MIXP: Efficient Deep Neural Networks Pruning for Further FLOPs Compression via Neuron Bond

Bin Hu*, Tianming Zhao†, Yucheng Xie‡, Yan Wang†, Xiaonan Guo‡, Jerry Cheng§ and Yingying Chen*
*Electrical & Computer Engineering, Rutgers University, USA
†Computer and Information Sciences, Temple University, USA
‡Computer and Information Technology, Indiana University-Purdue University Indianapolis, USA
§Engineering and Computing Sciences, New York Institute of Technology, USA
Email: bh439@scarletmail.rutgers.edu, tum94362@temple.edu, yx11@iupui.edu, y.wang@temple.edu
xg6@iupui.edu, yingche@scarletmail.rutgers.edu

2021 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-3900-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/IJCNN52387.2021.9533522

***Abstract*—Neuron networks pruning is effective in compressing pre-trained CNNs for their deployment on low-end edge devices. However, few works have focused on reducing the computational cost of pruning and inference. We find that existing pruning methods usually remove parameters without fine-grained impact analysis, making it hard to achieve an optimal solution. This work develops a novel mixture pruning mechanism, MIXP, which can effectively reduce the computational cost of CNNs while maintaining a high weight compression ratio and model accuracy. We propose to remove neuron bond that can effectively reduce convolution computations and weight size in CNNs. We also design an influence factor to analyze the importance of neuron bonds and weights in a fine-grained way so that MIXP could achieve precise pruning with few retraining iterations. Experiments with MNIST, CIFAR-10, and ImageNet datasets demonstrate that MIXP could achieve significantly fewer FLOPs and retraining iterations on four widely-used CNNs than existing pruning methods.**

***Index Terms*—pruning, deep learning, weights, CNN**

I. Introduction

Deep convolutional neural networks (CNNs) have achieved remarkable successes in a broad range of applications due to their performance scalability, and self adaptiveness [1]. For example, CNNs have enabled image recognitions [2], [3] and object identifications [4], [5] with high accuracy. CNNs have also enabled highly effective semantic segmentation [6], [7] and natural language processing [8]. With the recent emerging trend of using artificial intelligence (AI) and virtual reality (VR)/augmented reality (AR) on mobile devices, more mobile applications should benefit from using CNNs. However, state-of-art mobile devices still have limited memory size and computational capacity, which prevent them from running conventional CNNs. Thus, most mobile applications have employed cloud-based approaches, where mobile devices upload local sensing data to a cloud server to perform inferences tasks. However, such approaches could result in high processing delays, significant downgrade the system performance, and negative user experience, especially in time-sensitive applications (e.g., mobile healthcare and AR/VR-based applications). Therefore, it is highly desirable to design and develop efficient CNNs that can be deployed on mobile devices without performance degradation.

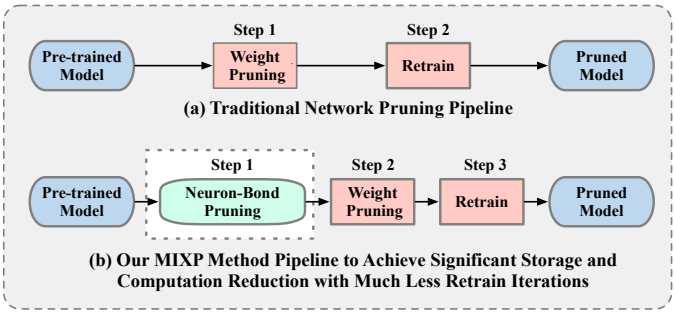

Fig. 1: Comparison of pruning pipeline between our MIXP mechanism and traditional pruning approaches. (a) Traditional weight pruning approaches only prune weights to reduce the size and computational cost of a CNN. (b) MIXP takes three steps to respectively prune neuron bonds and weights of a CNN so that the resulted model has significantly reduced computational cost (i.e., FLOPs and retrain) and size (i.e., numbers of neuron bonds and weights).

A typical way to obtain efficient CNNs for mobile devices is network compression [9], [10], which eliminates traditional CNNs' redundant weights and channels. Pruning is one of the most popular network compression techniques due to its effectiveness in reducing the network complexity and resolving the overfitting problem. Existing pruning methods can be categorized into non-structured pruning and structured pruning. Non-structured pruning [9], [11] can prune arbitrary weights in CNNs and achieve high pruning rates, but the resulting pruned models often have limited acceleration in real implementations since its sparse weight matrices and associated indices are less compatible with the parallel execution model of current computing processors. Structured pruning [12], [13], on the other hand, can directly remove filters, channels, or a set of weights while keeping the full matrix form. Therefore, it can achieve acceleration in real implementation and thus becomes more popular in recent research on network compression. However, it is challenging for pruned models to achieve a higher weight compression ratio. As a result, most of the existing pruning research focus on solving this problem [14]–[16]. Few works have been done to reduce the computational cost of pruning (i.e., retraining iterations) and inference (i.e., floating-point op-

978-0-7381-3366-9/21/$31.00 ©2021 IEEE

erations (FLOPs)). Since lowering computational cost would facilitate the deployment of CNNs on mobile devices directly, in this work, we develop a pruning mechanism targeting at reducing pruning iterations and FLOPs while maintaining a high weight compression rate and inference accuracy. As a result, the powerful and large deep learning models could be deployed on resources limited mobile devices.

To achieve these goals, we need to address a number of challenges. The existing pruning methods usually remove parameters without fine-grained analysis in each pruning iteration, resulting in a dramatic drop in the model accuracy. Furthermore, since the degree of significance of parameters depends heavily on network structure and inputs, it will change with the dynamic network structure after each pruning iteration. Many of the existing pruning methods recover pruned parameters at the last step, where the majority of important parameters may not be recovered. Consequently, the pruned network can not achieve the optimal accuracy [17]. In addition, the existing pruning approaches usually perform the iterations of parameter removal and accuracy recovery at each layer, making it hard to achieve a global optimum [14], [17] with few retraining iterations. This further increases the difficulty of designing pruning solutions with a low computational cost.

