An Efficient Sharing Grouped Convolution via Bayesian Learning

Tinghuan Chen, *Graduate Student Member, IEEE*, Bin Duan, Qi Sun, *Graduate Student Member, IEEE*, Meng Zhang, Guoqing Li, *Graduate Student Member, IEEE*, Hao Geng, Qianru Zhang, and Bei Yu, *Member, IEEE*

***Abstract*— Compared with traditional convolutions, grouped convolutional neural networks are promising for both model performance and network parameters. However, existing models with the grouped convolution still have parameter redundancy. In this article, concerning the grouped convolution, we propose a sharing grouped convolution structure to reduce parameters. To efficiently eliminate parameter redundancy and improve model performance, we propose a Bayesian sharing framework to transfer the vanilla grouped convolution to be the sharing structure. Intragroup correlation and intergroup importance are introduced into the prior of the parameters. We handle the Maximum Type II likelihood estimation problem of the intragroup correlation and intergroup importance by a group LASSO-type algorithm. The prior mean of the sharing kernels is iteratively updated. Extensive experiments are conducted to demonstrate that on different grouped convolutional neural networks, the proposed sharing grouped convolution structure with the Bayesian sharing framework can reduce parameters and improve prediction accuracy. The proposed sharing framework can reduce parameters up to 64.17%. For ResNeXt-50 with the sharing grouped convolution on ImageNet dataset, network parameters can be reduced by 96.875% in all grouped convolutional layers, and accuracies are improved to 78.86% and 94.54% for top-1 and top-5, respectively.**

***Index Terms*— Bayesian inference, convolutional neural networks (CNNs), grouped convolution, LASSO.**

I. Introduction

CONVOLUTIONAL neural networks (CNNs) have achieved impressive success in various applications of computer vision, such as object recognition [1], [2], object detection [3]–[5], and video analysis [6]. To handle complicated applications, CNN models become deeper and wider, which causes massive network parameters. The massive network parameters, however, bring huge challenges to model storage, data transfer, computation overhead, and energy consumption [7], [8]. Besides, the massive network parameters may contain redundancy, which causes overfitting and performance degradation.

The grouped convolution has been adopted to decrease parameter redundancy and improve accuracy in popular CNNs, such as AlexNet [9] and ResNeXt [10]. The vanilla grouped convolution is shown in Fig. 1(a), where the inputs, the weights, and the outputs are divided into several groups to perform the convolution operation. In practice, the grouped convolution is proved to be able to alleviate overfitting and improve the model accuracy, outperforming its counterpart, e.g., nongrouped ResNet versus grouped ResNeXt [10], [11]. Moreover, the grouped convolution is also proved to be more efficient and effective than wider and deeper networks [10].

Although the grouped convolution has the aforementioned advantages, the network parameters may still have redundancy. Various arts are proposed to reduce parameter redundancy. These methods can be classified into two types: model compression methods and architecture design methods [12]. Although existing compression methods have good compression performance in the traditional convolution models, they may lead to performance degradations while being applied to grouped convolutions since they ignore the diversities of importances and correlations (i.e., intergroup importance and intragroup correlation) among the different parameter groups. Without specific optimization techniques, directly training models with these group architectures may also degrade the performance.

To eliminate parameter redundancy and improve the efficiency of the grouped convolution, in this article, we propose a sharing grouped convolution structure, a novel and simple architecture, to reduce parameters, as shown in Fig. 1(b). A Bayesian grouped convolution sharing framework is proposed to transfer the vanilla grouped convolution to be the sharing structure. Intragroup correlation and intergroup importance are introduced into the prior of network parameters. We handle the Maximum Type II likelihood estimation problem of the intragroup correlation and intergroup importance by a group LASSO-type algorithm [13]. The prior mean is iteratively updated with the posterior mean and the intergroup importance learned in the previous iteration.

Manuscript received 17 June 2020; revised 25 February 2021; accepted 19 May 2021. Date of publication 10 June 2021; date of current version 1 December 2022. This work was supported in part by the National Key Research and Development Program of China under Project 2018YFB2202703, in part by the Research Grants Council of Hong Kong, SAR, under Project CUHK14209420, and in part by the Natural Science Foundation of Jiangsu Province under Project BK20201145. *(Corresponding authors: Bei Yu; Meng Zhang.)*

Tinghuan Chen, Qi Sun, Hao Geng, and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR (e-mail: byu@cse.cuhk.edu.hk).

Bin Duan is with the School of Microelectronics, Southeast University, Wuxi 214061, China.

Meng Zhang, Guoqing Li, and Qianru Zhang are with the School of Electronics Science and Engineering, Southeast University, Nanjing 210096, China (e-mail: zmeng@seu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3084900.

Digital Object Identifier 10.1109/TNNLS.2021.3084900

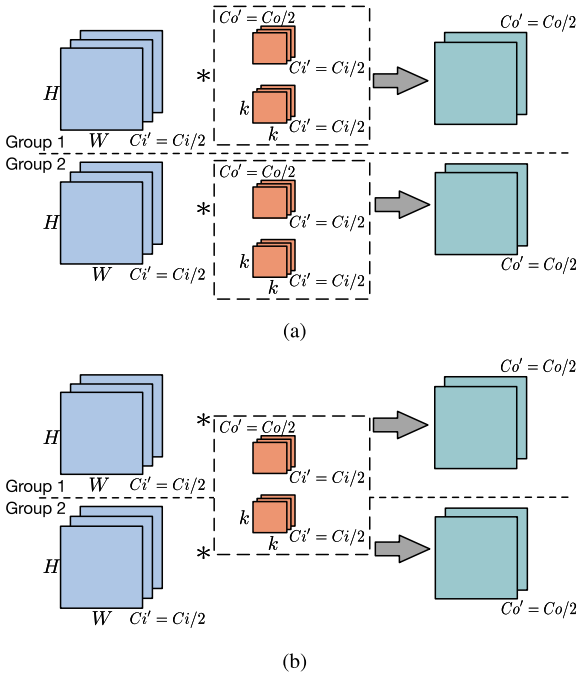


Fig. 1. Vanilla grouped convolution and our proposed sharing grouped convolution (two groups, the blue boxes are the input features, the orange boxes are the kernels, and the green boxes are the output features). H and W are the height and weight of the input features. Ci and Co are the numbers of the input and output channels. Ci' and Co' are the numbers of the input and output channels in each group. k is kernel size. (a) Vanilla grouped convolution. Each group has its own weights. (b) Our proposed sharing grouped convolution. All of these groups share the same weights.

We conduct experiments on three benchmarks, CIFAR-10, CIFAR-100 [14], and ImageNet [15], to validate our proposed sharing grouped convolution structure with Bayesian sharing framework. Three typical and popular grouped CNNs, ResNeXt [10], ShuffleNet [16], and DenseNet [17], are transferred to be the sharing structure under different grouped convolution configurations by using our proposed framework. Experiments demonstrate that our framework can reduce parameters significantly and improve model accuracies. The proposed sharing framework can reduce parameters up to 64.17%. Especially, for the sharing ResNeXt-50 on ImageNet dataset, the number of network parameters is reduced by 96.875% in the grouped convolutional layers and accuracies are improved to 78.86% and 94.54% for top-1 and top-5, respectively.

The rest of this article is organized as follows. In Section II, we provide a survey about model compression algorithms and efficient model architectures. In Section III, we propose a sharing grouped convolution structure. In Section IV, we consider intragroup correlation and intergroup importance and then propose a Bayesian sharing framework. Section V presents the experimental results with comparisons and discussions, followed by a conclusion in Section VI.

II. Related Works

A. Model Compression

The wider and deeper CNN models bring great challenges to model storage, computation, data communication, and system power consumption [18]. Therefore, many model compression methods were proposed to address these challenges. Consequently, the model compression can reduce the inference runtime. The inference runtime consists of memory access and computation [7], [8], where memory access is usually the runtime bottleneck [19].

