Accelerating Convolutional Neural Network via Structured Gaussian Scale Mixture Models: a Joint Grouping and Pruning Approach

Tao Huang, Weisheng Dong, *Member, IEEE*, Jinshan Liu , Fangfang Wu, Guangming Shi, *Senior member, IEEE*, and Xin Li, *Fellow, IEEE*

***Abstract*—Despite the great success of deep convolutional neural networks (DCNNs), their heavy computational complexity remains a key obstacle to the wide use in practical applications. To meet this challenge, DCNN pruning has been recently developed as a technique of compressing DCNNs to facilitate their applications in the real world. In this paper, we propose a *hybrid* network compression technique for exploiting the prior knowledge of network parameters by Gaussian scale mixture (GSM) models. Specifically, the collection of network parameters are characterized by GSM models and network pruning is formulated as a maximum a posteriori (MAP) estimation problem with a sparsity prior. The key novel insight brought by this work is that *groups of parameters associated with the same channel are similar*, which is analogous to the grouping of similar patches in natural images. Such observation inspires us to leverage powerful structured sparsity prior from image restoration to network compression - i.e., to develop a flexible filter-*grouping* strategy that not only promotes structured sparsity but also can be seamlessly integrated with the existing network *pruning* framework. Extensive experimental results on several popular DCNN models including VGGNet, ResNet and DenseNet have shown that the proposed GSM-based joint grouping and pruning method convincingly outperforms other competing approaches (including both pruning and non-pruning based methods).**

I. Introduction

In the past decade, deep convolutional neural networks (DCNNs) have achieved significant improvements over traditional shallow networks in various computer vision tasks - e.g., image classification [19], object detection [29], video tracking [25], and image super-resolution [20]. The success of those DCNN-based approaches mainly attribute to the deep network architecture with a large number of parameters, the availability of large-scale training datasets and the affordability of computational resources (i.e., powerful GPUs). For example, the VGGNet network [30] typically has 16 or 19 layers; while the ResNet [10] network has used 152 layers. From a real-world application perspective, DCNNs with so many layers often require prohibitive computational and memory resources, limiting their deployment on resource constrained devices such as mobile and Internet of Things (IoT) devices.

To meet this challenge, an emerging field of network compression (a.k.a. model compression) has advanced rapidly. The basic idea behind network/model compression is to obtain a more compact representation of DCNN without sacrifice on the classification accuracy. For instance, sparsity regularization [15], [22] (i.e., thresholding operators) can be conveniently enforced to the scaling factors during training for the purpose of eliminating/pruning unimportant channels. Along this line of reasoning, if a subset of inputs can produce similar outputs to the whole, it implies that the rest of inputs have negligible contributions to the outputs and therefore can be pruned [24]; additionally, all filters corresponding to removable inputs can be pruned. In addition to direct pruning [8], other network compression techniques include low-rank decomposition [17],[35], Bayesian compression [50], [51], [52], weight quantization [33], [4],[28], [53], sparse learning [1],[23],[36] as well as direct design of efficient networks (e.g., MobileNet [12], ShuffleNet [34]).

In this paper, we propose a *hybrid* approach toward network compression by combining two most popular ideas in the existing literature: *network pruning* [11] and *filter grouping* [13]. A key novel insight brought by this work is that through careful design of networks (e.g., from channel shuffling [34] to filter grouping [13]), we can make a network more prunable and therefore improve the efficiency of network compression. Unlike [11] (based on LASSO regression [42]), we leverage more powerful group/structured sparsity regularization [6] to joint compress a group of similar parameters associated with the same channel (conceptually analogous to a group of similar patches in images). Unlike [34] (channel shuffle for group convolutions), we propose a more flexible filter grouping strategy to promote group sparsity which can be seamlessly integrated with existing network pruning techniques.

Formally, we have implemented a hybrid network compression approach by the celebrated Gaussian Scale Mixture(GSM) [27] model (please refer to Fig. 1). In our formulation, the weights of convolutional filters are characterized by a GSM model - i.e., the weights are decomposed into a Gaussian vector $\boldsymbol{\alpha}$ and a hidden scalar multiplier θ. In view of the difficulty with optimizing a logarithmic function (the hidden scalar multiplier $\boldsymbol{\theta}$ for sparsity regularization) in CNN, we propose to exploit an alternative reweighted ℓ_1 norm [2] analogous to the logarithmic function in sparsity optimization.

This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1402603 and the Natural Science Foundation of China under Grant 61991451, Grant 61632019, Grant 61621005, and Grant 61836008. Xin Li's work is partially supported by the DoJ/NIJ under grant NIJ 2018-75-CX-0032, NSF under grant OAC-1839909 and the WV Higher Education Policy Commission Grant (HEPC.dsr.18.5).

Tao Huang, Weisheng Dong, Jinshan Liu, Fangfang Wu and Guangming Shi are with the School of Artificial Intelligence, Xidian University, Xi'an, 710071, China. The corresponding author is Dr. Weisheng Dong (e-mail: wsdong@mail.xidian.edu.cn).

Xin Li is with Lane Dep. of CSEE, West Virginia University, Morgantown WV 26506-6109, USA.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2020.2975610, IEEE Journal of Selected Topics in Signal Processing

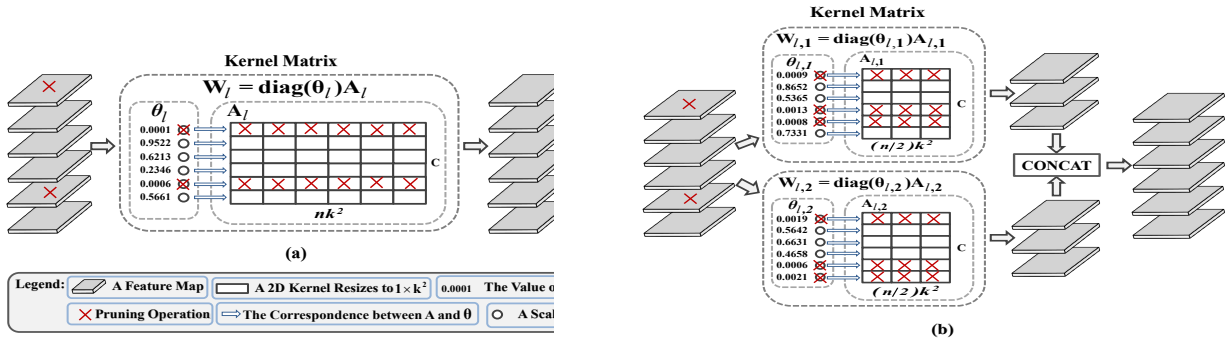

Fig. 1: The motivation behind the proposed GSM modeling and joint grouping and pruning approach. We decompose a filter weight $\mathbf{W}$ into a Gaussian matrix $\mathbf{A}$ and scalar multiplier vector $\boldsymbol{\theta}$ (we have borrowed the terminology "kernel matrix" from Fig. 1 of [21]). (a) network pruning strategy with GSM model only; (b) joint grouping and pruning strategy (we only show a toy example of $G = 2$ due to space limitation). Since both groups don't select the first and fifth channels as the inputs, these two channels will be pruned and the rest will be grouped together in the fine-tuning pruned network.

Our experimental results have shown that a reweighted ℓ_1 norm is more sparse than the original ℓ_1 norm and therefore can prune network even better. Under the framework of GSM modeling, We have designed an efficient network separating filters of each layers into G similar groups through channel shuffling [34] as shown in Fig. 1. Each group sharing similar inputs are selected through comparing the values of hidden scalar multipliers; by *grouping and pruning* similar filters, we can achieve further improved model compression efficiency.

