An Energy-Efficient Inference Method in Convolutional Neural Networks Based on Dynamic Adjustment of the Pruning Level

MOHAMMAD-ALI MALEKI, ALIREZA NABIPOUR-MEYBODI, and MEHDI KAMAL, University of Tehran
ALI AFZALI-KUSHA, University of Tehran and Institute for Research in Fundamental Sciences
MASSOUD PEDRAM, University of Southern California

In this article, we present a low-energy inference method for convolutional neural networks in image classification applications. The lower energy consumption is achieved by using a highly pruned (lower-energy) network if the resulting network can provide a correct output. More specifically, the proposed inference method makes use of two pruned neural networks (NNs), namely mildly and aggressively pruned networks, which are both designed offline. In the system, a third NN makes use of the input data for the online selection of the appropriate pruned network. The third network, for its feature extraction, employs the same convolutional layers as those of the aggressively pruned NN, thereby reducing the overhead of the online management. There is some accuracy loss induced by the proposed method where, for a given level of accuracy, the energy gain of the proposed method is considerably larger than the case of employing any one pruning level. The proposed method is independent of both the pruning method and the network architecture. The efficacy of the proposed inference method is assessed on Eyeriss hardware accelerator platform for some of the state-of-the-art NN architectures. Our studies show that this method may provide, on average, 70% energy reduction compared to the original NN at the cost of about 3% accuracy loss on the CIFAR-10 dataset.

CCS Concepts: • **Computer systems organization** → **Architectures**; **Other architectures**; **Neural networks**; • **Computing methodologies** → **Artificial intelligence**; **Computer vision**; **Computer vision problems**; **Object recognition**;

Additional Key Words and Phrases: Pruned neural network, online management, energy efficiency, DNN, image classification

ACM Reference format:
Mohammad-Ali Maleki, Alireza Nabipour-Meybodi, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram. 2021. An Energy-Efficient Inference Method in Convolutional Neural Networks Based on Dynamic Adjustment of the Pruning Level. *ACM Trans. Des. Autom. Electron. Syst.* 26, 6, Article 49 (July 2021), 20 pages.
https://doi.org/10.1145/3460972

Authors' addresses: M.-A. Maleki, A. Nabipour-Meybodi, and M. Kamal, School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran 19967-15433, Iran; emails: {m.a.maleki, nabipour, mehdikamal}@ut.ac.ir; A. Afzali-Kusha, School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran 19967-15433, Iran and Institute for Research in Fundamental Sciences, Tehran 19538 - 33511, Iran; email: afzali@ut.ac.ir; M. Pedram, Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089, USA; email: pedram@usc.edu.

1084-4309/2021/07-ART49 $15.00
https://doi.org/10.1145/3460972

1 INTRODUCTION

Deep neural networks (DNNs) are currently used in many applications that are using artificial intelligence. Self-driving cars [1], cancer detection from medical images [2], speech recognition [3], and automatic machine translation [4] are a few examples that heavily rely on DNNs. In recent years, an ever-increasing demand for higher accuracy in artificial intelligence and naturally in DNN has been observed mainly for two reasons. First, increasing the applicability of DNNs to a broad range of real-life applications and our frequent use of these applications requires accuracy levels that meet or exceed human expectations. Second, the availability of more powerful computational platforms has made the implementation of deeper and more complex networks practical, enabling us to obtain higher accuracy. Nevertheless, deeper networks need more computations and memory accesses in general [5, 6], which in turn leads to higher energy consumption. However, the limited power budget of embedded systems prohibits the deployment of such networks on these systems. In addition, the energy consumption of large and deep **neural network (NN)** models in enterprise settings (e.g., data centers and GPU clusters) has a direct impact on the operating cost of such enterprises mainly due to the dollar cost of electrical energy consumption [7]. This high cost has forced technology firms to move toward domain-specific hardware solutions to accelerate their NN computations while simultaneously improving their energy efficiency [6, 7].

The solutions to lower the energy consumption in NNs range from system-level algorithm optimization and compilation down to circuit- and device-level optimizations. Because typically the energy consumption of the memory accesses is the dominant part of the total energy consumption of the hardware implementation of the DNNs, encoding and compressing weights of filters (set of weights placed in a 2D structure) and input feature maps are the techniques that have received much attention from researchers (e.g., see [8–10]). Changing the dataflow and reordering the computations are also techniques that have been also proposed to reduce the memory accesses and thereby increase energy efficiency [5, 6]. In addition, reducing the bitwidth for the weights and/or activations is another method resulting in energy reduction for both memory accesses and computations [11–14]. The latter method can be applied to a vast range of platforms on which DNNs run. Truncation and quantization are two effective ways of bitwidth reduction. However, the intrinsic error-tolerance characteristic of NNs allows hardware designers to use approximate computing methods to achieve higher performance and lower energy consumption compared to the exact computing methods [15, 16].

Although larger and more structured networks have been utilized for complex applications, because such networks generally tend to achieve higher-quality results, not all parameters (weights) are intrinsically important for reaching a target accuracy figure. In other words, it is possible to remove some weights without any accuracy loss [6]. Removing weights leads to higher performance and lower energy consumption, and can be helpful in avoiding overfitting to the training data. Indeed, it has been shown that increasing the number of parameters may result in an overfitting problem and that pruning some of the weights can mitigate this problem [17]. Pruning is also has been observed in the mammalian brain where highly connected synapses in an infant's brain are gradually pruned during human growth, leading to a more efficient and capable adult brain [18, 19].

The pruning approach is accompanied by some amount of accuracy loss, limiting its utility in some applications. The loss depends strongly on the features of the input data. To maintain a target accuracy level, hence the number of pruned weights should be determined by the worst-case input data, which tends to limit the energy efficiency that is achievable through pruning. To enhance the energy savings through pruning, we propose an inference method, for image classification applications, where the pruning is performed based on the input data. In this method, a properly

pruned network, for performing the inference phase, is dynamically chosen during the runtime of the system. More specifically, two levels of pruning—mild pruning (with no or negligible accuracy loss but small energy reduction) and aggressive pruning (with large accuracy loss but high energy reduction)—are considered. By extracting features of the input data, a proper pruning level is chosen. To reduce performance and energy overheads of the dynamic selection approach, the feature selection for dynamic adjustment of the pruning level is shared by the feature selection (convolutional layers) of an aggressively pruned NN. This method, which results in a considerable energy reduction compared to the worst-case design, is independent of the employed pruning technique and may be utilized in any DNN hardware accelerator.

The rest of the article is organized as follows. In Section 2, some state-of-the-art methods for increasing the energy efficiency and performance of NNs are reviewed. Section 3 describes the motivation behind the proposed method, whereas in Section 4 the structure of the method is elaborated. The efficacy of the proposed method is evaluated in Section 5, and, finally, the article is concluded in Section 6.

2 RELATED WORKS

In this section, some approaches for improving the speed and reducing the energy consumption of the DNN inference phase are reviewed. Using image encoding algorithms, such as JPEG, on weights/filters have been suggested in the work of Ko et al. [8]. In the work of Guan et al. [9], all parameters (using the encoding algorithms like LZ77 and Huffman) have been compressed offline to reduce the required bandwidth for loading the weights from off-chip DRAM during the inference phase.

To reduce the size of storage needed for weights, a technique based on the combination of pruning, quantization, and encoding was suggested by Han et al. [10]. In the pruning phase, small value weights below a threshold were eliminated while the remaining weights were quantized. After applying each of these two steps, retraining was conducted to compensate for the accuracy loss. In the end, Huffman encoding was applied to all remaining weights. The use of approximate computing for large-scale NNs has been discussed in the work of Venkataramani et al. [20]. In this work, the neurons were divided into sensitive and resilient groups where the approximate computing technique was applied to the resilient ones. In the work of Venkataramani et al. [20] and Wang et al. [21], a reconfigurable neural architecture based on utilizing approximate computing was proposed. The technique made use of both software-level and hardware-level knobs to construct a balance between the quality of service and the quality of result defined by the user.

Caching the intermediate data for reducing the need for intermediate data storage has been proposed by Alwani et al. [22]. By merging the processing of multiple **convolutional neural network (CNN)** layers as well as modifying the input data fetch order, it has become possible to cache the intermediate data between the computations of adjacent CNN layers. The Eyeriss accelerator minimized data movement between off-chip and on-chip memory by proposing row-stationary dataflow [23]. The proposed dataflow approach optimized the energy efficiency by reducing expensive data movement, such as DRAM accesses, as well as maximizing data reuse in local memory near the processing elements.