Typical model compression methods can be categorized into model pruning, bit quantization, low-rank approximation, and knowledge distillation [20]. Model pruning methods can be used to prune parameters in different manners, e.g., channel-wise and depthwise prunings [21]–[27] or structural and non-structural prunings [28], [29]. In particular, attention statistics were adopted to evaluate the importance of channels so that channels can be pruned by the evaluation [30]. The low-rank approximation and the model parameters sparsification can accelerate inference and reduce model storage [31]–[37]. Due to the redundancy of the data precision, bit quantization approaches learn low-bit representations of features and parameters [38], [39]. Therefore, bit quantization approaches are very useful for model deployment tasks on domain-specific hardware [8]. Different from other categories, knowledge distillation methods facilitate the training of lightweight models by using knowledge learned from large networks [34], [37].

Although these compression methods have good compression performance in the traditional convolution, they may lead to performance degradation for the grouped convolution since they ignore the diversities of intergroup importance and intragroup correlation among the parameter groups.

B. Efficient Model Architecture

Considering the aforementioned drawbacks in model compression methods, some works adopt other ways to design efficient architectures to improve network performance. Various parameter normalization layers were proposed to avoid performance degradation, such as batch normalization, switchable normalization, and exemplar normalization [40]–[44]. However, these normalization schemes cannot remove parameter redundancy and even make the networks cumbersome. Besides, some tricky neural architecture search methods are proposed to determine network configurations [45]. Some works replace large filters with smaller ones [1], [17]. In order to further eliminate redundancies, some arts adopt separable convolution [46], that is, replacing a 3-D convolution with multiple 2-D convolutions. For example, a 3-D convolution is factorized to be two 2-D ones in Inception V3 [47]. The predefined sparse 2-D kernels are used to make a tradeoff between accuracy and energy consumption [18]. The depthwise separable convolution is adopted in MobileNets [48] and Xception [49].

In particular, the grouped convolution is an efficient architecture outperforming its counterpart, e.g., nongrouped ResNet *vs.* grouped ResNeXt [10], [11]. The grouped convolution was first proposed in AlexNet [9], which allocates models on two GPUs to facilitate parallelism. The grouped convolution adopts the sparse convolution connections between input and output channels, by dividing the input channels, output channels, and their connections into several groups as shown in Fig. 1(b). Compared with the traditional convolutions with

"fully connected" features and weights, the parameters and computation costs are reduced.

The success of grouped convolution has inspired its wide applications, e.g., ShuffleNet [16] and CondenseNet [17]. Recently, to improve the model accuracy, interleaved grouped convolutions were designed to further improve parameter efficiency and classification accuracy [50]. Wu and He [51] proposed a group normalization layer, where the mean and variance are computed within each group. A dynamic grouped convolution was designed with different numbers of channels in the same layer [11], [52]. A fully learnable grouped convolution was proposed to freely choose the grouping strategy in the training stage [12]. The optimal channel permutation was developed to explore group configuration [53]. However, the correlation among parameters and groups do not be considered so that these methods will cause performance degradation.

To further remove the parameter redundancy in grouped convolutional layers, we propose a sharing grouped convolution structure. To improve model performance, we propose a Bayesian sharing framework to transfer the vanilla grouped convolution to be the sharing structure, which is general and can be integrated into various current grouped convolution configurations. Our proposed framework is complementary to these model compression methods.

III. Sharing Grouped Convolution

In this section, a sharing grouped convolution structure is proposed to reduce parameter redundancy and improve parameter efficiency. Then, the number of parameters in it is analyzed to illustrate the compression performance.

To demonstrate the vanilla grouped convolution, the variation from ResNet to ResNeXt [10], [11] is taken as an example. Fig. 2 shows their basic block, which is repeatedly stacked with different configurations to the whole model. The basic block contains a shortcut and three convolutional layers, whose all kernels are represented by a box in each layer. In ResNet, the basic block is shown in Fig. 2(a), where there are three traditional convolutional layers. In order to transfer ResNet to ResNeXt, in each block, the second convolutional layer is transferred as the vanilla grouped convolution by dividing the 64 channels into 16 groups, and each group has four channels for this example, as shown in Fig. 2(b). Compared with the traditional convolution, the vanilla grouped convolution adopts the sparse convolution connections between input and output channels, by dividing the input channels, output channels, and their connections into several groups. According to Fig. 2, the parameter number for the second convolutional layer is reduced from $64 \times 3 \times 3 \times 64 = 36\,864$ of ResNet to $16 \times 4 \times 3 \times 3 \times 4 = 2304$ of ResNeXt in the convolutional layer.

In order to further reduce the parameter number and improve parameter efficiency in the vanilla grouped convolution, we propose a sharing grouped convolution structure. Specifically, all groups share the same parameters so that the same parameters can be used to extract features and pass information among different groups. It has the same manner with [50] to improve parameter efficiency. Then, in each basic block, the vanilla grouped convolution (the second layer) as shown in Fig. 2(b) will be transferred as the sharing grouped convolution, as shown in Fig. 3. The parameter number for the second convolutional layer is reduced from $16 \times 4 \times 3 \times 3 \times 4 = 2304$ to $4 \times 3 \times 3 \times 4 = 144$. Note that compared with the vanilla grouped convolution, our proposed sharing grouped convolution does not reduce computational complexity. However, as shown in Fig. 3, the parameters are shared among different groups. Therefore, the efficiency of parameters is improved and the parameter redundancy can be reduced. Besides, the sharing grouped convolution can facilitate the weights reusing strategy in the hardware-level implementations so that the actual number of memory accesses decreases significantly and the inference runtime reduces.

TABLE I

Numbers of Parameters of Basic Blocks in ResNet, ResNext, and the Sharing ResNeXt With $g = 16$

Type	ResNet	ResNeXt	Sharing ResNeXt
conv	$64 \times 1 \times 1 \times 64$	$64 \times 1 \times 1 \times 64$	$64 \times 1 \times 1 \times 64$
(g)conv	$64 \times 3 \times 3 \times 64$	$16 \times 4 \times 3 \times 3 \times 4$	$4 \times 3 \times 3 \times 4$
conv	$64 \times 1 \times 1 \times 128$	$64 \times 1 \times 1 \times 128$	$64 \times 1 \times 1 \times 128$
shortcut	$64 \times 1 \times 1 \times 128$	$64 \times 1 \times 1 \times 128$	$64 \times 1 \times 1 \times 128$
total	57344	22784	**20624**

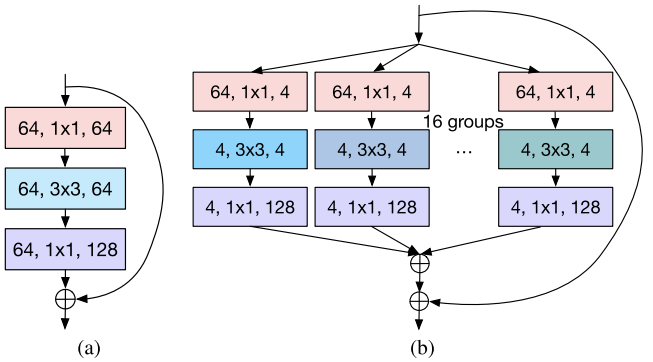


Fig. 2. Basic block contains a shortcut and three convolutional layers (the boxes indicate the convolutional kernels (#input channel, kernel size, and #output channel) for each layer). (a) Three convolutional layers in ResNet. (b) Two convolutional layers and one vanilla grouped convolutional layer with 16 groups in ResNeXt.

As a comparison, we show the numbers of parameters of basic blocks in ResNet, ResNext, and the sharing ResNeXt in Table I. Compared with ResNet, ResNeXt has fewer parameters, and the proposed sharing ResNeXt can further reduce the number of parameters.

Although the proposed sharing grouped convolution structure can reduce parameters and improve the efficiency of parameters, directly training models with the group convolution structure may cause performance degradation since the correlation among parameters and groups does not be considered.

IV. Bayesian Sharing Framework

To transfer the vanilla grouped convolution into the sharing structure, a naïve method is directly constructing a network with the proposed sharing grouped convolution structure and then training it. However, this method may cause performance

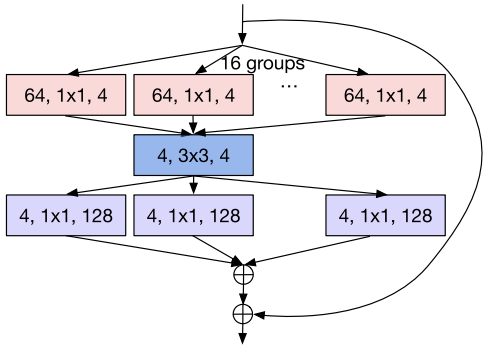


Fig. 3. Basic block contains a shortcut, two convolutional layers, and one sharing grouped convolutional layer with 16 groups in the sharing ResNeXt (the boxes indicate the convolutional kernels (#input channel, kernel size, and #output channel) for each layer).

