A Speed- and Energy-Driven Holistic Training Framework for Sparse CNN Accelerators

Yuanchen Qu*, Yu Ma* and Pingqiang Zhou
School of Information Science and Technology, ShanghaiTech University, Shanghai, P. R. China
{quych1, mayu, zhoupq}@shanghaitech.edu.cn

***Abstract*—Sparse convolution neural network (CNN) accelerators have shown to achieve high processing speed and low energy consumption by leveraging zero weights or activations, which can be further optimized by finely tuning the sparse activation maps in training process. In this paper, we propose a CNN training framework targeting at reducing energy consumption and processing cycles in sparse CNN accelerators. We first model accelerator's energy consumption and processing cycles as functions of layer-wise activation map sparsity. Then we leverage the model and propose a hybrid regularization approximation method to further sparsify activation maps in the training process. The results show that our proposed framework can reduce the energy consumption of Eyeriss by 31.33%, 20.6% and 26.6% respectively on MobileNet-V2, SqueezeNet and Inception-V3. In addition, the processing speed can be increased by 1.96×, 1.4× and 1.65× respectively.**

***Index Terms*—CNN Accelerator, Training, Activation Map Sparsification, Energy Consumption, Processing Speed**

I. Introduction

Last decade has witnessed the development of convolution neural networks (CNNs). CNNs achieve fantastic accuracy in different inference tasks such as image classification and object detection. These neural networks usually contain large amount of parameters, thus bring significant computation burden to the hardware, especially the mobile devices. Large computation requirements also pose significant challenges to the energy consumption and processing time of the hardware. In order to reduce the computation complexity, researchers have proposed various design techniques such as compact convolution neural networks (compact CNNs) [1]–[3], reducing precision of weights or activations [4] and sparse convolution neural networks (sparse CNNs) [5]. Among them, sparse CNNs have become a hot topic in recent literature [6]–[8].

In sparse CNNs, people have proposed to explore either weight or activation (input) sparsity to reduce their computation complexity. At the algorithm level, weight sparsity can be achieved by pruning methods [5], [9]. Activation sparsity naturally occurs in neural networks because of the widely used rectified linear unit (ReLU) activation function. The activation sparsity varies with different inputs, and thus is often regarded as uncontrolled features and draws less attention. There is only one kind of methods to sparsify the activation map – utilizing L0 regularization of activation maps in the training process [10], [11]. As L0 regularization is non-differentiable and can not be optimized with back propagation training method, [10] utilizes L1 regularization to approximation L0 regularization. [11] uses Hoyer regularization approximation to further increase the overall activation map sparsity.

At the hardware level, customized sparse CNN accelerators have been developed to reduce the computation complexity of CNNs by skipping or gating the operations with zero weights or activations. Sparse CNN accelerators utilize special hardware techniques such as sparse tensor compression and zero detection scheme [12]–[14]. On contrary, general platforms such as CPU, GPU can hardly make fully use of such sparse features. Evaluation results show that, compared with general platforms, such accelerators do consume less energy consumption and achieve faster processing speed in sparse CNN inference tasks [15]–[17].

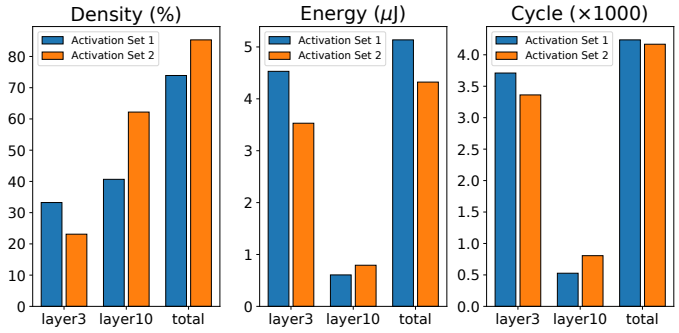

Fig. 1. Activation sparsity, energy consumption and processing cycles of layer 3 and layer 10 in MobileNet-V2, when applying different activations.

Although the state-of-the-art works [10], [11] try to develop activation sparsification methods to increase the overall activation sparsity, we find that the total sparsity doesn't work as an appropriate indicator of accelerator behaviours, such as energy consumption and processing speed in sparse CNN accelerators. We evaluate the impact of activation sparsity on different layers of MobileNet-V2 [2] by Timeloop [18], an accelerator simulator. Fig. 1 shows the accelerator's energy consumption and processing cycles of layer 3 and layer 10 fed by activations with different sparsity levels (marked as different colors). As shown in the figure, although the total activation density (indicated by 'Density', the percentage of non-zero values) of the blue one is smaller than that of the orange one, it still consumes more energy and takes more processing cycles in the accelerator. The reason is that the energy consumption in layer 3 is more likely to be influenced by the increasing activation sparsity. Such layer-wise difference is enlarged in compact CNNs which have convolution layers of various shapes

* These two authors contributed equally to this work.

