SmartDeal: Remodeling Deep Network Weights for Efficient Inference and Training

Xiaohan Chen, Yang Zhao, Yue Wang, Pengfei Xu, Haoran You, Chaojian Li, Yonggan Fu, Yingyan Lin, *Member, IEEE*, and Zhangyang Wang, *Senior Member, IEEE*

***Abstract*— The record-breaking performance of deep neural networks (DNNs) comes with heavy parameter budgets, which leads to external dynamic random access memory (DRAM) for storage. The prohibitive energy of DRAM accesses makes it nontrivial for DNN deployment on resource-constrained devices, calling for minimizing the movements of weights and data in order to improve the energy efficiency. Driven by this critical bottleneck, we present *SmartDeal*, a hardware-friendly algorithm framework to trade higher-cost memory storage/access for lower-cost computation, in order to aggressively boost the storage and energy efficiency, for both DNN inference and training. The core technique of *SmartDeal* is a novel DNN weight matrix decomposition framework with respective structural constraints on each matrix factor, carefully crafted to unleash the hardware-aware efficiency potential. Specifically, we decompose each weight tensor as the product of a small basis matrix and a large structurally sparse coefficient matrix whose nonzero elements are readily quantized to the power-of-2. The resulting sparse and readily quantized DNNs enjoy greatly reduced energy consumption in data movement as well as weight storage, while incurring minimal overhead to recover the original weights thanks to the required sparse bit-operations and cost-favorable computations. Beyond inference, we take another leap to embrace energy-efficient training, by introducing several customized techniques to address the unique roadblocks arising in training while preserving the *SmartDeal* structures. We also design a dedicated hardware accelerator to fully utilize the new weight structure to improve the real energy efficiency and latency performance. We conduct experiments on both vision and language tasks, with nine models, four datasets, and three settings (inference-only, adaptation, and fine-tuning). Our extensive results show that 1) being applied to inference, *SmartDeal* achieves up to 2.44× improvement in energy efficiency as evaluated using real hardware implementations and 2) being applied to training, *SmartDeal* can lead to 10.56× and 4.48× reduction in the storage and the training energy cost, respectively, with usually negligible accuracy loss, compared to state-of-the-art training baselines. Our source codes are available at: https://github.com/VITA-Group/SmartDeal.**

***Index Terms*— Data movement, deep network training, efficient machine learning, hardware accelerator.**

Manuscript received January 1, 2021; revised August 16, 2021 and November 18, 2021; accepted December 4, 2021. This work was supported by the National Science Foundation (NSF) through the Real-Time Machine Learning Program under Award 1937592 and Award 2053279. *(Xiaohan Chen and Yang Zhao contributed equally to this work.) (Corresponding author: Xiaohan Chen.)*

Xiaohan Chen and Zhangyang Wang are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA (e-mail: xiaohan.chen@utexas.edu; atlaswang@utexas.edu).

Yang Zhao, Yue Wang, Pengfei Xu, Haoran You, Chaojian Li, Yonggan Fu, and Yingyan Lin are with the Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA (e-mail: zy34@rice.edu; yw68@rice.edu; px5@rice.edu; hy34@rice.edu; cl114@rice.edu; yf22@rice.edu; yingyan.lin@rice.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2021.3138056.

Digital Object Identifier 10.1109/TNNLS.2021.3138056

I. Introduction

A. Background and Core Idea

THE performance breakthrough of deep neural networks (DNNs) motivates a growing demand to bring DNNs into storage- and energy-constrained edge devices, such as mobile phones, wearables, and IoT sensors, using domain-specific accelerators.

However, the excellent performance comes at a heavy parameter cost, which needs external dynamic random access memory (DRAM) for storage. The prohibitive energy of DRAM accesses makes DNN deployment on resource-constrained devices nontrivial. Take the accelerator in [1] as an example: >95% of the energy is consumed by DRAM. Thus, it is crucial to minimize weights and data movements in order to improve the energy efficiency of DNNs.

We present a holistic algorithm-hardware co-design framework, called *SmartDeal*, to aggressively reduce both the energy consumption of data movement and the storage for weights: the two major limiting factors for DNN on-device deployment. Our underlying philosophy is to seek a "smart deal": *trading the more "expensive" memory storage/access for "cheaper" computation, to eliminate the dominant data movement cost*. Our technical contributions highlight three major aspects: 1) inference algorithm: whose core is a novel *SmartDeal* weight representation and a unified optimization framework; 2) inference hardware accelerator, which is co-designed to maximize the benefit of our inference algorithm; and 3) training algorithm, which extends the *SmartDeal* benefits to energy-efficient training in nontrivial ways. We will go through each of them with more details in Section I-B.

This article significantly extends our previous conference version [2]. First, our prior work [2] only covers the inference algorithm and its hardware accelerator, while this work proposes *SmartDeal*-compatible training techniques for the first time, exploring the new horizon of energy-efficient training. We also extensively benchmark the efficient training performance, in two common edge-based training settings (adaptation and fine-tuning), using a training-specific hardware accelerator. Second, we offer a comprehensive suite of ablation studies to carefully investigate the impact of each proposed component in the *SmartDeal* weight decomposition. Besides, more datasets and application scenarios are included in experiments, such as a language-modeling dataset.

B. Overview of Major Technical Contributions

We start by introducing *SmartDeal* for inference, whose core is a new weight representation: a layer-wise weight matrix is decomposed as the product of a small *basis matrix* and a large *coefficient matrix* (see Fig. 1). We simultaneously enforce two important properties on the latter coefficient matrix.

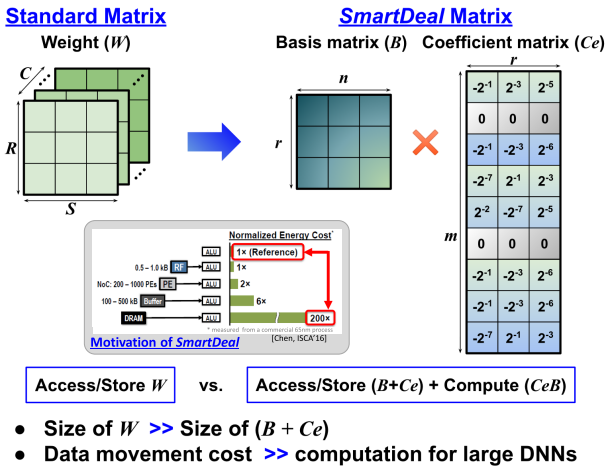


Fig. 1. Overview of the proposed weight restructuring in *SmartDeal*.

1) *(Structurally) Sparse:* Most elements are zero. Besides element-wise sparsity, we also exploit structured sparsity [3] for more hardware-friendliness.
2) *Specially Quantized:* The nonzero elements take only power-of-2 values, which not only have compact bit representations, but also (more importantly) turn the dot-product multiplication with the nonzero elements into much lower cost shift-and-add operations.

It is worth emphasizing that we never claim to have invented any new weight factorization, or quantization, or structured sparsity algorithm in this article. The *essential novelty* of *SmartDeal* lies in its unique angle and motivation, as well as the unified optimization framework dedicated to achieving our goal. The *core theme* of *SmartDeal* is trading higher cost memory storage/access for lower cost computation: this is beyond simply compressing weights to save storage or FlOPs. Our weight restructure could be interpreted as an innovative, well-motivated integration of *sparsification (or pruning)*, *factorization*, and *quantization* on DNN weights that can be solved under one unified optimization algorithm.

To fully unleash *SmartDeal* algorithm's potential, we further develop a dedicated DNN inference accelerator that takes advantage of the much reduced weight storage and readily quantized weights resulting from the algorithm to enhance hardware acceleration performance. Experiments show that the proposed accelerator outperforms state-of-the-art DNN inference accelerators in terms of acceleration energy efficiency and latency by up to 6.7× and 19.2×, respectively.

We then take another big leap to extend *SmartDeal* from inference to DNN training. The current practice of edge-based training typically starts from a pretrained and preloaded model, and then continues tuning the model with more data collected from the same training domain (denoted as *fine-tuning*), or from a different new domain (denoted as *adaptation*) due to customization or personalization [4]–[6]. Although fine-tuning or adaptation costs much less compared to training from scratch, their large resource consumption stands at odds with the limited computing and energy resources at the edge [7]. To enable *SmartDeal* for energy-efficient on-device training, we aim to preserve the *SmartDeal* restructured weight form during training. To do so, we have to cope with two roadblocks in the resulting optimization.

TABLE I
UNIT ENERGY COST PER 8-bit EXTRACTED FROM A COMMERCIAL 28 nm TECHNOLOGY

	DRAM	SRAM	MAC	multiplier	adder
Energy (pJ/8bit)	100	1.36−2.45	0.143	0.124	0.019

1) The weights, and therefore their sparse coefficient matrices, keep changing during training: that was known to be very hardware-unfriendly. To deal with this challenge, we perform a *SmartDeal* decomposition on the pretrained weight initializations, and then maintain the (structured) sparsity map (i.e., the locations of nonzero elements) unchanged during training, while their magnitudes can be updated. This greatly saves memory access and energy overhead with little impact on achievable performance.[1]
2) The other challenge arises from our enforced structure that nonzero coefficients could only take discrete values (power-of-2). The gradient descent cannot be directly applied to the discrete domain; meanwhile it is highly inefficient to keep a copy of continuous/higher-precision latent weight for updating. Instead, we design a special and lightweight update rule for the coefficient matrix that turns floating-point additions into quantization bucket switches, to be decided by gradient signs [8].

We evaluate *SmartDeal* on one state-of-the-art training accelerator [9] with necessary modifications to demonstrate the generality where *SmartDeal* achieves up to 4.48× improvement in energy efficiency over state-of-the-art competitors for training, at the negligible accuracy losses.

II. BACKGROUND AND RELATED WORK

A. A Motivating Example for SmartDeal

Table I summarizes the unit energy cost of accessing different-level memories with different storage capacities and computing an MAC/multiplication/addition (the main computation operation in DNNs) designed in a commercial 28-nm CMOS technology. We can see that the unit energy cost of memory accesses is much higher (≥9.5×) than that of the corresponding MAC computation. Therefore, it is promising in terms of more efficient acceleration if we can potentially enforce higher order of weight structures to more aggressively trade higher cost memory accesses for lower cost computations, motivating our *SmartDeal* idea. That is, the resulting higher structures in DNN weights' decomposed matrices, e.g., C_e in Fig. 1, will enable much reduced memory accesses at a cost of more computation operations (i.e., shift-and-add operations in our design), as compared to the vanilla networks.

B. Basics of Deep Neural Networks

Modern DNNs usually consist of a cascade of multiple convolutional (CONV), pooling, and fully connected (FC) layers through which the inputs are progressively processed.

[1]While it is not directly applicable to training from random scratch, such setting is unlikely in resource-constrained training.