The binary-weight network was introduced by Rastegari et al. [11]. In this network, by approximating the filters with binary values, a memory savings of 32× was achieved. This network is energy efficient; however, it suffers from low accuracy in large NNs. In the work of Rastegari et al. [11] and Zhou et al. [212], in addition to lowering the required bitwidth for the weights and activations in the inference phase, to accelerate the training of the NNs, the quantization of the gradients of the parameters in the backward phase was suggested. A method to train quantized NNs with extremely low precision weights and activation was introduced by Hubara et al. [13]. In the work

of Zhou et al. [14], the incremental network quantization method to convert any pretrained full-precision CNN into a low-precision one whose weights were constrained to a power of 2 or 0 was proposed.

To fit large networks to embedded systems, a three-step method for pruning the weights was suggested by Han et al. [24]. The method began by determining the important weights followed by pruning the unimportant ones. In the last step, the network was retrained to compensate for the accuracy loss resulting from removing the weights. In the work of Yang et al. [25], first, the network layers were prioritized based on their energy consumptions. Then, to reduce the probable accuracy loss, the weight pruning was started from the layer with the highest energy consumption while the remaining weights were locally fine-tuned by the least-square optimization on each filter. This process was continued until no layer was left unpruned. At this point, a global fine-tuning was conducted on all weights to further recover the loss in accuracy.

In the work of Molchanov et al. [26], a weight pruning approach in transfer learning was suggested. To estimate the effect of each weight omitting in the loss function, the method exploited a first-order approximation of the network loss function using Taylor expansion. Employing this ranking method and pruning the least important neurons followed by a fine-tuning of the remaining weights, the pruned network was obtained. A structured sparsity learning, in which the topology of the network was optimized, was suggested by Wen et al. [27]. Starting from a large-enough network size, the proposed network removed the unnecessary layers with respect to the ℓ_2-norm. After this step, each layer was explored for more possible filter pruning followed by channel pruning and filter shape learning of remaining filters. These steps also used ℓ_2-norm as their pruning criterion.

The pruning approach suggested by Ding et al. [28] pruned the convolutional features of a pretrained network with respect to the ℓ_1-norm ranking. The pruning rate was determined based on a suggested algorithm in a way that the accuracy did not violate the predefined accuracy constraint. In the work of Ding et al. [28] and Li et al. [29], different criteria for layer-by-layer pruning of the filters including ℓ_1-norm, ℓ_2-norm, mean, and mean of standard deviations of filters were explored. Based on this exploration, it was shown that ℓ_1-norm was a plausible choice due to its data independency. Moreover, different pruning rates, which were considered for different NN layers, were determined and applied offline. An NN framework that consists of a large (with a high number of layers) and a small (with a few layers) NNs was suggested by Park et al. [30]. In this framework, first, the input data was fed to the small network, where, based on the output of its softmax layer, the need for using the large network for guaranteeing the higher accuracy was determined. Obtaining the small network architecture was not systematic where the designer had to select it from a set of small NNs (provided by the designer), making this approach impractical for DNNs.

By defining a confidence value after each convolutional layer in a network, the proposed method of Panda et al. [31] decided whether the passed convolutional layer was enough for deciding on the input class or if it should be passed to the successor layers. In this method, there were inputs that could be classified correctly at the early stages of the convolutional network leading to lower latency and energy consumption for the classification process. A novel network architecture was proposed by Panda et al. [31] and Huang et al. [32] that targets resource-efficient image classification applications. The proposed network uses a cascade of intermediate classifiers throughout the network and utilizes a feature representation at multiple scales along with dense connectivity to solve the lack of coarse-level features and interference of early classifiers with later ones, respectively. The main focus of Yu et al. [33] is on the training of a single network capable of adjusting the width of its layers online. In this manner, a proper network size (named *switches*) can adaptively be adopted on the fly in the runtime considering the response time or resource constraints. Although the proposed trained model is able to change its needed computations for classifying,

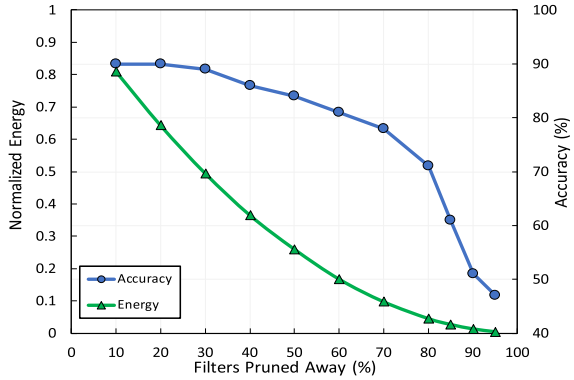

Fig. 1. The accuracy loss and energy reduction trend of VGG16 (on CIFAR-10) vs. percentage of the filters pruned away. The pruned network has undergone a retraining process for 10 epochs.

no solution for automatic selection of a proper network size during the runtime was suggested. A runtime pruning framework in which a recurrent NN (acts as a decision-maker) decides the level of pruning for each layer based on the input image was proposed by Lin et al. [34]. The training of the decision network consists of manual adjustment of some hyperparameters and deciding on some other parameters. The parameters include the number of the convolutional group that directly influences the possible pruning combinations and the difficulty level of reinforcement learning (used for the training of the decision-maker) by resizing the action space. A big/little DNN was proposed by Park et al. [35]. In this approach, first the little DNN is executed on all considered inputs. Then, the score margin metric of the output classification of the little DNN is extracted and compared with a threshold. This comparison is done by employing a binary classifier. The output of the classifier determines whether the execution of the big DNN is required or not to reach higher accuracy.

Contrary to the prior works, in which the inference phase is performed based on a predetermined level (amount) of pruning, we propose the idea of automatically choosing between two pruning levels, which leads to lower energy consumption. The selection, which is independent of the pruning method (as well as the NN architecture), is performed using the input data during the runtime. The proposed technique is a pruning-level classifier that determines the proper pruning level online based on the available pruned networks that can be obtained from any pruning method.

3 MOTIVATION

By increasing the amount of pruning, the accuracy of DNN is reduced where retraining the pruned network usually helps a partial accuracy recovery. As an example, Figure 1 shows the accuracy and normalized energy consumption (with respect to the unpruned network) of the VGG16 network trained on CIFAR-10 for different percentages of pruned filters (a.k.a. pruning points) for each layer. In this example, layer-by-layer pruning with respect to $\ell 1$-norm criteria was used. After pruning each layer, the network was retrained for 10 epochs. In this example, the accuracy drops after pruning 30% (30% as the pruning point) of the filters in each layer. In the filter pruning approach, all weights of the chosen filter are omitted.

Although by increasing the number of pruned filters the accuracy is reduced, there are still some inputs that the pruned network can correctly classify. For instance, in this study, with a filter pruning of 85%, 61% of the inputs still can be classified correctly by the pruned network. This means

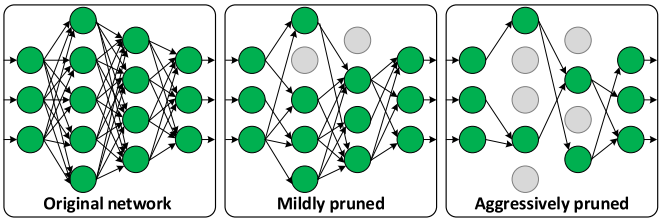

Fig. 2. The difference between the original, mildly, and aggressively pruned NNs.

that classification for these inputs can be faster and done with much lower energy consumption (about 75% lower energy compared to that of the original unpruned network). To take advantage of this point, the inputs should be classified into two distinct classes: inputs with the potential of being classified by the pruned network correctly (60% of the inputs in this example) and those that need an unpruned network for the correct classification (30% of the inputs). This has been the motivation for us to propose a method based on the input images that makes an online decision between using an aggressively pruned and a lightly pruned (or even unpruned) network to reduce the overall energy consumption of the inference phase. It should be mentioned that the remaining 10% (=100% − 60% − 30%) of the inputs cannot be classified by either the pruned or the unpruned networks.