TABLE II
LIST OF NOTATIONS

Name	Definition
$\boldsymbol{w}$	parameters in one grouped convolutional layer
g	# of groups in one grouped convolutional layer
$\boldsymbol{B}_i$	intra-group correlation of the group i
γ_i	inter-group importance of the group i
$\boldsymbol{w}_i$	parameters of the group i
$\boldsymbol{w}_b$	the sharing parameters of g groups
k	kernel size
Ci'	# of input channels in each group
Co'	# of output channels in each group
H	height of input feature
W	width of input feature

degradation. To avoid performance degradation, we adopt a separate-merge methodology [23], that is, updating independently parameters among all groups in the backpropagation stage and computing loss function value by the shared parameters in the forward propagation stage. Based on the separate-merge methodology, a typical method is indiscriminately averaging the parameters among different groups in the forward propagation stage. Given a pretrained model, we can directly average these parameters among different groups. This is quite straightforward, but it ignores the diversities of different groups.

In this section, to efficiently eliminate parameter redundancy and improve model performance, we introduce the intragroup correlation and intergroup importance of parameters. Then, we propose a Bayesian sharing framework. Some notations used in this article are listed in Table II and visualized in Fig. 1.

A. Intragroup Correlation and Intergroup Importance

To introduce intragroup correlation and intergroup importance, a prior distribution on model parameters $\mathcal{P}(\boldsymbol{w})$ is first introduced in our framework. Following previous arts [54]–[56], $\mathcal{P}(\boldsymbol{w})$ is defined to be a multivariate Gaussian distribution. For the convenience of expressions, network parameters are reshaped to be vectors. As shown in Fig. 4, in each group, the kernels in each channel are flattened to be a vector. Some notations are explained in Table II. Then, they are concatenated sequentially to be a vector. Considering that the features are extracted independently of different groups

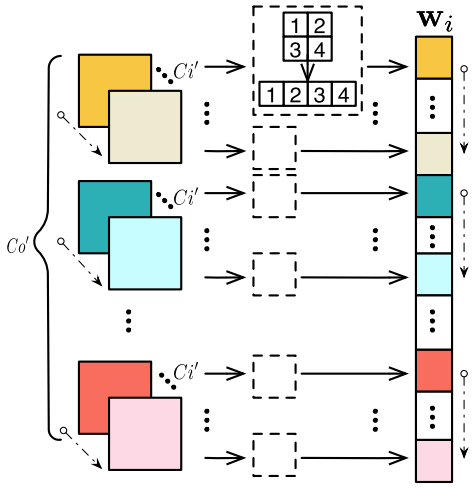


Fig. 4. Reshape the parameter tensor of one group as a vector, with Ci' input channels and Co' output channels. The kernel size is 2×2. The arrows show the flattening order.

in the vanilla grouped convolution as shown in Fig. 1(a), we assume that any two network parameters from different groups are independent, i.e., $\mathcal{P}(\boldsymbol{w}) = \prod_i^g \mathcal{P}(\boldsymbol{w}_i)$. Therefore, the prior distribution of network parameters in the group i is defined as follows:

$$\mathcal{P}(\boldsymbol{w}_i; \gamma_i, \boldsymbol{B}_i) \sim \mathcal{N}\big(\boldsymbol{\mu}_{\boldsymbol{w}_i}, \boldsymbol{\Sigma}_{\boldsymbol{w}_i}\big), \ \ \boldsymbol{\Sigma}_{\boldsymbol{w}_i} \triangleq \gamma_i \boldsymbol{B}_i \tag{1}$$

where $\boldsymbol{w}_i \in \mathbb{R}^{NCo'}$ and $\boldsymbol{B}_i \in \mathbb{R}^{NCo' \times NCo'}$ with $N \triangleq k^2 Ci'$. $\boldsymbol{B}_i$ is a positive definite matrix which captures the correlations of the parameters in group i, termed intragroup correlation. γ_i is a coefficient reflecting the relative importance of group i in comparison with other groups, termed intergroup importance. γ_i also indicates the importance of the group i while passing messages or knowledges in the model during inference. $\boldsymbol{\mu}_{\boldsymbol{w}_i}$ is the mean vector of the network parameters $\boldsymbol{w}_i$ in the group i. Also, $\boldsymbol{\Sigma}_{\boldsymbol{w}_i}$ is the covariance matrix of the network parameters $\boldsymbol{w}_i$ in the group i. For the convolutional layer with g groups, the prior distribution of network parameters is

$$\mathcal{P}(\boldsymbol{w}; \boldsymbol{B}, \boldsymbol{\gamma}) \sim \mathcal{N}\big(\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}\big) \tag{2}$$

where $\boldsymbol{\mu}_{\boldsymbol{w}} = [\boldsymbol{\mu}_{\boldsymbol{w}_1}^\top, \boldsymbol{\mu}_{\boldsymbol{w}_2}^\top, \ldots, \boldsymbol{\mu}_{\boldsymbol{w}_g}^\top]^\top$ is the mean vector of the network parameters $\boldsymbol{w}$. $\boldsymbol{\Sigma}_{\boldsymbol{w}} = \text{diag}[\boldsymbol{\Sigma}_{\boldsymbol{w}_1}, \boldsymbol{\Sigma}_{\boldsymbol{w}_2}, \ldots, \boldsymbol{\Sigma}_{\boldsymbol{w}_g}]$ is the covariance matrix of $\boldsymbol{w}$, which is a block-diagonal matrix with principal diagonal blocks being $\boldsymbol{\Sigma}_{\boldsymbol{w}_1}, \boldsymbol{\Sigma}_{\boldsymbol{w}_2}, \ldots, \boldsymbol{\Sigma}_{\boldsymbol{w}_g}$. $\boldsymbol{B} \triangleq \{\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_g\}$ and $\boldsymbol{\gamma} \triangleq [\gamma_1, \gamma_2, \ldots, \gamma_g]^\top$. The intragroup correlation $\boldsymbol{B}_i$ and the intergroup importance γ_i are determined by maximizing Type II likelihood [13] as shown in the following equation:

$$\max_{\boldsymbol{B}, \boldsymbol{\gamma}} \ \ \ln \int \mathcal{P}(\mathcal{Y}|\mathcal{X}, \boldsymbol{w}) \mathcal{P}(\boldsymbol{w}; \boldsymbol{B}, \boldsymbol{\gamma}) \mathrm{d}\boldsymbol{w} \tag{3}$$

where $\mathcal{Y}$ and $\mathcal{X}$ are the output and input features, respectively, and $\mathcal{P}(\boldsymbol{w}; \boldsymbol{B}, \boldsymbol{\gamma})$ satisfies multivariate Gaussian distribution with hyperparameters $\boldsymbol{B}$ and $\boldsymbol{\gamma}$ defined in (2).

According to formulation (3), to obtain the intragroup correlation $\boldsymbol{B}_i$ and the intergroup importance γ_i, we need to give a concrete form of $\mathcal{P}(\mathcal{Y}|\mathcal{X}, \boldsymbol{w})$. Moreover, the concrete form of $\mathcal{P}(\mathcal{Y}|\mathcal{X}, \boldsymbol{w})$ relies on the relationships among input

features $\mathscr{X}$, output features $\mathscr{Y}$ and model parameters $\boldsymbol{w}$ in one grouped convolutional layer.

Nevertheless, in practice, because of nonlinear operations in CNN models, it is hard to obtain the closed form of the likelihood function $\mathcal{P}(\mathscr{Y}|\mathscr{X}, \boldsymbol{w})$ and the integral of the marginal likelihood in formulation (3) is intractable in neural networks [57]. Like in [55], we consider the linear relationship between the input and the output features of each layer before a nonlinearity is applied.

B. Maximum Type II Likelihood Estimation

In this section, the Type II likelihood in (3) is transformed for the ease of the computations of the intragroup correlation $\boldsymbol{B}_i$ and intergroup importance γ_i.