The CONV and FC layers can be described as

$$\begin{aligned}&\boldsymbol{O}[c_o][e][f]\\&=\sigma\left(\sum_{c_i,k_r,k_s}^{C,R,S}\boldsymbol{W}[c_o][c_i][k_r][k_s]\cdot\boldsymbol{I}[c_i][eU+k_r][fU+k_s]\right.\\&\qquad\left.+\boldsymbol{B}[c_i]\right),\quad 0\le c_o<M,\ 0\le e<E,\ 0\le f<F\end{aligned}\tag{1}$$

where $\boldsymbol{W}$, $\boldsymbol{I}$, $\boldsymbol{O}$, and $\boldsymbol{B}$ denote the weights, input activations, output activations, and biases, respectively. In the CONV layers, C and M, E and F, R and S, and U stand for the number of input and output channels, the size of input and output feature maps, and the size of weight filters, and stride, respectively; while in the FC layers, C and M represent the number of input and output neurons, respectively; with σ denoting the activation function, e.g., a ReLU function ($\mathrm{ReLU}(x) = \max(x, 0)$). The pooling layers reduce the dimension of feature maps by average or max pooling. The recently emerging compact DNNs (e.g., MobileNet [10] and EfficientNet [11]) introduce depth-wise CONV layers and squeeze-and-excite layers, which can be expressed in the above description as well [12].

C. Overview of Efficient Deep Learning

To reduce the large quantity of weight parameters, numerous DNN compression techniques have been proposed to shrink the weight redundancy and accelerate the inference, including matrix decomposition [13], [14], quantization [15]–[17], pruning [3], [18], [19], knowledge distillation [17], [20], [21], and dynamic inference [22]–[25]. Combinations of two techniques have also been studied, e.g., decomposition with pruning [26], distillation with quantization [27], [28], and distillation with pruning [29]. Some latest works [30], [31] start to combine and jointly optimize three compression ideas, but for different motivations and applications (e.g., compressing robust models, and GANs). To our best knowledge, *SmartDeal* features a new joint formulation that combines the three ideas of weight pruning, matrix decomposition, and power-of-2 quantization: a unification never being considered by peer works. Also differently from [30], [31], *SmartDeal* is the first to associate with hardware co-design, especially with the unique goal to reduce the memory/storage-access costs. It is also the first of its kind to extend to efficient training.

Besides combining and jointly optimizing multiple compression techniques, another direction of work focuses on improving the performance of quantized models by utilizing more flexible quantization schemes. Mixed-precision training methods [15], [32], [33] quantize weights, activations, and gradients to different precisions instead of using single precision for the whole model. Quantization-aware training (QAT) co-optimizes the quantization schemes and the model parameters [34], [35]. *SmartDeal* is orthogonal to mixed precision and QAT methods, which can be easily integrated into *SmartDeal* to find its optimal precision configuration.

Moreover, current DNNs are typically trained in resource-rich servers or data centers. Nevertheless, we see a growing necessity for the model to continue learning and updating itself *in situ*, such as for user personalization, or incremental/lifelong learning in open-ended environments. On-device local learning can avoid communication forth-and-back between data centers and devices, reducing system latency and enhancing privacy protection. Despite a number of recent efforts [4]–[6], [36], limited progress has been witnessed so far in this field, partially due to the even larger gap between the training resource demands and the available on-device resources.

D. Compression-Aware DNN Accelerators

In general, the three typical compression approaches, weight *factorization*, data *quantization*, and weight *sparsification*, have been exploited in DNN accelerator's design to boost energy efficiency. Huang *et al.* [14] demonstrate DNNs with tensorized factorization using ASIC. For the weight sparsification accelerators, [19], [37], [38] have been proposed. Quantization is widely used by inference accelerators [14], [19], [37], [38]. In comparison, our proposed *SmartDeal* inference accelerator also unifies three techniques to simultaneously shrink the memory footprint and simplify the computations when recovering the weight matrix during runtime.

III. *SmartDeal* for Efficient Inference

In this section, we introduce the weight decomposition structure in *SmartDeal*, as can be naturally applied to *feed-forward inference* to reduce energy and time consumption of data movement as well as storage.

A. Problem Formulation

Given a weight matrix $W \in \mathbb{R}^{m\times n}$, we seek to decompose it as the product of a coefficient matrix $C_e \in \mathbb{R}^{m\times r}$ and a basis matrix $B \in \mathbb{R}^{r\times n}$, where $r \le \min\{m, n\}$, such that

$$W \approx C_e B. \tag{2}$$

In practice, n is usually set to be very small (thus small B), and $m \gg n$ (thus much larger C_e). Here we assume a 2-D weight matrix in a FC layer as an example for the simplicity of notation. We will later show that *SmartDeal* algorithm can be easily applied to weight tensors in CONV layers.

In addition to suppressing the reconstruction error (often defined as $||W - C_eB||_F^2$), we expect the decomposed matrix factors to display more favorable structures for compression/acceleration. For the much larger C_e, we enforce the following two structures simultaneously: 1) C_e needs to be highly sparse (a typical goal of pruning) and 2) the nonzero elements in C_e are exactly the powers of 2, so that their bit representations can be very compact and their involved multiplications to rebuild the original weights from B and C_e are simplified into extremely cheap shift-and-add operations. As a result, instead of storing the whole weight matrix, the new structure requires storing only a very small B; and a large, yet highly sparse and readily quantized C_e. We call this process *SmartDeal* (**SD**) decomposition and the resulting $\{C_e, B\}$ pair the *SD form* of W.

SmartDeal decomposition hence can be written as the following constrained optimization:

$$\begin{aligned}&\arg\min_{C_e,B}\ \|W - C_eB\|_F^2\\&\text{s.t.}\ \sum\nolimits_j \|C_e[:, j]\|_0 \le S_c\\&\qquad C_e[i, j] \in \Omega_P\quad \forall i, j,\ |P| \le N_p\end{aligned}\tag{3}$$

where $\Omega_P := \{0, \pm 2^p | p \in P\}$ with P being a chosen integer set that includes the possible degrees of the power-of-2 numbers and has cardinality no more than N_p, i.e.,

$|P| \leq N_p$. In (3), S_c controls the total number of nonzero elements in C_e, i.e., the sparsity, while N_p controls the bit-width required to represent an element in C_e. An innovative assumption of SD is to require nonzero elements in C_e to take one of a few *predefined, specifically picked discrete values*. That is different from previous compression using weight clustering, whose quantized values are adaptively learned from data [39], [40]. This special design is to facilitate 1) compact storage, and more crucially, and 2) extremely cheap weight reconstruction using only shift-and-add operations—the latter cannot be fulfilled by other arbitrary quantization.

B. SmartDeal Decomposition Algorithm

Solving (3) is nontrivial due to the nonconvex and integer set constraints. We propose a coordinate descent-type algorithm that iterates between objective fitting and feasible set projection, as outlined in Algorithm 1, and the explanation of three key steps to be iterated are discussed follows. Here, $\delta(C_e)$ is the difference between two iterates of C_e, tol is a small number indicating the convergence of the algorithm, and max_iter is the max number of iterations.

Algorithm 1 *SmartDeal* Decomposition Algorithm

1: Initialize C_e and B; k = 0
2: While $\|\delta(C_e)\| \geq tol$ or $k < max\_iter$:
3: **Step 1:** Quantizing C_e to powers of 2;
4: **Step 2:** Fitting B and C_e;
5: **Step 3:** Promoting (structured) sparsify C_e;
6: k = k + 1;
7: Re-quantize C_e and re-fit B.

We empirically find that simply initializing $C_e = W$ and $B = I$ can produce robust and good performing decomposition results. Hence, we use this initialization in all experiments. After the initialization, we iteratively perform the following three steps to gradually obtain better decompositions.

Step 1 (Quantizing C_e): The quantization step projects the nonzero elements in C_e to Ω_P. Specifically, we will first normalize each column in C_e to have a unit norm in order to avoid scale ambiguity. We will then round each nonzero element to its nearest power-of-two value. We define $\delta(C_e)$ to be the quantization difference of C_e.

Step 2 (Fitting B and C_e): We will first fit B by solving $\arg\min_B \|W - C_e B\|_F^2$, and then fit C_e by solving $\arg\min_{C_e} \|W - C_e B\|_F^2$. When fitting either one, the other is fixed to be its current updated value. The step simply deals with two unconstrained least squares.

Step 3 (Sparsifying C_e): We then prune the nonzero elements in C_e with smallest magnitudes to promote more sparsity. In practice, we use hard thresholds for element-wise and structured sparsity to zero out small magnitudes in C_e for implementation convenience. When promoting element-wise sparsity in C_e, as sparsity level S_c usually needs to be manually adjusted for each layer, we instead use a heuristic threshold θ to zero out elements. The structured sparsity is discussed in detail in Section III-D.

After sufficiently iterating between the above three steps (i.e., quantization, fitting and sparsification), we conclude the iterations by requantizing the nonzero elements in C_e to ensure $C_e[i, j] \in \Omega_P$ and then refitting B with the updated C_e. An example of how the C_e and B matrices evolve along the iterations is given in Appendix A in the Supplementary Material.

C. Applying the SmartDeal Algorithm to DNNs

1) SmartDeal Algorithm as Post-Processing: The selection of the dimensions of the coefficient matrix C_e and the basis matrix B is a design knob of *SmartDeal* for trading-off the achieved *compression rate* and model accuracy, i.e., a smaller r (see notations in Section III-A) favors a higher compression rate yet might cause a higher accuracy loss. Note that r is equal to the rank of the basis matrix B, i.e., $r = n$ when B is a full matrix, otherwise $r \leq n$. To minimize the memory storage, we set the basis matrix $B \in \mathbb{R}^{r \times n}$ to be small. In practice, we choose $n = R = S$ with $R \times S$ being the CONV kernel size. Since n is small, we choose $r = n = S$ too. We next discuss applying the proposed algorithm to the FC and CONV layers.

1) *SmartDeal on FC Layers:* Consider a fully connected layer $W \in R^{M \times C}$. We reshape each row of W into a new matrix $\tilde{W}_i \in R^{C/S \times S}$, and then apply *SmartDeal* algorithm. Specifically, zeros are padded if C is not divisible by S, and *SmartDeal* algorithm is applied to $\tilde{W}_i$, where $i = 1, \ldots, M$. When $C \gg S$, the reconstruction error might tend to be large due to the imbalanced dimensions. We alleviate it by slicing $\tilde{W}_i$ into smaller matrices along the first dimension.
2) *SmartDeal on CONV Layers:* Consider a convolutional layer W in the shape (M, C, R, S). *Case 1:* $R = S > 1$. We reshape the M filters in W into matrices of shape $(S \times C, S)$, on which *SmartDeal* algorithm is applied. The matrices can be sliced into smaller matrices along the first dimension if $S \times C \gg S$. *Case 2:* $R = S = 1$. The weight is reshaped into a shape of (M, C) and then is treated the same as an FC layer.

The above procedures are easily parallelized along the axis of the output channels for acceleration.

Applying *SmartDeal* algorithm to a VGG19 network[2] pretrained on the CIFAR-10 [41], with $\theta = 4 \times 10^{-3}$, tol $= 10^{-10}$ (introduced in Section III-B), and a maximum iteration of 30, the accuracy drop in the validation set is as small as 3.21% with an overall compression rate of over 10× without retraining after the decomposition. The *overall compression rate* of a network is defined as the ratio between the total number of bits to store the weights (including the coefficient matrix C_e, basis matrix B, and encoding overhead) and the number of bits to store the original FP32 weights.

2) Enhancing Accuracy With Retraining: After a DNN has been post-processed using *SmartDeal* algorithm, a retraining step can be used to remedy the accuracy drop. As the unregularized retraining will break the desired property of coefficient matrix C_e, we take an empirical approach to alternate between 1) retraining the DNN for one epoch and 2) applying *SmartDeal* algorithm to ensure the C_e structure. The default iteration number is 50 for CIFAR-10 [41] and 25 for ImageNet [42]. As shown in the ablation experiments in VI-B, the alternating retraining process improves the accuracy while maintaining the favorable weight structure. More analytic solutions will be explored in future work, e.g., incorporating *SmartDeal* algorithm as a regularization term [40].