4 PROPOSED INFERENCE METHOD

As discussed in the previous section, the accuracy of the pruned NN is data dependent, meaning it can respond correctly to a class of inputs. By separating and identifying these input types, it is possible to utilize the pruned network whenever this class of inputs is presented for the classification. Assuming that most of the inputs can be classified correctly with a (much smaller) pruned network, considerable energy savings may be achieved. For the rest of the article, for distinguishing different networks, the following terminology is used. The mildly pruned NN is the network that has been pruned to the point that there is no considerable accuracy loss (~1% in this work). In the provided example in Figure 1, this point is around 20%. The aggressively pruned NN is the network that has many more pruned filters compared to the mildly pruned NN. In the provided motivational example, this network is a network with more than 20% pruned filters. In addition, the untouched unpruned network is called the *original NN*. For our given example, Figure 2 demonstrates the differences between these networks.

The high-level structure of the proposed inference method is shown in Figure 3. Each input data is streamed to an input classifier that chooses the proper NN, either mildly or aggressively pruned NNs. The input classifier can be designed based on a range of possible solutions in statistics and machine learning algorithms. Because the target application is the image classification with CNNs, an NN-based classifier is a good choice. This network, which we call the **input classifier neural network (ICNN)**, should have two main characteristics. First, since the final accuracy of the proposed inference system depends strongly on ICNN accuracy, it should have a high accuracy in determining the appropriate pruned network for a given input. Second, its energy consumption should be considerably smaller than that of the mildly pruned NN. Therefore, the accuracy of the ICNN, as well as the energy consumptions of the ICNN, aggressively pruned and mildly pruned NNs, are the key factors determining the efficacy of the proposed inference method.

To extract the proper pruning level for the aggressively pruned NN, an effective pruning algorithm should be selected. The input dataset also must be prepared (labeled) for the training and

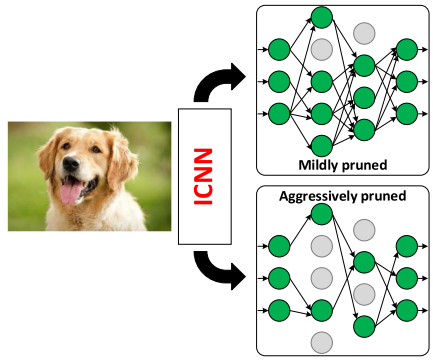

Fig. 3. The overall structure of the proposed inference method.

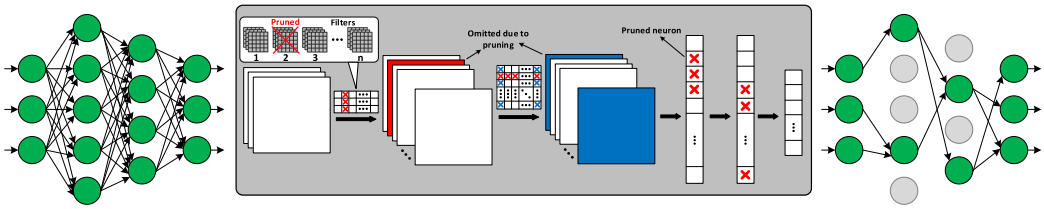

Fig. 4. The flow of pruning the filters in a sample two-layer convolutional network with two fully connected layers at the end as a classifier.

testing phases of the proposed system. In addition, the architecture of the ICNN should be determined. These issues are discussed in the following sections.

4.1 The Pruning Method

We choose the $\ell 1$-norm criterion for ranking the filters and pruning them accordingly [29]. Pruning the filters rather than weights results in a larger reduction in the number of parameters (network size), leading to lower energy consumption and higher performance due to lower memory accesses and computations. In addition, pruning a filter results in the removal of its corresponding output feature map in the next layer. The effect of filter pruning in two consecutive convolutional layers followed by two fully connected layers is depicted in Figure 4. The pruning is performed layer by layer (from the first to the last layer) followed by retraining with 10 epochs for recovering some of the accuracy loss. In the retraining process, parameters of the pruned layer and all of its predecessor layers become frozen, whereas other parameters will be fine-tuned freely. If there exists any feature in the inputs that possibly differentiate between members of the two input classes (i.e., low complexity and high complexity), the mentioned freezing in the retraining process will prevent excessive repeated changes to the filters that can detect these features in the inputs. Unlike the previous works on pruning (e.g., see [25–29]), to squeeze the network as much as possible, we continue the pruning to the fully connected layers. It should be emphasized that for the generation of the mildly and aggressively pruned networks, one may use any other pruning techniques.

4.2 Dataset Preparation

For training the side NN, in the training phase, the input dataset should be labeled as **low complexity input (LCI)** and **high complexity input (HCI)** based on the two available pruned networks for the inference. The label LCI (or HCI) determines that the input data should be injected

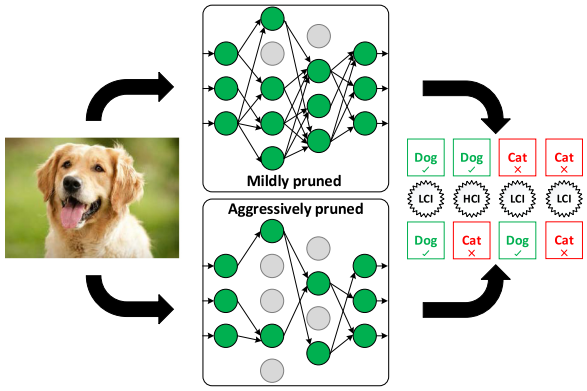

Fig. 5. Re-labeling the dataset for training the ICNN.

into the aggressively (or mildly) pruned NN. The preparation process of the ICNN training set is shown in Figure 5. After selecting the level of pruning for generating the aggressively pruned NN, the training dataset that is a part of the input dataset (e.g., ImageNet, CIFAR) is fed to the aggressively pruned and mildly pruned NN. If both networks recognize the input correctly, this input data is labeled as LCI. If only the mildly pruned NN classifies the input correctly, the input is labeled as HCI. When neither of the networks can classify the input, we use the label LCI to reduce the energy consumption in the inference phase, although the classification would be wrong anyway. Finally, if an input data is correctly categorized by only the aggressively pruned NN, it is labeled as LCI. After preparing the dataset, we should decide on the architecture of the ICNN (a.k.a. topology of the input classifier network). Finally, to prevent training from overfitting, in the training process, we considered balancing the dataset, adding drop-out layers, and tuning the hyperparameter, such as optimizer type, regularization parameters, learning rate, weight decay, and loss function.

4.3 The Architecture of the ICNN

It should be stressed that a low accuracy of the input classifier may result in lower accuracy for the whole system or an unnecessary increase in the energy consumption of the system. In the former case, the ICNN wrongly selects the aggressively pruned NN instead of the mildly pruned NN, whereas in the latter case, the mildly pruned NN is chosen in place of the aggressively pruned NN. However, using a larger ICNN for a higher accuracy that consumes more energy might eliminate the energy savings obtained through the proposed approach. This gives rise to a trade-off between the accuracy and energy consumption of the ICNN for an efficient design of the proposed inference method. As an example, Table 1 shows the accuracy of different network architectures considered as the ICNN for the CIFAR-10 dataset. As is observed from the figures, high accuracy is achieved at the cost of more convolutional layers and consequently higher energy consumption.

The convolutional layers of the aggressively pruned NN can extract the necessary features of the input dataset for differentiating between the two considered classes. This motivates us to employ these convolutional layers as the convolutional layers of the ICNN as well. More specifically, inputs go through these convolutional layers in the ICNN. If the ICNN points us to use aggressively pruned NN, the outputs of these layers will be used in the aggressively pruned network again without any need to perform the recomputation. Combining convolutional layers of the ICNN with those of the aggressively pruned NN leads to a considerable reduction in the energy overhead of the ICNN. In addition, to reduce the complexity of designing the ICNN, we use the

Table 1. Accuracy of the Input Classifier for Different Sample Architectures by the Modified Dataset

Model Name	Architecture**	Accuracy
LeNet	$C^5{}_{18}$/ M^2/ $C^3{}_{48}$/ M^2/ F_{360}/ F_{4096}/ F_2	65%
Residual LeNet	$C^5{}_{48}$/ M^2/ [$C^1{}_{48}$/ $C^3{}_{48}$/ $C^1{}_{48}$] / M^2/ F_{360}/ F_{4096}/ F_2	70%
ResNet10	$C^5{}_{48}$/ M^3/ [$C^1{}_{48}$/ $C^3{}_{48}$/ $C^1{}_{48}$]/ [$C^1{}_{48}$/ $C^3{}_{48}$/ $C^1{}_{48}$]/[$C^1{}_{48}$/ $C^3{}_{48}$/ $C^1{}_{48}$]/ M^2/ F_2	71%
VGG16	$C^3{}_{64}$/$C^3{}_{64}$/M^2/$C^3{}_{128}$/$C^3{}_{128}$/M^2/$C^3{}_{256}$/$C^3{}_{256}$/$C^3{}_{256}$/M^2/$C^3{}_{512}$/$C^3{}_{512}$/$C^3{}_{512}$/M^2/$C^3{}_{512}$/ $C^3{}_{512}$/ $C^3{}_{512}$/ M^2/F_{4096}/ F_{4096}/ F_2	82%

** C_x^y refers to the convolutional layer with x filters with the size of $y \times y$. M^K is a max-pooling layer with the size of $k \times k$. F_h points to a fully connected layer with h neurons. [C/C/C] defines an identity block.