As mentioned above, we consider the linear relationship between the input and the output features of each layer before a nonlinearity is applied. To represent the vanilla grouped convolution in the form of matrix–vector multiplication, we reshape the input features, as shown in Fig. 5.

1) In Step 1, we reshape the input features of one group to be a block-diagonal matrix. As the parameter kernel window slides on the input feature, the corresponding features are flattened to be a vector with length k^2. Therefore, we flatten the feature in one channel to be an $HW \times k^2$ matrix. Then, the feature matrices of all Ci' channels are reshaped to be a block-diagonal matrix.
2) In Step 2, we duplicate the input feature block matrix by Co' times to generate a larger block-diagonal matrix.
3) In Step 3, we place the block-diagonal matrices of the g groups at the diagonal of the final feature matrix. The parameters are also reshaped in the same manner, as mentioned in Fig. 4.

For each group, the matrix–vector multiplication with model error $\boldsymbol{v}_i$ can be represented as $\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{w}_i + \boldsymbol{v}_i$, where $\boldsymbol{y}_i \in \mathbb{R}^{MCo'}$ and $\boldsymbol{X}_i \in \mathbb{R}^{MCo' \times NCo'}$ represent the reshaped outputs and inputs respectively, with $M \triangleq HW$. For a layer with g groups, the vanilla grouped convolution is

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{v} \tag{4}$$

where $\boldsymbol{y} = [\boldsymbol{y}_1^\top, \cdots, \boldsymbol{y}_g^\top]^\top$, $\boldsymbol{X} = \text{diag}[\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_g]$, and $\boldsymbol{v} = [\boldsymbol{v}_1^\top, \cdots, \boldsymbol{v}_g^\top]^\top$. The model error $\boldsymbol{v}$ is assumed to follow the independent identical Gaussian distribution, i.e., $\mathcal{P}(\boldsymbol{v}) \sim \mathcal{N}(\boldsymbol{0}, \lambda\boldsymbol{I})$, where λ is a hyperparameter controling the precision of model error and $\boldsymbol{I}$ is an identity matrix. The concrete form of the likelihood function in formulation (3) can be obtained as follows:

$$\mathcal{P}(\mathscr{Y}|\mathscr{X}, \boldsymbol{w}) = \mathcal{P}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}; \lambda) \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{w}, \lambda\boldsymbol{I}). \tag{5}$$

According to Fig. 5, the size of matrix $\boldsymbol{X}$ is $MCo'g \times NCo'g$. However, on one hand, note that the vanilla grouped convolution adopts the sparse convolution connections between input and output channels, by dividing the input channels, output channels, and their connections into several groups. Thus, the matrix–vector multiplication can be performed group by group, i.e., $\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{w}_i + \boldsymbol{v}_i$, where $\boldsymbol{X}_i$'s size is $MCo' \times NCo'$. Moreover, $\boldsymbol{X}_i$ is also a block-diagonal matrix, whose each block size is $M \times N$, which can facilitate the matrix–vector multiplication output channel by output channel. Besides, in popular vanilla grouped CNNs, for example, ResNeXt, small kernels (e.g., $k = 3$ or $k = 1$) are widely adopted. On the other hand, our proposed sharing framework is performed layer by layer in grouped convolutional layers, instead of all convolutional layers, in popular CNN models. Thus, in practice, this reshaping input features does not bring a horrible memory footprint. In addition, this reshaping is only performed in the training stage, while the memory footprint does not increase in the inference stage.

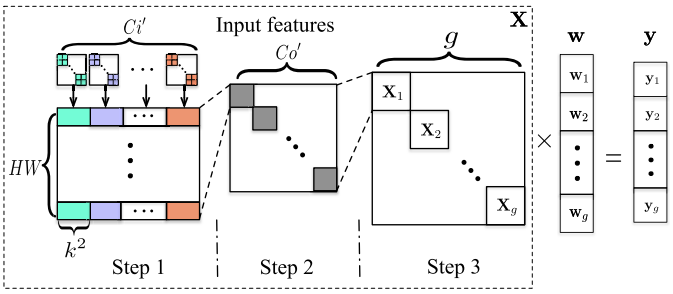


Fig. 5. Reshape input features. The vanilla grouped convolution operation is transformed as matrix–vector multiplication.

According to the network parameters prior $\mathcal{P}(\boldsymbol{w}_i; \gamma_i, \boldsymbol{B}_i)$ defined in (1) and the likelihood function $\mathcal{P}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})$ defined in (5), the posterior of network parameters also follows the multivariate Gaussian distribution $\mathcal{P}(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\gamma}, \boldsymbol{B}, \lambda) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where the mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ are represented as follows [13]:

$$\begin{aligned} \boldsymbol{\mu} &= \boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top\left(\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top\right)^{-1}\left(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}\right) \\ \boldsymbol{\Sigma} &= \left(\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1} + \frac{1}{\lambda}\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1} \end{aligned} \tag{6}$$

where $\boldsymbol{\mu} \triangleq [\boldsymbol{\mu}_1^\top, \ldots, \boldsymbol{\mu}_g^\top]^\top$ and $\boldsymbol{\Sigma} \triangleq \text{diag}[\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_g]$. $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ are the posterior mean and the covariance matrix of network parameters in the group i, respectively.

Now, to determine the intragroup correlation $\boldsymbol{B}_i$ and the intergroup importance γ_i, we can transform formulation (3) as follows:

$$\max_{\boldsymbol{B}, \boldsymbol{\gamma}, \lambda} \quad \ln \mathcal{P}(\boldsymbol{y}|\boldsymbol{X}; \boldsymbol{B}, \boldsymbol{\gamma}, \lambda) \tag{7}$$

where the marginal likelihood function $\mathcal{P}(\boldsymbol{y}|\boldsymbol{X}; \boldsymbol{B}, \boldsymbol{\gamma}, \lambda)$ is defined as follows [58], [59]:

$$\mathcal{P}(\boldsymbol{y}|\boldsymbol{X}; \boldsymbol{B}, \boldsymbol{\gamma}, \lambda) = \int \mathcal{P}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}; \lambda)\mathcal{P}(\boldsymbol{w}; \boldsymbol{B}, \boldsymbol{\gamma})\mathrm{d}\boldsymbol{w}. \tag{8}$$

Then, formulation (7) can be transformed to the equivalent formulation as follows:

$$\min_{\boldsymbol{B}, \boldsymbol{\gamma}, \lambda} \quad \mathcal{L}(\boldsymbol{B}, \boldsymbol{\gamma}, \lambda) \tag{9}$$

where

$$\begin{aligned} \mathcal{L}(\boldsymbol{B}, \boldsymbol{\gamma}, \lambda) &\triangleq -2\ln \mathcal{P}(\boldsymbol{y}|\boldsymbol{X}; \boldsymbol{B}, \boldsymbol{\gamma}, \lambda) \\ &= \ln|\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top| \\ &\quad + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}})^\top(\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top)^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}). \end{aligned} \tag{10}$$

Since it has the ability to adaptively learn and exploit intragroup correlation for better performance and only takes few iterations, in Section IV-C, we illustrate how to use a group LASSO-type method to handle formulation (9) so that the intragroup correlation $\boldsymbol{B}_i$, the intergroup importance γ_i, and the hyperparameter λ can be well determined.

C. Optimization via Group LASSO-Type Algorithm

In this section, we follow the work [60] and use a group LASSO-type algorithm to determine hyperparameters so that it can achieve fast convergence. The main idea is shown as follows. First, we find the upper bound of the cost function $\mathcal{L}(\boldsymbol{B}, \boldsymbol{\gamma}, \lambda)$ defined in (10). Then, the upper bound can be transformed to be a group LASSO problem. As a result, we can solve it with a typical group LASSO solver more efficiently.