We have validated the proposed hybrid model compression approach on several well-known datasets (CIFAR-10, CIFAR-100, SVHN and ImageNet) and three popular architectures (VGGNet, ResNet and DenseNet). Extensive experimental results demonstrate that our method can compress architectures more efficiently and achieve better accuracy than most state-of-the-art methods. For instance, on all datasets and for all architectures in our experiments, we have achieved modest to moderate gain over network slimming[22], networking pruning [21], sparse structure selection [15], [37] and channel pruning [41]. We have also observed convincing gain over non-pruning based network compression methods (e.g., Automl for model compression [44] and Cascaded Projection [45]) on selected dataset and architecture.

II. Related Works

Low rank decomposition uses several low rank matrices to approximate the weight matrix in the network. In [5], the SVD method was used to decompose the matrix to reduce the parameters of the weight matrix. This method compresses the convolution layer by 2-3 times and the fully connected layer by 5-13 times. In [31], an algorithm based on a structured matrix was proposed for low rank decomposition, which employed a Toeplitz matrix to approximate the original weight matrix. Such low-rank approximation can achieve the compression ratio of 2-3 times on some small data sets. The final accuracy can even exceed that of the network without the compression. Most recently, cascaded projection using low-rank method was proposed for network compression in [45].

Weight quantization selects several *representatives* from the weight matrix, and these *representatives* are used to represent the specific values of a certain type of weight. The *representative* is stored in the codebook, while the original weight matrix stores the corresponding index, which greatly reduces the storage space. In [9], the weight parameter is quantized to 8 bits, which not only saves model storage space but also obtains a fairly good acceleration with negligible loss on accuracy. In [7], K-Means method is used to cluster all the weights to obtain k cluster centers. The original weight matrix only stores the cluster index. This method compresses the model storage space of Alexnet and VGGNet by 35 times and 49 times respectively. Although this method saves a lot of storage space, it does not bring acceleration during the testing stage. In BinaryNet [3], XNOR-Net [28], the single precision weight is quantized to binary, which greatly saves model storage and memory footprint. However, the downside of this approach is lacking of effective training algorithms, which leads to the loss of precision.

Designing compact model achieves the goal of reducing model parameters and calculations by designing a lightweight network structure. Squeezenet [16] uses 1×1 convolution to design a small network. MobileNet [12] architecture is based on depth-wise separable convolution which factorize a standard convolution into a depth-wise convolution and a 1×1 (point-wise) convolution. Shufflenet [34] combined group convolution, channel shuffle, skip connections which achieves $13\times$ actual speedup over AlexNet while maintaining comparable accuracy. The bottleneck structure in ResNet [10] has been proposed to reduce the number of parameters.

Network pruning has recently received increasingly more attention. The key lies in how to measure the importance of neurons, filters, etc. A neuron-level pruning algorithm was proposed in [7]; but the network structure is destroyed so that special hardware equipment is needed to achieve theoretical speedup ratio, which seriously limits the versatility of the model after pruning. Therefore, the filter-level pruning method becomes an important research direction in recent years. [21] calculates the sum of the absolute values of the weight matrix to prune the filter. [22] uses the parameters of the BN layer and add sparse constraints to select the channel. [15] introduces

a scaling factor to the output of specific structures, such as neurons, groups or residual blocks. More recent works such as [41], [39] have mostly focused on smarter ways of pruning.

III. Grouping and Pruning Filters

Network compression aims at throwing away most redundant parameters and reducing computations without sacrifice on the accuracy. Although both strategies of grouping and pruning have been successfully applied for model compression, they have not been combined in the open literature to the best of our knowledge. In this section, we propose to progressively combine the strategies of grouping and pruning filters, which exploit the best of both worlds.

A. DCNN Learning with Gaussian Scale Mixture Modeling

Pruning is a direct and effective method that eliminates unimportant connections and redundant parameters. It is usually composed of a pipeline - i.e., *train, prune and fine-tune*. The key to effective pruning is how to estimate the importance of connections and weights. Several effective methods have been developed recently. In [21], network pruning is based on the sum of absolute values of each filter's weights (pruned filters have small sums). In [22] and [15], the ℓ_1 norm is used to enforce the sparsity on scaling factors in the batch normalization layers. While these methods are heuristic in exploiting the sparsity of model parameters, we propose a principled approach of leveraging Gaussian scale mixture (GSM) models [27], [6] to DCNN pruning under a maximum a posterior (MAP) estimation framework.

For a given dataset $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$, we aim to optimize network parameters by maximizing the posterior

$$\begin{aligned}\log p(\mathbf{W}|\mathcal{D}) &\propto \log p(\mathcal{D}|\mathbf{W}) + \log p(\mathbf{W}) \\ &= \sum_{i=1}^{N} \log p(\boldsymbol{y}_i|\boldsymbol{x}_i, \mathbf{W}) + \log p(\mathbf{W}),\end{aligned} \tag{1}$$

where $p(\mathcal{D}|\mathbf{W})$ denotes a DCNN model with parameters $\mathbf{W}$, which maps an input image $\boldsymbol{x}_i$ to its corresponding output $\boldsymbol{y}_i$, and $p(\mathbf{W})$ denotes the prior distribution of $\mathbf{W}$. If a Gaussian model is adopted for $p(\mathbf{W})$, the corresponding MAP estimation becomes the commonly used DCNN objective function

$$\mathbf{W} = \underset{\mathbf{W}}{\operatorname{argmin}} \sum_{i=1}^{N} L_1(\boldsymbol{y_i}, f(\mathbf{W}, \boldsymbol{x_i})) + \lambda \sum_{l=1}^{L} ||\mathbf{W}_l||_F^2, \tag{2}$$

where L_1 is the data loss, $f(\cdot)$ is the function of the DCNN, $\mathbf{W}_l = [\boldsymbol{w}_{l,1}^\top; \cdots; \boldsymbol{w}_{l,C}^\top] \in \mathbb{R}^{C \times nk^2}$ is the parameters of the l-th layer, C and n are the number of input channels and filters for the l-th layer respectively, k is the spatial size of filters, and L is the number of total layers.

In this paper, we propose to characterize the weight matrix with GSM models. The GSM model decomposes each coefficient $w_{l,c,j}$ of $\mathbf{W}_l$ into the product of a zero-mean Gaussian variable $\alpha_{l,c,j}$ with unitary variance and a hidden scalar multiplier $\theta_{l,c}$ - i.e., $w_{l,c,j} = \alpha_{l,c,j}\theta_{l,c}$, where $\theta_{l,c}$ is the positive scaling variable with probability $p(\theta_{l,c})$ and $j = 1, 2, ..., nk^2$. Thus, the GSM prior of $p(w_{l,c,j})$ can be expressed as

$$p(w_{l,c,j}) = \int_0^\infty p(w_{l,c,j}|\theta_{l,c}) p(\theta_{l,c}) d\theta_{l,c}. \tag{3}$$

For most of $p(\theta_{l,c})$, there is no analytical expression of $p(w_{l,c,j})$. However, we can still use the MAP to estimate $w_{l,c,j}$ by jointly estimating $w_{l,c,j}$ and $\theta_{l,c}$, as will be described later. Considering that the filter coefficients corresponding to the same channel (i.e., $\boldsymbol{w}_{l,c} \in \mathbb{R}^{nk^2}$, $c = 1, \cdots, C$) have strong correlations, they should be characterized by the same distribution with the same $\theta_{l,c}$. Thus, we decompose $\boldsymbol{w}_{l,c}$ as $\boldsymbol{w}_{l,c} = \theta_{l,c}\boldsymbol{\alpha}_{l,c}$, where $\boldsymbol{\alpha}_{l,c} \in \mathbb{R}^{nk^2}$. The GSM modeling of $\mathbf{W}_l$ can then be written as

$$\mathbf{W}_l = \mathbf{\Lambda}_l \mathbf{A}_l, \tag{4}$$

where $\mathbf{\Lambda}_l = \operatorname{diag}(\boldsymbol{\theta}_l) \in \mathbb{R}^{C \times C}$ is a diagonal matrix, and $\mathbf{A}_l = [\boldsymbol{\alpha}_{l,1}^\top; \cdots; \boldsymbol{\alpha}_{l,C}^\top] \in \mathbb{R}^{C \times nk^2}$. Considering that each $\boldsymbol{w}_{l,c}$ is i.i.d., the GSM prior of $\mathbf{W}_l$ can be written as

$$p(\mathbf{W}_l) = \prod_{c=1}^{C} p(\boldsymbol{w}_{l,c}) = \prod_{c=1}^{C} \int_0^\infty p(\boldsymbol{w}_{l,c}|\theta_{l,c}) p(\theta_{l,c}) \, \mathrm{d}\theta_{l,c}, \tag{5}$$

where $p(\boldsymbol{w}_{l,c})$ is the Gaussian model with standard deviation $\theta_{l,c}$. For $p(\theta_{l,c})$, we use the noninformative Jeffrey's prior for $p(\theta_{l,c})$, which is given by

$$p(\theta_{l,c}) \propto \frac{1}{\theta_{l,c} + \epsilon}, \tag{6}$$

where ϵ is introduced for numerical stability.