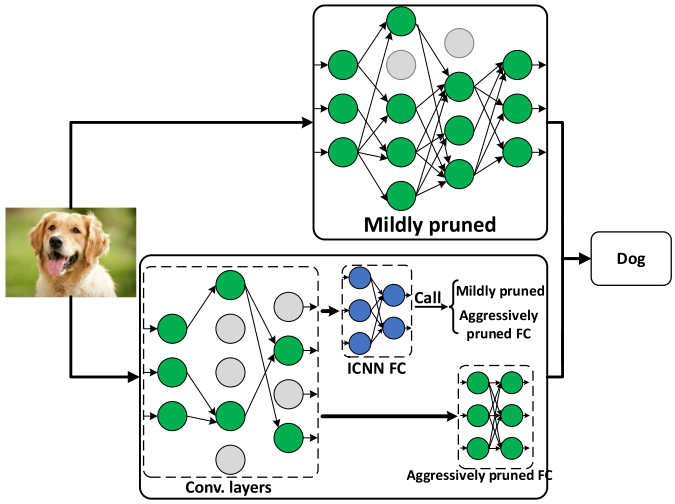

Fig. 6. The structure of the proposed inference method, composed of the input classifier and aggressively pruned networks merged together. FC refers to fully connected layers.

same number of fully connected layers and neurons in each layer, except for the last layer, as those of the aggressively pruned NN. For the last layer of the ICNN, we utilize two neurons for separating the two defined classes (HCI and LCI). Whenever this ICNN architecture classifies the input as LCI, the processing of the convolutional layers of the aggressively pruned NN has been completed by the ICNN while only the processing of its fully connected layers remains to be executed. Thus, the output of the last convolutional layer of the ICNN should be stored for being used for completing the processing of the aggressively pruned NN. The overall structure of our proposed inference method, in which most of the layers of the input classifier and aggressively pruned NNs are unified, is depicted in Figure 6. In the next section, an algorithm for designing the inference system is proposed.

4.4 Generating the Aggressively Pruned NN

As mentioned before, the pruning of the network provides some energy efficiency at the cost of some accuracy loss. Normally, the accuracy level should not go below a certain threshold determined by the application. This sets an optimization problem for the aggressively pruned NN with the objective of minimizing the energy consumption subject to a minimum expected accuracy. Alternatively, the problem may be formulated in a way that we do not consider any minimum accuracy limitation (maximum pruning level) and obtain the characteristic of the

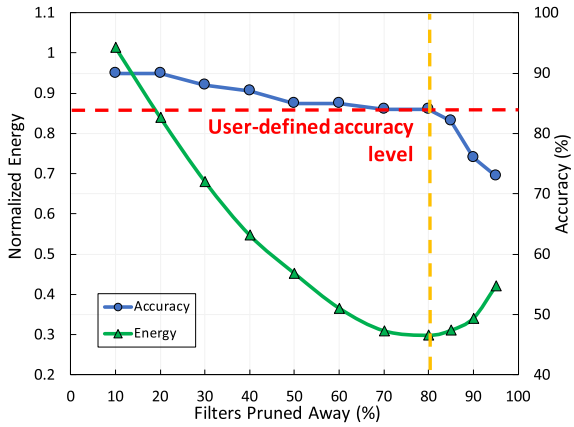

Fig. 7. The variations of the normalized energy as well as the accuracy of the proposed inference method under the pruning point sweeping.

energy consumption versus the pruning level. Since the characteristic has a convex behavior, there exists a minimum energy point that can be considered as the optimum pruning point from an energy consumption point of view (Figure 7). Therefore, as we will discuss in Section 5, we consider two scenarios to cover the aforementioned cases. It should be mentioned that the mildly pruned NN is devised before the aggressively pruned NN and independent from it. The accuracy of the mildly pruned network should be very close to that of the original NN. Next, we discuss the procedure for conceiving a proper aggressively pruned NN. In the first step, we should model the energy consumption of the proposed method for the inference phase. In this phase, the ICNN is called for each input data, whereas the mildly pruned NN (fully connected layers of the aggressively pruned NN) is (are) called when the ICNN classifies the input data as HCI (LCI). Thus, the overall energy consumption of our proposed system (E) is modeled as

$$E = E_{Conv_{IC,AP}} + E_{FC_{IC}} + (1-\beta)\, E_{FC_{AP}} + \beta \times E_{MP}, \tag{1}$$

where $E_{Conv_{IC,AP}}$ is the energy consumption of the set of the convolutional layers shared between the input classifier and the aggressively pruned NN, $E_{FC_{IC}}$ ($E_{FC_{AP}}$) is the energy consumption of the fully connected layers of the input classifier (aggressively pruned) NN, E_{MP} is the energy consumption of the mildly pruned NN, and, β is the probability that the ICNN classifies the input as HCI. Now, denoting the energy consumption of the aggressively pruned NN as E_{AP} ($E_{AP} = E_{Conv_{IC,AP}} + E_{FC_{AP}}$), the energy of ICNN as ($E_{IC} = E_{Conv_{IC}} + E_{FC_{IC}}$), and defining α_{AP} and α_{IC} as $\frac{E_{FC_{AP}}}{E_{AP}}$ and $\frac{E_{FC_{IC}}}{E_{IC}}$, respectively, (1) can be rewritten as

$$E = (1 + \alpha_{IC} - \beta\alpha_{AP})\, E_{AP} + \beta \times E_{MP}. \tag{2}$$

Since the energy consumption of the fully connected layers are considerably smaller than those of the convolutional layers in DNNs [5, 6], in (2), one can replace α_{IC} by α_{AP}. In addition, the amount of the energy consumption reduction of the aggressively and mildly pruned NNs with respect to the energy consumption of the original network (E_O) is a function ($f()$) of the chosen pruning point (φ).

$$E_{AP,MP} = f(\varphi) \times E_O \tag{3}$$

By considering φ_{AP} (φ_{MP}) as the amount of pruning for generating the aggressively (mildly) pruned NN and based on the preceding explanation, (2) can be rewritten as

$$E = [(1 + (1 - \beta)\,\alpha_{AP}) \times f\,(\varphi_{AP}) + \beta \times f\,(\varphi_{MP})]\,E_O. \tag{4}$$

In (4), for E_O, we used the energy consumption of the Eyeriss given in the work of Alwani et al. [22] and Chen et al. [23]. It should be reminded that in this work, we used Eyeriss as the platform for evaluating the energy consumption of the proposed inference technique. The parameter φ_{MP} is specified one time at the beginning of designing the proposed inference system. The parameter α_{AP} is determined after constructing the aggressively pruned NN. In addition, the parameter β is specified after training the ICNN when the network is being evaluated by the test portion of the input dataset in the testing phase. Since the ICNN utilizes the convolutional layers of the aggressively pruned NN, indeed β is determined after generating the aggressively pruned NN. To reach a higher energy efficiency without violating the target accuracy level, a proper value for φ_{AP} also should be selected. As an example, Figure 7 shows the normalized energy consumption (normalized to that of the original NN) and the accuracy of the proposed inference method versus φ_{AP} for the VGG16 network on the CIFAR-10 dataset. As expected, the accuracy of the inference system is reduced with increasing φ_{AP}. The accuracy degradation slope is lower for smaller φ_{AP} values. This is due to the fact that for large φ_{AP} values, the accuracy of the aggressively pruned NN degrades considerably.