This work was supported by the National Natural Science Foundation of China under the Grant No. 62074100.

978-3-9819263-7-8/DATE23/© 2023 EDAA

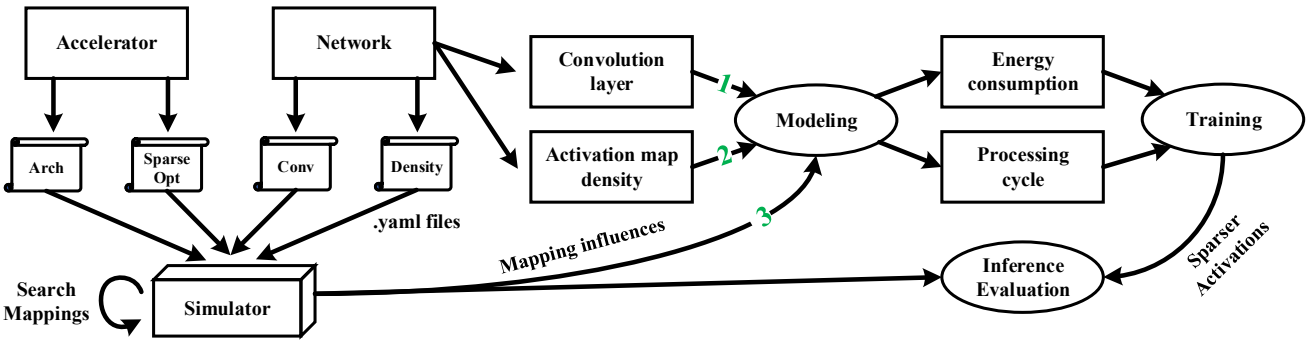

Fig. 2. The Proposed Framework.

and sizes, such as SqueezeNet [3] and MobileNet [19].

Motivated by the aforementioned observation, *we explore the layer-wise sparsity of a given CNN in the training process to reduce the energy consumption and also increase the speed of sparse CNN accelerators.* We propose a modeling method which translates the layer-wise activation sparsity in CNNs into measurable hardware improvements, such as the reduction of energy consumption and processing cycles. Our model not only calculates the relative computation cost across the layers, but also considers the mapping (dataflow) influences in accelerators. Then we utilize the proposed model in the training process, together with the improved hybrid L1-Hoyer regularization approximation method, to achieve layer-wise activation sparsification and evaluate the hardware implementation, aiming at reducing the energy consumption and processing cycles in sparse CNN accelerators. To the best of our knowledge, we are the first to optimize the accelerators by sparsifying layer-wise activations in the training process. The main contributions of this paper can be summarized as follows:

- We analyze the relationship between the activation map sparsity and the performance of CNN accelerators, and find that higher total sparsity of neural networks doesn't always leads to lower energy consumption and higher processing speed of CNN accelerators. Based on our analysis and observation, we propose a layer-wise sparse CNN training framework targeting at reducing the processing cycles and energy of accelerators when processing CNNs.
- In order to further improve the effect of our training framework, we propose a hybrid L1-Hoyer regularization method to improve the activation optimization capacity.
- We evaluate our proposed method on three different compact CNNs with Cifar-100 dataset. The experimental results show that our proposed method achieves $1.67X$ average processing speedup and 26.2% average energy reduction in Eyeriss [13], a sparse CNN accelerator.

The rest of the paper is organized as follows. Section II introduces our proposed modeling and training framework. The experiment results are shown in Section III. At last, we draw conclusions in Section IV.

II. The Proposed Method

A. Framework

As shown in Fig. 2, our proposed framework consists of two parts, modeling and training. For the modeling part, as illustrated by arrows 1 and 2, we exploit some network features, together with convolution layer types and layer-wise activation maps, to construct a primary model that captures the theoretical computation cost. Besides, we extract the influences of mapping (dataflow) and complete the modeling procedure, as illustrated by arrow 3. The modeling details are discussed in Section II-B.