Since it is difficult to obtain an analytical form of $p(\mathbf{W}_l)$, we propose to pursue a joint estimation of $(\mathbf{W}_l, \boldsymbol{\theta}_l)$ through MAP estimation, where $\boldsymbol{\theta}_l = [\theta_{l,1}, \cdots, \theta_{l,C}]^\top$. By substituting $p(\mathbf{A}_l, \boldsymbol{\theta}_l)$ into Eq. (1), we obtain the following GSM-based objective function for DCNN

$$\begin{aligned}(\mathbf{A}_l, \boldsymbol{\theta}_l) = \underset{\mathbf{\Omega}}{\operatorname{argmin}} &\sum_{i=1}^{N} L_1(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \mathbf{\Omega})) \\ &+ 4\lambda \sum_{l=1}^{L} \log(\boldsymbol{\theta}_l + \epsilon) + \lambda \sum_{l=1}^{L} ||\mathbf{A}_l||_F^2,\end{aligned} \tag{7}$$

where $\mathbf{\Omega} = \{\mathbf{A}_l, \boldsymbol{\theta}_l\}_{l=1}^L$ denotes the set of network parameters and $\mathbf{W}_l = \mathbf{\Lambda}_l \mathbf{A}_l$.

This way, the original optimization problem with respect to $\mathbf{W}_l$ has been translated into the joint estimation of $\mathbf{A}_l$ and $\boldsymbol{\theta}_l$ through the adopted GSM model. Under the context of model compression, we expect $\mathbf{W}_l$ to be sparse because most unimportant filters can be eliminated. It follows that the sparsity constraint can be enforced on $\boldsymbol{\theta}_l$ instead of $\mathbf{W}_l$ in Eq. (7). Note that the slope of the logarithmic function is steeper than that of the ℓ_1 norm regularization, consequently it is often more challenging to work with the logarithmic penalty function than the conventional ℓ_1-regularization.

Fortunately, the above difficulty can be resolved by considering a surrogate function proposed in [2]. Let $g(\boldsymbol{\theta}_l) =$

$\sum_{c=1}^{C} \log(\theta_{l,c} + \epsilon)$. $g(\boldsymbol{\theta}_l)$ can be approximated by the first order Taylor expansion, as

$$g(\boldsymbol{\theta}_l) = g(\boldsymbol{\theta}_l^t) + \nabla g(\boldsymbol{\theta}_l^t) \cdot (\boldsymbol{\theta}_l - \boldsymbol{\theta}_l^t). \tag{8}$$

where $\boldsymbol{\theta}_l^t$ denote the solution obtained in the t-th iteration. Thus, we can solve Eq. (7) by iteratively minimizing

$$\begin{aligned}(\mathbf{A}_l^{t+1}, \boldsymbol{\theta}_l^{t+1}) = &\underset{\boldsymbol{\Omega}}{\operatorname{argmin}} \sum_{i=1}^{N} L_1(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\Omega})) \\ &+ 4\lambda \sum_{l=1}^{L} \sum_{c=1}^{C} \frac{\theta_{l,c}}{\theta_{l,c}^t + \epsilon} + \lambda \sum_{l=1}^{L} ||\mathbf{A}_l||_F^2,\end{aligned} \tag{9}$$

where we have used $\nabla g(\boldsymbol{\theta}_l^t) = \sum_{c=1}^{C} \frac{1}{\theta_{l,c}^t + \epsilon}$ and ignored the constant in Eq. (8). As shown in [2], we can get more sparse $\boldsymbol{\theta}_l$ by using the reweighted ℓ_1 norm than the original ℓ_1 norm. Experimental comparison is referred to Table III. Different from [22] scaling the outputs of convolution layers, we have characterized the weights of convolution filters by the parametric GSM model as shown in Fig. 1 (a). When compared against previous method [22], our GSM-based approach is arguably more principled and powerful.

B. Filter Pruning and Fine-tuning

By training the DCNN with GSM-based regularization, we can obtain a model with sparse multipliers $\theta_{l,c}$. Those channels with multipliers $\theta_{l,c}$ close to zero can then be pruned - i.e., the associated filter weights in both the current layer and the previous layer can be pruned. The pruned channels can be selected by applying a hard-thresholding operation to the multiplier with a threshold. The threshold is obtained by first sorting all the multiplier and selecting the threshold as a percentile (e.g., 80%) of all the multiplier values. After pruning the filter weights, a compact DCNN with much fewer parameters can be obtained, which can be further fine-tuned to compensate the loss of prediction accuracy due to pruning (e.g., the weights of the compact DCNN can be updated via the common DCNN objective function Eq. (2) which does not have any sparsity constraint).

C. Learning Grouped Convolutions with GSM Model

Enforcing the constraint on connection patterns is tight for one group because all output features are generated by the same inputs. By contrast, multiple groups have more flexibility with selecting appropriate filter inputs. ShuffleNet[34] is the representative of efficient lightweight models where pointwise group convolution is introduced to reduce computational complexity. Based on the observation that not all inputs have equally significant contributions to outputs, CondenseNet [13] proposed a learned group convolution method to automatically select different inputs for multiple groups on DenseNet. The key idea behind CondenseNet is to split the filters of a layer into multiple groups and gradually prune the connections to less important features for each group during the training.

In this subsection, we propose to extend the proposed structured GSM model by a new strategy called filter grouping. In the proposed GSM-based modeling of $\mathbf{W}_l$, we assume that all the filter coefficients corresponding to the same channel in each layer share the same distribution prior (i.e., the same $\theta_{l,c}$). This assumption may be too strong and can be relaxed by dividing the filters of each layer into several groups and modeling the weights with different $\theta_{l,c}$ for different groups. Let G be the number of divided groups, and there are $m = [n/G]$ filters in each group. Then, the weights in each group can be modeled by the GSM model as $\mathbf{W}_{l,g} = \boldsymbol{\Lambda}_{l,g}\mathbf{A}_{l,g}$, $\mathbf{A}_{l,g} \in \mathbb{R}^{C \times mk^2}$, $\boldsymbol{\Lambda}_{l,g} = \operatorname{diag}(\boldsymbol{\theta}_{l,g}) \in \mathbb{R}^{C \times C}$, $g = 1, \cdots, G$. It can be easily verified that DCNN learning based on a grouped GSM prior can be expressed by

$$\begin{aligned}(\mathbf{A}_l, \boldsymbol{\theta}_l) = &\underset{\boldsymbol{\Omega}}{\operatorname{argmin}} \sum_{i=1}^{N} L_1(\boldsymbol{y}_i, f(\boldsymbol{x}_i; \boldsymbol{\Omega})) \\ &+ 4\lambda \sum_{l=1}^{L} \sum_{g=1}^{G} \log(\boldsymbol{\theta}_{l,g} + \epsilon) + \lambda \sum_{l=1}^{L} \sum_{g=1}^{G} ||\mathbf{A}_{l,g}||_F^2,\end{aligned} \tag{10}$$

Similar to Eq. (7), the above objective function can be iteratively solved using the reweighted ℓ_1-norm trick.

