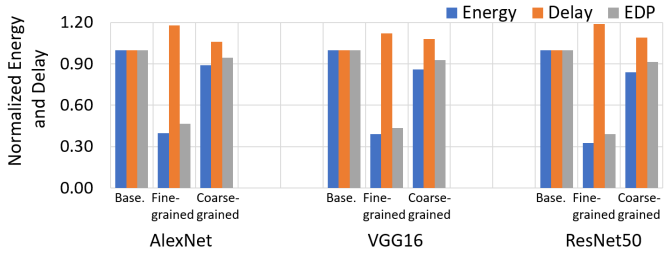

Fig. 13. Energy, delay and EDP for the fine-grained and coarse-grained DVFS Schemes, compared with the baseline.

The PE array size is 16×16 for both Cambricon-S and the proposed architecture while Eyeriss V2 only consists of 128 PEs due to the SIMD-based PE design with two MACs to retain the peak throughput constraint. The original throughput for Eyeriss V2 and Cambricon-S is 153.6 GOP/s with 200MHz and 512 GOP/s with 1GHz respectively. To match the V/F settings (from 1.2V/363 MHz to 0.8V/555 MHz in table I), we scaled up the frequency for Eyeriss-V2 and scaled down for Cambricon-S. The energy efficiency is presented as the number of images processed in a second. It must be noted that the result is lower when compared to the original design(e.g., 664.6 inference/J in Eyeriss-V2 for sparse AlexNet). The main reason is the different working frequency and process technology simulation library. The total on-chip SRAM consists of global buffer and register files in each PE. Our proposed architecture has a higher ratio of SRAM to the total area because of extra buffers that are required to synchronize data flow through the PEs under different V/F domains and to restore the weights and activations after the PE is completely power gated. The last two rows present the overheads of the specific sparsity components, which refer to the DVFS predictor, dynamic voltage and frequency generator and sparsity detector (from Figure 1 and 2). Although the fine-grained DVFS control at each PE involves higher hardware overheads, it is still more energy efficient than Cambricon-S because the energy savings brought by power-gating and low V/F pair dominates in highly sparse NNs. Moreover, the hardware cost for adapting the generic PE to sparse weights with irregularity is non-trivial as in Cambricon-S (36.8% of total area). Eyeriss V2 uses compressed sparse column (CSC) format for both on-chip processing and off-chip accesses. Only CSC decorder and encoder are implemented within the PE design, which occupies smaller portion in the PE area breakdown [4]. It is proved in Table III that Eyeriss V2 achieves the lowest area and power overheads for sparsity components when compared to Cambricon-S and our proposed DVFS work.

TABLE III
COMPARISON WITH STATE-OF-ART DESIGNS.

Items	Everiss V2	Cambricon-S	DVFS
Area Cost	2.12(mm^2)	3.24(mm^2)	2.58(mm^2)
Peak Throughput	256GOPs	256GOPs	256GOPs
Number of PEs	128	256	256
On-chip SRAM	164KB	26.5KB	528KB
Power	500.2 (mW)	433.8 (mW)	329.0 (mW)
AlexNet Energy Efficiency	346 (inference/J)	399 (inference/J)	526 (inference/J)
Sparsity Components Area	<1%	36.8%	13.4%
Sparsity Components Power	<1%	24.3%	5.8%

IV. Related Work

A. Sparsity

Researchers have found promising parameter redundancy in dense DNNs. Various efforts have been made to identify and prune the unnecessary overfitting parameters with no loss in accuracy, including algorithm-level (e.g., Deep Compression [23] and [24] and architecture-level (e.g., short-bits representation and approximate computing). Before exploiting sparsity, most DNN accelerators focus on addressing the dataflow and data reuse to improve energy efficiency. Eyeriss-v1 [25] fully discusses the dataflow in CNN layers and proposes row stationary dataflow for both neuron and synapse reuse. ShiDianNao [2] maximizes the neuron reuse and eliminates the DRAM. Although these accelerators can achieve high throughput and low energy, they can not fit with modern sparse and compressed neural networks. Therefore, sparsity-related accelerators are proposed to exploit the sparsity to skip data movement and computation of the pruned synapses with corresponding neurons. Cambricon-X [9] develops the key Index Module to select the necessary neurons based on the index of compressed synapses, but it fails to exploit dynamic neuron sparsity (DNS). The proposed DVFS based accelerator benefits from both dynamic and static sparsity regardless of irregularity from pruning techniques, thus a more efficient design.

B. DVFS

DVFS has been applied at different levels of granularity to various fields, from personal computers, laptops to data warehouse servers. The core of DVFS is the adjustment of voltage and frequency settings on an electrical component to optimize resource allotment for tasks and maximize the energy-saving [26], [27]. For DNN implementation, prior works apply the DVFS technique to trade-off performance and power. [28] exploits DVFS setting of GPU platform, monitors the server's load, and adjusts the precision to optimize the power consumption, but it is a task level and coarse DVFS

scheme, more precisely, a job scheduler. It takes no consideration of the sparsity of DNN. The DVFS technique is an intuitive approach to addressing the dynamic workloads to save energy at a very fine level. Sparse workloads based DVFS scheme has not been applied to hardware accelerators to the best of our knowledge.