After constructing the model of energy consumption and processing cycles with layer-wise activation, we further leverage the proposed model to finely sparsify the activation maps in the training part. We also propose a hybrid L1-Hoyer regularization method to support the back propagation of activation map sparsity. The details of the training process are presented in Section II-C.

B. Analysis and Modeling

The energy consumption and processing speed of CNN accelerators are mainly affected by three factors. The theoretical computational cost $Conv(l)$ evaluates the operation number of a single convolution layer l. Data mapping $Map(l)$ describes the influences when convolution layers are mapped on the accelerator hardware. The sparsity influence $Den(l)$ (short for 'Density') shows the sparsity related optimization in the accelerator. Theoretical factor $Conv(l)$ can be derived explicitly without the knowledge of accelerator hardware, while the other two factors are closely related to hardware designs and thus require the help of simulated results. In all, we combine these three factors to build our model, which evaluates the energy consumption and processing cycles of CNN accelerators. The model can be expressed as:

$$\begin{aligned} E &= \sum_{l=1}^{L} Conv(l) \cdot Map_E(l) \cdot Den_E(l) \\ C &= \sum_{l=1}^{L} Conv(l) \cdot Map_C(l) \cdot Den_C(l) \end{aligned} \tag{1}$$

where E and C represent total energy consumption and processing cycles of accelerators, l and L stand for the convolution layer index and the total number of the convolution layers.

We illustrate the meaning of the three factors and how to identify them, respectively.

Computation Cost Analysis: We first analyze the layer-wise differences of computation cost which stands for the operation

number of a single CNN layer. The computation cost directly influences the energy and processing cycles of the accelerator and distinguishes the differences among all the convolution layers in the same network.

Many parameters of CNN affect operation number thus change the computation cost. As shown in Equation 2, the computation cost ($Conv(l)$) is proportional to the output feature map height (H_{out}) and width (W_{out}), input channel (C_{in}), output channel (C_{out}), kernel height (H_{ker}) and width (W_{ker}) and batch size (B). Note that group convolution is commonly used in the middle bottleneck layer of some compact CNNs such as MobileNet [2], [19]. In group convolution layer, the input and output channels are split into multiple groups. Because the parameters C_{in} and C_{out} represent the input and output channel sizes in a single group, we should additionally multiply the number of the groups (G). To conclude, the theoretical computation cost $Conv(l)$ can be expressed as:

$$Conv(l) = H_{out} \cdot W_{out} \cdot C_{in} \cdot C_{out} \cdot H_{ker} \cdot W_{ker} \cdot B \cdot G \quad (2)$$

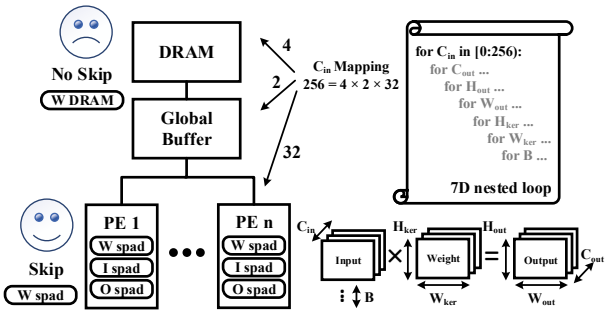

Fig. 3. Sparsity optimization, mapping example in Eyeriss [13].

Data Mapping Analysis: The second factor which influences the energy and cycles is the data mapping (or called dataflow), the way we schedule operations and data on the accelerator architecture. Take Eyeriss [13] as an example. As shown in Fig. 3, Eyeriss uses a three-level storage structure – DRAM, global buffer, and spad (buffer inside processing elements (PE)). A convolution problem can be described as a 7D nested loop over H_{out}, W_{out}, H_{ker}, W_{ker}, C_{in}, C_{out} and B [18]. Thus, to map the convolution problems, we should break down the detailed loops and allocate them to different levels of the storage. Fig. 3 shows an example allocating 256 input channels. We apply the number of loops to storage levels (4 for DRAM, 2 for global buffer, 32 for PE spad in this example).

Different data mapping results in different hardware optimization. Besides, there are also some mapping constraints to make sure the validity of current data mapping. Thus, searching for the optimal mapping is typically complicated. To characterize the mapping influences $Map(l)$, we use Timeloop to search for the best valid mapping result in the mapspace [18]. For a given accelerator architecture and convolution problem, Timeloop searches for the best mapping, which means the best way to schedule the convolution problem on the accelerator.