During the training, the scalar multiplier $\boldsymbol{\theta}$ serves as a natural candidate of progressively selecting varying inputs for different groups. There are two key differences between ours and CondenseNet [13]. First, CondenseNet learns the group convolution through grouping filter weights of 1×1 convolutional layer directly and pruning small weights in each group; while we opt to characterize the filters' weights by the GSM model and conduct network pruning based on scalar multiplier $\boldsymbol{\theta}$ instead. Second, CondenseNet adopts a multi-stage process in which training and pruning operations have to be repeated for multiple times which requires a lot of computations. Our strategy only needs to train and prune once during the process of network compression, which is more computationally efficient. In summary, exploiting the group strategy can learn multiple connection patterns and reduce more parameters and computing operations to produce more efficient networks.

Putting things together, the proposed network compression algorithm based on GSM model and joint grouping and pruning is summarized in **Algorithm 1**. Note that by simply replacing the objective function Eq. (10) in **Algorithm 1** with Eq. (9), we can obtain a network pruning algorithm based on GSM only (without grouping). As a toy example, Fig. 1 shows the difference between conventional pruning (with GSM only) and our joint grouping and pruning strategy ($G = 2$). During the training process, we train all parameters without pruning and enforce the sparsity constraint with scalar multipliers $\boldsymbol{\theta}$. After training is done, we sort all $\boldsymbol{\theta}$ and prune those small values. Accordingly, the useless entries in Gaussian matrix $\mathbf{A}$ are pruned too. Finally, the pruned network will be fine-tuned to reach a higher accuracy (similar to Step 3 of **Algorithm 1**).

IV. Experiment Results

We have implemented the proposed compression method using PyTorch and evaluated it on popular datasets such as CIFAR-10, CIFAR-100 [18], SVHN [26] and ImageNet [43]. The network architectures in our experiments include VGGNet [30], ResNet [10] and DenseNet [14].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2020.2975610, IEEE Journal of Selected Topics in Signal Processing

(a). Test Errors on CIFAR-10

Network	Method	Params	Params Pruned	FLOPs	FLOPs Pruned	Test error (%)
VGGNet-16	Baseline	14.77M	-	6.27×10^8	-	6.21
	Pruned [21]	5.40M	64.0%	4.12×10^8	34.2%	6.60
	Pruned [39]	2.35M	84.0%	2.74×10^8	56.2%	7.26
	SSS [15]	8.19M	44.5%	4.44×10^8	29.8%	6.24
	Pruned [46]	3.92M	73.3%	3.80×10^8	39.1%	6.82
	GAL_0.1 [47]	2.67M	82.2%	3.44×10^8	45.2%	6.58
	Ours	1.64M	88.9%	3.23×10^8	48.5%	**6.06**
VGGNet-19	Baseline	20.05M	-	7.97×10^8	-	6.24
	Slimming [22]	2.30M	88.5%	3.91×10^8	51.0%	6.20
	Ours	1.80M	91.0%	3.16×10^8	60.4%	**6.02**
ResNet-20	Baseline	0.27M	-	0.82×10^8	-	**8.66**
	SSS [15]	0.15M	44.4%	0.35×10^8	57.3%	9.17
	Pruned [41]	0.18M	33.3%	-	-	9.10
	Ours	0.14M	48.1%	0.41×10^8	50.0%	8.81
ResNet-56	Baseline	0.86M	-	2.52×10^8	-	**6.61**
	Pruned [11]	-	-	1.26×10^8	50.0%	8.20
	Pruned [21]	0.73M	13.7%	1.82×10^8	27.6%	6.94
	TPRD [38]	-	-	1.49×10^8	40.9%	6.86
	KSE (G=5) [49]	0.36M	58.1%	1.00×10^8	60.3%	7.12
	AMC[44]	-	-	1.26×10^8	50.0%	8.10
	NISP [48]	0.49M	42.4%	1.62×10^8	35.5%	6.99
	CaP [45]	-	-	1.27×10^8	49.8%	6.78
	Ours	0.35M	59.3%	1.23×10^8	51.2%	6.83
ResNet-110	Baseline	1.73M	-	4.98×10^8	-	**6.08**
	Pruned [21]	1.68M	2.3%	4.26×10^8	15.9%	6.45
	GAL_0.5 [47]	0.95M	44.8%	2.60×10^8	48.5%	7.26
	Ours	0.57M	67.2%	2.37×10^8	53.3%	6.23
ResNet-164	Baseline	1.74M	-	5.07×10^8	-	5.81
	Slimming [22]	1.44M	14.9%	3.81×10^8	23.7%	5.08
	SSS [15]	1.62M	6.9%	3.88×10^8	23.1%	5.25
	Ours	0.95M	45.1%	3.00×10^8	39.8%	**4.93**
DenseNet-40	Baseline	1.08M	-	5.68×10^8	-	5.99
	Slimming [22]	0.66M	35.7%	3.81×10^8	28.4%	**5.19**
	Pruned [46]	0.42M	59.7%	3.12×10^8	44.8%	6.84
	GAL_0.01 [47]	0.67M	35.6%	3.66×10^8	35.3%	5.39
	KSE (G=3) [49]	0.63M	41.7%	3.40×10^8	40.1%	**5.19**
	Ours	0.66M	39.3%	3.93×10^8	30.9%	5.23

(b). Test Errors on CIFAR-100

Network	Method	Params	Params Pruned	FLOPs	FLOPs Pruned	Test error (%)
VGGNet-16	Baseline	14.77M	-	6.27×10^8	-	26.81
	Pruned [39]	4.50M	69.5%	3.82×10^8	39.1%	27.99
	SSS [15]	2.50M	83.1%	3.08×10^8	50.6%	27.05
	Ours	2.50M	83.1%	3.51×10^8	44.2%	**26.41**
VGGNet-19	Baseline	20.09M	-	7.97×10^8	-	27.15
	Slimming [22]	5.00M	75.1%	5.01×10^8	37.1%	26.52
	Compressed [40]	-	-	4.78×10^8	40.0%	26.35
	Ours	4.72M	76.5%	4.96×10^8	37.8%	**25.82**
ResNet-164	Baseline	1.75M	-	4.99×10^8	-	24.51
	SSS [15]	1.63M	6.3%	3.90×10^8	21.8%	23.71
	Slimming [22]	1.21M	29.7%	2.47×10^8	50.6%	23.91
	Ours	1.15M	34.3%	3.26×10^8	34.7%	**23.59**
DenseNet-40	Baseline	1.13M	-	5.68×10^8	-	25.95
	Slimming [22]	0.66M	37.5%	3.71×10^8	30.3%	25.28
	Ours	0.61M	45.6%	3.47×10^8	39.0%	**25.22**

TABLE I: Results without grouping filters on CIFAR-10/100, using VGGNet-16/19, ResNet-20/56/110/164 and DenseNet-40.