[2]https://github.com/chengyangfu/pytorch-vgg-cifar10

3) Time Complexity of SmartDeal: *SmartDeal* algorithm is efficient in terms of running time with fully parallelized implementation. In practice, running one pass of the parallelized SmartDeal algorithm takes less than 30 s for VGG-19 and ResNet-18 networks and less than 2 min for ResNet-50. In combination with the alternating retraining approach mentioned above, the computational cost of *SmartDeal* is negligible compared to the cost for training on data. The above running time is measured on Intel Xeon Platinum 8168 CPU platform (we only implement *SmartDeal* on CPU currently).

D. SmartDeal With Structured Sparsity

Customized accelerators for sparse models can utilize the sparsity to reduce the associated computations and memory accesses [37], [38]. However, the irregularity in element-wise/unstructured sparse models prevents hardware from fully leveraging reduction. Coarse-grained sparsity brings more regular sparsity pattern, making it easier for hardware acceleration [43]. Hereby, when we design the dedicated hardware accelerator for *SmartDeal*, we introduce two types of structured sparsity, *channel-wise* and *vector-wise* sparsity, to C_e.

1) We first prune channels whose corresponding scaling factor in batch normalization layers is lower than a threshold, which is manually controlled for each layer. In practice, we only apply channel-wise sparsifying at the first training epoch once, given the observation that the pruned channel structure will not change much.
2) We then zero out elements in C_e based on the magnitudes to meet the vector-wise sparsity constraint: $\sum_j \|C_e[:, j]\|_0 \leq S_c$, where S_c is manually controlled per layer.

Structured sparsity being more regular and hardware efficient, it also brings much more aggressive constraints on the model capacity and thus more performance degradation. This is empirically verified in the ablation study in Section VI-A. Although vector-wise sparsity has been investigated in [44], we are the first to consider vector sparsity in a unified framework with quantization, sparsification, and decomposition.

After we obtain C_e and B via *SmartDeal* algorithm, we further desire storage-economic and hardware-friendly representations to store them on-device. We explain the encoding schemes of C_e and B in Appendix B in the Supplementary Material. We also discuss more on the rationale behind our algorithm, in Appendix C in the Supplementary Material.

IV. *SmartDeal* for Efficient Training

Extending *SmartDeal* to energy-efficient training is highly nonstraightforward. As pointed out in Section I, the dynamic sparsity pattern and the discrete values in C_e constitute two grand challenges. Also, directly involving optimization like solving (3) into training is not acceptable due to its own complexity. We hereby discuss how to transplant the methodology of *SmartDeal* to energy-efficient training, with two dedicated techniques presented to address the two challenges. The two key points of the proposed techniques are 1) to optimize the decomposed C_e and B matrices in the special discrete space constrained by sparsity and power-of-2 quantization so that the model preserves *SmartDeal* structure and thus the economic memory access throughout training and 2) to avoid using any "shadow weights," i.e., the latent high-precision copies of weights that are actually trained and will introduce large overhead in model storage.

Basic Routine [SD-Training (SD-T)]: Considering the practice of on-device learning, we assume a pretrained DNN to start with, where the goal is either fine-tuning or adaptation. For a layer with pretrained weight W, we first perform one-pass SD to get the initial C_e and B. We use C_e and B as hidden weights that will be used to reconstruct W run-time during a *feed-forward* pass. Note that the reconstruction step introduces little overhead but significantly lowers the data movement cost due to the fixed sparsity structure and the power-of-2 nonzero values in C_e.

During the *back-propagation pass*, the gradient is passed back to the intermediate W as usual. The gradients of B and C_e are calculated with W's gradient and matrix multiplications. We update B using the standard gradient descent. For updating C_e, we explicitly require C_e to be within its original feasible domain: a (structurally) sparse matrix with power-of-2 nonzero elements. Such a challenging requirement is met thanks to the next two customized techniques.

1) *For C_e Zeros (Fixed Sparsity Mask):* We fix the sparsity pattern in C_e throughout training: the initial zero entries in C_e are "frozen" to zero, while the initial nonzero entries can be updated to either zero or nonzero flexibly. Experiments show that such a fixation does not noticeably impact the tuned/adapted model accuracy. This fixed sparsity brings in twofold advantages: 1) fixing the sparse pattern of C_e saves the gradient computations of its zero elements during training, which cannot be skipped in classic pruning with a dynamic sparse pattern and 2) the static sparsity pattern can be utilized to avoid the dynamic indexing sparse weights and/or adjusting processing schedules. Extensive experiments in Section VI-C show that the fixed sparsity implementation is effective across different models and datasets for saving more energy.
2) *For C_e Nonzeros (Bucket Switch Updating):* The next dilemma is on updating nonzero power-of-2 elements: if we update C_e using floating-point add operations, the floating-point numbers have to be recovered before updating incurring overheads. Instead, we propose a *Bucket Switch* updating scheme for C_e nonzero updating: inspired by [8] showing that taking only gradient signs (i.e., 1-bit gradients) suffices to training DNNs, we refer to gradient signs to guide the switch of nonzero values, from one discrete "bucket" to another.

Specifically, if the gradient w.r.t. a nonzero element (whose current value is 2^p) is positive, then we will switch up its value from 2^p to 2^{p+1} (or no change if $p = P$ already reaches the upper range bound). Similarly, a negative gradient will switch 2^p down to 2^{p-1}, and a zero gradient will not change it. In this way, the nonzero element in C_e is directly updated over the discrete domain, without any overhead of floating-point number operations. Notice that we never aggregate high-precision updates, and there also exists no high-precision latent weight for C_e in our implementation, because such will go against our goal of saving storage and energy for efficient training. Instead, we only record update directions (signs) and accumulate using an *integer* counter. Once the counter's integer record passes a threshold, we "switch the bucket."

While the above gradient quantization rule could suffer from high variance to learning rates, in practice, we apply two methods to mitigate the effect of noisy gradients. Before taking the sign, gradients whose magnitudes are below a threshold θ_g are zeroed out. We also adopt an "update delay and aggregation" scheme, in which we only really update an entry in C_e (switch the bucket) when it receives switch signals in the same direction for enough times. The number of required times is controlled by an integer hyperparameter θ_c. That is inspired by the lazy update and trajectory smoothing in gradient-based optimization [45], [46]: applying the similar idea to bucket switch is found to help training stability (as our update directions are "noisy"). The detailed description of this strategy is laid out in Appendix D in the Supplementary Material. We note that a similar idea of "bucket switch" and "update decay and aggregation" was coincidentally found useful in another latest work on optimizing binary neural networks (BNNs) [47]. We leave more discussions on the potential theoretical underpinnings for future work. We also apply the stochastic weight averaging (SWA) technique [45] that could stabilize training too without incurring noticeable overhead.

V. Dedicated Hardware-Algorithm Codesign

In this section, we present our proposed *SmartDeal* accelerator. We first introduce the design principles and considerations (Section V-A) for fully making use of the proposed *SmartDeal* algorithm's properties to maximize energy efficiency and minimize latency, and then describe the proposed accelerator (Section V-B) in detail.

A. Design Principles and Considerations

1) Minimizing Overhead of Rebuilding Weights: Thanks to the sparse and readily quantized coefficient matrices resulting from the *SmartDeal* algorithm, the memory storage and data movements associated with these matrices can be greatly reduced (see Section VI-B; e.g., up to 80×). Meanwhile, to fully utilize the advantages of the *SmartDeal* algorithm, the overhead of rebuilding weights should be minimized. To do so, it is critical to ensure that the location and time of the rebuilding units and process are properly designed. Specifically, for a *SmartDeal* accelerator: 1) the rebuild engine (RE) that restores weights using both the basis matrix and corresponding weighted coefficients should be located close to the PEs for minimizing the data movement costs of the rebuilt weights and 2) as the basis matrices are reused most frequently, the dataflow for these matrices should be weight stationary, i.e., once being fetched from the memories, they stay in the REs until all the corresponding weights are rebuilt.

2) Taking Advantage of the Structured Sparsity: The enforced *vector-wise* sparsity in the *SmartDeal* algorithm's coefficient matrices offers benefits of 1) *vector-wise* skipping both the memory accesses and computations of the corresponding activations [see Fig. 2(a)] and 2) reduced coefficient matrix encoding overhead [see Fig. 2(b)]. Meanwhile, there is an opportunity to make use of the *vector-wise/bit-level* sparsity of activations for improving efficiency.

First, one promising benefit of the *SmartDeal* algorithm's enforced *vector-wise* sparsity in the coefficient matrices is the possibility to *vector-wise* skip both the memory accesses and computations of the corresponding activations [see Fig. 2(a)].

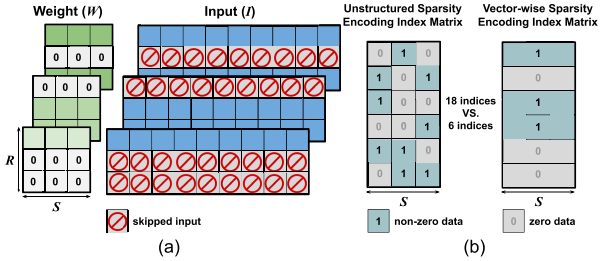


Fig. 2. Illustration of (a) vector-wise skipping the corresponding activations and (b) reduced indexing overhead, thanks to the enforced *vector-wise* weight sparsity of the *SmartDeal* algorithm.

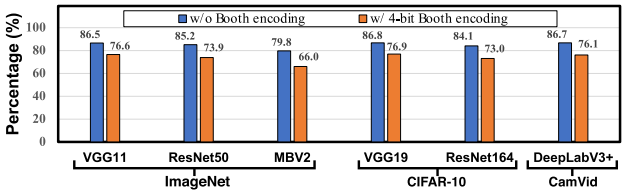


Fig. 3. Bit-level sparsity in activations for six models on three datasets.

This is because those *vector-wise* sparse coefficient matrices' corresponding weight vectors naturally carry their *vector-wise* sparsity pattern/location, offering the opportunity to directly use the sparse coefficient matrices' encoding index to identify the weight sparsity and skip the corresponding activations' memory accesses and computations. Such a skipping can lead to large energy and latency savings because weight vectors are shared by all activations of the same feature maps in CONV operations, see Fig. 2(b).

Second, commonly used methods for encoding weight sparsity, such as the 1-bit direct weight indexing [48], compressed row storage (CRS) [19], and Huffman encoding, store both the values and sparsity encoding indexes of weights. Our *SmartDeal* algorithm's *vector-wise* weight sparsity reduce both the sparsity encoding overhead [see Fig. 2(b)] and skipping control overhead. The resulting energy and latency benefits depend on the sparsity ratio and pattern, and hardware constraints (e.g., memory bandwidths).

Third, the accelerator can further make use of *bit-level* and *vector-wise* sparsity of activations to improve energy efficiency and reduce latency, where the *bit-/vector-wise* sparsity means the percentage of the zero activation bits/rows over the total activation bits/rows. Fig. 3 shows the *bit-level* sparsity of activations w/ and w/o 4-bit Booth encoding [49] in popular DNNs, including VGG11, ResNet50, and MobileNetV2 on ImageNet, VGG19 and ResNet164 on CIFAR-10, and DeepLabV3+ on CamVid. We can see that the bit-level sparsity is 79.8% under an 8-bit precision and 66.0% using the corresponding 4-bit Booth encoding even for a compact model such as MobileNetV2; for vector-wise sparsity, it can be widely observed among the CONV layers with 3×3 kernel size, e.g., up to 27.1% in the last several CONV layers of MobileNetV2 and up to 32.4% in ResNet164.