Therefore, we develop a novel global **MIX**ture **P**runing mechanism, **MIXP**, which aims to reduce the computational cost of CNNs during the inference and retraining phase while maintaining a high weight compression ratio and model accuracy. Different from the existing pruning methods that focus on reducing the size of weights or neurons, we define *Neuron Bonds* which are the paths between filter and output map in the convolution layer and the paths between two fully connected layers. By pruning the neuron bond, we can skip corresponding conventional computation for generating each pixel of the output map. We utilize *Neuron Bonds* pruning to effectively reduce the convolution computations and size of weights in CNNs. We develop a metric parameter, *Importance*, to measure the global importance of each parameter. We further design an *influence factor* based on the parameter's importance to enable fine-grained analyses on the parameter's importance. In contrast to simply removing a parameter in each pruning iteration, we utilize the influence factor to adjust the parameter's involvement and gradually determine its importance to the network. As a result, MIXP could significantly reduce the pruning iterations by only performing a one-shot pruning when the total number of unimportant parameters reaches the compress ratio. It also reduces the computational cost of powerful DNNs so that they can run on low-end mobile devices.

Specifically, MIXP consists of three steps: *Neuron Bond Pruning*, *Weight Pruning*, and *Retrain*. By utilizing the influence factor, MIXP first prunes neuron bonds of the input model, focusing on reducing its computational cost while removing partial weights. Then, MIXP performs the weight pruning to optimize the size of the network by removing more weights based on their global importance. Since MIXP already removes the neuron bonds with associated weights in the first step, the weight pruning process in the second step has fewer weights to examine and therefore requires fewer pruning iterations. Once a target weight compression ratio is reached, MIXP performs the third step (Retrain) to recover the model's accuracy. Since the unimportant neuron bonds and weights are identified and removed based on the fine-grained parameters' importance and influence factor, the identification process is more accurate in finding the target parameters and the accuracy drop in the model caused by removed parameters is smaller than the traditional directly pruning approach. Therefore, our approach requires fewer retraining iterations to recover the accuracy of the network. Figure 1 illustrates our MIXP pipeline in comparison with a traditional weight pruning pipeline: MIXP exploits the mixture of two pruning steps to optimize the computational cost and size of a CNN model whereas traditional pruning methods only focus on pruning weights.

The main contributions of this work are as follows:

- We propose a new sparsity dimension measurement of CNN, namely neuron bond. Based on neuron bond, we develop the first pruning mechanism that can significantly reduce computational cost (i.e., FLOPs) while maintaining a high model compression rate and inference accuracy
- We design an influence factor to analyze the importance of a particular network parameter in a fine-grained way so that MIXP could achieve precise pruning with fewer retraining iterations. The fewer retaining iterations would effectively save time and computational cost in running a large and powerful pruned models on the mobile devices.
- We demonstrate that our MIXP could significantly reduce the FLOPs and achieve comparable compression ratio and inference accuracy on small and simple dataset (i.e., MNIST), small but more complex dataset (i.e., CIFAR-10), and larger and complex dataset (i.e., ImageNet) for the four widely used DNNs - early models (i.e., LeNet-5), deeper and larger models (i.e., VGG-16), dense models (i.e., ResNet), and lightweight models (i.e., MobileNet V1/V2).

II. Related Work

Existing research on CNN model pruning can be categorized as non-structured pruning and structured pruning. Non-structured pruning directly prunes weights independently of each neuron. For example, Han *et al.* [9] propose a weight pruning method by learning the importance of connections. Zhang *et al.* [11] formulate the weight pruning problem of DNN models as a non-convex optimization problem with combinatorial constraints specifying the sparsity requirements. Ren *et al.* [18] propose a joint framework of DNN weight pruning and quantization using alternating direction method of multipliers (ADMM) to solve non-convex optimization problems. Lee *et al.* [19] propose to prune a given network once before training by introducing a saliency criterion based on the importance of parameters. Although non-structured pruning can achieve high compression ratio and retain decent accuracy, its results have irregular weight matrics, which increase model execution overhead due to irregular memory access and

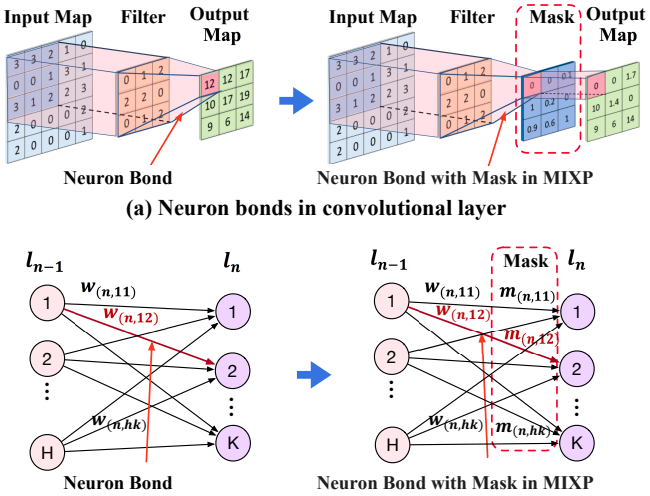

Fig. 2: Illustration of neuron bond definition in a convolutional layer and a fully connected layer.

degraded support for parallelism. Thus, non-structured pruning approaches are not suitable for mobile devices with limited memory and computation capacity.

Structured pruning keeps the model with a regular structure after pruning and is thus more suitable for regular computing hardware on mobile devices. A few filter and channel pruning solutions have been proposed to accelerate the pruned networks while minimizing the network's accuracy loss. For example, Zhao *et al.* [20] re-formulate the channel pruning problem within a Bayesian probabilistic learning framework to directly operate on a redefined scaling factor in Batch Normalization. Lin *et al.* [14] globally prune the unimportant filters across all layers first, then recover the mistakenly pruned filters to improve the model accuracy. These approaches are evaluated in an end-to-end manner, which are difficult to recover the original accuracy. The layer-by-layer pruning approaches are proposed to achieve better performance. For instance, He *et al.* [21] prune each layer of a CNN model by minimizing reconstruction error on its output feature maps. Aghasi *et al.* [22] prune a trained model with removing connections at each layer trying to keep the layer inputs and outputs consistent with the original model.