In order to find an appropriate upper bound of $\mathcal{L}(\boldsymbol{B}, \boldsymbol{\gamma}, \lambda)$, we introduce a temporary function $h(\boldsymbol{\alpha}) \triangleq [(1/\lambda)||\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}-\boldsymbol{X}\boldsymbol{\alpha}||_2^2 + \boldsymbol{\alpha}^\top\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\boldsymbol{\alpha}]$. $\boldsymbol{\alpha}$ is defined as a temporary variable, which is different from the model parameters $\boldsymbol{w}$. Note that the function $h(\boldsymbol{\alpha})$ is convex. Therefore, there is a global minimum $\boldsymbol{\alpha}_0$, i.e., $h(\boldsymbol{\alpha}_0) \leq h(\boldsymbol{\alpha})$, with the first derivative $h(\boldsymbol{\alpha}_0)' = \boldsymbol{0}$, where

$$\boldsymbol{\alpha}_0 = (\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1} + \boldsymbol{X}^\top\boldsymbol{X})^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}})^\top\boldsymbol{X}. \tag{11}$$

Substituting (11) into the function $h(\boldsymbol{\alpha})$ and using the Woodbury matrix identity [61] lead to

$$h(\boldsymbol{\alpha}_0) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}})^\top(\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top)^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}). \tag{12}$$

Thus, for the right-hand side in (10), we have

$$\begin{aligned}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}})^\top(\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top)^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}) \\ \equiv \min_{\boldsymbol{\alpha}}\left[\frac{1}{\lambda}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}} - \boldsymbol{X}\boldsymbol{\alpha}||_2^2 + \boldsymbol{\alpha}^\top\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\boldsymbol{\alpha}\right].\end{aligned} \tag{13}$$

With this, (10) is upper bounded by

$$\begin{aligned}\mathcal{UL}(\boldsymbol{\alpha}, \boldsymbol{\gamma}, \ \boldsymbol{B}, \lambda) = \ln|\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top| \\ + \frac{1}{\lambda}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}} - \boldsymbol{X}\boldsymbol{\alpha}||_2^2 + \boldsymbol{\alpha}^\top\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\boldsymbol{\alpha}.\end{aligned} \tag{14}$$

Here, we temporarily fix $\boldsymbol{B}$ and λ. Then, instead of directly optimizing formulation (9), we minimize the upper bound in (14) with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ as follows:

$$\min_{\boldsymbol{\alpha},\boldsymbol{\gamma}} \quad \mathcal{UL}(\boldsymbol{\alpha}, \boldsymbol{\gamma}). \tag{15}$$

Furthermore, considering that the term $(1/\lambda)||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}} - \boldsymbol{X}\boldsymbol{\alpha}||_2^2$ is independent of $\boldsymbol{\gamma}$ in (14), formulation (15) can be handled in two steps alternatively and iteratively. In the first step, we optimize formulation (15) with respect to $\boldsymbol{\gamma}$ as follows:

$$f(\boldsymbol{\alpha}) \triangleq \min_{\boldsymbol{\gamma}\geq\boldsymbol{0}}\left[\ln|\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top| + \boldsymbol{\alpha}^\top\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\boldsymbol{\alpha}\right]. \tag{16}$$

In the second step, we optimize formulation (15) with respect to $\boldsymbol{\alpha}$ as follows:

$$\min_{\boldsymbol{\alpha}} \quad ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}} - \boldsymbol{X}\boldsymbol{\alpha}||_2^2 + \lambda f(\boldsymbol{\alpha}). \tag{17}$$

In the first step, since $g(\boldsymbol{\gamma}) \triangleq \ln|\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top|$ is nondecreasing and concave, $g(\boldsymbol{\gamma})$ can be expressed with its concave conjugate $g^*(\boldsymbol{z})$ as follows [62]:

$$g(\boldsymbol{\gamma}) \triangleq \ln|\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top| = \min_{\boldsymbol{z}\geq\boldsymbol{0}} \boldsymbol{z}^\top\boldsymbol{\gamma} - g^*(\boldsymbol{z}) \tag{18}$$

where the concave conjugate is given as follows:

$$g^*(\boldsymbol{z}) = \min_{\boldsymbol{\gamma}\geq\boldsymbol{0}} \boldsymbol{z}^\top\boldsymbol{\gamma} - \ln|\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top|. \tag{19}$$

Therefore, (16) can be transformed as follows:

$$\begin{aligned}f(\boldsymbol{\alpha}) &= \min_{\boldsymbol{\gamma},\boldsymbol{z}\geq\boldsymbol{0}} \boldsymbol{\alpha}^\top\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\boldsymbol{\alpha} + \boldsymbol{z}^\top\boldsymbol{\gamma} - g^*(\boldsymbol{z}) \\ &= \min_{\boldsymbol{\gamma},\boldsymbol{z}\geq\boldsymbol{0}} \ \sum_{i=0}^{g}\left(\frac{\boldsymbol{\alpha}_i^\top\boldsymbol{B}_i^{-1}\boldsymbol{\alpha}_i}{\gamma_i} + z_i\gamma_i\right) - g^*(\boldsymbol{z})\end{aligned} \tag{20}$$

where $\boldsymbol{z} = [z_1, z_2, \ldots, z_g]^\top$. Minimizing (20) with respect to $\boldsymbol{\gamma}$, we have

$$\gamma_i = z_i^{-\frac{1}{2}}\sqrt{\boldsymbol{\alpha}_i^\top\boldsymbol{B}_i^{-1}\boldsymbol{\alpha}_i}, \quad i = 1, 2, \ldots, g. \tag{21}$$

However, γ_i relies on z_i. According to (18) and the duality property [62], we can obtain

$$z_i = \mathrm{Tr}[\boldsymbol{B}_i\boldsymbol{X}_i^\top(\lambda\boldsymbol{I} + \boldsymbol{X}_i\boldsymbol{\Sigma}_{\boldsymbol{w}_i}\boldsymbol{X}_i^\top)^{-1}\boldsymbol{X}_i]. \tag{22}$$

According to (21) and (22), $\boldsymbol{\gamma}$ relies on $\boldsymbol{z}$ and $\boldsymbol{z}$ relies on $\boldsymbol{\gamma}$ ($\boldsymbol{\Sigma}_{\boldsymbol{w}}$). Therefore, in the first step, we optimize formulation (16) by updating $\boldsymbol{\gamma}$ and $\boldsymbol{z}$ alternatively.

In the second step, after $\boldsymbol{\gamma}$ and $\boldsymbol{z}$ are determined, formulation (17) is transformed as follows:

$$\min_{\boldsymbol{\alpha}} \quad ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}} - \boldsymbol{X}\boldsymbol{\alpha}||_2^2 + \lambda\sum_{i=1}^{g} 2z_i^{\frac{1}{2}}\sqrt{\boldsymbol{\alpha}_i^\top\boldsymbol{B}_i^{-1}\boldsymbol{\alpha}_i}. \tag{23}$$

Formulation (23) is an implicit group LASSO formulation. To make it more clear, we transform it to be formulation (24) as follows:

$$\min_{\boldsymbol{d}} \quad ||\boldsymbol{t} - \boldsymbol{H}\boldsymbol{d}||_2^2 + \lambda\sum_{i=1}^{g}||\boldsymbol{d}_i||_2 \tag{24}$$

where $\boldsymbol{d}_i = 2z_i^{1/2}\boldsymbol{B}_i^{-1/2}\boldsymbol{\alpha}_i$, $\boldsymbol{d} = [\boldsymbol{d}_1^\top, \boldsymbol{d}_2^\top, \ldots, \boldsymbol{d}_g^\top]^\top$, $\boldsymbol{t} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}$, and $\boldsymbol{H} = \boldsymbol{X}\cdot\mathrm{diag}[\boldsymbol{B}_1^{1/2}/(2z_1^{1/2}), \boldsymbol{B}_2^{1/2}/(2z_2^{1/2}), \ldots, \boldsymbol{B}_g^{1/2}/(2z_g^{1/2})]$. Formulation (24) is a standard group LASSO formulation, which can handled by calling classical group LASSO solver (e.g., [63]) to determine $\boldsymbol{\alpha}$.