3) Support for Compact Models: The recently emerged compact models, such as MobileNet [10] and EfficentNet [11], often adopt depth-wise CONV and squeeze-and-excite layers other than the traditional 2-D CONV layers to restrict the model size, which reduces the data reuse opportunities. Taking a depth-wise CONV layer as an example, it has an "extreme" small number of CONV channels (i.e., 1), reducing the input

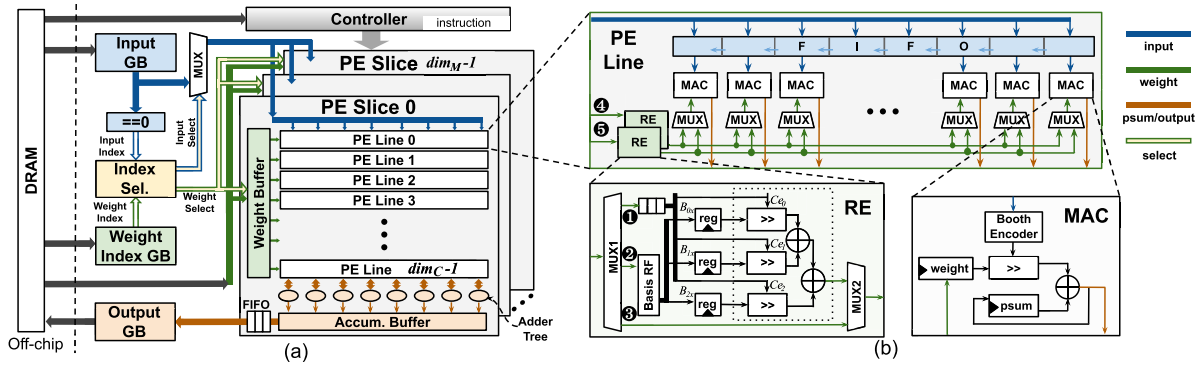


Fig. 4. Illustration of the proposed *SmartDeal* accelerator. (a) Architecture and (b) lock diagram of the PE line, each of which includes two REs and eight MAC units.

reuse over the standard CONV layers; for squeeze-and-excite, similar to that of FC layers, there are no weight reuse opportunities in squeeze-and-excite layers. On-device efficient accelerators should consider these features of compact models for their wide adoption and leveraging compact models for more efficient processing.

B. Architecture of the SmartDeal Accelerator

1) Architecture Overview: Fig. 4(a) shows the architecture of the proposed *SmartDeal* accelerator, which consists of a 3-D PE array with a total of dim_M PE slices, input/index/output global buffers (see the blocks named Input GB, Weight Index GB, and Output GB, where GB denotes global buffer) associated with an index selector for sparsity (see the blocks named Index sel.), and an controller. The accelerator communicates with an off-chip DRAM through DMA (direct memory access) [38]. Following the aforementioned design principles and considerations (see Section V-A), the proposed accelerator features the following properties: 1) *an RE design*, which is inserted within PE lines to reduce the rebuilding overhead [see the top part of Fig. 4(b)]; 2) *a hybrid dataflow*: a 1-D row stationary dataflow is adopted within each PE line for maximizing weight and input reuses, while each PE slice uses an output stationary dataflow for maximizing output partial sum reuses; 3) *an index selector* [named Index Sel. in Fig. 4(a)] to select the nonzero coefficient and activation vector pairs as inspired by [48]. This is to skip not only computations but also data movements associated with the sparse rows of the coefficients and activations. The index selector design in *SmartDeal* is the same as that of [48] except that Huffman encoding is used here and the index values of 0/1 stand for vector (instead of scalar) sparsity; and 4) *a data-type driven memory partition* in order to use matched bandwidths (e.g., a bigger bandwidth for the weights/inputs and a smaller bandwidth for the outputs) for different types of data to reduce the unit energy cost of accessing the SRAMs, which is used to implement the GB blocks [50]. We adopt separated centralized GBs to store the inputs, outputs, weights, and indexes, respectively, and distributed SRAMs [see the Weight Buffer unit in Fig. 4(a)] among PE slices to store weights (including the coefficients and basis matrices); and 5) *a bit-serial multiplier-based MAC array* in each PE line to make use of the activations' bit-level sparsity together with a Booth Encoder as inspired by [49].

2) PE Slices and Dataflow: We here describe the design of the PE Slice unit in the 3-D PE slice array of Fig. 4(a):

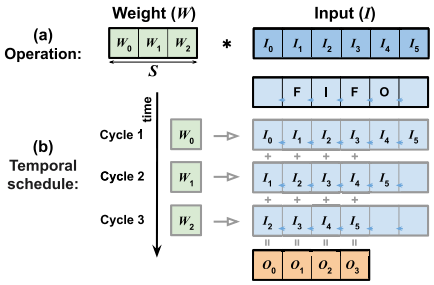


Fig. 5. Illustration of the proposed 1-D row stationary along each PE slice (in this particular example, FIFO size is 5, and in general it should be $\text{dim}_F + S - 1$). (a) 1-D CONV and (b) processing flow of 1-D row stationary.

a) First, the 3-D PE Slide array: Our *SmartDeal* accelerator enables paralleled processing of computations associated with the same weight filter using the PE slice array of size dim_M (with each PE slice having dim_C PE lines) and dim_C number of input channels, where the resulting partial sums are accumulated using the adder trees at the bottom of the PE lines [see the bottom right side of Fig. 4(a)]. In this way, a total of dim_M consecutive output channels (i.e., dim_M weight filters) are processed in parallel to maximize the reuse of input activations. Note that this dataflow is employed to match the way we reshape the weights as described in Section III-C.

b) Second, the PE line design: Each PE line in Fig. 4 includes an array of dim_F MACs, one FIFO (using double buffers), and two RE units, where the REs at the left restore the original weights in a row-wise manner. During operations, each PE line processes one or multiple 1-D CONV operations, similar to the 1-D row stationary in [51] except that we stream each rebuild weight of one row temporally along the MACs for processing one row of input activations. In particular, the 1-D CONV operation is performed by shifting the input activations along the array of MACs within the PE line (see Fig. 5) using an FIFO; this 1-D CONV computation is repeated for the remaining 1-D CONV operations to complete one 2-D CONV computation in $\leq(S \times R)$ cycles (under the assumption of w/ sparsity and w/o bit-serial multiplication) with 1) each weight element being shared among all the MACs in each cycle and 2) the intermediate partial sums of the 2-D CONV operations are accumulated locally in each MAC unit [see the bottom right part of Fig. 4(b)].

c) Third, the RE design: As shown in the bottom left corner of Fig. 4(b), an RE unit includes an RF (register file) of size $S \times S$ to store one basis matrix and a shift-and-add unit to rebuild weights. The time division multiplexing unit at the left, i.e., MUX1, is to fetch the ❶ coefficient

matrices, ❷ basis matrices, or ❸ original weights. This design enables the accesses of these three types of data to be performed in a time division manner in order to reduce the weight bandwidth requirement by taking advantage of the fact that it is not necessary to fetch these three types of data simultaneously. Specifically, the basis matrix is fetched first and stored stationary within the RE until the associated computations are completed; the weights are then rebuilt in an RE where each row of a coefficient matrix stays stationary until all its associated computations are finished. The third path of MUX1 ❸ for the original weights is to handle DNNs' layers where *SmartDeal* is not applied on.

d) Fourth, the handling of compact models: When handling compact models, we consider an adjusted dataflow and PE line configuration for improving the utilization of both the PE slice array and the MAC array within each PE line. Specifically, for depth-wise CONV layers, since the number of CONV channels is only 1, the dim_C PE lines will no longer correspond to input channels. Instead, we map the R number of 1-D CONV operations along the dimension of the weight height to these PE lines. For squeeze-and-excite/FC layers, each PE line's MAC array of dim_F MACs can be divided into multiple clusters [e.g., two clusters for illustration in the top part of Fig. 4(b)] with the help of the two REs in one PE line (denoted as ❹ and ❺) and multiplexing units at the bottom of the MAC array, where each cluster handles computations corresponding to a different output pixel in order to improve the MAC array's utilization and thus latency performance. In this way, the proposed *SmartDeal* accelerator's advantage is maintained even for compact models, thanks to this adjustment together with 1) our adopted 1-D row stationary dataflow within PE lines; 2) the employed bit-serial multipliers; and 3) the possibility to heavily quantized coefficients.

3) Buffer Design: For making use of DNNs' (filter-/vector-wise or bit-level) sparsity for skipping corresponding computations/memory accesses, it in general requires a larger buffer (than that of corresponding dense models) due to the unknown dynamic sparsity patterns. We here discuss how we balance between the skipping convenience and the increased buffer size. Specifically, to enable the processing with sparsity, the row pairs of nonzero input activations and coefficients are selected from the Input GB and the Index GB (using the corresponding coefficient indexes), respectively, as inspired by [48], which are then sent to the corresponding PE lines for processing with the resulting outputs being collected to the output GB.

a) First, input GB: To ensure a high utilization of the PE array, a vanilla design requires $(\text{dim}_C \times \text{dim}_F \times \text{bits}_{\text{input}})\times$ input activation rows (than that of the dense model counterpart) to be fetched for dealing with the dynamic sparsity patterns, resulting in $(\text{dim}_C \times \text{dim}_F \times \text{bits}_{\text{input}})\times$ increased input GB bandwidth requirement. In contrast, our design leads to a $\geq 1/S$ reduction of this required input GB bandwidth, with $\text{dim}_C \times \text{dim}_F \times \text{bits}_{\text{input}}$ inputs for every (S + "Booth encoded nonzero activation bits") cycles. This is because all the FIFOs in the PE lines are implemented in a ping-pong manner using double buffers, thanks to the fact that 1) the adopted 1-D row stationary dataflow at each PE line helps to relieve this bandwidth requirement, because each input activation row can be reused for S cycles and 2) the bit-serial multipliers takes ≥ 1 cycles to finish an element-wise multiplication.

b) Second, weight/index/output buffer: Similar to that of the input GB, weight/index buffer bandwidth needs to be expanded for handling activation sparsity, of which the expansion is often small thanks to the common observation that the vector-wise activation sparsity ratio is often relatively low. Note that because basis matrices need to be fetched and stored into the RE before the fetching of coefficient matrices and the weight reconstruction computation, computation stalls occur if the next basis matrix is fetched after finishing the coefficient fetching and the computation corresponding to the current basis matrix. Therefore, we leverage the two REs (❹ and ❺ paths) in each PE line to operate in a "ping-pong" manner to avoid the aforementioned computation stalls. For handling the output data, we adopt an FIFO to buffer the outputs from each PE slice before writing them back into the GB, i.e., a cache between the PE array and the output GB. This is to reduce the required output GB bandwidth by making use of the fact that each output is calculated over several clock cycles.

VI. Experiments

In this section, we present a thorough evaluation of *SmartDeal*. We lay out our plan below.