A. Datasets

The CIFAR-10 and CIFAR-100 datasets both have 50,000 images for training and 10,000 images for testing (all sized by 32×32). The CIFAR-10 dataset has 10 independent classes; while the CIFAR-100 dataset has 100 different classes. Before training, we pre-process images with a standard data-augmentation scheme (i.e., zero-padding, cropping and random flipping along horizontal/vertical/diagonal, and normalization). The Street View House Number (SVHN) dataset has a large number of color images of size 32×32. The SVHN dataset consists of 604,388 training images including all the extra set and 26,032 testing images. The ImageNet dataset consists of over 1.2 million training images and 50,000 validation images drawn from 1000 classes. A standard pre-processing procedure is adopted. The input images are first resized to 256×256. Then the training set is created by random cropping to 224×224 patches followed by random flipping; the testing set adopts a similar 224×224 cropping at the center.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2020.2975610, IEEE Journal of Selected Topics in Signal Processing

Algorithm 1 Network compression based on GSM model and Joint Grouping and Pruning

- **Initialization**:
  (a) Set parameters $\mathbf{A}$, $\boldsymbol{\theta}$ and λ;
  (b) Set basic configurations: mini-batch size(bs), training epochs (T_1), fine-tune epochs (T_2), pruning ratio(pr).
- **Input**: $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$.
- **Output**: A compact DCNN with comparable accuracy.
- **Step-1**: TRAIN
  for $t = 1, 2, \ldots, T_1$
    for $k = 1, 2, \ldots, \lfloor \frac{N}{bs} \rfloor$
      Update parameters $\mathbf{A}$, $\boldsymbol{\theta}$ via Eq. (10);
- **Step-2**: PRUNE
  (a) Sort all multipliers $\boldsymbol{\theta}$ and obtain the pruning threshold according to the pruning ratio pr;
  (b) Prune the unimportant $\mathbf{A}$ and $\boldsymbol{\theta}$;
  (c) Compute $\mathbf{W}$ via Eq. (4) and feed it to the new compact DCNN.
- **Step-3**: FINE-TUNE
  for $t = 1, 2, \ldots, T_2$
    for $k = 1, 2, \ldots, \lfloor \frac{N}{bs} \rfloor$
      Update weights $\mathbf{W}$ of the new compact DCNN via

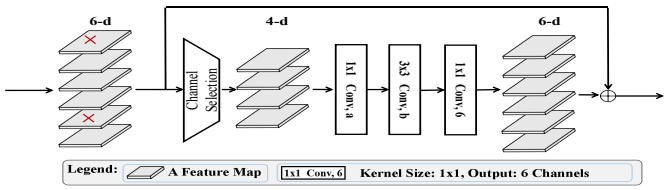

Fig. 2: The structure of the residual block (note that the values of "a" and "b" are not fixed).

B. Network Architectures and Configurations

We have evaluated the proposed model compression method on both CIFAR and SVHN datasets with several variations of three popular networks. We pruned two kinds of VGGNet (i.e., VGGNet-16 and VGGNet-19) with only one fully connected layer. We also pruned ResNet with 20/56/110/164 layers and DenseNet with 40 layers. On the ImageNet dataset, we have evaluated our compression method on VGGNet-A which has three fully connected layers. For the grouping strategy, we have only conducted experiments with VGGNet so far. In our implementation, we have trained randomly initialized networks from the scratch: θ is initialized to be 1; α is initialized to a normal distribution with zero mean and standard deviation of $\sqrt{2/n}$ where n denotes the number of elements in α. We trained all models using similar basic configurations to previous work of [22]. Note that network slimming [22] represents the current state-of-the-art in the open literature; other benchmark methods such as filter pruning [21] and Structured Sparsity Learning (SSL) [32] were less competitive as shown in [22]).

In our experiments, we have also pruned the parameters of the residual block. Similar to [22], we add a channel selection layer before the first convolutional layer of residual block as shown in Fig. 2. The channel selection layer can mask out insignificant channels. The number of parameters in the channel selection layer is equal to the number of input channels of the residual block and is negligible. Thanks to the introduction of the channel selection layer, we can further prune the convolutional layers in the residual block. To protect the structure of ResNet, we have fixed the number of output channels in the last convolutional layers of the residual block. We have also added the channel selection layer to the dense block for DenseNet, which shares a similar observation.

C. Sparsity of the Reweighted ℓ_1-norm

In Eq. (9) and (10), we have replaced the original logarithmic penalty function by a surrogate function of reweighted ℓ_1 norm. Each $\theta_{l,c}$ has a different regularization parameter. The regularization parameter is given by

$$\lambda_{l,c}^t = \frac{\eta}{\theta_{l,c}^t + \epsilon}. \tag{11}$$

In our experiments, $\lambda_{l,c}^t$ is related to both sparsity and accuracy. In Eq. (11), there are two parameters (η, ϵ) affecting the value of $\lambda_{l,c}^t$: ϵ is a small positive value assuring the stability of implemented algorithm; η is related to $\lambda_{l,c}^t$ controlling the sparsity of $\boldsymbol{\theta}$ directly. Fig. 3 shows the distributions of $\boldsymbol{\theta}$ with different η on VGGNet. It can be observed that the distributions of $\boldsymbol{\theta}$ are indeed more and more skewed with the increasing of η. Note that the frequency of updating batch sizes or $\lambda_{l,c}^t$ also has an effect on sparsity and accuracy. In our experiments, we update the regularization parameters at the first epoch and then update it every 30 epochs on CIFAR dataset and every 4 epochs on SVHN dataset. We have selected η in the range of $[5e-6, 5e-5]$, where higher values of η lead to larger pruning ratios.

It should be noted that we have pruned all convolutional layers including the first layer and the last layer according to the value of θ. How much weights of the first layer and the last layer are pruned depends on the value of θ and the set threshold. Table II shows the architecture of our pruned VGGNet on CIFAR-10. It can be observed from Table II that 512 feature map layers are heavily pruned and the first 5 convolutional layers are slightly pruned. The first layer and the last layer also have redundant parameters. Thanks to structured sparsity constraints, the information and features of the inputs can be efficiently learned by our pruned network.

In order to show that our pruning method benefits from the strong sparsity of reweighted ℓ_1-norm, we have also compared reweighted ℓ_1-norm and conventional ℓ_1-norm on Cifar-10 dataset with VGGNet-19 and ResNet-56. For a fair comparison, we have included the pruning results of reweighted ℓ_1-norm vs. ℓ_1-norm under various pruning ratios with $\eta = 5e-5$. From Table III, we can see that the pruning results by reweighted ℓ_1-norm have fewer parameters, less FLOPs and higher accuracies than ℓ_1-norm counterpart. For the pruning ratio 0.8 with VGGNet-19 and 0.6 with ResNet-56, several layers of the model are pruned completel by ℓ_1-norm (i.e., failure); while reweighted ℓ_1-norm still achieves good performance.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2020.2975610, IEEE Journal of Selected Topics in Signal Processing

Pruned Ratio	Architecture (VGGNet)	Params Pruned	Flops Pruned	Test error (%)
0	[64, 64, M, 128, 128, M, 256, 256, 256, 256, M, 512, 512, 512, 512, M, 512, 512, 512, 512]	-	-	6.24
0.1	[63, 64, M, 126, 128, M, 253, 249, 242, 232, M, 455, 444, 460, 455, M, 445, 454, 443, 440]	20.80%	12.30%	6.05
0.2	[62, 64, M, 126, 128, M, 251, 243, 225, 214, M, 399, 379, 389, 403, M, 381, 387, 370, 376]	39.16%	23.17%	6.17
0.3	[62, 64, M, 126, 126, M, 248, 236, 208, 191, M, 343, 318, 321, 336, M, 313, 319, 319, 322]	55.01%	33.12%	5.97
0.4	[56, 64, M, 126, 126, M, 246, 229, 190, 172, M, 283, 261, 264, 264, M, 252, 259, 243, 267]	67.98%	42.28%	5.97
0.5	[53, 63, M, 126, 126, M, 238, 220, 176, 150, M, 229, 206, 199, 199, M, 183, 189, 189, 205]	78.30%	49.81%	6.11
0.6	[49, 63, M, 125, 126, M, 234, 216, 162, 131, M, 173, 140, 120, 132, M, 121, 123, 137, 149]	86.03%	56.09%	5.95
0.7	[43, 63, M, 124, 126, M, 234, 208, 151, 112, M, 108, 69, 57, 75, M, 59, 58, 72, 92]	91.07%	59.85%	6.02
0.8	[39, 63, M, 122, 126, M, 231, 198, 123, 64, M, 36, 11, 9, 4, M, 2, 2, 16, 55]	93.87%	64.87%	6.18

TABLI layer.