While these pruning methods could achieve high weight compression ratios, they overlooked the computational costs of the inference and retrain, which are also critical for enabling deep learning on mobile devices. Different from the existing approaches, our MIXP leverages a novel mixture pruning mechanism, including neuron bonds pruning and weights pruning to significantly reduce FLOPs while preserving high weight compression ratio and accuracy. On the contrary, traditional pruning approaches (e.g., [19]) only perform the weights to reduce the computational cost. The major of our advantage is that we can remove neuron bonds to effectively reduce the convolution computations and size of weights beside the weight pruning in CNNs. Moreover, In MIXP, the importance of the parameters is computed by the partial derivation of the loss function with respect to each mask of the parameter instead of respect to weights. The advantage is that MIXP could find the target parameters more precisely since the Neuron Bond Importance is computed independently to the weight. In addition, MIXP designs an influence factor to perform the fine-grained analysis on the importance of parameters, resulting in accurate pruning and reduced retraining iterations.

III. Methodology

A. Neuron Bond and Mask

Different from traditional pruning methods only removing weights or neurons in deep learning networks, MIXP removes unimportant links and weights, namely neuron bond, to reduce FLOPs, which would facilitate the deployment of DNN networks on mobile devices with constrained computing resources. There are two types of neuron bonds in a CNN model: the path connecting a filter and one entry of the output feature map in a convolutional layer and the weight connecting two neurons in two fully connected layers. Figure 2 demonstrates these two types of neuron bonds. Figure 2 (a) shows the neuron bond involving the input map, filter, and output feature map of a convolutional layer. Each element in the output feature map corresponds to a neuron bond. Figure 2 (b) shows the neuron bond that connects every pair of neurons between two fully connected layers.

We can see that when MIXP prunes an unimportant neuron bond, it removes corresponding entries in the output feature map. Since the number of FLOPs is positively correlated to the size of the output feature map, MIXP could reduce significantly more FLOPs than the existing work using weight pruning. Moreover, we design a mask array denoted as M to accurately prune the neuron bonds for each convolutional layer. The mask is a middle layer (as shown in Figure 2) designed to dynamically estimate the importance of each neuron bond during the pruning process. MIXP eventually prunes the unimportant neuron bonds based on the value of each M (e.g., remove a neuron bond if its corresponding mask is less than a threshold or keep it otherwise). We discuss how to update M based on the importance of neuron bond in Section III-E.

B. Overview of MIXP Mechanism

The basic idea of MIXP Mechanism is to analyze the impact of each neuron bond and weight in an input CNN model in a fine-grained way. We design two steps to prune the neuron bonds and weights in convolutional layers and fully connected layers, which could reduce the computational cost of the model in both pruning and inference phases while preserving high weight compression ratio and inference accuracy. The flow of the MIXP mechanism is illustrated in Figure 3. First, MIXP iteratively performs the *Neuron Bond Pruning* on the pre-trained model in both convolutional and fully connected layers to reduce the model computational cost of the input model. A mask is employed to gradually track the importance of every neuron bond in each pruning iteration. To capture the importance of each neuron bond globally, we propose the neuron bond importance in Section III-D with three levels

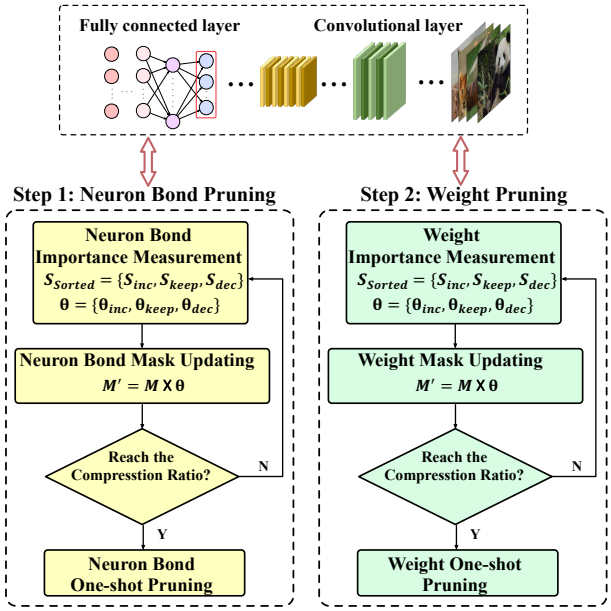

Fig. 3: Illustration of the MIXP mechanism.

(i.e., high, medium and low) corresponding to the impacts on the model's performance. To enable a fine-grained analysis on neuron bonds' importance, we update the associated masks using different strategies (i.e., multiplying different influence factors) according to their different levels of importance. MIXP identifies the unimportant neuron bonds by examining the masks with a threshold approach. The pruning iteration stops when the number of unimportant neuron bonds achieves a target compression ratio. Next, MIXP performs a one-time pruning to remove the identified unimportant neuron bonds.

Next, MIXP iteratively performs the *Weight Pruning* to improve the weight compression ratio. Similar to the neuron bond importance, we propose the weight importance in Section III-D with three levels of importance for different mask updating strategies. Eventually, MIXP identifies the unimportant weights by examining the masks with a threshold approach, stops the pruning iteration when the number of unimportant weights reaches a target compression ratio and removes the identified unimportant weights. After these two pruning steps, MIXP iteratively performs the Retrain to recover the model accuracy by fine-tuning the compressed mode.