Note that during the above process, we fix the intragroup correlation $\boldsymbol{B}$ and the hyperparameter λ. In fact, the hyperparameter λ can be automatically determined by a group LASSO solver [63]. Besides, according to [60], since $\boldsymbol{\alpha}$ has the approximate covariance with $\boldsymbol{w}$, $\boldsymbol{B}$ can be approximately estimated by $\boldsymbol{\alpha}$ from the previous iteration, that is

$$\boldsymbol{B}_i = \frac{1}{\gamma_i}\boldsymbol{\Sigma}_{\boldsymbol{w}_i} \approx \frac{1}{\gamma_i}\mathbb{E}\big[(\boldsymbol{\alpha}_i - \mathbb{E}(\boldsymbol{\alpha}_i))(\boldsymbol{\alpha}_i - \mathbb{E}(\boldsymbol{\alpha}_i))^\top\big]. \tag{25}$$

In particular, according to [64], the first-order auto-regressive process corresponding to the Toeplitz matrix is more sufficient to capture intragroup correlation.

Therefore, the intragroup correlation matrix $\boldsymbol{B}_i$ is replaced by $\hat{\boldsymbol{B}}_i$ as follows:

$$\hat{\boldsymbol{B}}_i = \text{Toeplitz}\left(\left[1, r, \ldots, r^{\text{NCo}'-1}\right]\right) \tag{26}$$

where $r = \bar{m}_1/\bar{m}_0$ and $\bar{m}_0$ and $\bar{m}_1$ are the averages of elements along the main diagonal and the main subdiagonal of $\boldsymbol{B}_i$, respectively.

In summary, the developed group LASSO-type algorithm flow is shown in Algorithm 1. We do not directly optimize formulation (9). Instead, we find the upper bound of the function $\mathcal{L}(\boldsymbol{B}, \boldsymbol{\gamma}, \lambda)$ as shown in (14). Then, the upper bound function $\mathcal{UL}(\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{B}, \lambda)$ is minimized by two steps. In the first step, we alternatively optimize formulation (16) to obtain the hyperparameter $\boldsymbol{\gamma}$ in (21) and $\boldsymbol{z}$ in (22). In the second step, we transform formulation (17) as an equivalent group LASSO formulation as shown in formulation (24). After calling the group LASSO solver, we can determine $\boldsymbol{d}$ ($\boldsymbol{\alpha}$) and λ simultaneously. In addition, we use (25) and (26) to update the hyperparameters $\boldsymbol{B}$ and $\hat{\boldsymbol{B}}_i$. The above process is iteratively performed until convergence. Then, the intragroup correlation $\boldsymbol{B}$, the intergroup importance $\boldsymbol{\gamma}$, and the hyperparameter λ are determined.

In practice, it only takes few iterations in Algorithm 1 (2–5 iterations). In each iteration, any efficient group LASSO algorithm can be used, which bring much faster and suitable to the sharing parameters in the grouped convolution.

Algorithm 1 Group LASSO to Handle Formulation (9)

Require: $\boldsymbol{X}$, $\boldsymbol{y}$ from one grouped convolutional layer, network parameters $\boldsymbol{w}$.

1: Initialize $\boldsymbol{B}$, $\boldsymbol{\gamma}$, $\boldsymbol{z}$ and λ.
2: **repeat**
3: Update $\boldsymbol{\gamma}$ by Equation (21);
4: Update $\boldsymbol{z}$ by Equation (22);
5: Solve Formulation (24) (Formulation (23)) to obtain $\boldsymbol{d}$ ($\boldsymbol{\alpha}$) and λ;
6: Update $\boldsymbol{B}_i = \hat{\boldsymbol{B}}_i$ by Equations (25) and (26);
7: **until** Convergence
8: **return** hyper-parameters $\boldsymbol{B}$, $\boldsymbol{\gamma}$ and λ.

D. Overall Flow

In this section, we will give an overall flow about how to share parameters among different groups so that the vanilla grouped convolution can be transferred as the sharing structure.

After $\boldsymbol{B}$, $\boldsymbol{\gamma}$, and λ are determined by Algorithm 1, in each group, model parameters $\boldsymbol{w}_i$ can be determined by the posterior mean as shown in (6), that is, $\boldsymbol{w}_i = \boldsymbol{\mu}_i$. To share the parameters among different groups in one grouped convolutional layer, the mean of the sharing parameters $\boldsymbol{\mu}_{\boldsymbol{w}_b}$ is defined as a prior mean as follows:

$$\boldsymbol{\mu}_{\boldsymbol{w}_b} = \frac{\sum_i^g \gamma_i \boldsymbol{w}_i}{\sum_i^g \gamma_i}. \tag{27}$$

The mean is the weighted average of all network parameters obtained in the last iteration, with the intergroup importance γ_i. Then, in (10) and (14), the prior mean is $\boldsymbol{\mu}_{\boldsymbol{w}} = \mathbf{1}_g \otimes \boldsymbol{\mu}_{\boldsymbol{w}_b} = [\boldsymbol{\mu}_{\boldsymbol{w}_b}^\top, \boldsymbol{\mu}_{\boldsymbol{w}_b}^\top, \ldots, \boldsymbol{\mu}_{\boldsymbol{w}_b}^\top]^\top$, and $\mathbf{1}_g \in \mathbb{R}^g$ is a vector whose all elements are 1. $\otimes$ represents the Kronecker product. The sharing process is shown in Algorithm 2. As shown in Fig. 6, initially, all groups have different parameters. After few iterations, parameters will gradually become the same by our proposed Bayesian sharing framework. In particular, the mean sharing method is a special case of our proposed Bayesian sharing method, i.e., $\boldsymbol{\gamma} \equiv \mathbf{1}_g$.

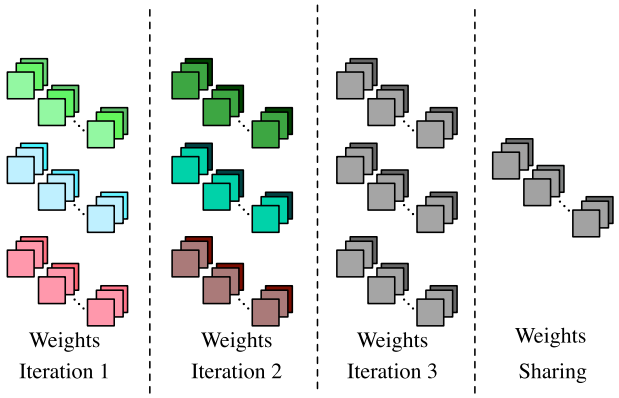


Fig. 6. Sharing process of grouped convolution parameters. Green, blue, and red boxes represent parameters (kernels) in three groups. After few iterations, all groups have the same kernels (gray boxes), which are shared.

Algorithm 2 Bayesian Sharing Framework

Require: $\boldsymbol{X}$, $\boldsymbol{y}$ from one grouped convolutional layer, network parameters $\boldsymbol{w}$.

1: Initialize $\boldsymbol{\mu}_{\boldsymbol{w}_b} = \sum_i^g \boldsymbol{w}_i/g$, $\boldsymbol{\mu}_{\boldsymbol{w}} = \mathbf{1}_g \otimes \boldsymbol{\mu}_{\boldsymbol{w}_b}$;
2: **repeat**
3: Update $\boldsymbol{B}$, $\boldsymbol{\gamma}$ and λ by Algorithm 1;
4: Update model parameters $\boldsymbol{w}_i$ by the posterior mean in Equation (6);
5: Update the sharing model parameters $\boldsymbol{\mu}_{\boldsymbol{w}_b}$ by Equation (27) and $\boldsymbol{\mu}_{\boldsymbol{w}} = \mathbf{1}_g \otimes \boldsymbol{\mu}_{\boldsymbol{w}_b}$;
6: **until** Convergence
7: **return** The sharing weights $\boldsymbol{w}_b = \boldsymbol{\mu}_{\boldsymbol{w}_b}$.

For the whole CNN model, we adopt a separate-merge methodology [23] to share weights in all grouped convolutional layers, that is, separately updating parameters by loss function in the backpropagation stage and computing loss function value in the forward propagation stage. Given a pretrained CNN model, we fix model parameters in non-grouped convolutional layers and update model parameters in all grouped convolutional layers by our proposed Bayesian sharing method as shown in Algorithm 2 from front layers to back layers sequentially in the forward propagation stage.

The loss value is calculated by all updated shared grouped convolution parameters and other fixed model parameters. Then, the loss value is used to update all model parameters. By performing this sharing process for few epochs, the final sharing model can be obtained.

Note that instead of all layers in CNN model, our proposed Bayesian sharing framework is performed in grouped convolutional layers to transfer to be the sharing structure.