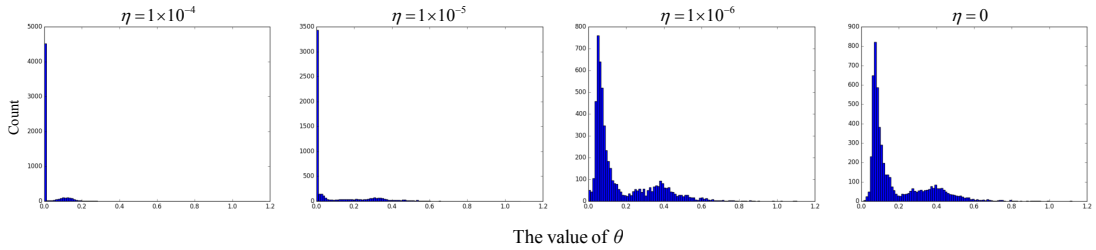

Fig. 3: The distributions of $\boldsymbol{\theta}$ with different η on VGGNet. The x and y axis denote the values and counts of $\boldsymbol{\theta}$ respectively.

VGGNet-19						
Norm / PR	ℓ_1			Reweighted ℓ_1		
	Error	Params	FLOPs	Error	Params	FLOPs
0.5	6.13%	4.82M	0.51B	6.11%	4.35M	0.40B
0.6	5.93%	3.40M	0.47B	5.95%	2.80M	0.35B
0.7	6.24%	2.47M	0.44B	6.02%	1.79M	0.32B
0.8	-	-	-	6.18%	1.23M	0.28B
ResNet-56						
Norm / PR	ℓ_1			Reweighted ℓ_1		
	Error	Params	FLOPs	Error	Params	FLOPs
0.3	7.05%	0.44M	0.17B	6.59%	0.47M	0.16B
0.4	7.06%	0.39M	0.14B	6.73%	0.37M	0.14B
0.5	6.98%	0.34M	0.12B	6.76%	0.32M	0.12B
0.6	-	-	-	7.06%	0.28M	0.10B

TABLE III: Performance comparison between reweighted ℓ_1-norm and ℓ_1-norm on CIFAR-10, using VGGNet-19 and ResNet-56. "PR" denotes the ratio of the pruned number to the total number of $\boldsymbol{\theta}$. M/B means (10^6/10^9), respectively.

D. Results of one-group GSM modeling

In Table I, we have compared our method with existing network pruning techniques [11], [15], [21], [22], [38], [39], [41], [46], [47], [48] and non-pruning techniques [49], [44], [45], [40] on CIFAR datasets. Tables IV and VII are the compression results on SVHN and ImageNet dataset, respectively.

CIFAR. From Table I (a) and (b), we can observe that, when compared with other methods, our method can prune more parameters, save more FLOPs and still achieve lower testing errors. On VGGNet-16/19, we can prune 88.9%/91.0% parameters on CIFAR-10 and 83.1%/76.5% on CIFAR-100 without accuracy loss. The testing error of the pruned VGGNet-19 by our method is 0.7% lower than that of [22] on CIFAR-100 with even smaller number of parameters and FLOPs. With 30.2% fewer parameters and 16.1% less FLOPs, our pruned ResNet-164 achieves 0.15% lower testing error than [22] on CIFAR-10. Compared with [15], our pruned VGGNet-16 and ResNet-164 respectively obtain 0.18% and 0.32% decrease in testing error on CIFAR-10 with 44.4% and 38.2% fewer parameters and 18.7% and 16.7% fewer FLOPs. Compared with [21], our pruned ResNet-56 and ResNet-110 respectively obtain 0.11% and 0.22% decrease in testing error on CIFAR-10 with 45.6% and 64.9% fewer parameters and 23.6% and 37.4% fewer FLOPs. With 8.1% fewer parameters and 8.7% fewer FLOPs, our pruned DenseNet achieves nearly the same testing accuracy with [22] on CIFAR-100. As shown in Table I (a) and (b), when compared to [11], [38], [39], [40], [41], our pruned models achieve the best results on CIFAR dataset.

SVHN. Table IV shows the results on SVHN dataset. As shown in Table IV, the testing accuracies on SVHN dataset can reach to 98% and near saturation. Our testing errors are almost same as [22] and [49], but with much fewer parameters and FLOPs. For the SVHN dataset, our pruned VGGNet-19 obtains comparable performance with that pruned by [22] with 6.9% fewer parameters and 29.4% fewer FLOPs. With 38.9% fewer parameters and 29.9% fewer FLops, our pruned ResNet obtains same test error as the ResNet pruned by [22]. When compared to [49], our pruned DenseNet-40 saved more than 13.2% parameters and 15.4% FLOPs with only 0.06% accuracy degradation. In Table V, we have compared our method with [22] on a mini-version of SVHN dataset, which only 73,257 training images without the extra set and 26,032 testing images. From Table V, we can see that our pruned VGGNet-19 achieves comparable test accuracy with [22] using 7.6% less parameters and 39.4% less FLOPs. For ResNet, with 25% less parameters and 18.3% less FLOPs, our pruned ResNet also achieves similar accuracy to [22]. With 9.3% less parameters and 9.8% less FLOPs, our pruned DenseNet obtains 0.11% lower test errors than that pruned by [22].

ImageNet. Table VI and Table VII have shown the top-1 results on ImageNet dataset using VGGNet-A with different experimental settings (e.g., whether to distinguish between FC and Conv layers, high vs. low compression ratios). The

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2020.2975610, IEEE Journal of Selected Topics in Signal Processing

Network	Method	Params	Params Pruned	FLOPs	FLOPs Pruned	Test error (%)
VGGNet-19	Baseline	20.04M	-	7.97×10^8	-	2.21
	Slimming [22]	3.04M	84.8%	3.98×10^8	50.1%	**2.06**
	Ours	1.66M	91.7%	1.64×10^8	79.5%	2.09
ResNet-164	Baseline	1.74M	-	4.98×10^8	-	1.91
	Slimming [22]	1.46M	14.5%	3.44×10^8	31.1%	**1.85**
	Ours	0.81M	53.4%	1.94×10^8	61.0%	**1.85**
DenseNet-40	Baseline	1.08M	-	5.68×10^8	-	1.80
	Slimming [22]	0.44M	56.6%	2.67×10^8	49.8%	1.81
	KSE (G=5)[49]	0.42M	61.1%	2.60×10^8	54.2%	**1.75**
	Ours	0.28M	74.3%	1.73×10^8	69.6%	1.81

TABLE IV: Results without grouping filters on SVHN, using VGGNet-19, ResNet-164 and DenseNet-40.

Network	Method	Params	Params Pruned	FLOPs	FLOPs Pruned	Test error (%)
VGGNet-19	Baseline	20.04M	-	7.97×10^8	-	3.89
	Slimming [22]	3.04M	84.8%	4.31×10^8	45.9%	3.86
	ours	1.52M	92.4%	1.17×10^8	85.3%	**3.85**
ResNet-164	Baseline	1.74M	-	4.99×10^8	-	3.35
	Slimming [22]	0.98M	43.4%	2.36×10^8	52.7%	**3.07**
	ours	0.55M	68.4%	1.45×10^8	71.0%	3.13
DenseNet-40	Baseline	1.09M	-	5.70×10^8	-	3.46
	Slimming [22]	0.47M	56.2%	2.96×10^8	48.1%	3.32
	ours	0.38M	65.5%	2.39×10^8	57.9%	**3.21**

TABLE V: Results without grouping filters on mini-SVHN dataset, using VGGNet-19, ResNet-164 and DenseNet-40.

VGGNet-A has 8 convolutional layers and 3 fully connected layers without dropout layers. The parameters are concentrated in the fully connected layers, while the computations are concentrated in the convolutional layers. The number of parameters of the first fully connected layer is nearly 103 million, which is 77.5% of the total parameters. The number of FLOPs of the fourth and sixth convolutional layers add up to 7.4 billion, which is nearly 48.7% of the total FLOPs. In our experiments, we also modeled the parameters of fully connected layers through the GSM model. We sorted all $\boldsymbol{\theta}$ and pruned all layers. From Table VII, we can see that our pruned VGGNet-A achieved 0.14% lower testing error than [22] with 8.47% fewer parameters. Note that the FLOP results of baseline and slimming for VGGNet-A in Table VII are directly cited from [22]; according to our own calculation, the baseline FLOP of VGGNet-A is around $15.2B$ (rather than $45.7B$) which is more consistent with the well-known $30.9B$ baseline result for VGGNet-16 given by [8](since VGGNet-A has 5 fewer convolutional layers than VGGNet-16).