C. Problem Formulation

We consider the network pruning as an optimization problem that aims to minimize the cross-entropy loss between the prediction results and ground truth for a neural network after pruning its redundant parameters. The optimization problem is as follows:

$$\min_{\{M,\mathcal{P}\}} \mathcal{L}(M \odot \mathcal{P}; \mathcal{D}) = \min_{\{M,\mathcal{P}\}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(M \odot \mathcal{P}; (x_i, y_i)),$$
$$s.t.\ \mathcal{P} \in \mathbb{R}^J, \| \mathcal{P} \|_0 \leqslant \lambda, M \in \{0,1\}^k, \tag{1}$$

where $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, x and y are the input sample and its label, $\mathcal{L}(\cdot)$ denote loss functions (e.g., cross-entropy loss), $\mathcal{P}$ is the target parameters (i.e., neuron bonds or weights) of the neural network to be pruned, J is the total number of parameters, $||\cdot||_0$ is the L_0 norm, λ is a target compression ratio, and $\odot$ denotes the Hadamard product. Specifically, we need to remove a part of $\mathcal{P}$ based on their importance to the neural network model, measured by a mask M which is a vector of auxiliary indicator variables $m_{ij} \in \{0, 1\}$ for every parameter in $\mathcal{P}$. With this, we utilize M to track parameters' importance and decide whether to prune them or not based on their corresponding mask values in M. The mask values are updated after every iteration based on the importance of the parameters. We discuss how to update the mask values in each iteration in Section III-E.

D. Neuron Bond Importance and Weight Importance

The key to solving the above optimization problem is to accurately identify unimportant parameters in $\mathcal{P}$ for pruning, with a metric that could effectively measure the significance of a parameter with a low computational cost. In this work, we propose to use the difference derived by the following loss functions to describe the impact of performance before and after removing a parameter.

$$\Delta\mathcal{L}_j(\mathcal{P}; \mathcal{D}) = \mathcal{L}(M_j^* \odot \mathcal{P}; \mathcal{D}) - \mathcal{L}(M_j' \odot \mathcal{P}; \mathcal{D}), \tag{2}$$

where M_j^* and M_j' are the masks with and without the j^{th} parameter, respectively. While the concept is straightforward, Equation 2 is computationally expensive as it requires millions of forward passes over the dataset. To solve this problem with reasonable computing cost, we use the partial derivation of the loss function with respect to the j^{th} mask M to approximate $\Delta\mathcal{L}_j(\mathcal{P}; \mathcal{D})$: $\Delta\mathcal{L}_j(\mathcal{P}; \mathcal{D}) \approx \Phi_j(\mathcal{P}; \mathcal{D}) = \partial\mathcal{L}(M \odot \mathcal{P}; \mathcal{D})/\partial M_j$.

Based on $\Phi_j(\mathcal{P}; \mathcal{D})$, we define the *Neuron Bond Importance and Weight Importance* s_j as:

$$s_j = \frac{|\Phi_j(\mathcal{P}; \mathcal{D})|}{\sum_{k=1}^{m} \Phi_k(\mathcal{P}; \mathcal{D})}. \tag{3}$$

As the higher magnitude of $\Phi_j(\mathcal{P}; \mathcal{D})$ indicates the bigger difference in the network loss, the parameter's importance s_j would reflect the importance of the j^{th} parameter to the network [19]. We note that Equation 3 could also be applied to weights of the network to derive the weights' importance, which will be used by MIXP to perform the weight pruning in the second step.

E. Mask Update Using Influence Factor

Traditional pruning methods determine the importance of parameters by checking the accuracy changes with and without parameters. They use binary masks with the value of 0 indicating that the corresponding parameter can be removed, and 1 indicating otherwise. Such approaches have coarse-grained resolutions on discovering parameters' importance and may result in degraded performance with possible loss of essential features by mistake. Moreover, the degraded performance requires more iterations in the retraining step to recover the accuracy. In contrast to the existing approaches, we analyze the parameters' importance in a fine-grained way by gradually updating the mask based on the parameters' importance. As a result, our approach can ensure minimal accuracy drop while

TABLE I: Comparison of different pruning methods for compressing LeNet-300 and LeNet-5 on the MNIST dataset.

	Method	Weight CR(%)	FLOPs CR(%)	Re-Iters	ΔAcc(%)
LeNet-300-100	LC [15]	99.0	-	100K	-1.5
	LWC [9]	92.3	92.0	96K	0.0
	GSM [16]	98.3	92.3	61K	0.0
	DNS [23]	98.2	89.2	34K	-0.3
	FDNP [24]	99.2	-	20K	0.0
	L-OBS* [25]	98.5	-	510	-0.3
	AUTO [26]	98.6	91.0	27K	0.0
	SNIP [19]	98.0	-	25K	-0.7
	MIXP	**98.8**	**92.5**	**4k**	**-0.3**
LeNet-5	NISP [13]	74.4	71.6	100K	-0.02
	GAL [27]	93.0	95.6	30K	-0.2
	LWC	92.0	84.0	96K	0.0
	DNS	99.1	86.4	47K	0.0
	L-OBS*	99.1	-	841	-0.5
	FDNP	99.2	-	20K	0.0
	GSM	99.2	94.1	61K	0.0
	AUTO	99.4	93.0	27K	0.0
	SNIP	99.0	-	25K	-0.2
	MIXP	**99.4**	**96.8**	**4k**	**-0.3**

removing the parameters, facilitating iteration reduction in the retraining process.

Specifically, in each pruning iteration, we derive the importance s_j for each parameter using Equation 3 and obtain a list of parameter importance $S_O = \{s_1, \cdots, s_j, \cdots, s_J\}$. Next, MIXP sorts S_O by the descending order of s_j's magnitude and obtains the sorted list $S_D = \{s'_1, \cdots, s'_j, \cdots, s'_J\}$. We propose to divide the importance in S_D into three groups based on their index and design different mask updating strategies. In implementation, we use two indices, α^* and β^*, to split S_D into three groups: $S_{inc} = \{s'_1, \cdots, s'_{\alpha^*}\}$, $S_{keep} = \{s'_{\alpha^*+1}, \cdots, s'_{\beta^*}\}$, and $S_{dec} = \{s'_{\beta^*+1}, \cdots, s'_J\}$. J is the number of parameters $\mathcal{P}$. We define α is the ratio of α^* over J and β is the ratio of β^* over J . Given a α and a β, we can compute the α^* and β^* based on the J. (i.e., $\alpha^* = \alpha \times J$ and $\beta^* = \beta \times J$).