On the algorithm level, as *SmartDeal* unifies three mainstream model compression ideas: *sparsification/pruning*, *decomposition*, and *quantization* into one framework, we first present a carefully designed ablation study in Section VI-A that investigates the effects of different components in *SmartDeal* and the interactions among them.

We then perform extensive experiments (benchmark over two structured pruning and four quantization, i.e., state-of-the-art compression techniques on four standard DNN models with two datasets) to validate its superiority. In addition, we evaluate *SmartDeal* on two compact DNN models (MobileNetV2 [52] and EfficientNet-B0 [11]) on the ImageNet [42] dataset, one segmentation model (DeepLabv3+ [53]) on the CamVid [54] dataset, two MLP models on MNIST, and language-modeling tasks.

Following the validation of *SmartDeal* inference, we evaluate *SmartDeal* for training on *fine-tuning* and *adaptation* tasks given pretrained models, which is the common practice in edge-based training [55], [56]. Extensive training experiments in Section VI-C1 of two lightweight networks show the storage and energy efficiency of *SmartDeal* during training without trading too much model performance. We also provide evaluation over state-of-the-art GPUs in terms of their energy efficiency in Section VI-C2 by using a state-of-the-art training accelerator [9].

On the hardware level, as the goal of the proposed *SmartDeal* is to boost hardware acceleration energy efficiency and speed, we evaluate *SmartDeal*'s algorithm-hardware co-design results with state-of-the-art DNN accelerators in terms of energy consumption and latency when processing representative DNN models and benchmark datasets. Furthermore, to provide more insights about the proposed *SmartDeal*, we perform various ablation studies to visualize and validate the effectiveness of *SmartDeal*'s component techniques.

A. An Ablation Study on SmartDeal's Building Blocks

We first investigate the influence of different components of the *SmartDeal* algorithm: 1) *Decomposition*—decomposing

TABLE II

ABLATION STUDY OF THE COMPONENTS IN *SmartDeal* USING RESNET18 ON CIFAR-10 [41] AND FPGA ENERGY RESULTS. EXP. 7, 8, SD, SD† USE HUFFMAN CODING REPRESENTATIONS FOR C_e AND 8-bit FIXED-POINT REPRESENTATIONS FOR B AND ACTIVATIONS. OTHER EXPERIMENTS USE 32-bit FLOATING-POINT REPRESENTATIONS

No.	DE	Q2	US	SS	RT	Acc	FLOPs* (M)	Size‡ (MB)	W/C_e Sparsity□	Norm. E.E.
1						94.73%	1110.8	42.59	100.0%/ -	1×
2	✓					94.73%	1177.8	43.28	100.0%/100.0%	0.79×
3	✓	✓				94.16%	1177.8	11.34	100.0%/100.0%	2.49×
4^{+}			✓			52.98%	212.0	9.46	19.09%/ -	2.14×
5	✓		✓			93.36%	689.6	9.46	37.09%/20.60%	1.71×
6		✓	✓			90.42%	440.5	3.97	39.66%/ -	3.77×
7	✓	✓	✓			92.29%	400.0	2.95	67.22%/46.44%	3.63×
8	✓	✓		✓		43.96%	290.9	2.43	41.47%/29.19%	3.97×
8$_{\triangle}$	✓	✓		✓		11.30%	233.9	1.91	20.53%/13.76%	4.26×
9^{+}		✓	✓		✓	93.02%	141.7	2.18	12.75%/ -	4.31×
SD	✓	✓	✓		✓	94.32%	261.2	1.78	22.12%/12.77%	4.15×
SD^{-}	✓	✓	✓		✓	94.60%	552.2	2.99	48.27%/36.36%	-
SD†	✓	✓		✓	✓	93.30%	209.8	1.74	17.16%/10.53%	4.38×
SD$^{\dagger}_{\triangle}$	✓	✓		✓	✓	92.54%	189.6	1.67	13.14%/ 7.75%	4.40×
Improv. (SD† vs. baseline 1)						**-1.43%**	**-81.11%**	**-95.91%**	**-82.84%/-89.47%**	**4.38×**

† Full form of *SmartDeal* with structured sparsity.
‡ Representations of C_e and B and encoding overheads are all considered for *SmartDeal* models.
+ The model is pruned to be of the same model size compared with its counterpart.
− A smaller threshold θ is used in *SmartDeal* for performance comparable to the baseline in experiment No. 1.
□ We provide the sparsity of C_e coefficients as long as DE is involved, in which case we also report the sparsity of rebuilt weight matrices. Sparsity refers to the ratio of **non-zeros**.
△ Use larger layer-wise structural pruning ratios. See Tab. X.

each layer W into B and C_e with *no constraint* on C_e; 2) *Quantization*—requiring C_e elements to either power-of-2 values or zero; 3) *Unstructured Sparsification*—promoting element-wise sparsity in C_e; 4) *Structured Sparsity*—incorporating structured sparsity for C_e; and 5) *Retraining*—iteratively performing retraining and *SmartDeal*. For simplicity, we term the above five components as **DE** (**DE** composition), **Q2** (**Q**uantization to Power-of-**2**), **US** (**U**nstructured **S**parsity), **SS** (**S**tructured **S**parsity), and **RT** (**R**e-**T**raining), respectively.

Table II summarizes the results of ResNet18 [57] on CIFAR-10 dataset [41], with the accuracy, FLOPs,[3] the model storage (total memory size to save all parameters including index if necessary), sparsity levels, and normalized energy efficiency results measured on an edge FPGA (i.e., Ultra 96 FPGA [59]) reported. For US, we pick the sparsity threshold $\theta = 8 \times 10^{-3}$. For RT, we iteratively perform retraining and *SmartDeal* algorithm (with fixed θ) for 80 epochs and report best performing models among. For SS, the pruning ratios for different layers are manually tuned. We provide the reproducible details for selecting layer-wise pruning ratios in Appendix E in the Supplementary Material, as well as the general rule how we select them. We use Huffman coding representations for C_e (refer to Section B) and 8-bit fixed-point representations for B and input-output activations for *SmartDeal* (Exp. 7, 8, SD, and SD†). Other experiments use standard 32-bit floating-point representations for input/activations. We further use a Ultra96 FPGA [59] platform for real-device energy measurement and report the energy efficiency (normalized to the implementation without any components of the *SmartDeal* algorithm). The difference in utilized computation resource on the FPGA board [i.e., digital signal processing (DSP) unit and lookup table (LUT)] among all implementations is within 5% so that the influence of computation resources is negligible.

[3]We use the method in [58] to compare fixed-point and 32-bit floating-point where the highest bit precision of the operands determines the equivalent floating-point operation. Only CONV and FC layers are considered.

1) *Quantization:* Comparing Exp. 2 and 3, the power-of-2 quantization has aggressive advantage in reducing the model size, without incurring much accuracy loss: *>3.7× reduction in model size while incurring <0.6% accuracy loss*. Note that because we don't quantize B in Exp. 3, the equivalent FLOPs is the same as Exp. 2.
2) *(Structured) Sparsity:* Comparing Exp. 7 with 3, sparsity trades slightly more accuracy drop for higher storage saving and has an immediate influence on FLOPs. Structured sparsity can bring much more aggressive size/FLOPs advantage but will suffer from significant performance loss as shown in Exp. 8. Fortunately, however, model performance can be recovered by retraining in both cases (see last three rows).
3) *Decomposition:* The benefit brought by decomposition can be supported by comparing multiple pairs of experiments in different cases. For example, when we compare Exp. 6 and 7 (we compress the models to around the same FLOPs number for fair comparison), the model in Exp. 7 (with decomposition) achieves near 2% higher accuracy with even lower FLOPs.
4) *Comparison With Sparsity-Only Models:* Comparing (4^{+}, 5) and (9^{+}, SD) shows that under the same tight storage budget, DE induces a much smaller accuracy drop (93.36% vs. 52.98% and 94.32% vs. 93.02%) at the cost of more FLOPs, which further supports the important role that DE plays in *SmartDeal* by identifying a higher-order structure. The higher FLOPs of Exp. 5 and SD are mainly due to lower sparsity in reconstructed W, which has direct influence on FLOPs. Overheads for rebuilding W only account for <1% of

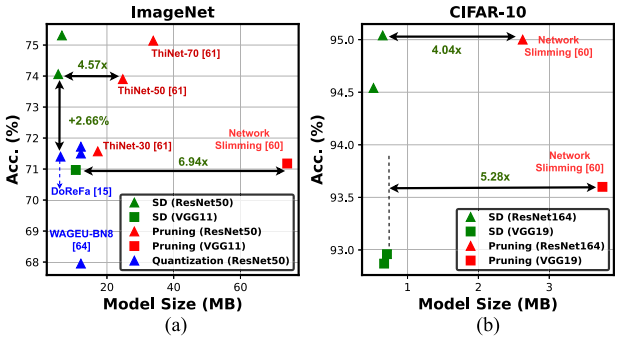


Fig. 6. Accuracy versus model size comparison of *SmartDeal* (SE) and SOTA compression techniques on (a) ImageNet and (b) CIFAR-10, where different colors differentiate the SE and baseline techniques. We use 16-bit floating weight representation for the pruning-alone methods.

total FLOPs thanks to the high sparsity and readily quantized-to-2 structure of C_e.

5) *Energy Efficiency on FPGA: SmartDeal* trades higher cost memory storage/access for lower cost computation. Therefore, energy efficiency is a better way than FLOPs to show its power. We can see that SD enjoys near 2% better accuracy than Exp. 9^+ and comparable energy efficiency despite higher FLOPs. When favored with structured sparsity, $\text{SD}^\dagger$ still outperforms Exp. 9^+ with smaller model size and further improves the energy efficiency.
6) *Controlling Pruning Ratios for Structured Sparsity:* Exp. 8 from Table II shows that structured sparsity improves energy efficiency but at the cost of a drastic accuracy drop, although retraining can recover most of the accuracy in Exp. $\text{SD}^\dagger$. Here, we use the additional Exp. $8_\triangle$ and $\text{SD}^\dagger_\triangle$, in which we use larger pruning ratios for structured sparsity, to show that we can easily trade-off between storage and energy efficiency with accuracy by controlling the pruning ratios.

B. Evaluation of the SmartDeal Algorithm

1) Experiment Settings: To evaluate the algorithm performance of *SmartDeal* algorithm, we conduct experiments on 1) a total of six DNN models using both the CIFAR-10 [41] and ImageNet [42] datasets; 2) one segmentation model on the CamVid [54] dataset; and 3) two MLP models on the MNIST dataset and compare the performance with state-of-the-art compression techniques in terms of accuracy and model size, including two structured pruning techniques (Network Slimming [60] and ThiNet [61]),[4] Four quantization techniques (scalable 8-bit (S8) [62], FP8 [63], WAGEUBN [64], and DoReFa [15]), one power-of-2 quantization technique [65], and one pruning and quantization technique [48].