In this work, to determine the proper (near-optimal) value for φ_{AP}, we suggest an exhaustive algorithm provided as Algorithm 1. The inputs of this algorithm are the original NN (denoted as O_NN), the mildly pruned NN (denoted as MP_NN), and the predefined minimum expected inference output

ALGORITHM 1: THE PROCEDURE OF PRUNING POINT SELECTION

1: **function** ***PRUNING_POINT_SELECTION*** (O_NN, φ_{MP}, ζ_{EXP}):
2: $MP\_NN \leftarrow$ pruning (O_NN, φ_{MP})
$i=0$
3: **for** (φ_{AP}=0.1;0.1;1):
4: $AP\_NN_i \leftarrow$ pruning (O_NN, φ_{AP})
5: Labeling the input images based on MP_NN and AP_NN_i
Generating IC_NN_i
6: Training the last FC layer of IC_NN_i
7: Estimating the accuracy (Acc_i) and energy (E_i) of the system
i++
8: **END**
9: **If** ($\zeta_{EXP} < Acc_i$)
10: return (AP_NN_i and IC_NN_i with lowest E_i)
11: **Else**
return (AP_NN_i and IC_NN_i with lowest E_i while $Acc_i > \zeta_{EXP}$)
END
12: **end function**

accuracy (denoted as ζ_{EXP}). The outputs of this algorithm are the ICNN (denoted as IC_NN) and the aggressively pruned NN (denoted as AP_NN). As mentioned before, the aggressively pruned network is designed for a given minimum expected inference output accuracy (when $\zeta_{EXP} \geq 0$). Thus, the accuracy of the proposed inference method should be estimated based on the accuracies

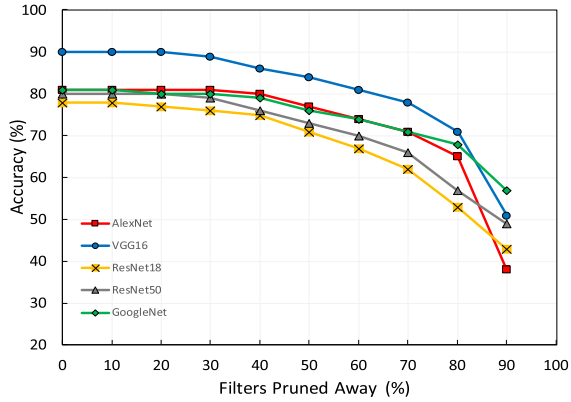

Fig. 8. The accuracy loss trends for different networks (on CIFAR-10) vs. different pruning percentages. The pruned network has undergone a retraining process for 10 epochs.

of the mildly and aggressively pruned NN as well as the probabilities of correct labeling of the input data by the ICNN. Therefore, we estimate the accuracy of the proposed system (Acc) as

$$Acc = \xi_{HH} Acc_{MP} + \xi_{LL} Acc_{AP} + \xi_{LH} Acc_{MP}, \tag{5}$$

where Acc_{MP} (Acc_{AP}) is the accuracy of the mildly (aggressively) pruned NN. ξ_{HH} and ξ_{LL} are the probabilities that the label of the input data is HCI or LCI (the labels are extracted during the dataset preparation) and the ICNN labels it correctly, and ξ_{LH} is the probability that the label of the input data is LCI, whereas the ICNN labels it incorrectly. The probabilities are estimated after applying the test part of the input dataset to the trained ICNN.

In the proposed algorithm, φ_{AP} is swept from 0 to 1 using steps of 0.1 where for each φ_{AP} value, the corresponding aggressively pruned and ICNNs are designed. Then, the ICNN is trained where, based on which, the accuracy and energy of the proposed inference method are estimated. After determining a set of aggressively pruned NNs and corresponding ICNNs, the best aggressively pruned NN and its corresponding ICNN meeting the accuracy constraint are returned. Finally, since almost all of the layers of the ICNN are the same as those in the aggressively pruned NN, only its fully connected layers should be trained. In addition, the retraining process of the aggressively pruned NN for increasing the accuracy as much as possible is set to 10 epochs, limiting the required processing time. The maximum runtime of this algorithm for the considered networks in this work was about 4 days on a system with Core i7 4790 as the CPU and a GeForce1080Ti GPU. Although the proposed method may be mapped on CPUs, GPUs, and hardware accelerators, the improvement level highly depends on how efficiently the networks are implemented and how effectively the platform can switch dynamically between the pruned networks. Compared to GPUs and CPUs, hardware accelerators are more appropriate from this aspect, and, hence, we chose Eyeriss as our DNN platform.

5 RESULTS AND DISCUSSION

5.1 Experimental Setup

To assess the effectiveness of the proposed inference method, seven well-known networks including AlexNet, VGG16, ResNet50, ResNet18, GoogLeNet, MobileNetV2, and EfficientNetB0 were employed on three different datasets of CIFAR-10, CIFAR-100, and Tiny ImageNet. The accuracies of these networks for different pruning percentages, when the images from the CIFAR-10 dataset were considered as the input, are shown in Figure 8. The networks were retrained after pruning.

Table 2. Accuracy, Energy Savings, and Pruning Points of the Networks in the Proposed Inference Method for the Considered Benchmarks Under the First Scenario for the CIFAR-10 Dataset

Model	Acc_O (%)	φ_{MP} (%)	Acc_{MP} (%)	φ_{AP} (%)	Reduction in # of AP MACs (%)	Acc_{AP} (%)	Acc (%)	Energy Savings (%)	
								Comp. to Original	Comp. to MP
AlexNet	81	30	80	85	97.7	53	79	73	46
VGG16	90	20	90	80	96.0	71	85	71	54
ResNet18	78	20	77	90	99.0	44	76	64	49
ResNet50	80	20	80	85	97.7	54	76	69	52
GoogLeNet	81	20	80	85	97.7	64	78	71	55

The results in this figure reveal that even for aggressive pruning of the filters, there are still some inputs that can be classified correctly (about 40% to 60% correct classification even when 90% of filters are pruned away in each layer). The amount of accuracy recovery by retraining depends on the network, pruning method, and the hyperparameters in the retraining procedure like the optimizer type, regularization parameters, learning rate, weight decay, loss function, and number of epochs. It also should be mentioned that in this work, all training and retraining procedures were performed from scratch where all considered hyperparameter values were derived from hyperparameter optimization techniques. As mentioned before, we have used Eyeriss as the platform for realizing the network where its chip energy model was employed for the estimation of the network energy consumption.

In this model, the energy was calculated as the sum of the energy consumed in each memory element in the memory hierarchy and the processing array. More specifically, based on the dataflow employed in Eyeriss (row stationary dataflow), the number of accesses to the memories in the memory hierarchy of the accelerator has been extracted; the energy of each access was reported in the work of Alwani et al. [22] and Chen et al. [23].

In addition, by counting the MAC operations, the energy consumption of the processing array has been determined. This requires modeling of the row stationary dataflow for which we have used an in-house-developed tool. The tool has been validated by the online tool presented by the Eyeriss team [36]. All training and testing was performed using Keras API on a machine with a Core i7 4790 CPU and a GeForce1080Ti GPU.

The output of the model provides energy/throughput/latency/performance of the input NN, whereas architectural components such as buffer sizes and processing array sizes, the configuration, and connection between processing elements are provided as inputs. These architectural inputs were kept unchanged during all simulations performed in this work (the exact values used in the work of Alwani et al. [22] are utilized). It also should be stressed that we used the same model for estimating the energy consumption of all networks including the original unpruned one and then normalized the energies to that of the original network.

5.2 Results

In this study, for determining the pruning point of the aggressively pruned NN (φ_{AP}) in the proposed inference method, two different scenarios were considered. In the first (second) scenario, the point with the lowest energy consumption without (with) considering any restriction on the accuracy loss (i.e., without (with) a predefined minimum expected inference output accuracy (ζ_{EXP}) was considered. In the case of the first scenario, Table 2 shows the accuracies and energy savings as well as the pruning points in the considered networks for the CIFAR-10 dataset.

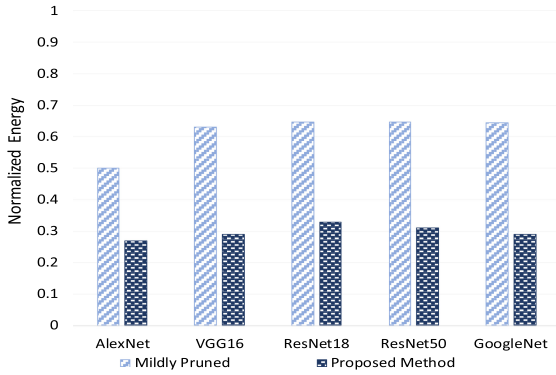

Fig. 9. The energy of mildly pruned and proposed system normalized to the energy of the original network for five different benchmarks on the CIFAR-10 dataset.