We design three mask updating strategies (i.e., increase, decrease, and hold) based on S_D to ensure that m_j can track the importance of each parameter in a fine-grained way with low computational complexity. Specifically, we define an *influence factor* θ_j to apply these mask updating strategies as follows:

$$m'_j = \theta_j \times m_j = \begin{cases} \theta_{j,inc} \times m_j & , s'_j \ in \ S_{ins} \\ \theta_{j,keep} \times m_j & , s'_j \ in \ S_{keep} \\ \theta_{j,dec} \times m_j & , s'_j \ in \ S_{dec}, \end{cases} \quad (4)$$

where m'_j is the updated mask, $\theta_{j,inc} > 1$, $\theta_{keep} = 1$, and $0 < \theta_{dec} < 1$. m'_j is set to 1 if $m'_j > 1$. After mask updating, MIXP calculates the ratio of unimportant parameters based on the masks that are less than a cut-off threshold (i.e., γ_c for neuron bonds and γ_w for weights). If the ratio exceeds a target compression ratio (i.e., λ_c for neuron bonds and λ_w for weights), MIXP stops updating the masks and perform a one-shot pruning on the unimportant parameters corresponding to the masks that are less than the cut-off threshold. Eventually, MIXP finishes pruning both neuron bonds and weights and performs the retraining process to fine-tune the model accuracy by recovering certain neuron bonds and weights. In this work, we empirically determine the influence factors, dividing indices, cut-off thresholds, and target compression ratios based on input models. The values of these parameters are discussed in Section IV.

IV. Experiments

Benchmark Datasets. To compare the existing works fairly, we evaluate the performance of MIXP on MNIST datasets with LeNet-5 and LeNet-300-100, CIFAR-10 datasets with VGG-16, MobileNet-v1/v2, and ResNet-56, and ImageNet datasets with VGG-16, MobileNet-v1/v2, and ResNet-50, respectively. We compare them in terms of FLOPs compression rate (FLOPs CR), weight compression rate (weight CR) and retraining iterations (Re-Iters).

MIXP Parameters. We utilize the Random Search [28] to determine the parameters of MIXP one by one. For each dataset, we derive a parameter setting $Par = \{\lambda_c, \lambda_w, \gamma_c, \gamma_w, \alpha, \beta, \theta_{j,inc}, \theta_{j,dec}\}$. The range of each parameter is set to $\lambda_c \in [0.3, 0.998]$, $\lambda_w \in [0.3, 0.998]$, $\gamma_c \in [0.2, 0.7]$, $\gamma_w \in [0.2, 0.7]$, $\alpha \in [0.01, 0.3]$, $\beta \in [0.1, 0.7]$, $\theta_{j,inc} \in [1.05, 1.2]$, $\theta_{j,dec} \in [0.35, 0.95]$.

A. LeNet on MNIST

We first conduct experiments with LeNet-300-100 and LeNet-5-Caffe on the MNIST dataset. The parameter setting for LeNet-5/LeNet-300 is $Par_{LeNet\_MNIST} = \{0.9, 0.993, 0.3, 0.3, 0.01, 0.10, 1.1, 0.90\}$. We train both models with SGD optimization method with mini-batch size 256, weight decay 0.0005 and momentum 0.9. For the LeNet-300-100, we do not use λ_c and γ_c). Training is started by a learning rate 0.1, decayed by 0.9 at every 5 epochs, and stopped after 30 epochs.

We compare our MIXP with state-of-the-art pruning methods in Table I. We can observe that MIXP overall outperforms the existing methods on LeNet-5 model. In particular, MIXP achieves about 23% higher weight CR compared to NISP and about 10% higher weight CR in comparison with GAL with no accuracy change. In addition, MIXP can significantly reduce FLOPs and achieve the highest FLOPs CR of 96.8%. It is also obvious that MIXP has a low Re-Iters of $4k$ with a comparably high weight CR of 99.4% and ΔAcc of -0.4. We note that although L-OBS* exhibits the fewest Re-Iters in the table, it suffers from much more pruning iterations than MIXP (i.e., $O(J^3)$ (L-OBS*) v.s. $O(J^2)$ (MIXP)). The results of LeNet-300-100 show that MIXP can achieve high weight CR of 98.8% without loss of accuracy using only $4k$ retraining iterations. The experimental results demonstrate our MIXP approach can be applied to the fully connected neural network with high weight CR. We note that the FLOPs CR is similar to other methods since we do not prune the neuron bonds on LeNet-300-100. Compared to the SNIP, MIPX achieves the higher Weight CR on both LeNet-5 and LeNet-300 models with obvious fewer retain iterations. The results clearly demonstrate that MIXP could significantly reduce FLOPs and retraining iterations while maintaining a high weight compression rate and model accuracy compared to existing pruning methods on small and simple datasets.

TABLE II: Comparison of different pruning methods for compressing VGG-16, MobileNet-V1/V2 and ResNet-56 on the CIFAR-10 dataset.