2) SmartDeal Versus Existing Compression Techniques: As *SmartDeal* unifies the three mainstream ideas of pruning, decomposition, and quantization, we evaluate the *SmartDeal* algorithm performance by comparing it with state-of-the-art pruning-alone and quantization-alone algorithms,[5] under four DNN models and two datasets. Note that we use 16-bit floating-point weight representation for the pruning-alone algorithms for more fair comparison. The experiment results are shown in Fig. 6. *SmartDeal* in general outperforms all other pruning-alone or quantization-alone competitors in terms of the achievable trade-off between the accuracy and the model size. Taking ResNet50 on ImageNet as an example, the quantization algorithm DoReFa [15] seems to aggressively shrink the model size yet unfortunately cause a larger accuracy drop; while the pruning algorithm ThiNet [61] maintains competitive accuracy at the cost of larger models. In comparison, *SmartDeal* combines the best of both worlds: it obtains almost as high accuracy as the pruning-only ThiNet [61], which is 2.66% higher than the quantized-only DoReFa [15]; and on the other hand, it keeps the model as compact as DoReFa [15]. SmartDeal also outperforms more recent and competitive model compression works [a1, a3]. When compared with the state-of-the-art quantization method LSQ [66] that uses 3-bit weight representation, SmartDeal costs much less storage for a ResNet50 network than using LSQ (6.51 MB vs. 9.61 MB) while inducing less accuracy drop (0.82% vs. 1.10%) on ImageNet. Moreover, SmartDeal also outperforms CLIP-Q [67], a state-of-the-art joint pruning-quantization method, yielding smaller model size than Clip-q (6.51 MB vs. 6.70 MB) but much higher top-1 accuracy (75.31% vs. 73.80%).

Apart from the aforementioned works, we also evaluate the *SmartDeal* algorithm with a state-of-the-art power-of-2 quantization algorithm [65] based on the same MLP model with a precision of 8 bits: while having a significantly higher compression rate of 188.3× (vs. 128× in [65]), *SmartDeal* achieves a comparable accuracy (97.32% vs. 97.35%), even if

TABLE III

RESULT SUMMARY OF THE PROPOSED *SmartDeal* WITH RETRAINING ON 1) VGG11 AND RESNET50 USING THE IMAGENET DATASET [42]; 2) VGG19 AND RESNET164 USING THE CIFAR-10 DATASET [41]; AND 3) MLP-1 [65] AND MLP-2 [48] USING THE MNIST DATASET

Model	Top-1 (%)	Top-5 (%)	CR (×)	Param. (MB)	B (MB)	C_e (MB)	Spar. (%)
VGG11	71.18%	90.08%	-	845.75	-	-	-
VGG11_{SD}	70.97%	89.88%	79.26	10.67	1.67	7.46	86.00
ResNet50	76.13%	92.86%	-	102.40	-	-	-
$\text{ResNet50}_{\text{SD}}$	75.31%	92.33%	15.73	6.51	1.40	4.40	45.00
$\text{ResNet50}_{\text{SD}}$	74.06%	91.53%	18.93	5.41	1.40	3.30	58.60
VGG19	93.66%	-	-	80.13	-	-	-
VGG19_{SD}	92.96%	-	112.3	0.71	0.27	0.37	92.80
VGG19_{SD}	92.87%	-	119.6	0.67	0.27	0.33	93.70
ResNet164	94.58%	-	-	6.75	-	-	-
$\text{ResNet164}_{\text{SD}}$	95.04%	-	10.38	0.65	0.25	0.34	37.60
$\text{ResNet164}_{\text{SD}}$	94.54%	-	12.87	0.52	0.25	0.21	61.00
MLP-1	98.47%	-	-	14.125	-	-	-
MLP-1_{SD}	97.32%	-	188.3	0.075	0.01	0.065	82.34
MLP-2	98.50%	-	-	1.07	-	-	-
MLP-2_{SD}	98.11%	-	66.88	0.016	0.00	0.021	93.33

1. The baseline models use 32-bit floating-point representations for the weights and input/output activations, so as to benchmark with the best achievable accuracy results in the literature.
2. The proposed *SmartDeal* models use 8-bit fixed-point representations for the input/output activations; the Huffman coding representations for the coefficient matrices; and 8-bit basis matrices, respectively.

[4]We reimplement ThiNet [61] and report the best numbers that we reproduce in the same computation environment for fair comparison, because we obtain better baseline accuracies than the original ThiNet article. Note that we report model size in megabytes (MB) (with 16-bit weight representation) while *the number of parameters* in million (M) is reported in [61].

[5]We did not include decomposition-alone algorithms since their results are not as competitive and also less popular.

SmartDeal is not specifically dedicated for FC layers while the power-of-2 quantization [65] does. In addition, compared with the pruned and quantized MLP model in [48], *SmartDeal* achieves a higher compression rate of 66.88× (vs. 40× in [48]) with a comparable accuracy (98.11% vs. 98.42%).

A more extensive set of evaluation results is summarized in Table III, in order to show the maximally achievable gains (and the incurring accuracy losses) by applying *SmartDeal* over the original uncompressed models. In Table III, "CR" means the *compression rate* in terms of the overall parameter size; "Param.," "B," and "C_e" denote the total size of the model parameters, the basis matrices, and the coefficient matrices, respectively; "Spar." denotes the ratio of the pruned and total parameters (the higher the better). Without too much surprise, *SmartDeal* compresses the VGG networks by 80× to 120×, all with negligible (less than 1%) top-1 accuracy losses. For ResNets, *SmartDeal* is still able to achieve a solid >10× compression ratio. For example, when compressing ResNet50, we find *SmartDeal* to incur almost no accuracy drop, when compressing the model size by 15× to 18×.

3) SmartDeal Applied on Compact Models: We also validate that the proposed SmartDeal algorithm remains to be beneficial when adopted for well-known compact models, i.e., MobileNetV2 (MBV2) [52] and EfficientNet-B0 (Eff-B0) [11]. SmartDeal only incurs ∼2% top-1 accuracy and 1% top-5 accuracy losses when compressing MBV2 and Eff-B0 for 7.69∼7.82× CR, in contrast to a 7.07% top-1 accuracy loss with 8× compression (4-bit quantization) of MBV2 as reported in the latest work [68]. Detailed experimental settings and empirical results can be found in Appendix F in the Supplementary Material.

4) Extending SmartDeal Beyond Classification and Computer Vision: We further demonstrate the effectiveness of SmartDeal is beyond one specific task setting by evaluating it on the semantic segmentation task with CamVid [54] dataset and on the character-level language modeling task with Penn Treebank dataset [69]. Refer to the Appendix F in the Supplementary Material for detailed experiment settings and the empirical results.

C. SmartDeal Training Evaluation

1) Fine-Tuning and Adaptation Study: We present experiment results of SD-T in fine-tuning and adaptation tasks to support the efficacy of *SmartDeal* for resource-constrained on-device training. We run training experiments in two settings on partitioned CIFAR-10/100 datasets. Our experiments are based on ResNet18 [57] and MobileNetV2 [52]. We compare SD-T (with and without structured sparsity) with standard training. We denote SD-T as **SD** and SD-T with structured sparsity as **SS**. Besides accuracies, we also compare the model sizes and normalized energy efficiency.

We consider two datasets split strategies on CIFAR-10/100, corresponding to two scenarios, *fine-tuning* and *adaptation* in which a dataset is divided into two parts and we pre-train a model on one part and then fine-tune on the other. However, in fine-tuning, both parts contain all classes, while in adaptation the two parts contain non-overlapping classes. Refer to the Appendix E in the Supplementary Material for detailed splitting strategies.

Results of fine-tuning and adaptation experiments of ResNet18 and MobileNetV2 on CIFAR-10 dataset are shown in Table IV. Refer to the Appendix E for detailed experiment settings and the Appendix H for results on CIFAR-100 dataset in the Supplementary Material. We evaluate the hardware quantified benefits of *SmartDeal* for training using a state-of-the-art accelerator [9] using the method in [70]. We can see SD-T saves energy during fine-tuning/adaption for both networks. When combined with structured sparsity, SD-T can further save the energy cost with tolerable accuracy drop. We also include the convergence analysis of SD-T in the Appendix G in the Supplementary Material.

TABLE IV
RESULT OF RESNET18 AND MOBILENET-V2 EXPERIMENTS FOR FINE-TUNING AND ADAPTATION TASKS ON CIFAR-10

Task		Acc	Norm. Energy Efficiency	Size(MB)
		ResNet18		
Fine Tuning	VA	94.67%	1	42.59
	SD	93.62%	1.36	2.85
	SS	92.40%	1.50	2.44
Adapt-ation	VA	95.80%	1	42.59
	SD	93.60%	1.13	2.52
	SS	93.32%	1.34	2.34
		MobileNet-V2		
Fine Tuning	VA	91.26%	1	8.63
	SD	90.98%	1.29	1.03
	SS	90.38%	1.40	0.99
Adapt-ation	VA	93.66%	1	8.64
	SD	92.52%	1.31	1.00
	SS	92.07%	1.43	0.98

2) Hardware-Quantified Benefits of SmartDeal On-Device Training: In this set of experiments, we deploy SD-T on a state-of-the-art training accelerator [9] and compare the accelerator's performance over SOTA GPUs in terms of energy efficiency. Here, we use the state-of-the-art training accelerator [9] instead of designing a dedicated one in order to demonstrate the generality of SD-T algorithm, which can potentially improve the performance of training process regardless of the deployment hardware design. We follow the evaluation method in [70] to evaluate the accelerator by implementing a cycle-accurate simulator, aiming to model the Register-Transfer-Level (RTL) behavior of the hardware circuits [9]. As summarized in Table V, by running SD-T-based ResNet18, [9] achieves 1.3× and 4.48× improvement in energy efficiency (E.E.) over powerful GPUs with vanilla ResNet18, which are designed in more advanced technologies.

D. Evaluation of the Dedicated SmartDeal Accelerator

In this section, we present experiments to evaluate the performance of the dedicated *SmartDeal* accelerator. Specifically, we first introduce the experiment setup and methodology, and then compare *SmartDeal* accelerator with four state-of-the-art DNN accelerators (covering a diverse range of design considerations) on seven DNN models (including four standard DNNs, two compact models, and one segmentation model) in terms of energy consumption and latency when running on three benchmark datasets. Details and ablation studies of the accelerator can be referred from Section V of our prior work [2].

TABLE V

PEAK ENERGY EFFICIENCY OF ASIC TRAINING ACCELERATOR [9] WITH SD-T-BASED RESNET18 VERSUS SOTA EDGE [71] AND CLOUD GPU [72] WITH RESNET18

	Freq. (MHz)	Tech. (nm)	E. E. (GOP/s/W)
Edge GPU [71]	1455	40	120
Cloud GPU [72]	1300	22	400
SD-T [9]	500	65	538
Improv.			**1.34-4.48×**

TABLE VI

DESIGN CONSIDERATIONS OF THE BASELINE AND OUR ACCELERATORS

Accelerator	Design Considerations
DianNao [1]	Dense models
Cambricon-X [38]	Unstructured weight sparsity
SCNN [37]	Unstructured weight sparsity + Activation sparsity
Bit-pragmatic [73]	Bit-level activation sparsity
Ours	Vector-wise weight sparsity + Bit-level and vector-wise activation sparsity

1) Experiment Setup and Methodology:

a) Baselines and configurations: We benchmark the *SmartDeal* accelerator with four state-of-the-art accelerators: DianNao [1], SCNN [37], Cambricon-X [38], and Bit-pragmatic [73]. These representative accelerators have demonstrated promising acceleration performance, and are designed with a diverse design considerations as summarized in Table VI. Specifically, DianNao [1] is a classical architecture for DNN inference, which is reported to be over 100× faster and over 20× more energy efficient than those of CPUs. While DianNao considers dense models, the other three accelerators take advantage of certain kinds of sparsity in DNNs. To ensure fair comparisons, we assign the *SmartDeal* accelerator and baselines with the same computation resources and on-chip SRAM storage in all experiments, as listed in Table VII. For example, the DianNao, SCNN, and Cambricon-X accelerators use 1K 8-bit nonbit-serial multipliers and *SmartDeal* and Bit-pragmatic employ an equivalent 8K bit-serial multipliers.