Sparsity Analysis: In addition to the computation cost and data mapping, sparsity of each layer also affects the total energy and processing cycles of accelerators. To model the impact of the activation map sparsity, we should figure out how the accelerator architecture leverages the sparse activation maps during processing. Sparse design optimization mainly focuses on skipping multiply-accumulate (MAC) operations, reducing access to storage unit whenever zero inputs are detected. To further evaluate this density related factor, we model Den as

$$Den_E(l) = e_1 \cdot D(l) + e_0, \;\; Den_C(l) = c_1 \cdot D(l) + c_0 \quad (3)$$

where$D(l)$ stands for the percentage of non-zero values in the l-th convolution layer, e_1, e_0, c_1, c_0 are accelerator hardware related parameters.

We use $e_1 \cdot D(l)$ to denote the energy patterns that tend to be reduced by the sparse optimization methods. On the contrary, there also exists inherent energy e_0 which is not influenced by the activation sparsity. As shown in the example of Eyeriss (Fig. 3), skipping the operation according to activation sparsity will directly reduce the energy consumption in MAC of PEs. In addition, accessing to several storage patterns such as the innermost weight storage (weight spad) can also be skipped. However, the access to the outermost weight storage (weight DRAM) remains the same, which contributes to the inherent energy consumption e_0.

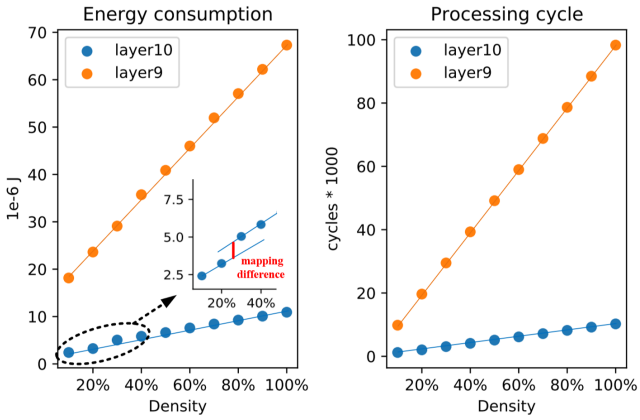

Fig. 4. Verify the model on Timeloop [18]: energy consumption (left) and processing cycles (right) in SqueezeNet.

The modeling of the processing cycles is similar to the ones of the energy. Note that the inherent processing cycles pattern c_0 is relatively small compared with that of energy consumption. The reason is that processing operations consume most of the cycles in the accelerator, while accessing to DRAM memory takes fewer cycles, resulting in smaller c_0.

Unlike $Conv(l)$ which can be determined by Equation (2), parameters such as $Map(l)$, e_1, e_0, c_1 and c_0 are related to the accelerator architecture and the mapping. Thus, we use Timeloop to help with the parameter settings and evaluate the correctness of our model. We determine these parameters by linear fitting of evaluated results. We use unpruned network to avoid the optimization influences brought by weight sparsity. For example, Fig. 4 shows the evaluated energy consumption and processing cycles of Eyeriss, fed by convolution layer 9 and layer 10 in Squeezenet [3]. Both the energy and processing cycles show linear relationship with activation map density. The slope of the fitting line in layer 9 is much larger than the one

in layer 10, which is mainly caused by the relative theoretical computation cost ($Conv(9)/Conv(10)$). In addition, the relative layer-wise mapping influence ($Map(9)/Map(10)$) also causes the differences across the layers.

Note that, the mapping influences not only contribute to the layer-wise differences, but also affect the linear relationship in a single layer. Taking layer 10 as an example, we enlarge several points in Fig. 4 to illustrate the mapping influences. When the density is lower than 30%, the best mapping allocates 32 input channels to the PE array (Fig. 3). However, when the density increases, the current mapping is invalid because the input channels applied to PE array extend the capacity of the restricted compression metadata buffer. In order to preserve valid mapping, we must allocate more input channels to the global buffer. Such mapping difference may cause small gaps marked as red in Fig. 3. However, the mapping difference doesn't affect the parameters much and could be ignored.

C. Sparse Activation Aware Training

The proposed model shows the layer-wise differences and motivates the sparse activation aware training. Based on the model as we constructed in Section II-B, we can modify the loss function in the training process targeting at reduction of energy consumption and processing cycles. We directly add the total energy and processing cycles into the loss function as

$$Loss(w) = Loss_0(w) + \frac{1}{N}\sum_{n=1}^{N}\sum_{l=0}^{L}\alpha E(x_{l,n}) + \beta C(x_{l,n}) \quad (4)$$

where w stands for the network weights, $Loss(w)$ and $Loss_0(w)$ denote the modified and original loss function, respectively. N denotes the number of training batches. $x_{l,n}$ represents the activation map of layer l with batch size n. α and β are preset parameters to balance the impact strength between energy and cycles.