V. Conclusions

This paper discusses a DVFS based technique applied to DNNs, along with a hardware accelerator architecture that efficiently implements a DVFS scheme to improve energy-efficiency. The experimental results show that the DVFS based accelerator can significantly reduce total energy cost by as much as 67% when compared to the baseline. Various DVFS models with different voltage-workload settings are used for comparative purposes and to highlight the trade-offs between the energy efficiency and DVFS model costs. Two DVFS implementation schemes (coarse-grained and fine-grained scheme) are also explored to evaluate the trade-offs between the energy saving efficiency and the area cost. Moreover, our proposed research shows how to combine the non-blocking power-gated scheme and the smart proactive DVFS state selection model to achieve energy-efficiency. We also demonstrate ways of exploring the machine learning algorithm of DVFS state prediction to further improve the performance of the DVFS model.

Acknowledgment

This research was partially supported by NSF grants CCF-1513606, CCF-1703013, and CCF-1901192. We sincerely thank the anonymous reviewers for their excellent feedback.

References

[1] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[2] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, 2015, pp. 92–104.

[3] T. Luo, S. Liu, L. Li, Y. Wang, S. Zhang, T. Chen, Z. Xu, O. Temam, and Y. Chen, "Dadiannao: A neural network supercomputer," *IEEE Transactions on Computers*, vol. 66, no. 1, pp. 73–88, 2017.

[4] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.

[5] B. Asgari, R. Hadidi, T. Krishna, H. Kim, and S. Yalamanchili, "Alrescha: A lightweight reconfigurable sparse-computation accelerator," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 249–260.

[6] N. Srivastava, H. Jin, S. Smith, H. Rong, D. Albonesi, and Z. Zhang, "Tensaurus: A versatile accelerator for mixed sparse-dense tensor computations," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 689–702.

[7] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "Scnn: An accelerator for compressed-sparse convolutional neural networks," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 27–40.

[8] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 1–13.

[9] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, "Cambricon-x: An accelerator for sparse neural networks," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016, pp. 1–12.

[10] E. Qin, A. Samajdar, H. Kwon, V. Nadella, S. Srinivasan, D. Das, B. Kaul, and T. Krishna, "Sigma: A sparse and irregular gemm accelerator with flexible interconnects for dnn training," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 58–70.

[11] J. Yu, A. Lukefahr, D. Palframan, G. Dasika, R. Das, and S. Mahlke, "Scalpel: Customizing dnn pruning to the underlying hardware parallelism," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 548–560.

[12] H. Jung and M. Pedram, "Supervised learning based power management for multicore processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 9, pp. 1395–1408, 2010.

[13] M. Clark, Y. Chen, A. Karanth, B. Ma, and A. Louri, "Dozznoc: Reducing static and dynamic energy in nocs with low-latency voltage regulators using machine learning," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020, pp. 1–11.

[14] Y. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 367–379.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[18] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[19] P. Kurup and T. Abbasi, *Logic synthesis using Synopsys®*. Springer Science & Business Media, 2012.

[20] R. Thapa, S. Ataei, and J. E. Stine, "Wip. open-source standard cell characterization process flow on 45 nm (freepdk45), 0.18 µm, 0.25 µm, 0.35 µm and 0.5 µm," in *2017 IEEE International Conference on Microelectronic Systems Education (MSE)*, 2017, pp. 5–6.

[21] C. Sun, C. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L. Peh, and V. Stojanovic, "Dsent - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, 2012, pp. 201–210.

[22] X. Zhou, Z. Du, Q. Guo, S. Liu, C. Liu, C. Wang, X. Zhou, L. Li, T. Chen, and Y. Chen, "Cambricon-s: Addressing irregularity in sparse neural networks through a cooperative software/hardware approach," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2018, pp. 15–28.

[23] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *International Conference on Learning Representations*, 2016.

[24] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally, "Exploring the regularity of sparse structure in convolutional neural networks," *arXiv preprint arXiv:1705.08922*, 2017.

[25] Y. Chen, J. Emer, and V. Sze, "Using dataflow to optimize energy efficiency of deep neural network accelerators," *IEEE Micro*, vol. 37, no. 3, pp. 12–21, 2017.

[26] R. Jain, P. R. Panda, and S. Subramoney, "Machine learned machines: Adaptive co-optimization of caches, cores, and on-chip network," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 253–256.

[27] X. Chen, Z. Xu, H. Kim, P. V. Gratz, J. Hu, M. Kishinevsky, U. Ogras, and R. Ayoub, "Dynamic voltage and frequency scaling for shared resources in multicore processor designs," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2013, pp. 1–7.

[28] S. M. Nabavinejad, H. Hafez-Kolahi, and S. Reda, "Coordinated dvfs and precision control for deep neural networks," *IEEE Computer Architecture Letters*, vol. 18, no. 2, pp. 136–140, 2019.