Table 3. Accuracy, Energy Savings, and Pruning Points of the Networks in the Proposed Inference Method for the Considered Benchmarks Under the Second Scenario for the CIFAR-10 Dataset

Model	φ_{AP} (%)	Reduction in # of AP MACs (%)	Acc_{AP} (%)	ζ_{EXP} (%)	Energy Savings (%)	
					Comp. to Original	Comp. to MP
AlexNet	85	97.7	53	79	73	46
VGG16	40	64	86	88	46	15
ResNet18	90	99	44	76	64	49
ResNet50	70	91	66	78	65	45
GoogLeNet	70	91	71	79	64	48

As the results show, for the best case, which belonged to AlexNet, the proposed method led to just 2% accuracy loss compared to the original network ($Acc - Acc_o$) while achieving 73% energy savings. On average, the proposed inference method resulted in 70% energy savings at the cost of up to 3% accuracy loss compared to those of the original networks. In addition, utilizing the proposed method resulted, on average, in a 51% reduction in energy consumption in the inference phase compared to the mildly pruned NN. The normalized energy consumption of the proposed system and mildly pruned NNs for the proposed inference method is plotted in Figure 9. On average, the energy consumption of the original NN was 3.3× and 2× larger than those of the aggressively and mildly pruned NN, respectively.

As the reported figures in Table 2 show, the value of the parameter φ_{MP} (φ_{AP}) in the considered benchmarks was around 20% (85%). The accuracy loss of the proposed method can be suppressed by allowing lower energy savings. To elaborate on this further, we considered the maximum accuracy loss of $loss_{EXP} = 2\%$ (compared to the accuracy of the original network) in the case of the second scenario. The results of evaluating the proposed method in this scenario are reported in Table 3, which indicates, on average, that 62% (40%) energy savings was achieved compared to that of the original (mildly pruned) NN.

Due to the considered accuracy loss limitation, the φ_{AP} value was, on average, about 71%, which was about 15% smaller than the value of φ_{AP} in the first scenario.

Figure 10 demonstrates the energy (normalized to the energy consumption of the original network) and the accuracy of the proposed inference method along with the selected pruning points in the considered first (❶) and second (❷) scenarios versus φ_{AP} ($loss_{EXP} = 2\%$). From these energy characteristics, one may observe that the energy consumption decreases continuously until a

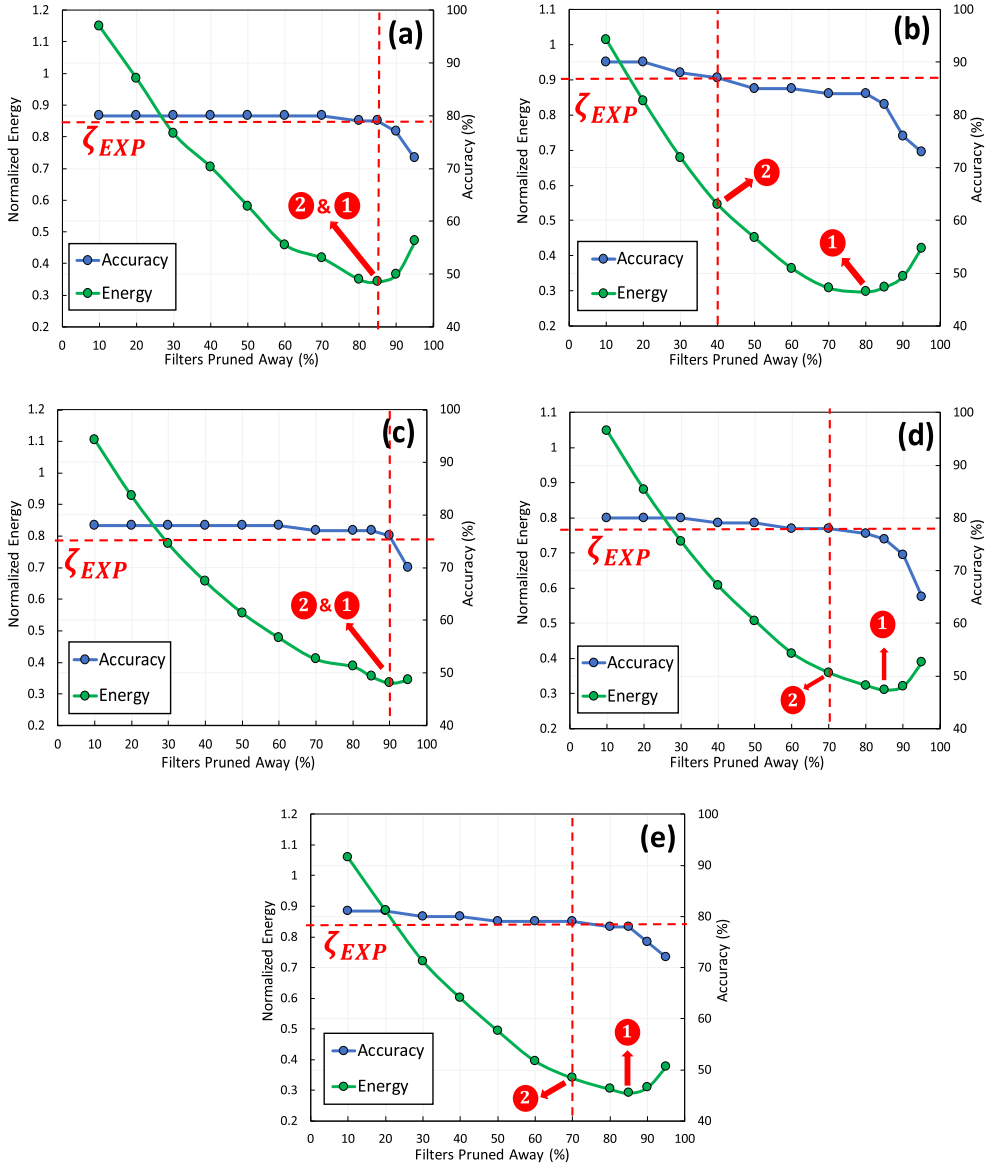

Fig. 10. The normalized energy and accuracy variation of the proposed inference method by sweeping φ_{AP} for AlexNet (a), VGG16 (b), ResNet18 (c), ResNet50 (d), and GoogLeNet (e) on the CIFAR-10 dataset.

minimum point reached where, after this point, the energy increases. This is attributed to the fact that overpruning of the network leads to a significant accuracy loss resulting in a large number of calling the mildly pruned NN.

The normalized latency of the proposed inference method (under the selected pruning points demonstrated in Figure 10), as well as the mildly pruned network with respect to that of the original network, is illustrated in Figure 11. As the results indicate, the proposed inference method leads to 68% (60%) and 51% (63%) latency reductions compared to the original unpruned and mildly pruned networks when the pruning point in the proposed inference method has been selected under the first (second) scenarios.

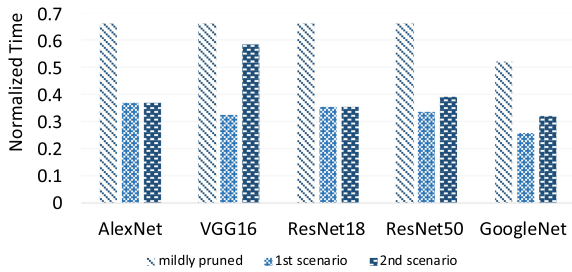

Fig. 11. The normalized latency of the mildly pruned and the proposed inference method in considered scenarios with respect to that of the original unpruned network for the CIFAR-10 dataset.