To further apply the model in Equation (1), we drop out the inherent terms e_0 and c_0 which can not be optimized by activation sparsification. Then we rewrite the loss function as

$$Loss(w) = Loss_0(w) + \frac{1}{N}\sum_{n=1}^{N}\sum_{l=0}^{L}(\alpha e_l + \beta c_l)\lambda_l \|x_{l,n}\|_0 \quad (5)$$

where

$$e_l = Conv(l)\cdot Map_E(l)e_1,\ \ c_l = Conv(l)\cdot Map_C(l)c_1 \quad (6)$$

Note that activation map density is represented by L0 regularization, the number of non-zero values. λ_l denotes the regularization parameter of layer l.

Even though L0 regularization shows to be the best description of the activation map density, it is non-differentiable for further training approach. To approximate L0 regularization, researchers have utilized L1 regularization (the sum of non-zero values) or Hoyer regularization for activation sparsification. The derivative of L1 regularization is constant. Thus, L1 regularization achieves the same magnitude of sparsification of the whole activation map. On the other hand, Hoyer regularization, defined as $\|x\|_{Hoyer} = \frac{(\sum_{i=1}^{d}|x_i|)^2}{\sum_{i=1}^{d}(|x_i|)^2}$, has a decreasing derivative. The derivative around zero activation values is comparative large, while the derivative decreases to zero around larger activation values. Thus, Hoyer regularization tends to sparsify small values while preserve large values in order to prevent accuracy loss.

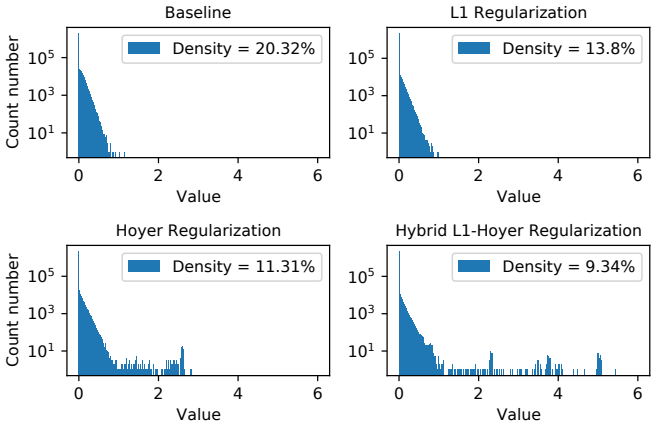

Fig. 5. Activation map distribution of a convolution layer (MobileNet-V2, layer 35) of different regularization approximation methods.

Normally, eliminating smaller elements in the activation map suffers from less accuracy penalty, because smaller elements contribute less to output patterns. Thus, Hoyer regularization shows better sparsification performance than L1 regularization with the same accuracy loss. However, Hoyer regularization is too conservative to sparsify layers with high level features, which have more large values. Such layers always have large input and output channel sizes, and thus gain more energy and cycles reduction compared with other layers. Therefore, in our work, we propose a hybrid regularization approximation, which strengthens the sparsification magnitude on such layers

$$\lambda_l\|x_{l,n}\|_0 \approx \lambda_l\|x_{l,n}\|_{Hybrid} = \lambda_l^{'}\|x_{l,n}\|_1 + \lambda_l^{''}\|x_{l,n}\|_{Hoyer} \quad (7)$$

Note that we break the original regularization parameter λ_l into two independent regularization parameters, $\lambda_l^{'}$ and $\lambda_l^{''}$. Then Equation (5) can be rewritten as:

$$Loss(w) = Loss_0(w) + \frac{1}{N}\sum_{n=1}^{N}\sum_{l=0}^{L}(\alpha e_l + \beta c_l)(\lambda_l^{'}\|x_{l,n}\|_1 + \lambda_l^{''}\|x_{l,n}\|_{Hoyer}) \quad (8)$$

The hybrid regularization approximation tends to sparsify smaller values and still preserves smaller sparsification magnitude on larger values. The impact of the proposed hybrid L1-Hoyer regularization approach is shown in Fig. 5. We select the layer 35 in the MobileNet-V2 as an example, which contributes to larger energy consumption and processing cycles than the majority of the layers in the network (See Fig. 6 in Section III). The proposed hybrid regularization preserves less large values than Hoyer regularization method. In addition, it still has a decreasing derivative and thus sparsifies more small values compared with L1 regularization method. To conclude, the proposed hybrid regularization method achieves the lowest activation map density without additional accuracy loss.