E. Results of multi-groups GSM modeling

For multi-groups GSM modeling, we have obtained the network compression results for VGGNet-19 on CIFAR and SVHN datasets. The filters in each layer were divided into G groups ($G = 2, 4$ in our experiment). The results are reported in Table VIII. The two rows for group=2,4 represent different pruning ratio results of the same model. From Table VIII, it can be seen that the multi-groups GSM modeling method can prune more parameters than its one-group counterpart. For CIFAR10, with 3.2% fewer parameters and 11.7% fewer FLOPs, the VGGNet pruned by the proposed method with 4-groups achieves slightly better accuracy than its one-group counterpart. For CIFAR100, the 2-groups GSM method saved more than 6.2% parameters and 11.6% FLOPs over one-group, with only 0.43% accuracy degradation. For the SVHN dataset, the multi-groups GSM method also leads to better network pruning performance.

F. Comparison against other competing approaches

Finally, we want to demonstrate the superiority of the proposed approach to other competing methods such as non-pruning based and variational Bayesian (VB) based network compression. Table IX includes the compression performance comparison against three non-pruning based approaches: adversarial network compression (ANC) [54], Automl for model compression (AMC)[44] and Cascaded Projection (CaP) [45] - on ResNet-56 and CIFAR-10. It can be observed that out joint-grouping-and-pruning convincingly outperforms the two benchmark schemes (higher pruning ratio and comparable error performance). Table X includes the compression performance comparison against several VB-based methods ([55], [56] and [57]). In order to compare to [50], [55] and [57], we have pruned LeNet-300-100, LeNet-5-Caffe on MNIST and VGGNet on CIFAR-10. It can be seen from Table X that our proposed method can prune more parameters, save more FLOPs and still achieve lower testing errors than other VB-based methods, especially VGGNet on CIFAR-10. With 18.14% and 12.71% less FLOPs, our pruned VGGNet achieves 2% and 2.4% lower testing errors than BC-GNJ and BC-GHS (VB-based method [50] with two different priors) respectively.

V. Conclusion

In this paper, we have proposed a hybrid approach of compressing DCNNs by joint pruning and grouping based on structured GSM models. To effectively exploit the high dependencies among network parameters, we have developed a principled solution to model a group of parameters sharing the same GSM prior by structured GSMs. Under this new structured sparsity framework, a collection of less important parameters associated with the same channel can be grouped first and then effectively pruned together. On several well-known benchmark datasets and network architectures, our

Model	Total		Conv		FC		Top-1 Errors
	Param	FLOPs	Param	FLOPs	Param	FLOPs	
Baseline	132.9M	15.2B	9.22M	14.97B	123.63M	0.25B	36.75%
Pruned_t	12.0M 90.97%	11.7B 23.03%	6.86M 25.60%	11.68B 21.98%	5.10M 95.87%	0.01B 96.00%	36.52%
Pruned_s	14.58M 89.03%	6.39B 57.96%	3.28M 64.43%	6.37B 57.45%	11.26M 90.89%	0.02B 92.00%	39.82%
Pruned_h	6.14M 95.38%	4.00B 73.68%	1.95M 78.85%	3.99B 73.34%	4.16M 96.63%	0.01B 96.00%	44.20%

TABLE VI: Results on ImageNet (1-group) using VGGNet-A. Here, M/B means (10^6/10^9), respectively. "Pruned_t" denotes the method where we pruned the FC and Conv layers together, "Pruned_s" denotes the method where we pruned FC and Conv separately, and 'Pruned_h" denotes the method where we pruned the FC and Conv together on high compression ratio.

Method	Params	Pruned	FLOPs	Pruned	Errors
Baseline [22]	132.9M	-	45.7B	-	36.69%
Slimming [22]	23.2M	82.50%	31.8B	30.40%	36.66%
Baseline (Ours)	132.9M	-	15.2B	-	36.75%
Ours	12.0M	90.97%	11.7B	23.03%	**36.52%**

TABLE VII: Results without grouping filters on ImageNet, using VGGNet-A. Here, M/B means (10^6/10^9), respectively.

Dataset	Groups	Params	Pruned	FLOPs	Pruned	Errors
C-10	1	1.80M	91.0%	0.32B	60.4%	6.02%
		1.23M	93.9%	0.28B	64.9%	6.52%
	2	1.80M	91.0%	0.23B	71.4%	6.38%
		0.89M	95.6%	0.19B	76.5%	6.50%
	4	1.77M	91.2%	0.25B	68.3%	6.05%
		1.17M	94.2%	0.22B	72.1%	**5.99%**
C-100	1	4.72M	76.5%	0.50B	37.8%	**25.82%**
		3.64M	81.9%	0.47B	41.0%	26.56%
	2	5.24M	73.9%	0.45B	44.0%	26.10%
		3.48M	82.7%	0.40B	49.6%	26.25%
	4	4.71M	76.6%	0.40B	50.4%	26.04%
		3.63M	82.0%	0.36B	55.1%	26.47%
SVHN	1	1.66M	91.7%	0.16B	79.5%	2.09%
		0.90M	95.5%	0.13B	83.7%	2.25%
	2	2.01M	90.0%	0.28B	64.5%	**2.01%**
		0.87M	95.7%	0.16B	79.5%	2.15%
	4	1.93M	90.4%	0.24B	69.6%	2.03%
		0.79M	96.1%	0.13B	83.7%	2.08%

TABLE VIII: Results of Multi-Group GSM modeling for VGGNet-19 pruning. Here, M/B means (10^6/10^9), respectively.

proposed method have achieved improved tradeoff between computational savings and classification accuracy (sometimes the acceleration is significant without noticeable sacrifice on accuracy). We will report additional experimental results on other datasets and network architectures (e.g., ResNet on ImageNet) in the future.

REFERENCES

[1] J. M. Alvarez and M. Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pages 2270–2278, 2016. 1

Method	Params	Pruned	FLOPs	Pruned	Errors
ANC[54]	0.27M	68.6%	-	-	8.08%
AMC[44]	-	-	1.26×10^8	50.0%	8.10%
CaP [45]	-	-	1.27×10^8	49.8%	6.78%
Ours	0.35M	59.3%	1.20×10^8	51.2%	6.83%

TABLE IX: Comparison of ResNet-56 with other accelerating non-pruning approaches on CIFAR-10.

[2] E. J. Candes, M. B. Wakin, and S. P. Boyd. Enhancing sparsity by reweighted l_1 minimization. *Journal of Fourier analysis and applications*, 14(5-6):877–905, 2008. 1, 3, 4
[3] M. Courbariaux and Y. Bengio. Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or- 1. arxiv: 1602.02830, 2016, 2017. 2
[4] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 1
[5] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014. 2
[6] W. Dong, G. Shi, Y. Ma, and X. Li. Image restoration via simultaneous sparse coding: Where structured sparsity meets gaussian scale mixture. *International Journal of Computer Vision*, 114(2-3):217–232, 2015. 1, 3
[7] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 2
[8] 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*, pages 1135–1143, 2015. 1, 8
[9] S. J. Hanson and L. Y. Pratt. Comparing biases for minimal network construction with back-propagation. In *Advances in neural information processing systems*, pages 177–185, 1989. 2
[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2, 4
[11] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, volume 2, 2017. 1, 5, 7
[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1, 2
[13] G. Huang, S. Liu, L. van der Maaten, and K. Q. Weinberger. Condensenet: An efficient densenet using learned group convolutions. *group*, 3(12):11, 2017. 1, 4
[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017. 4
[15] Z. Huang and N. Wang. Data-driven sparse structure selection for deep neural networks. *arXiv preprint arXiv:1707.01213*, 2017. 1, 2, 3, 5, 7
[16] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 2
[17] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014. 1
[18] A. Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Tront*, 2009. 4
[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1
[20] 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. *arXiv preprint arXiv:1609.04802*, 2016. 1
[21] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning

Model & size & (Params,FLOPs) & Error	Method	Pruned architecture	Params	Params pruned	FLOPs	FLOPs pruned	Error
LeNet-300-100 784-300-100 (266610, 533220) 1.6%	Sparse VD (impl. by [50])	512-114-72	67492	74.69%	134984	74.69%	1.8%
	SBP	245-160-55	48775	81.71%	97550	81.71%	1.6%
	BC-GNJ	278-98-13	28769	89.21%	57538	89.21%	1.8%
	BC-GHS	311-86-14	28200	89.42%	56400	89.42%	1.8%
	OURS	251-94-33	27163	89.81%	54326	89.81%	1.8%
LeNet-5-Caffe 20-50-800-500 (431520, 4.586M) 0.6%	Sparse VD (impl. by [50])	14-19-242-131	40292	90.66%	1.320M	71.22%	1.0%
	SBP	3-18-284-283	84684	80.38%	0.436M	90.49%	0.9%
	BC-GNJ	8-13-88-13	4186	99.03%	0.566M	87.66%	1.0%
	BC-GHS	5-10-76-16	2806	99.35%	0.301M	93.44%	1.0%
	OURS	4-14-74-11	2424	99.44%	0.296M	93.55%	1.0%
VGG (2 x 64)-(2 x 128)- -(3 x 256)-(8 x 512) (19.43M, 720.78M) 6.1%	SBP (impl. by [56])	47-50-91-115-227-160-50- -72-51-12-34-39-20-20-272	0.94M	95.16%	199.20M	72.36%	9.0%
	BC-GNJ	63-64-128-128-245-155-63- -26-24-20-14-12-11-11-15	1.00M	94.85%	283.08M	60.73%	8.6%
	BC-GHS	51-62-125-128-228-129-38- -13-9-6-5-6-6-6-20	0.82M	95.78%	243.91M	66.16%	9.0%
	RBP	50-63-123-108-104-57-23- -14-9-8-6-7-11-11-12	0.39M	97.99%	179.21M	75.14%	9.0%
	OURS	24-57-100-106-148-114-86- -57-18-11-12-4-4-12-34	0.60M	96.91%	152.30M	78.87%	6.6%

TABLE X: The pruning results for our method and other Variational Bayesian methods (Sparse VD [55], SBP [57], BC-GNJ [50], BC-GHS [50] and RBP [56]). Here, M means 10^6. BC-GNJ and BC-GHS denote the Bayesian Compression (BC) with group normal-Jeffreys (BC-GNJ) and group horseshoe (BC-GHS) priors proposed by [50], respectively.

filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 2, 3, 5, 6, 7

[22] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2755–2763. IEEE, 2017. 1, 2, 3, 4, 5, 6, 7, 8, 9

[23] B. Liu, M. Wang, H. Foroosh, M. Tappen and M. Pensky. Sparse convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814. IEEE, 2015. 1

[24] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. *arXiv preprint arXiv:1707.06342*, 2017. 1

[25] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical convolutional features for visual tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 3074–3082, 2015. 1

[26] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, number 2, page 5, 2011. 4

[27] J. Portilla, V. Strela, M. Wainwright, and E. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Trans. Signal Process.*, 12(11):1338–1351, Nov. 2003. 1, 3

[28] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016. 1, 2

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 4

[31] V. Sindhwani, T. Sainath, and S. Kumar. Structured transforms for small-footprint deep learning. In *Advances in Neural Information Processing Systems*, pages 3088–3096, 2015. 2

[32] 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*, pages 2074-2082, 2016. 6

[33] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016. 1

[34] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: an extremely efficient convolutional neural network for mobile devices (2017). arxiv preprint. *arXiv preprint arXiv:1707.01083*. 1, 2, 4

[35] X. Zhang, J. Zou, X. Ming, K. He, and J. Sun. Efficient and accurate approximations of nonlinear convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1984–1992, 2015. 1

[36] H. Zhou, J. M. Alvarez, and F. Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer, 2016. 1

[37] Y. Li, S. Lin, B. Zhang, J. Liu, D. Doermann, Y. Wu, F. Huang and R. Ji. Exploiting Kernel Sparsity and Entropy for Interpretable CNN Compression. *arXiv preprint arXiv:1812.04368*, 2018. 2

[38] S. Kang, J. Yu and K. Choi. Tapered-Ratio Compression for Residual Network. In *2018 International SoC Design Conference (ISOCC)*, pages=72–73, IEEE, 2018. 5, 7

[39] Y. Hu, S. Sun, J. Li, X. Wang and Q. Gu. A novel channel pruning method for deep neural network compression. *arXiv preprint arXiv:1805.11394*, 2018. 3, 5, 7

[40] B. Peng, W. Tan, Z. Li, S. Zhang, D. Xie, S. Pu. Extreme network compression via filter group approximation. In European Conference on Computer Vision (ECCV), pages 300–316, 2018. 5, 7

[41] J. Ye, X. Lu, Z. Lin, J. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *arXiv preprint arXiv:1802.00124*, 2018. 2, 3, 5, 7

[42] Robert. Tibshirani, Regression shrinkage and selection via the lasso. In *Journal of the Royal Statistical Society: Series B (Methodological)*, 58.1 (1996): 267-288. 1

[43] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012. 4

[44] Y. He, J. Lin, Z. Liu, H. Wang, L. Li, and S. Han. Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European Conference on Computer Vision (ECCV), pages 784–800, 2018. 2, 5, 7, 8, 9

[45] M. Breton, and A. Savakis. Cascaded Projection: End-to-End Network Compression and Acceleration. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 2, 5, 7, 8, 9

[46] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian. Variational Convolutional Neural Network Pruning. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 5, 7

[47] 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 The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 5, 7

[48] R. Yu, A. Li, C. Chen, J. Lai, V. Morariu, X. Han, M. Gao, C. Lin, and L. Davis. NISP: Pruning Networks Using Neuron Importance Score Propagation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 5, 7

[49] Y. Li, S. Lin, B. Zhang, J. Liu, D. Doermann, Y. Wu, F. Huang and R. Ji. Exploiting Kernel Sparsity and Entropy for Interpretable CNN Compression. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. 5, 7, 8

[50] C. Louizos, K. Ullrich, and M. Welling. Bayesian compression for deep learning. In Neural Information Processing Systems (NIPS), 2017. 1, 8, 10

[51] C. Louizos and M. Welling. Multiplicative normalizing flows for

variational bayesian neural networks. In International Conference on Machine Learning (ICML), 2017. 1

[52] M. Federici, K. Ullrich, and M. Welling. Improved bayesian compression. In arXiv:1711.06494, 2017. 1

[53] J. Achterhold, J. M. Kohler, A. Schmeink and T. Genewein. Variational network quantization. In Inter. Conf. on Learning Representation (ICLR), 2018. 1

[54] V. Belagiannis, A. Farshad, and F. Galasso. Adversarial network compression. In Proceedings of the European Conference on Computer Vision (ECCV), 2018. 8, 9

[55] M. Dmitry, A. Arsenii and V. Dmitry. Variational dropout sparsifies deep neural networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 2498–2507, 2017 8, 10

[56] Y. Zhou, Y. Zhang, Y. Wang, and Q. Tian. Accelerate CNN via Recursive Bayesian Pruning. In The IEEE International Conference on Computer Vision (ICCV). IEEE, 2019. 8, 10

[57] N. Kirill, M. Dmitry, A. Arsenii and V. Dmitry P. Structured bayesian pruning via log-normal multiplicative noise In Advances in Neural Information Processing Systems, pages 6775–6784, 2017. 8, 10