Table 4. Accuracy, Energy Savings, and Pruning Points of the Networks in the Proposed Inference Method for the Considered Benchmarks Under the First Scenario

Model	Dataset	Acc_O (%)	φ_{MP}(%)	Acc_{MP}(%)	φ_{AP}(%)	Reduction in # of AP MACs (%)	Acc_{AP} (%)	Acc (%)	Energy Savings (%)	
									Comp. to Original	Comp. to MP
VGG16	CIFAR-100	68.00	30	64	95	99.7	3	49.0	63	25
ResNet18	CIFAR-100	53.37	20	51	85	97.7	27	40.0	70	54
ResNet50	CIFAR-100	53.11	20	51	80	96.0	s33	42.0	69	52
GoogLeNet	CIFAR-100	59.65	10	58	80	96.0	27	48.0	48	36
MobileNetV2	CIFAR-100	70.95	30	70	85	97.7	27	57	66	32
EfficientNetB0	CIFAR-100	76.02	30	76	85	97.7	29	61	66	32
ResNet50	TinyImageNet	56.04	10	53	85	97.7	18	38.0	57	47
InceptionV3	TinyImageNet	58.00	10	57	90	99.0	5	42.5	48	35

Table 5. Accuracy, Energy Savings, and Pruning Points of the Networks in the Proposed Inference Method for the Considered Benchmarks Under the Second Scenario

Model (Dataset)	φ_{AP} (%)	Reduction in # of AP MACs (%)	Acc_{AP} (%)	ζ_{EXP} (%)	Energy Savings (%)	
					Comp. to Original	Comp. to MP
VGG16 (CIFAR-100)	80	96	32	64	56	12
ResNet18 (CIFAR-100)	60	84	46	51	56	31
ResNet50 (CIFAR-100)	60	84	44	48	56	32
GoogLeNet (CIFAR-100)	60	84	46	54	43	30
MobileNetV2	70	91	47	62	64	28
EfficientNetB0	80	96	32	63	64	27
ResNet50 (Tiny ImageNet)	60	64	36	45	51	40
InceptionV3 (TinyImageNet)	50	75	23	48	30	14

To show the effectiveness of the proposed approach on larger datasets, we employed the CIFAR-100 and Tiny ImageNet datasets. Tables 4 and 5 show the accuracies and energy savings as well as the pruning points in the considered networks for the first and second scenarios, respectively. The results reveal that when compared to the energies of original and mildly pruned networks, on average, the energy reductions of 63.6% (52.5%) and 38.5% (41%) are obtained, respectively, in the first scenario on CIFAR-100 (Tiny ImageNet). These numbers are 56.5% (40.5%) and 26.6% (27%) in the second scenario where a minimum expected accuracy was imposed to be met in our proposed system for the CIFAR-100 (Tiny ImageNet) dataset.

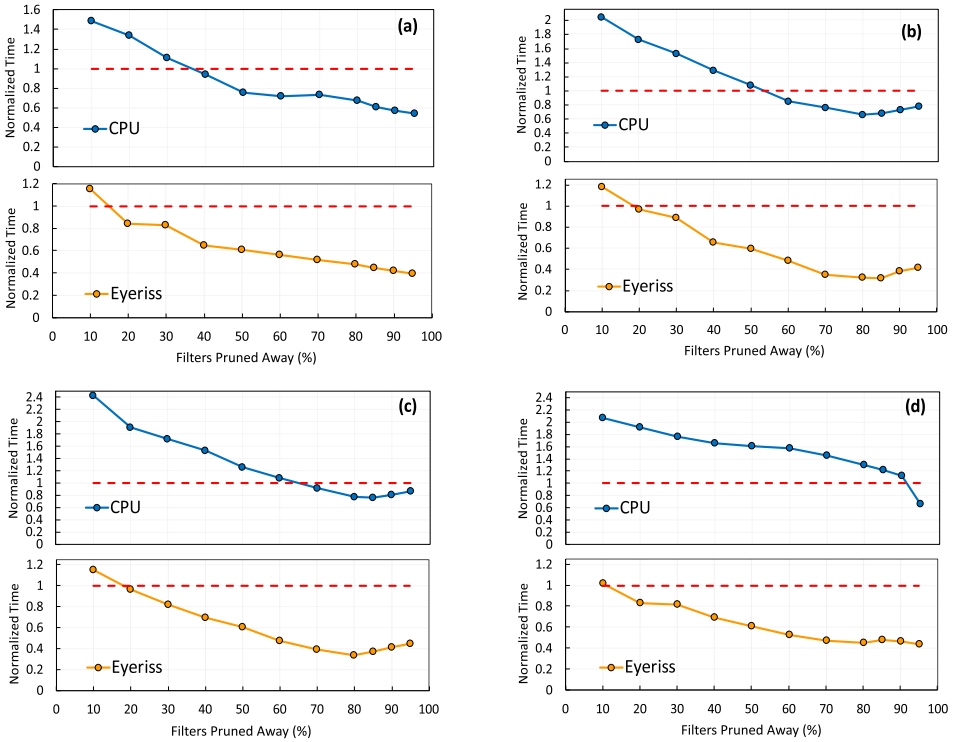

Fig. 12. The normalized (with respect to the unpruned original network) latency of our proposed system vs. a different amount of pruning evaluated on CPU/Eyeriss platforms for VGG16 (a), ResNet18 (b), ResNet50 (c), and GoogLeNet (d) on the CIFAR-100 dataset. The red lines show the latency time of the original unpruned network.

In addition, we estimated the latency of the proposed approach for different percentages of the filters pruned away (i.e., φ_{AP}) on Eyeriss alongside a CPU platform (there were no tools available for estimating the energy of running our proposed inference method on the GPU). The results of this study, which are illustrated in Figure 12, show that under some φ_{AP} values, the latency of the proposed approach becomes lower than the unpruned network (the red lines in the plots show the latency of the unpruned network on the CPU/Eyeriss platform). As expected, when there is a small number of situations where the aggressively pruned network is chosen, the total system mainly has the latency of the mildly pruned network plus the latency of the classifier network (ICNN).

For these cases, the use of the proposed method would not be efficient in terms of the overall latency. However, for cases in which the aggressively pruned network is selected more often, the total latency of the system would be mainly the sum of the latencies of the classifier and the aggressively pruned network, which is smaller than that of the original network without the input classifier. In addition, please note that the convolutional layers of the aggressively pruned network and the ICNN are the same, meaning that no recomputation is required for those layers in the aggressively pruned network when realized using the Eyeriss platform. In the case of the CPU implementation, however, the aggressively pruned network was implemented independent of the ICNN depriving of the gain achievable through the shared computation.

In this work, the efficacy of the proposed method is compared with the techniques suggested in other works [31, 34, 35]. In the work of Park et al. [30], the energy and latency efficiencies were reported for networks of AlexNet, ResNet50, and LeNet using the CIFAR-10 and Tiny ImageNet

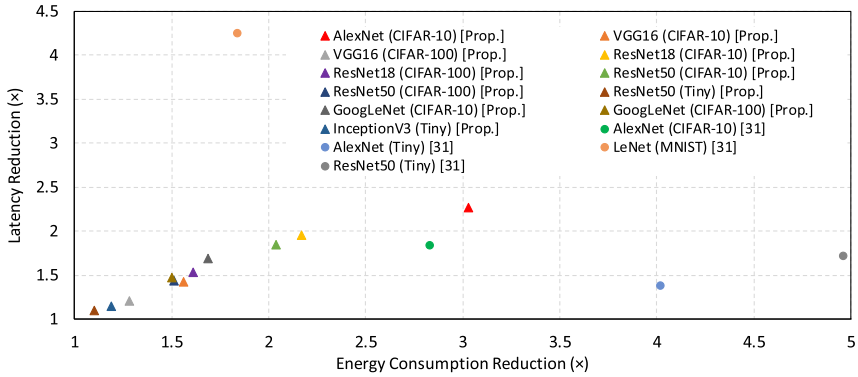

Fig. 13. The energy and latency reductions of the proposed method and the one suggested by Panda et al. [31] on different NN architectures with respect to that of the original unpruned network.

datasets. For the sake of comparison, we calculated the energy (latency) reduction of our approach with respect to the energy (latency) of the original network in the pruning points that our proposed system maintains the original accuracy (since Panda et al. [31] maintain the original accuracy).

In the work of Panda et al. [31], the energy improvements were extracted by synthesizing the proposed NN architecture in a 45-nm technology where the authors used specific hardware description for each studied NN. For our approach, the improvements were evaluated based on the Eyeriss platform. There are two common networks (with the same dataset) between Panda et al. [31] and our work, which are AlexNet and ResNet50. The results show that the energy (latency) reduction for AlexNet under the CIFAR-10 dataset in the work of Panda et al. [31] was 2.83× (1.83×), whereas our method led to 3.03× (2.27×) reduction in the case of this network. However, the proposed approach in the work of Park et al. [30] and Panda et al [31] provided 4.96× (1.71×) energy (latency) gain in the case of ResNet50 under the Tiny ImageNet dataset, whereas our approach led to 1.1× (1.1×) energy (latency) improvement. The energy and latency reductions of our approach and the one suggested by Panda et al. [31] with respect to the energy and latency of the original network for the studied networks are provided in Figure 13.