For handling the dynamic sparsity in the *SmartDeal* accelerator, the on-chip input GB bandwidth and weight GB bandwidth with each PE slice are set to be four and two times of those in the corresponding dense models, respectively, which are empirically found to be sufficient for handling all the considered models and datasets. Meanwhile, because the computation resources for the baseline accelerators may be different from their original articles, the bandwidth settings are configured accordingly based on their articles' reported design principles. Note that 1) we do not consider FC layers when benchmarking the *SmartDeal* accelerator with the baseline accelerators (see Figs. 7–9) for a fair comparison as the SCNN [37] baseline is designed for CONV layers, and similarly, we do not consider EfficientNet-B0 for the SCNN accelerator as SCNN is not designed for handling the squeeze-and-excite layers adopted in EfficientNet-B0 and 2) our ablation studies consider all layers in the models can be referred from Section V of our prior work [2].

b) Benchmark models, datasets, and precision: We use seven representative DNNs (ResNet50, ResNet164, VGG11, VGG19, MobileNetV2, EfficientNet-B0, and DeepLabV3+) and three benchmark datasets (CIFAR-10 [41], ImageNet [42], and CamVid [54]). Regarding the precision, we adopt 1) 8-bit activations for both the baseline-used and *SmartDeal*-based DNNs and 2) 8-bit weights in the baseline-used DNNs, and 8-bit precision for the basis matrices and Huffman coding for the coefficient matrices in the *SmartDeal*-based DNNs.

c) Technology-dependent parameters: For evaluating the performance of the *SmartDeal* accelerator, we implemented a custom cycle-accurate simulator, aiming to model the RTL behavior of synthesized circuits, and verified the simulator against the corresponding RTL implementation to ensure its correctness. Specifically, the gate-level netlist and SRAM are generated based on a commercial 28-nm technology using the Synopsys Design Compiler and Arm Artisan Memory Compilers, proper activity factors are set at the input ports of the memory/computation units, and the energy is calculated using a state-of-the-art tool PrimeTime PX [74]. Meanwhile, thanks to the clear description of the baseline accelerators' articles and easy representation of their works, we followed their designs and implemented custom cycle-accurate simulators for all the baselines. In this way, we can evaluate the performance of both the baseline and our accelerators based on the same commercial 28-nm technology. The resulting designs operate at a frequency of 1 GHz and the performance results are normalized over that of the DianNao accelerator, where the DianNao design is modified to ensure

TABLE VII

SUMMARY OF THE COMPUTATION AND STORAGE RESOURCES IN THE *SmartDeal* AND BASELINE ACCELERATORS. THE AREA OVERHEAD FOR THE RE MODULES IN *SmartDeal* IS AROUND 0.64%

***SmartDeal* and Bit-pragmatic [73]**			
dim_M	64	**Input GB**	16KB×32Banks
dim_C	16	**Output GB**	2KB×2Banks
dim_F	8	**Weight Buff./slice**	2KB×2Banks
# of bit-serial mul.	8K	**Precision**	8 bits
DianNao [1], SCNN [37], and Cambricon-X [38]			
The same total on-chip SRAM storage as *SmartDeal*			
# of 8-bit mul.	1K	**Precision**	8 bits

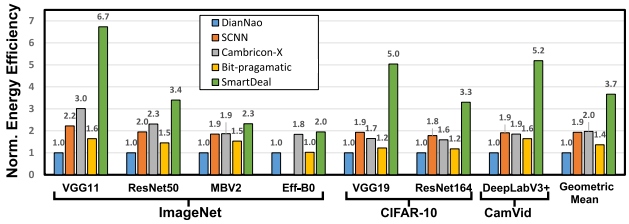


Fig. 7. Normalized energy efficiency (over DianNao) achieved by the *SmartDeal* accelerator over the four state-of-the-art baseline accelerators on seven DNN models and three datasets.

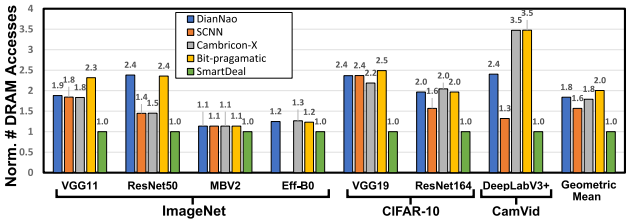


Fig. 8. Normalized number of DRAM accesses (over the *SmartDeal* accelerator) of the *SmartDeal* and four state-of-the-art baseline accelerators on seven DNN models and three datasets.

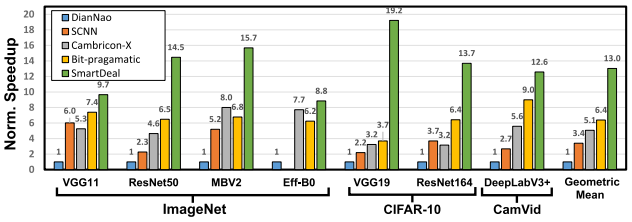


Fig. 9. Normalized speedup (over DianNao) achieved by the *SmartDeal* accelerator over the four state-of-the-art baseline accelerators on seven DNN models and three datasets.

that all accelerators have the same hardware resources (see Table VII). We refer to [70] for the unit energy of DRAM accesses, which is 100 pJ per 8 bit, and the unit energy costs for computation and SRAM accesses are listed in Table I.

2) SmartDeal Versus State-of-the-Art Accelerators:

a) Energy efficiency over that of the baseline accelerators: Fig. 7 shows the normalized energy efficiency of the *SmartDeal* and the baseline accelerators. It is shown that the *SmartDeal* accelerator consumes the least energy under all the considered DNN models and datasets, achieving an energy efficiency improvement ranging from 2.0× to 6.7×. The *SmartDeal* accelerator's outstanding energy efficiency performance is a result of *SmartDeal*'s algorithm-hardware co-design effort to effectively trade the much higher cost memory storage/accesses for the lower cost computations (i.e., rebuilding the weights using the basis and coefficient matrices at the least costly RF and PE levels versus fetching them from the DRAM). Note that *SmartDeal* nontrivially outperforms all baseline accelerators even on the compact models (i.e., MobileNetV2 and EfficientNet-B0) thanks to both the *SmartDeal* algorithm's higher compression ratio and the *SmartDeal* accelerator's dedicated and effective design (see Section V-B) of handling depth-wise CONV and squeeze-and-excite layers that are commonly adopted in compact models.

Fig. 8 shows the normalized number of DRAM accesses for the weights and input-output activations. We can see that: 1) the baselines always require more (1.1× to 3.5×) DRAM accesses than the *SmartDeal* accelerator, e.g., see the ResNet and VGG models on the ImageNet and CIFAR-10 datasets as well as the segmentation model DeepLabV3+ on the CamVid dataset; 2) *SmartDeal*'s DRAM-access reduction is smaller when the models' activations dominate the cost (e.g., compact DNN models); and 3) the *SmartDeal* accelerator can reduce the number of DRAM accesses over the baselines by up to 1.3× for EfficientNet-B0, indicating the effectiveness of our dedicated design for handling the squeeze-and-excite layers (see Section V-B).

b) Speedup over that of the baseline accelerators: Similar to benchmarking the *SmartDeal* accelerator's energy efficiency, we compare its latency of processing one image (i.e., batch size is 1) over that of the baseline accelerators on various DNN models and datasets, as shown in Fig. 9. We can see that the *SmartDeal* accelerator achieves the best performance under all the considered DNN models and datasets, achieving a latency improvement ranging from 8.8× to 19.2×. Again, this experiment validates the effectiveness of *SmartDeal*'s algorithm-hardware co-design effort to reduce the latency on fetching both the weights and the activations from the memories to the computation resources. Since the *SmartDeal* accelerator takes advantage of both the weights' *vector-wise* sparsity and the activations' *bit-level* and *vector-wise* sparsity, it has a higher speedup over all the baselines that make use of only one kind of sparsity. Specifically, the *SmartDeal* accelerator has an average latency improvement of 3.8×, 2.5×, and 2.0× over SCNN [37] and Cambricon-X [38], which consider unstructured sparsity, and Bit-pragmatic [73], which considers the *bit-level* sparsity in activations, respectively.

VII. Conclusion

The trade-off between the model performance and its speed/computational cost pervasively exists in the research community and industry. In this article, we propose a more smart trade-off strategy, *SmartDeal*, an algorithm-hardware co-design framework with a unique goal to trade higher cost memory storage/access for lower cost computation, in order to achieve storage- and energy-efficient DNN inference and training. Extensive experiments including both algorithm and hardware aspects show that *SmartDeal* can effectively resolve the practical bottleneck of high-cost memory storage/access and can aggressively trim down energy cost and model size, while incurring minimal accuracy drops, for both inference and training. Moreover, *SmartDeal* provides a handy way for the users to freely balance such trade-off by controlling hyperparameters in *SmartDeal* to favor either better performance or higher speed. Our immediate future goal is to automate the hyper parameter tuning in *SmartDeal* using AutoML.

References

[1] T. Chen *et al.*, "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGPLAN Notices*, vol. 49, no. 4, pp. 269–284, Feb. 2014.

[2] Y. Zhao *et al.*, "SmartExchange: Trading higher-cost memory storage/access for lower-cost computation," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit. (ISCA)*, May 2020, pp. 954–967.

[3] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2074–2082.

[4] Y. Wang *et al.*, "E2-train: Training state-of-the-art CNNs with over 80% less energy," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 5139–5151.

[5] H. You *et al.*, "Drawing early-bird tickets: Towards more efficient training of deep networks," 2019, *arXiv:1909.11957*.

[6] C. Li, T. Chen, H. You, Z. Wang, and Y. Lin, "Halo: Hardware-aware learning to optimize," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2020, pp. 500–518.

[7] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5687–5695, doi: 10.1109/CVPR.2017.643.

[8] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "SignSGD: Compressed optimisation for non-convex problems," 2018, *arXiv:1802.04434*.

[9] C. Kim, S. Kang, D. Shin, S. Choi, Y. Kim, and H.-J. Yoo, "A 2.1 TFLOPS/W mobile deep RL accelerator with transposable PE array and experience compression," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 136–138.

[10] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[11] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*.

[12] Y.-H. Chen, T.-J. Yang, J. S. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 2, pp. 292–308, Jun. 2019.

[13] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 442–450.

[14] H. Huang, L. Ni, K. Wang, Y. Wang, and H. Yu, "A highly parallel and energy efficient three-dimensional multilayer CMOS-RRAM accelerator for tensorized neural network," *IEEE Trans. Nanotechnol.*, vol. 17, no. 4, pp. 645–656, Jul. 2018.

[15] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.

[16] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6 TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 218–220.

[17] H. You *et al.*, "ShiftAddNet: A hardware-inspired deep network," 2020, *arXiv:2010.12785*.

[18] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[19] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Architecture (ISCA)*, Jun. 2016, pp. 243–254.

[20] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2654–2662.

[21] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[22] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 409–424.

[23] J. Shen, Y. Wang, P. Xu, Y. Fu, Z. Wang, and Y. Lin, "Fractional skipping: Towards finer-grained dynamic CNN inference," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 4, pp. 5700–5708, Apr. 2020.