It is worth mentioning that our proposed Bayesian sharing framework is not only compressing models but also regularizing parameters. The regularization technique can encourage learning a more simple model to avoid the risk of overfitting so that the accuracy can be improved [13]. The most common regularization techniques are ride regression and LASSO regression, which can force the model parameters to decay toward zeros so that model complexity is reduced and the model generalizes better [13]. Note that directly learning a sharing grouped convolution structure has the same manner with directly forcing the model parameters to zeros, which brings performance degradation. In the proposed Bayesian sharing framework, the model parameters are imposed to be the same among different groups by adaptively learning the intragroup correlation $\boldsymbol{B}$ and the intergroup importance $\boldsymbol{\gamma}$. Thus, the intuition behind this technique has the same manner as the common regularizations, which adaptively force the model parameters to decay toward zeros. As a result, we expect that the proposed Bayesian sharing framework can improve the accuracy of original models.

V. Experimental Results

In this section, we apply our Bayesian sharing framework on some popular grouped CNNs, including ResNeXt [10], ShuffleNet [16], and G-DenseNet [17], [65]. We test them on CIFAR-10, CIFAR-100 [14], and ImageNet [15]. As an ablation study, to clarify the impact of the proposed Bayesian sharing framework, the directly trained sharing grouped CNNs and the mean sharing method are also implemented for comparison. The direct training method constructs a network with the proposed sharing structure and then trains it. The mean sharing method trains the model from scratch and each group has its own weights. At some certain training epochs, e.g., 80 and 100 epochs, we average the weights and then continue the training process. In the experimental results, "$-D$" represents the results of directly trained sharing grouped CNNs, "$-M$" represents the results of mean sharing, and "$-B$" represents the Bayesian sharing.

A. Implementation Details and Experimental Settings

1) Training Settings: On CIFAR-10 and CIFAR-100, we test all of these three methods. The initial learning rate is set as 0.1. For ResNeXt and ShuffleNet, the batch size is 128 and the learning rate is gradually divided by 10 at 81 and 122 epochs, with 164 training epochs in total. For G-DenseNet, the batch size is 64, and the learning rate is divided by 10 at 150 and 225 epochs, with a total of 300 training epochs. CIFAR-10 and CIFAR-100 are shorted as C-10 and C-100 in the result tables.

We test the sharing ResNeXt On ImageNet. The learning rate is initially set to 0.1, divided by 10 at 50 and 70 epochs. There are 90 training epochs, and the batch size is 256.

Our optimizer uses momentum optimizer, with momentum 0.9 and weight decay 2×10^{-4}.

2) Evaluation Metrics: Parameter volume, model accuracy, and grouped convolution compression ratio (GCR) are considered as the evaluation metrics. Parameter volume, abbreviated as "#P", counts all the parameters in the model, including grouped convolutional layers and other linear or nonlinear layers. GCR is only for grouped convolutional layers, i.e., volume of the sharing layer divided by the original volume before sharing. The compression ratio of the baseline model is also 100%. For a grouped convolutional layer with g groups, after sharing, the compression ratio is $1/g$. Therefore, our compression ratio relies on the number of groups. For ImageNet, we report Top-1 and Top-5 accuracies. The number of floating-point operations (FLOPs) and runtime is also attached.

B. Experiments on CIFAR Dataset

Our sharing method is applied to some baseline models, i.e., ResNeXt, ShuffleNet, and G-DenseNet to test CIFAR-10 and CIFAR-100, with some necessary model modifications in Tables III and IV. For ResNeXt-35 and RexNeXt-50, to test the cardinality, some tests are conducted on grouped convolutional layers with 4, 8, and 16 groups, while the kernel size is 3×3. The pointwise convolutional layers are not considered here since they are not in the grouped convolutional layers of these two models. For ShuffleNet, grouped convolutional layers with four and eight groups are tested. Different from ResNeXt, the pointwise (1×1) convolutions in ShuffleNet are grouped convolutional layers. Some experiments are conducted on ShuffleNet with 1×1 convolutions to further demonstrate the effectiveness of our sharing method. DenseNet contains both 3×3 and 1×1 convolutional layers, which are both tested to further validate the compatibility of our method.

As ablation studies, to clarify the impacts of our proposed Bayesian sharing framework, we compare the directly trained model with sharing grouped convolution, the mean sharing, and the proposed Bayesian sharing. The results are shown in Tables III and IV. For all of these tests, compared with the corresponding baseline models, the performance degradations occur in all directly trained models. The mean sharing method can achieve slight accuracy improvements in the most cases but G-DenseNet-86 since it is able to combine parameters among different groups without discrimination, i.e., $\boldsymbol{\gamma} \equiv \mathbf{1}_g$ in (27). Compared with the mean sharing method, our Bayesian sharing framework can bring significant accuracy improvements, mostly more than 2%, since it considers the intragroup correlation and the intergroup importance to combine parameters among different groups with discriminations. In other words, it is able to discriminately combine parameters to achieve message passing to different features according to the importantly learned from maximum likelihood estimation in (9). Some tests achieve higher improvements, e.g., in Table III, ResNeXt-50-B with eight groups on CIFAR-100 improves the accuracy by $76.11\% - 73.16\% = 2.95\%$ with the less parameter volume. As a result, the proposed Bayesian sharing framework can improve the parameter efficiency, reduce the parameter redundancy, and alleviate the overfitting issue.

Our Bayesian sharing method can result in impressive compression and runtime performance. Since for grouped con-

TABLE III
RESNEXT ON THE CIFAR DATASET

Model	g	RexNeXt-35 #P (M)	RexNeXt-35 Acc. (%) C-10	RexNeXt-35 Acc. (%) C-100	RexNeXt-35 FLOPs (M)	RexNeXt-35 Time (ms)	RexNeXt-50 #P (M)	RexNeXt-50 Acc. (%) C-10	RexNeXt-50 Acc. (%) C-100	RexNeXt-50 FLOPs (M)	RexNeXt-50 Time (ms)	GCR (%)
ResNeXt (baseline)	4	1.29	92.87	72.91	202	33.4	2.01	93.67	73.08	279	42.8	100.00
ResNeXt-D	4	1.19 (↓)	92.02 (↓)	72.17 (↓)	202	26.2	1.58 (↓)	92.89 (↓)	72.86 (↓)	279	33.3	25.00
ResNeXt-M	4	1.19 (↓)	93.31 (↑)	73.44 (↑)	202	26.2	1.58 (↓)	93.90 (↑)	73.55 (↑)	279	33.3	25.00
ResNeXt-B	4	**1.19** (↓)	**94.15** (↑)	**74.56** (↑)	**202**	**26.2**	**1.58** (↓)	**94.93** (↑)	**75.46** (↑)	**279**	**33.3**	**25.00**
ResNeXt (baseline)	8	1.31	93.00	73.07	214	43.3	2.04	93.22	73.16	291	47.3	100.00
ResNeXt-D	8	1.18 (↓)	92.32 (↓)	72.51 (↓)	214	38.5	1.70 (↓)	93.14 (↓)	72.71 (↓)	291	42.3	25.00
ResNeXt-M	8	1.18 (↓)	93.42 (↑)	73.62 (↑)	214	38.5	1.70 (↓)	93.78 (↑)	73.93 (↑)	291	42.3	12.50
ResNeXt-B	8	**1.18** (↓)	**94.08** (↑)	**74.97** (↑)	**214**	**38.5**	**1.70** (↓)	**95.04** (↑)	**76.11** (↑)	**291**	**42.3**	**12.50**
ResNeXt (baseline)	16	1.35	92.93 (↑)	73.23 (↑)	222	48.7	2.12	93.23 (↑)	73.31 (↑)	309	55.0	100.00
ResNeXt-D	16	1.26 (↓)	92.48 (↓)	72.57 (↓)	222	43.8	1.88 (↓)	93.22 (↓)	72.93 (↓)	309	52.6	6.25
ResNeXt-M	16	1.26 (↓)	93.38 (↑)	73.64 (↑)	222	43.8	1.88 (↓)	93.78 (↑)	73.79 (↑)	309	52.6	6.25
ResNeXt-B	16	**1.26** (↓)	**94.77** (↑)	**75.88** (↑)	**222**	**43.8**	**1.88** (↓)	**95.17** (↑)	**75.87** (↑)	**309**	**52.6**	**6.25**