	Method	Weight CR(%)	FLOPs CR(%)	Re-Iters	ΔAcc(%)
VGG-16	VCNN [20]	73.3	39.1	70K	-0.1
	GAL	77.6	39.6	140K	-0.2
	DCP [12]	47.9	50.0	300K	-0.17
	ASP [29]	92.7	67.0	140K	-0.6
	PFS [30]	48.3	50.0	300K	+0.19
	DINP [31]	75.0	-	120K	-0.8
	SM [32]	90.0	88.1	300K	-6.7
	SNIP	95.0	-	250K	-0.45
	MIXP	**91.8**	**92.8**	**45K**	**-0.42**
MobileNet-V1	1× baseline	0.0	0.0	150K	0.0
	0.75× baseline	43.0	43.5	150K	-0.7
	0.50× baseline	74.9	74.7	150K	-1.2
	0.25× baseline	93.8	93.6	150K	-5.8
	DCP	58.0	73.8	300K	+0.41
	MIXP	**89.2**	**90.2**	**45K**	**+0.3**
MobileNet-V2	1× baseline	0.0	0.0	150K	0.0
	0.75× baseline	41.0	45.1	150K	-0.9
	0.50× baseline	78.9	73.1	150K	-1.8
	DCP	56.0	57	300K	+0.22
	MIXP	**68.6**	**78.9**	**50K**	**+0.15**
ResNet-56	GAL	65.9	60.2	350K	-1.23
	NISP	42.0	35.5	300K	-6.99
	VCNN	20.4	20.3	280K	-0.8
	DCP	66.4	-	320K	-0.96
	AMC [33]	54.5	50.0	300K	-2.8
	MIXP	**67.1**	**75.8**	**60K**	**-0.9**

TABLE III: Comparison of different filter pruning and weight methods for compressing VGG-16, MobileNet-V1/V2 and ResNet-50 on the ImageNet dataset.

	Method	Weight CR(%)	FLOPs CR(%)	Re-Iters	ΔAcc(%)
VGG-16	DDS	76.8	39.1	1.5M	-2.0
	Thinet [34]	66.7	36.3	1.2M	-0.4
	RBP [35]	-	83	1.2M	-1.8
	ASP	86.2	78.2	1.5M	-0.6
	AMC	80.0	50.0	1.5M	-1.4
	DNS [36]	92.5	77.6	1.07M	-28.6
	DNP [37]	-	81.3	1.2M	-1.4
	LDRF [38]	-	81.3	1.2M	-1.4
	MIXP	**88.3**	**89**	**400K**	**-1.2**
MobileNet-V1	1× baseline	0.0	0.0	1.2M	0.0
	0.75× baseline	16.0	42.8	1.2M	-3.8
	0.50× baseline	69.2	73.6	1.2M	-8.1
	AMC	42.8	50.1	1.2M	-0.2
	PFC	52.5	50.1	1.2M	0.9
	MIXP	**69.2**	**65.2**	**450K**	**-0.3**
MobileNet-V2	1× baseline	0.0	0.0	1.2M	0.0
	0.75× baseline	25.0	30.3	1.2M	-5.4
	AMC	34.2	50.0	1.2M	-0.2
	PFC	25.7	30.0	1.2M	-2.8
	MIXP	**30.0**	**56.3**	**480K**	**-0.2**
ResNet-50	VCNN	37.3	36.7	1.2M	-0.1
	NISP	46.2	41.8	1.3M	-1.54
	GDF [39]	-	51.3	1.2M	-1.9
	GAL	43.3	50.2	1.3M	0.6
	DDS	53.5	45.3	1.2M	-3.19
	PFC	52.3	50.3	1.04M	-1.2
	DN [14]	46.2	51.2	1.5M	-0.1
	MIXP	**52.1**	**62.8**	**500K**	**-0.4**

B. VGG-16, MobileNet and ResNet on CIFAR-10

We test four popular models, VGG-16, MobileNet-V1/V2 and ResNet-56 on the CIFAR-10 dataset to demonstrate the effectiveness of MIXP with more complex dataset. The parameter setting for VGG-16 is $Par_{VGG\_CIFAR} = \{0.8, 0.90, 0.35, 0.35, 0.01, 0.15, 1.1, 0.6\}$. The parameter setting for MobileNet-V1/V2 is $Par_{Mobi\_CIFAR} = \{0.75, 0.90, 0.35, 0.35, 0.01, 0.1, 1.1, 0.6\}$. The parameter setting for ResNet-56 is $Par_{Res\_CIFAR} = \{0.6, 0.70, 0.4, 0.3, 0.01, 0.1, 1.1, 0.6\}$. We train these models with SGD optimization method with mini-batch size 64, weight decay 0.0005 and momentum 0.9. Training is started by a learning rate 0.1, decayed by 0.9 at every 30 epochs, and stopped after 150 epochs.

Table II shows the comparison results between MIXP and state-of-the-art pruning methods on these four models. We can observe that MIXP has significantly higher FLOPs CR than others on VGG-16. Particularly, compared to existing pruning methods, MIXP achieves the highest FLOPs CR of 92.8%, which is 5% more than the second-highest achieved by SM, but with 7 times fewer retrain iterations. MIXP achieves the comparative weight CR with SNIP but with 6 times fewer retrain iterations. The results from MobileNet also show that MIXP effectively improves the weight CR and FLOPs CR by more than 20%. We compare the accuracy performance of baseline models (width multipliers 1.0, 0.75, 0.5, and 0.25) with the DCP (width multipliers 1.0) in Table II on the CIFAR-10 dataset. Our MIXP has 20% and 30% higher weight CR and FLOPs CR than the 0.5 baseline MobileNet-V1, respectively. Similarly, it achieves 23% and 22% higher weight CR and FLOPs CR than the 0.75 baseline MobileNet-V2, respectively. MIXP also outperforms DCP in weight and FLOPs CR. Our MIXP has 53% and 23% higher weight CR and FLOPs CR than the DCP on MobileNet-V2 and 21% and 36 $\%$ higher weight CR and FLOPs CR than the DCP on MobileNet-V2, respectively. Note that MIXP improves the accuracy of MobileNet-V1 and -V2 by 0.3% and 0.15 as DCP does. The results of ResNet-56 show that MIXP achieves the comparative weight CR and higher Flops CR in comparison with other pruning approaches with fewer retrain iterations. Specifically, MIXP achieves 10% FLOPs CR than the second-highest DCP with 6% fewer retrain iterations. We note that though the weight CR of MIXP is similar to DCP, MIXP still achieves higher FLOPS reduction. The results demonstrate that MIXP effectively reduces the computational cost and weight size for complex datasets on deeper, light and dense models.