In the work of Lin et al. [34], the results for the network VGG16 tested on CIFAR-100 are the only common results that can be compared with those of our work. They achieved a speedup of 2× at the cost of 2% accuracy drop. In our approach, by setting a maximum accuracy drop of 2% (considering the original network accuracy as the reference), our proposed method reaches to 2.15× speedup. The same as our proposed method, the proposed method of Lin et al. [34] is input dependent, and hence, to provide a better comparison, we reported the speedup by looking at the average reduction in the number of MAC operations over the whole dataset (Lin et al. [34] also followed the same approach). The proposed approach of Park et al. [35] was evaluated under two configurations where the big DNNs were chosen to be LeNet (for the MNIST dataset), and VGG19 (for the ImageNet dataset). The most efficient combination (in terms of energy reduction) achieved 53.7% (47.5%) energy reduction with the accuracy loss of 0.9% (0.98%) when using the static (dynamic) threshold. However, the energy reduction of our proposed system in VGG19 on Tiny ImageNet was about 55% (compared to the energy of the original unpruned net) at the cost of 1.33% accuracy reduction.

Finally, it should be stressed that the proposed approach targets reducing the energy consumption of a given NN without inducing large/unacceptable accuracy loss when compared to that of the baseline model. More specifically, starting from a baseline model with a baseline accuracy of Acc_0 and energy of E_0, the proposed approach by utilizing an ICNN along with an aggressively

pruned NN reduces the energy to $E_1 = E_0 - \Delta E$ where $\frac{\Delta E}{E_0}$ should be as large as possible and accuracy to $Acc_1 = Acc_0 - \Delta Acc$ where $\frac{\Delta Acc}{Acc_0}$ should be as small as possible. In the proposed approach, ΔAcc should be small, meaning that the higher Acc_0 is, the higher Acc_1 will be. As a specific example, our training of the ResNet-18 network from scratch provided the baseline accuracy of 78% on the dataset CIFAR-10 (see Table 2). When our approach was applied to this network, an energy decrease of 64% (compared to the baseline network) was achieved while the accuracy deteriorated to 76% (see Table 2). Next, by fine-tuning the hyperparameters and using the transfer learning approach, a baseline accuracy of 92.71% was obtained. Having applied our proposed approach, the energy was lowered to about 65% of its baseline value while the accuracy became 87.12%. It should be mentioned that the accuracies in both cases could be increased further by spending much more time and using higher computation resources. Since the goal of our purposed approach was lowering the energy consumption without much degradation of the baseline accuracy, we sufficed to the aforementioned accuracies (i.e., the accuracies reported in Tables 2 and 4).

6 CONCLUSION

In this work, we proposed an energy-efficient inference method that consisted of three networks: an aggressively and a mildly pruned NN as well as an ICNN. Based on the input image, the input classifier determined which of the aggressively or mildly pruned network was to be employed for classifying the input image. To reduce the energy overhead of the ICNN, its convolutional layers were those of the AP network. To assess the efficacy of the proposed NN framework, several benchmarks under two optimization scenarios of considering and not observing a maximum accuracy loss constraint were considered. The evaluation study showed that the proposed inference method resulted in about 60% lower energy consumption compared to the original unpruned network at the cost of an acceptable accuracy loss.

REFERENCES

[1] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. 2015. DeepDriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision* 2722–2730.

[2] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. 2017. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* 542, 7639 (2017), 115–118.

[3] Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, et al. 2013. Recent advances in deep learning for speech research at Microsoft. In *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'13)*. 8604–8608.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. 3104–3112.

[5] NVIDIA. 2015. *GPU-Based Deep Learning Inference: A Performance and Power Analysis*. White Paper. NVIDIA.

[6] Vivienne Sze, Yu Hsin Chen, Tien Ju Yang, and Joel S. Emer. 2017. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE* 105, 12 (2017), 2295–2329. https://doi.org/10.1109/JPROC.2017.2761740

[7] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, et al. 2017. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA'17)*. 1–17.

[8] J. H. Ko, D. Kim, T. Na, J. Kung, and S. Mukhopadhyay. 2017. Adaptive weight compression for memory-efficient neural networks. In *Proceedings of the 2017 Design, Automation, and Test in Europe Conference and Exhibition (DATE'17)*. 199–204.

[9] Yijin Guan, Ningyi Xu, Chen Zhang, Zhihang Yuan, and Jason Cong. 2017. Using data compression for optimizing FPGA-based convolutional neural network accelerators. In *Proceedings of the International Workshop on Advanced Parallel Processing Technologies*. 14–26.

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

[11] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *Proceedings of the European Conference on Computer Vision*. 525–542.

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

[13] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research* 18, 1 (2017), 6869–6898.

[14] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. 2017. *Incremental network quantization: Towards lossless CNNs with low-precision weights*. arXiv:1702.03044.

[15] Ali BanaGozar, Mohammad Ali Maleki, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram. 2017. Robust neuromorphic computing in the presence of process variation. In *Proceedings of the 2017 Design, Automation, and Test in Europe Conference and Exhibition (DATE'17)*. 440–445.

[16] Cesar Torres-Huitzil and Bernard Girau. 2017. Fault and error tolerance in neural networks: A review. IEEE Access 5 (2017), 17322–17341.

[17] Yann LeCun, John S. Denker, and Sara A. Solla. 1990. Optimal brain damage. In *Advances in Neural Information Processing Systems*. 598–605.

[18] J. P. Rauschecker. 1984. Neuronal mechanisms of developmental plasticity in the cat's visual system. *Human Neurobiology* 3, 2 (1984), 109–114.

[19] Christopher A. Walsh. 2013. Peter Huttenlocher (1931–2013). *Nature* 502, 7470 (Oct. 2013), 172. https://doi.org/10.1038/502172a

[20] Swagath Venkataramani, Ashish Ranjan, Kaushik Roy, and Anand Raghunathan. 2014. AxNN energy-efficient neuromorphic systems using approximate computing. In *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*. 27–32.

[21] Ying Wang, Huawei Li, and Xiaowei Li. 2017. Real-time meets approximate computing: An elastic CNN inference accelerator with adaptive trade-off between QoS and QoR. In *Proceedings of the 54th ACM/EDAC/IEEE Design Automation Conference (DAC'17)*. 1–6.

[22] Manoj Alwani, Han Chen, Michael Ferdman, and Peter Milder. 2016. Fused-layer CNN accelerators. In *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture*. 22.

[23] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze. 2017. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits* 52, 1 (2017), 127–138.

[24] S. Han, J. Pool, J. Tran, and W. Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*. 1135–1143.

[25] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. 2017. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5687–5695.

[26] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2016. *Pruning convolutional neural networks for resource efficient transfer learning*. arXiv:1611.06440.

[27] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*. 2074–2082.

[28] Xiaohan Ding, Guiguang Ding, Jungong Han, and Sheng Tang. 2018. Auto-balanced filter pruning for efficient convolutional neural networks. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

[29] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. *Pruning filters for efficient ConvNets*. arXiv:1608.08710.

[30] Eunhyeok Park, Dongyoung Kim, Soobeom Kim, Yong Deok Kim, Gunhee Kim, Sungroh Yoon, and Sungjoo Yoo. 2015. Big/little deep neural network for ultra low power inference. In *Proceedings of the 2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'15)*. 124–132. https://doi.org/10.1109/CODESISSS.2015.7331375

[31] Priyadarshini Panda, Abhronil Sengupta, and Kaushik Roy. 2017. Energy-efficient and improved image recognition with conditional deep learning. *ACM Journal on Emerging Technologies in Computing Systems* 13, 3 (2017), 33.

[32] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. 2017. *Multi-scale dense networks for resource efficient image classification*. arXiv:1703.09844.

[33] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2018. *Slimmable neural networks*. arXiv:1812.08928.

[34] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. 2017. Runtime neural pruning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2178–2188.

[35] Eunhyeok Park, Dongyoung Kim, Soobeom Kim, Yong Deok Kim, Gunhee Kim, Sungroh Yoon, and Sungjoo Yoo. 2015. Big/little deep neural network for ultra low power inference. In *Proceedings of the 2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'15)*. 124–132. https://doi.org/10.1109/CODESISSS.2015.7331375

[36] MIT. 2017. Deep neural network energy estimation tool. Retrieved June 4, 2021 from https://energyestimation.mit.edu.

Received October 2020; revised March 2021; accepted April 2021