[24] T.-K. Hu, T. Chen, H. Wang, and Z. Wang, "Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference," 2020, *arXiv:2002.10025*.

[25] Y. Wang *et al.*, "Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 623–633, May 2020.

[26] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7370–7379.

[27] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018.

[28] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," 2018, *arXiv:1802.05668*.

[29] M. Wang, Q. Zhang, J. Yang, X. Cui, and W. Lin, "Graph-adaptive pruning for efficient inference of convolutional neural networks," 2018, *arXiv:1811.08589*.

[30] S. Gui, H. Wang, C. Yu, H. Yang, Z. Wang, and J. Liu, "Adversarially trained model compression: When robustness meets efficiency," 2019.

[31] H. Wang, S. Gui, H. Yang, J. Liu, and Z. Wang, "GAN slimming: All-in-one GAN compression by a unified optimization framework," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2020, pp. 54–73.

[32] P. Micikevicius *et al.*, "Mixed precision training," in *Proc. Int. Conf. Learn. Represent.*, 2018. [Online]. Available: https://openreview.net/forum?id=r1gs9JgRZ

[33] B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda, and K. Keutzer, "Mixed precision quantization of ConvNets via differentiable neural architecture search," 2018, *arXiv:1812.00090*.

[34] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8612–8620.

[35] S. A. Tailor, J. Fernandez-Marques, and N. D. Lane, "Degree-quant: Quantization-aware training for graph neural networks," in *Proc. Int. Conf. Learn. Represent.*, 2021. [Online]. Available: https://openreview.net/forum?id=NSBrFgJAHg

[36] Y. Fu *et al.*, "Fractrain: Fractionally squeezing bit savings both temporally and spatially for efficient dnn training," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 12127–12139.

[37] A. Parashar *et al.*, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 27–40.

[38] S. Zhang *et al.*, "Cambricon-X: An accelerator for sparse neural networks," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2016, p. 20.

[39] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," 2014, *arXiv:1412.6115*.

[40] J. Wu, Y. Wang, Z. Wu, Z. Wang, A. Veeraraghavan, and Y. Lin, "Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions," 2018, *arXiv:1806.09228*.

[41] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," M.S. thesis, Univ. Toronto, Toronto, ON, Canada, 2009.

[42] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, Jun. 2009, pp. 248–255.

[43] H. Mao *et al.*, "Exploring the regularity of sparse structure in convolutional neural networks," 2017, *arXiv:1705.08922*.

[44] J. Yu, A. Lukefahr, D. Palframan, G. Dasika, R. Das, and S. Mahlke, "Scalpel: Customizing dnn pruning to the underlying hardware parallelism," *ACM SIGARCH Comput. Archit. News*, vol. 45, no. 2, pp. 548–560, 2017.

[45] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," 2018, *arXiv:1803.05407*.

[46] G. Yang, T. Zhang, P. Kirichenko, J. Bai, A. G. Wilson, and C. De Sa, "SWALP : Stochastic weight averaging in low-precision training," 2019, *arXiv:1904.11943*.

[47] K. Helwegen, J. Widdicombe, L. Geiger, Z. Liu, K.-T. Cheng, and R. Nusselder, "Latent weights do not exist: Rethinking binarized neural network optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7531–7542.

[48] X. Zhou *et al.*, "Cambricon-S: Addressing irregularity in sparse neural networks through a cooperative software/hardware approach," in *Proc. 51st Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2018, pp. 15–28.

[49] A. D. Lascorz *et al.*, "Bit-tactical: A software/hardware approach to exploiting value and bit sparsity in neural networks," in *Proc. 24th Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Apr. 2019, pp. 749–763.

[50] Z. Du *et al.*, "Shidiannao: Shifting vision processing closer to the sensor," *ACM Sigarch Comput. Archit. News*, vol. 43, no. 3, pp. 92–104, 2015.

[51] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proc. ACM/IEEE 43th Annu. Int. Symp.*, Jun. 2016, pp. 367–379.

[52] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[53] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.

[54] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *Proc. Eur. Conf. Comput. Vis.* Berlin, Germany: Springer, 2008, pp. 44–57.

[55] T. Chen *et al.*, "The lottery ticket hypothesis for pre-trained BERT networks," 2020, *arXiv:2007.12223*.

[56] T. Chen *et al.*, "The lottery tickets hypothesis for supervised and self-supervised pre-training in computer vision models," 2020, *arXiv:2012.06908*.

[57] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[58] (2020). N. Advances in Neural Information Processing Systems Workshop. *Micronet Challenge*. [Online]. Available: https://micronet-challenge.github.io/

[59] Xilinx Inc. *AvNet Ultra96*. Accessed: Sep. 1, 2019. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/1-vad4rl.html

[60] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2736–2744.

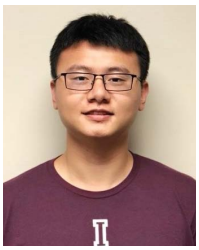
[61] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 5058–5066.

[62] R. Banner, I. Hubara, E. Hoffer, and D. Soudry, "Scalable methods for 8-bit training of neural networks," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 5151–5159.

[63] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018, pp. 7675–7684. [Online]. Available: https://papers.nips.cc/paper/7994-training-deep-neural-networks-with-8-bit-floating-point-numbers.pdf

[64] Y. Yang, S. Wu, L. Deng, T. Yan, Y. Xie, and G. Li, "Training high-performance and large-scale deep neural networks with full 8-bit integers," *Neural Netw.*, vol. 125, pp. 70–82, 2019.

[65] Z. Qin *et al.*, "Accelerating deep neural networks by combining block-circulant matrices and low-precision weights," *Electronics*, vol. 8, no. 1, p. 78, Jan. 2019.
[66] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," 2019, *arXiv:1902.08153*.
[67] F. Tung and G. Mori, "CLIP-Q: Deep network compression learning by in-parallel pruning-quantization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7873–7882.
[68] R. Gong *et al.*, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 4852–4861.
[69] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn treebank," *Comput. Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.
[70] X. Yang *et al.*, "Interstellar: Using Halide's scheduling language to analyze DNN accelerators," 2018, *arXiv:1809.04070*.
[71] NVIDIA. *NVIDIA Jetson TX2 Delivers Twicethe Intelligence to the Edge*. Accessed: Sep. 9, 2019. [Online]. Available: https://devblogs.nvidia.com/jetson-tx2-delivers-twice-intelligence-edge/
[72] NVIDIA. *NVIDIA Tesla V100 Tensor Core GPU*. Accessed: Sep. 9, 2019. [Online]. Available: https://www.nvidia.com/en-us/data-center/tesla-v100/
[73] J. Albericio *et al.*, "Bit-Pragmatic deep neural network computing," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2017, pp. 382–394.
[74] Synopsys. *PrimeTime PX: Signoff Power Analysis*. Accessed: Aug. 6, 2019. [Online]. Available: https://www.synopsys.com/support/training/signoff/primetimepx-fcd.html
[75] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.



Xiaohan Chen received the B.S. degree in applied mathematics from the University of Science and Technology of China, Hefei, Anhui, China, in 2017. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA.

His research interests include machine learning, sparse optimization, and sparse neural networks. He is currently focused on automated learning-based optimization, lottery ticket hypothesis, and efficient deep learning.

Yang Zhao received the B.S. and M.S. degrees from Fudan University, Shanghai, China, in 2012 and 2015, respectively. She is currently pursuing the Ph.D. degree with the Efficient and Intelligent Computing Laboratory (EIC), Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA.

Her current research interests include algorithm-hardware co-design for efficient DNN systems and computer architecture.

Yue Wang received the M.S. degree in electrical and computer engineering from Rice University, Houston, TX, USA, in 2020.

He is currently a Research Engineer with Ford Motor Company, Dearborn, MI, USA. His research concentrates on computer vision and efficient machine learning.

Pengfei Xu received the B.S. degree in information engineering from Shanghai Jiao Tong University, Shanghai, China, in 2017, and the master's degree in ECE from Rice University, Houston, TX, USA, in 2019. He is currently pursuing the Ph.D. degree in computer science with the University of California at Santa Barbara, Santa Barbara, CA, USA.

His research interests include deep learning and computer architecture.

Haoran You received the bachelor's degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China, in 2019. He is currently pursuing the Ph.D. degree in machine learning realm with the Electronic and Computer Engineering Department, Rice University, Houston, TX, USA.

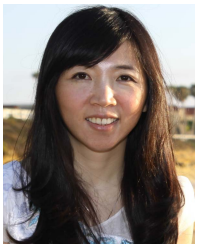
His research interests include resource-constrained machine learning, computer vision, deep learning, and algorithm/accelerator co-design.

Chaojian Li received the B.S. degree from Tsinghua University, Beijing, China, in 2019. He is currently pursuing the Ph.D. degree with the Efficient and Intelligent Computing Laboratory, Rice University, Houston, TX, USA, under the supervision of Dr. Y. Lin.

His research interests include neural architecture search, DNN deployment tools, and graph neural networks.

Yonggan Fu received the bachelor's degree in applied physics and computer science (dual major) from the School of Gifted Young, University of Science and Technology of China, Hefei, Anhui, China, in 2019. He is currently pursuing the Ph.D. degree with the Efficient and Intelligent Computing Laboratory, Electrical and Computer Engineering Department, Rice University, Houston, TX, USA, under the supervision of Dr. Y. Lin.

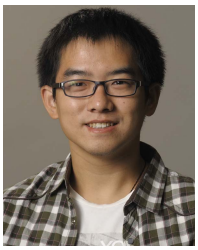
His current research interests include but are not limited to efficient deep neural network (DNN) training and Inference algorithms, vulnerability of machine learning systems and robust learning algorithms, and algorithm/hardware co-design for efficient and robust DNN deployment.



Yingyan Lin (Member, IEEE) received the Ph.D. degree in ECE from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2017.

She is currently an Assistant Professor with the Department of Electrical and Computer Engineering (ECE), Rice University, Houston, TX, USA. Her research focuses on embedded machine learning, which is to develop efficient algorithms, accelerators, and automated tools for enabling ubiquitous on-device intelligence and promoting green AI.

Dr. Lin was a recipient of a Best Student Paper Award at the 2016 IEEE International Workshop on Signal Processing Systems (SiPS 2016), the 2016 Robert T. Chien Memorial Award at UIUC for Excellence in Research, and was selected as a Rising Star in EECS by the 2017 Academic Career Workshop for Women at Stanford University. She received the NSF CAREER Award, the IBM Faculty Award, and the Facebook Research Award. She served as the Program Co-Chair of the 32nd IEEE International Conference on Application-specific Systems, Architectures, and Processors (ASAP 2021), and is currently a Track Chair for the ACM/EDAC/IEEE Design Automation Conference (DAC) and an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS.



Zhangyang Wang (Senior Member, IEEE) received the B.E. degree in EEIS from USTC, Hefei, Anhui, China, in 2012 and the Ph.D. degree in ECE from UIUC, Urbana, IL, USA, in 2016.

He was an Assistant Professor of CSE with TAMU, from 2017 to 2020. He is currently an Assistant Professor of ECE with UT Austin, Austin, TX, USA. His research interests are in the fields of machine learning, computer vision, optimization, and their interdisciplinary applications. His latest interests focus on automated machine learning (AutoML), learning-based optimization, machine learning robustness, and efficient deep learning.