C. VGG-16, MobileNet and ResNet on ImageNet

To evaluate the effectiveness of our MIXP approach on large-scale datasets, we further conduct experiments to prune VGG-16, MobileNet V1/V2 and ResNet-50 on the ImageNet ILSVRC-12 dataset. The parameter setting for VGG-16 is $Par_{VGG\_Img} = \{0.5, 0.9, 0.35, 0.35, 0.05, 0.3, 1.1, 0.65\}$. The parameter setting for MobileNet V1 is $Par_{Mobi1\_Img} = \{0.5, 0.7, 0.35, 0.33, 0.1, 0.3, 1.1, 0.75\}$. The parameter setting for MobileNet V2 is $Par_{Mobi2\_Img} = \{0.5, 0.4, 0.35, 0.33, 0.1, 0.3, 1.1, 0.75\}$. The parameter setting for ResNet-50 is $Par_{Res\_Img} = \{0.5, 0.5, 0.35, 0.32, 0.05, 0.4, 1.1, 0.3\}$. We train both models

TABLE IV: Performance with different pruning components in MIXP on LeNet-5.

	Neuron bond CR(%)	Weight CR(%)	FLOPs CR(%)
Neuron bond pruning only	90	86	65
Weight pruning only	-	98	92
Both	90	99	**97**

with SGD optimization method with mini-batch size 64, weight decay 0.0005 and momentum 0.9. Training is started by a learning rate 0.01, decayed by 0.9 at every 30 epochs, and stopped after 300 epochs.

We compare our method with state-of-the-art pruning methods in Table III. We observe that MIXP outperforms the existing methods in FLOPs CR on four models with fewer retrain iterations. Compared to pruning methods on VGG-16, though MIXP achieves lower weight CR than DNS, it achieves 8% higher FLOPs CR than it. MIXP also achieves fewer retain iterations than other methods. The results also show that MIXP has 15% and 24% higher weight CR and FLOPs CR than the 0.5 baseline MobileNet-V1, respectively. Similarly, it achieves 15% and 19% higher weight CR and FLOPs CR than the 0.75 baseline MobileNet-V2, respectively. MIXP also outperforms AMC and PFC in weight and FLOPs compression ratio. Our MIXP has 22% and 14% higher weight compression ratio and FLOPs compression ratio than the PFC on MobileNet-V1 and 21% and 18% higher weight and FLOPs compression ratio than the PFC on MobileNet-V2. The results of ResNet-50 shows that MIXP achieves comparable weight CR and higher Flops CR in comparison with other pruning approaches with fewer retrain iterations. Specifically, MIXP achieves 9% FLOPs CR than the second-highest G-SGD with 3% fewer retrain iterations. We note that the MIXP has a similar weight CR with PFC but achieves 10% higher FLOPs CR and 3% fewer retrain iterations. The results on ImageNet show that the effectiveness of our MIXP approach for complex, light and dense modes on large-scale datasets.

D. Ablation Evaluation

Performance with Different Pruning Components. To gain insight on the effects of different pruning components on the overall performance, we set the MIXP parameter set $Par_{LeNet\_MNIST} = \{0.9, 0.993, 0.3, 0.3, 0.005, 0.85, 1.1, 0.90\}$ and prune the Lenet-5 on the MNIST using the partial pruning components in MIXP (neuron bond pruning and weight pruning), respectively. We find that using the neuron bond pruning achieves 86% weight CR and 65% FLOPs CR, while using the weight pruning achieves 98% weight CR and 92% FLOPs CR. In contrast, the full MIXP achieves 99% weight CR and 97% FLOPs CR, indicating that both neuron bond and weight pruning play a significant role in improving the performance of MIXP. Specifically, pruning can achieve an additional 10% than pure weight pruning. The results demonstrate the effectiveness of neuron bond pruning that can further reduce the FLOPs.

Performance with Different Parameters. We also study the impact of parameters by experimenting with Lenet-5 on MNIST using different settings in Par. Figure 4 shows that

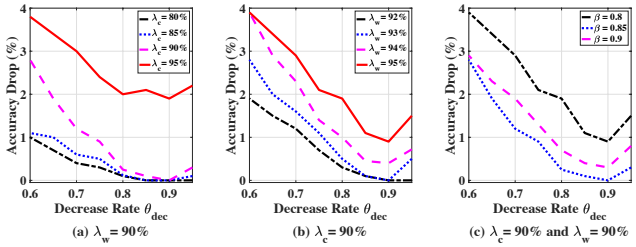

Fig. 4: Performance with different λ_c, λ_w, β and θ_{dec} in MIXP on LeNet-5.

when $\theta_{dec} = 0.9$, MIXP generally achieves the minimum accuracy drop. Figure 4 (a) and (b) also show that given the minimum accuracy drop, there is a trade-off between λ_c and λ_w (i.e., λ_c cannot exceed 90% when λ_w is 90%, and λ_w cannot exceed 93% when λ_c is 90%.) We observe that for the more complex datasets, we need to reduce the λ_c and λ_w to achieve higher performance. That because with the increase of complexity for the models and datasets, we could not prune much more neuron bond as it in the MNIST dataset. In addition, we observe that when both λ_c and λ_w are 90%, MIXP archives the best performance when $\beta = 0.85$. We also observe that the γ_c and γ_w have a little bit different values for different models from 0.25 to 0.35. The α, β, $\theta_{j,inc}$ and $\theta_{j,dec}$ have the similar values for different models. The accuracy loss would significantly increase with these values increasing. That is because MIXP prunes the unimportant parameters from the fine-grained manner to the coarse manner. When these values decrease, the retrain iterations would significantly increase because of the significant increase of iterations of unimportant parameters identification. The results suggest that we can focus on adjusting the parameters α, β, $\theta_{j,inc}$ and $\theta_{j,dec}$ since they have higher correlations with the performance of MIXP than other parameters.

E. Discussion

The results on MNIST, CIFAR-10 and ImageNet datasets clearly demonstrate the effectiveness of our MIXP mechanism for different datasets and network structures. MIXP performs better than most of pruning methods, because we utilize the neuron bond importance and influence factor to gradually update masks of neuron bond and weights, which enables a fine-grained analysis of weights' impacts on the model and accurate pruning. Such high-accurate pruning also leads to significantly fewer retraining iterations for recovering important parameters.