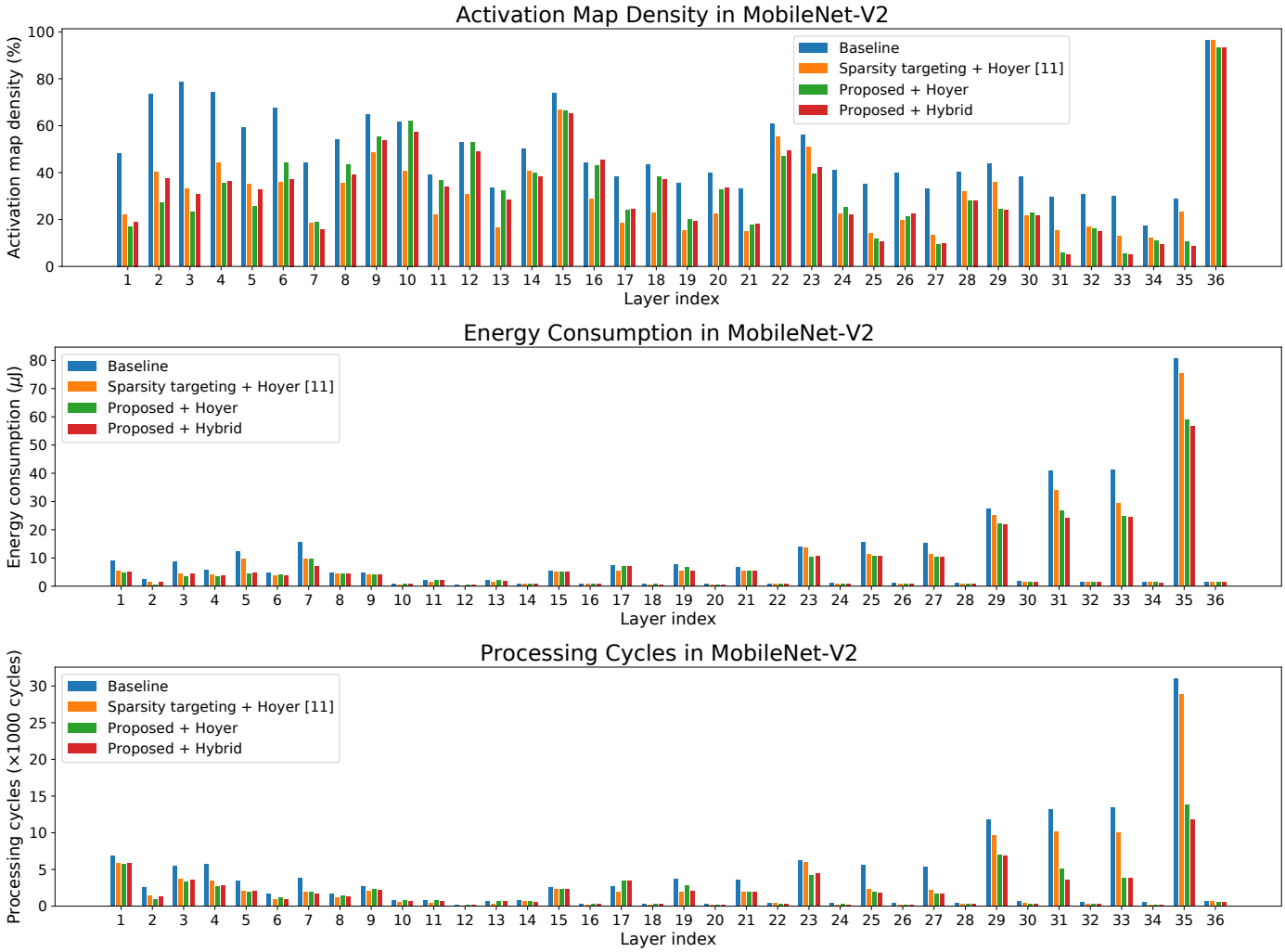

Fig. 6. Layer-wise results: activation map density (top), energy consumption (middle), processing cycles (bottom) in MobileNet-V2.

III. Experimental Results

A. Experiment Setup

In our experiments, we implement our performance-aware training framework in Pytorch. To evaluate the energy consumption and processing cycles on accelerators, we use Timeloop [18], an NN accelerator simulator supporting sparsity optimization evaluation. Timeloop has shown to be an accurate NN accelerator simulator which achieves less than 8% error on different types of popular cited accelerators such as Eyeriss serious [13], [15], SCNN [16], DSTC [20], compared with the baseline designs [21].

We carry our experiments on three commonly used compact CNNs, MobileNet-V2 [2], SqueezeNet [3] and Inception-V3 [22]. These networks cover different types of convolution layers and a large range of network sizes. Some over-parameterized CNN such as AlexNet [23] and VGG [24] are shown easy to compress, so they are not evaluated in our work. The experiments are carried on CIFAR-100 dataset [25]. Cifar is shown to be the most commonly used deep learning computer vision datasets [26]. Related work [10] shows similar activation sparsification effects among different datasets, such as MINST, Cifar and ImageNet. Thus, Cifar is capable enough to evaluate our framework. In the future work, we plan to evaluate our experiments in ImageNet, whose training takes longer time (days) compared with Cifar (hours).

As for the accelerator, we use the popular accelerator Eyeriss which leverages activation map sparsity in their architecture design. However, our proposed framework doesn't rely on specific networks and accelerators and can handle most of the cases.

B. Evaluation of Our Method

In this section, we present the effectiveness of our method. We compare the following four approaches:

- the *baseline* network without any sparsification method,
- the *sparsity targeting* method which sparsifies the average activation in the network [11],
- our proposed method which aims at sparsifying the energy consumption and processing cycles in the accelerator, with Hoyer regularization approximation method,
- our proposed method which aims at sparsifying the energy consumption and processing cycles in the accelerator, with L1-Hoyer hybrid regularization approximation method.

Fig. 6 shows layer-wise results of MobileNet-V2. The top most subfigure shows the activation map density across 36

sparse convolution layers in the MobileNet-V2. The sparsity targeting method achieves the best result in highest activation sparsity. However, as discussed in Section I, the average sparsity may not be directly related with the overall hardware performance. The next two figures show the evaluated energy consumption and processing cycles, respectively. We can observe that the convolution layers with even indexes consume much less energy and processing cycles. These convolution layers correspond to the group convolution layers in the bottleneck architecture [2], whose channel size is 1. Thus, the other layers with even indexes are more sensitive to the sparsification of the activation map. Our proposed method guides the training process to strengthen the magnitude of sparsifying even layers. In summary, our proposed method (marked as green) achieves additional 9.8% energy reduction and $0.31X$ processing speed across all convolution layers, compared with the sparsity targeting method (marked as orange).

The comparison between the last two methods (marked as green and red in Fig. 6) shows the effects of proposed hybrid L1-Hoyer regularization approximation approach. As mentioned in Section II-C, Hoyer regularization are conservative to cut off larger values in the activation map. Meanwhile, our proposed hybrid regularization method is more aggressive to cut off such larger values and achieves less density in last few layers, which reduces much more energy consumption and processing cycles. In all, our proposed hybrid L1-Hoyer regularization approximation method additionally achieves 1.5% energy reduction and $0.05X$ processing speed improvement.

TABLE I
COMPARISONS OF ENERGY AND PROCESSING SPEED.

	Sparsity targeting [11]		Proposed method with Hoyer		Proposed method with hybrid	
Network	Energy	Speed	Energy	Speed	Energy	Speed
MobileNet-V2	80.0%	$1.61X$	70.2%	$1.92X$	$\mathbf{68.7\%}$	$\mathbf{1.97X}$
SqueezeNet	88.0%	$1.16X$	80.1%	$1.37X$	$\mathbf{79.4\%}$	$\mathbf{1.40X}$
Inception-V3	82.6%	$1.35X$	74.6%	$1.59X$	$\mathbf{73.4\%}$	$\mathbf{1.65X}$

Due to the limitation on paper length, layer-wise details of SqueezeNet and Inception-V3 are not presented in this paper. Table I summarizes the results on these three networks, applied with sparsity targeting method [11] and our proposed method. All the results are compared with the baseline. Note that for both methods, the regularization parameters are tuned, which adds an intensive search over the parameter space to ensure similar and acceptable accuracy loss (less than 1% Top1 accuracy loss compared with the pre-trained model). As shown in Table I, SqueezeNet is harder to sparsify compared with the other two networks. Inception-V3 network has complicated network architectures with much more various sizes of convolutions. Our proposed method achieves $1.67X$ average processing speedup and 26.2% average energy reduction in all these three compact CNNs, which shows to be more effective compared with sparsity target method.