We are aware that the masks that our method uses would introduce a small number of FLOPs, which equal to the size of the masks. According to the definition of FLOPs [40], the additional FLOPs introduced by our system are negligible compared to the significant amount of FLOPs reduced by the neuron bond pruning. For example, our method can reduce $8,874,368$ FLOPs for a convolutional layer 3-4 of VGG-16 while only introducing $16,384$ additional FLOPs.

We also notice that the rules of handling the weight importance in MIXP may vary with different networks and datasets. A thorough study of such impacts of networks and datasets could help accelerate the convergence and reduce the accuracy loss.

V. Conclusion

We develop a novel mixture pruning mechanism, MIXP, focusing on reducing the computational cost of CNNs while maintaining a high weight compression ratio and model accuracy. By employing a novel neuron bond pruning, MIXP removes unimportant neuron bonds in convolutional layers and vastly reduces FLOPs of the pruned network. An influence factor is designed to enable a fine-grained analysis of network parameters' impacts and accurate pruning, facilitating significant reductions of training iterations. Comprehensive experiments on three benchmark datasets demonstrate the effectiveness of the proposed mixture approach for model compression.

ACKNOWLEDGMENT

This work was partially supported by the NSF Grants CCF1909963, CCF2028876, CCF2000480, CCF2028858, CCF2028873, CNS1954959, CCF2028894, CNS1815908, CNS1717356 and ARO Grants W911NF-18-1-0221,W911NF-17-S-0002 and W911NF-20-S-0009.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[2] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *CVPR*, 2017.

[3] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing cnn architectures using the genetic algorithm for image classification," *IEEE Transactions on Cybernetics*, 2020.

[4] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154–6162.

[5] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *CVPR*, 2019.

[6] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "Ccnet: Criss-cross attention for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 603–612.

[7] J. Fu, J. Liu, Y. Wang, J. Zhou, C. Wang, and H. Lu, "Stacked deconvolutional network for semantic segmentation," *IEEE Transactions on Image Processing*, 2019.

[8] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *ieee Computational intelligenCe magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[9] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.

[10] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in neural information processing systems*, 2016, pp. 2074–2082.

[11] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *ECCV*, 2018.

[12] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Advances in Neural Information Processing Systems*, 2018.

[13] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *CVPR*, 2018.

[14] S. Lin, R. Ji, Y. Li, Y. Wu, F. Huang, and B. Zhang, "Accelerating convolutional networks via global & dynamic filter pruning." in *IJCAI*, 2018.

[15] M. A. Carreira-Perpinán and Y. Idelbayev, ""learning-compression" algorithms for neural net pruning," in *CVPR*, 2018.

[16] X. Ding, X. Zhou, Y. Guo, J. Han, J. Liu *et al.*, "Global sparse momentum sgd for pruning very deep neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 6382–6394.

[17] A. Morcos, H. Yu, M. Paganini, and Y. Tian, "One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers," in *Advances in Neural Information Processing Systems*, 2019, pp. 4932–4942.

[18] A. Ren, T. Zhang, S. Ye, J. Li, W. Xu, X. Qian, X. Lin, and Y. Wang, "Admm-nn: An algorithm-hardware co-design framework of dnns using alternating direction methods of multipliers," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 925–938.

[19] N. Lee, T. Ajanthan, and P. Torr, "SNIP: SINGLE-SHOT NETWORK PRUNING BASED ON CONNECTION SENSITIVITY," in *International Conference on Learning Representations*, 2019.

[20] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational convolutional neural network pruning," in *CVPR*, 2019.

[21] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *ICCV*, 2017.

[22] A. Aghasi, A. Abdi, N. Nguyen, and J. Romberg, "Net-trim: Convex pruning of deep neural networks with performance guarantee," in *Advances in Neural Information Processing Systems*, 2017, pp. 3177–3186.

[23] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Advances in neural information processing systems*, 2016, pp. 1379–1387.

[24] Z. Liu, J. Xu, X. Peng, and R. Xiong, "Frequency-domain dynamic pruning for convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 1043–1053.

[25] X. Dong, S. Chen, and S. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," in *Advances in Neural Information Processing Systems*, 2017, pp. 4857–4867.

[26] X. Xiao, Z. Wang, and S. Rajasekaran, "Autoprune: Automatic network pruning by regularizing auxiliary parameters," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 681–13 691.

[27] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured cnn pruning via generative adversarial learning," in *CVPR*, 2019.

[28] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[29] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates." in *AAAI*, 2020, pp. 4876–4883.

[30] Y. Wang, X. Zhang, L. Xie, J. Zhou, H. Su, B. Zhang, and X. Hu, "Pruning from scratch." in *AAAI*, 2020, pp. 12 273–12 280.

[31] B. Mussay, M. Osadchy, V. Braverman, S. Zhou, and D. Feldman, "Data-independent neural pruning via coresets," in *International Conference on Learning Representations*, 2019.

[32] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: Faster training without losing performance," 2020. [Online]. Available: https://openreview.net/forum?id=ByeSYa4KPS

[33] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[34] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[35] Y. Zhou, Y. Zhang, Y. Wang, and Q. Tian, "Accelerate cnn via recursive bayesian pruning," in *ICCV*, 2019.

[36] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *ECCV*, 2018.

[37] Y. Wang, X. Zhang, X. Hu, B. Zhang, and H. Su, "Dynamic network pruning with interpretable layerwise channel selection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020.

[38] W. Chen, Y. Zhang, D. Xie, and S. Pu, "A layer decomposition-recomposition framework for neuron pruning towards accurate lightweight networks," in *AAAI*, 2019.

[39] Z. You, K. Yan, J. Ye, M. Ma, and P. Wang, "Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 2133–2144.

[40] R. Hunger, *Floating point operations in matrix-vector calculus*. Munich University of Technology, Inst. for Circuit Theory and Signal ..., 2005.