IV. CONCLUSION

In this paper, we analyze the relationship between layer-wise activation map density and hardware improvements of sparse CNN accelerators. Based on the analysis, we propose a sparse activation aware training framework targeting at reducing the processing cycles and energy consumption for accelerators executing CNNs. We evaluate our method on three compact CNNS, MobileNet-V2, SqueezeNet and Inception-V3 using Timeloop. Results show that our proposed training framework can achieve 31.33%, 20.6% and 26.6% energy reduction and $1.96X$, $1.4X$, $1.65X$ speedup respectively compared with the baseline models. In our future work, we will evaluate our proposed method on larger dataset, ImageNet, with more accelerator architectures.

REFERENCES

[1] C. Szegedy *et al.*, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[2] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[3] F. N. Iandola *et al.*, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *CVPR*, 2016.

[4] A. Zhou *et al.*, "Incremental network quantization: Towards lossless cnns with low-precision weights," *ICLR*, 2017.

[5] S. Han *et al.*, "Learning both weights and connections for efficient neural network," *NeurIPS*, vol. 28, 2015.

[6] B. Li *et al.*, "Sctn: Sparse convolution-transformer network for scene flow estimation," in *AAAI*, vol. 36, no. 2, 2022, pp. 1254–1262.

[7] L. Song *et al.*, "Serpens: a high bandwidth memory based accelerator for general-purpose sparse matrix-vector multiplication," in *DAC*, 2022, pp. 211–216.

[8] S. Feng *et al.*, "Cosparse: A software and hardware reconfigurable spmv framework for graph analytics," in *DAC*. IEEE, 2021, pp. 949–954.

[9] W. Wen *et al.*, "Learning structured sparsity in deep neural networks," *NeurIPS*, vol. 29, 2016.

[10] Georgiadis *et al.*, "Accelerating convolutional neural networks via activation map compression," in *CVPR*, 2019, pp. 7085–7095.

[11] M. Kurtz *et al.*, "Inducing and exploiting activation sparsity for fast inference on deep neural networks," in *ICML*, 2020, pp. 5533–5543.

[12] S. Dave *et al.*, "Hardware acceleration of sparse and irregular tensor computations of ml models: A survey and insights," *Proceedings of the IEEE*, vol. 109, no. 10, pp. 1706–1752, 2021.

[13] Y.-H. Chen *et al.*, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *JSSC*, vol. 52, no. 1, pp. 127–138, 2016.

[14] J.-F. Zhang *et al.*, "Snap: A 1.67—21.55 tops/w sparse neural acceleration processor for unstructured sparse deep neural network inference in 16nm cmos," in *VLSIC*. IEEE, 2019, pp. C306–C307.

[15] Y.-H. Chen *et al.*, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *JESTCS*, vol. 9, no. 2, pp. 292–308, 2019.

[16] A. Parashar *et al.*, "Scnn: An accelerator for compressed-sparse convolutional neural networks," *ACM SIGARCH*, vol. 45, no. 2, pp. 27–40, 2017.

[17] C. Deng *et al.*, "Gospa: an energy-efficient high-performance globally optimized sparse convolutional neural network accelerator," in *ISCA*. IEEE, 2021, pp. 1110–1123.

[18] A. Parashar *et al.*, "Timeloop: A systematic approach to dnn accelerator evaluation," in *ISPASS*. IEEE, 2019, pp. 304–315.

[19] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CVPR*, 2017.

[20] Y. Wang *et al.*, "Dual-side sparse tensor core," in *ISCA*. IEEE, 2021, pp. 1083–1095.

[21] Y. N. Wu *et al.*, "Sparseloop: An analytical approach to sparse tensor accelerator modeling," *MICRO*, 2022.

[22] C. Szegedy *et al.*, "Rethinking the inception architecture for computer vision," in *CVPR*, 2016, pp. 2818–2826.

[23] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," *NeurIPS*, vol. 25, 2012.

[24] K. Simonyan *et al.*, "Very deep convolutional networks for large-scale image recognition," *CVPR*, 2014.

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

[26] D. Macêdo *et al.*, "Enhancing batch normalized convolutional networks using displaced rectifier linear units: A systematic comparative study," *Expert Systems with Applications*, vol. 124, pp. 271–281, 2019.