TABLE IV
SHUFFLENET AND G-DENSENET ON THE CIFAR DATASET

Model	g	#P (M)	Acc. (%) C-10	Acc. (%) C-100	FLOPs (M)	Time (ms)	GCR (%)
ShuffleNet-1x (baseline)	4	0.62	91.65	71.48	106	23.0	100.00
ShuffleNet-1x-D	4	0.28 (↓)	90.78 (↓)	70.56 (↓)	106	17.6	25.00
ShuffleNet-1x-M	4	0.28 (↓)	92.47 (↑)	72.29 (↑)	106	17.6	25.00
ShuffleNet-1x-B	4	**0.28** (↓)	**93.56** (↑)	**73.83** (↑)	**106**	**17.6**	**25.00**
ShuffleNet-2x (baseline)	4	1.34	91.48	71.65	123	29.8	100.00
ShuffleNet-2x-D	4	0.48 (↓)	90.32 (↓)	70.49 (↓)	123	22.9	25.00
ShuffleNet-2x-M	4	0.48 (↓)	92.68 (↑)	72.07 (↑)	123	22.9	25.00
ShuffleNet-2x-B	4	**0.48** (↓)	**93.79** (↑)	**73.89** (↑)	**123**	**22.9**	**25.00**
ShuffleNet-1x (baseline)	8	1.35	92.29	72.12	204	33.4	100.00
ShuffleNet-1x-D	8	0.60 (↓)	91.19 (↓)	71.57 (↓)	204	28.1	12.50
ShuffleNet-1x-M	8	0.60 (↓)	93.16 (↑)	72.26 (↑)	204	28.1	12.50
ShuffleNet-1x-B	8	**0.60** (↓)	**94.00** (↑)	**72.98** (↑)	**204**	**28.1**	**12.50**
G-DenseNet-86 (baseline)	4	0.62	93.21	73.89	102	69	100.00
G-DenseNet-86-D	4	0.33 (↓)	92.89 (↓)	73.14 (↓)	102	53	25.00
G-DenseNet-86-M	4	0.33 (↓)	93.78 (↑)	73.76 (↓)	102	53	25.00
G-DenseNet-86-B	4	**0.33** (↓)	**94.91** (↑)	**75.12** (↑)	**102**	**53**	**25.00**

volutional layers with g groups, the GCR is $1/g$, more groups mean a better compression ratio. According to Tables III and IV, as the group number increases, our method achieves higher compression ratios. Convolutional layers with four groups have the minimal GCR, i.e., compressed to 0.25 times. Dividing to 16 groups can bring the maximal compression ratio, i.e., 0.0625 times. Except for grouped convolutional layers, a typical neural network contains many other linear or nonlinear layers. The models with more grouped convolutional layers have better compression performance for parameter volume by using our Bayesian sharing method. In Table III, for ResNeXt models with the limited number of 3×3 convolutional layers, we can achieve up to 21% $[(2.01-1.58)/2.01]$ overall volume reduction. In Table IV, for the sharing G-DenseNet-86, the parameter volume is reduced by 46.77% $[(0.62-0.33)/0.62]$. The sharing ShuffleNet-1x reduces the parameter volume by 54.8% $[(0.62-0.28)/0.62]$, and the parameter volume in ShuffleNet-2x reduces more than 64.17% $[(1.34-0.48)/1.34]$. The proposed sharing method can achieve the more significant parameter reductions for CNN with the more grouped convolutional layers. Generally, the deeper and larger models suffer from higher risks of overfitting. With our Bayesian sharing framework, we can alleviate this problem by reducing parameter volume.

In particular, compared with these baseline methods, our proposed sharing grouped convolution does not reduce FLOPs in the inference stage. However, as shown in Fig. 3, the parameters are shared among different groups. The sharing parameter strategy can reduce the actual number of memory accesses so that the inference time can be reduced, as shown in Tables III and IV. The runtime results are tested on one Kaggle Nvida Tesla P100 (16-GB memory and 720-GB/s bandwidth). It is believed that we can achieve better run performances on field-programmable gate array (FPGA) with dataflow optimizations [19].

We also compare our method with the state-of-the-art methods on ResNet, MobileNet, and DenseNet on CIFAR dataset, as shown in Tables V and VI. These methods include efficient

TABLE V

OUR PROPOSED SHARING RESNEXT-50 AND SHUFFLENET, IN COMPARISON WITH THE STATE-OF-THE-ART MODELS ON CIFAR

Model	Method	#P (M)	Acc. (%)	FLOPs (M)	Dataset
ResNet-50	CP [24]	6.44	94.15	1620	C-10
	CaP [35]	1.60	92.67	-	
	Genetic [45]	7.71	93.60	954	
	Gaussian [45]	5.94	94.00	735	
ResNeXt-50	-B (g=4)	**1.58**	94.93	**279**	
	-B (g=8)	1.70	95.04	291	
	-B (g=16)	1.88	**95.17**	309	
ResNet-50	FP [26]	7.83	73.60	616	C-100
	CP [24]	9.24	74.10	1740	
	PCAS [30]	4.02	73.84	475	
ResNeXt-50	-B (g=4)	**1.58**	75.46	**279**	
	-B (g=8)	1.70	**76.11**	291	
	-B (g=16)	1.88	75.87	309	
MobileNet-v2	α-1 [18]	2.30	90.20	89	C-10
	α-0.75 [18]	1.73	87.80	**60**	
	FLGC(g=2) [12]	1.18	94.11	158	
	FLGC(g=3) [12]	0.85	**94.20**	122	
	FLGC(g=4) [12]	0.68	94.16	103	
	FLGC(g=8) [12]	0.43	93.09	76	
ShuffleNet	1x-B (g=4)	**0.28**	93.56	106	
	2x-B (g=4)	0.48	93.79	123	
	1x-B (g=8)	0.60	94.00	204	

model architecture methods [18], [32], [36], [45], [46] and compression methods at various levels, such as filter pruning [26] and channel pruning [12], [24], [25], [30], [35], [53]. It is worth mentioning that the group pruning [12], [53] is a special channel pruning since the input channels, output channels, and their connections are divided into several groups.

According to Tables V and VI, our method outperforms all of the current efficient model architecture methods [18], [32], [36], [45], [46] since they do not consider correlations among parameters in the training stage so that the model performance is significantly degraded. In particular, it is hard to make a better tradeoff between accuracy and parameter volume, even though some advanced neural architecture search methods are adopted to determine model configurations [45]. Unlike these traditional compression methods [12], [24]–[26], [30], [35], [53], our proposed Bayesian sharing framework does not completely prune channels or filters (kernels). Instead, compared with the grouped convolution, we reserve all filters (kernels) and channels and discriminately combine parameters to achieve message passing to features with different importances. The reused weights, structures of model parameters, and the adaptive importance learning strategy can reduce the parameter redundancy, improve efficiency, and alleviate the overfitting issue. In particular, compared with the state-of-the-art group pruning methods that prune several unimportant groups [12], [53], our method reserves all groups, thus striking a better balance between accuracy and parameter volume.

C. Experiments on ImageNet

To further evaluate the impact of our Bayesian sharing framework for model performance on large datasets, we test the sharing ResNeXt-50 on the ImageNet dataset. We follow the configurations in [10]. The number of groups is 32.

TABLE VI

OUR SHARING G-DENSENET, IN COMPARISON WITH DENSENET ON CIFAR

Model	#P (M)	Acc. (%) C-10	Acc. (%) C-100	FLOPs (M)
DenseNet-40-pruned [25]	0.66	94.81	74.72	190
DenseDsc(k=36) [46]	0.47	94.05	74.24	123
DVN-77 [32]	0.40	93.09	72.60	-
DenseNet-100-D [36]	0.70	93.12	72.39	269
DenseNet-86 (g=4) [53]	0.59	94.06	74.04	132.7
G-DenseNet-86-B (g=4)	**0.33**	**94.91**	**75.12**	**102**

TABLE VII

OUR PROPOSED SHARING RESNEXT-50 IN COMPARISON WITH THE STATE-OF-THE-ART MODELS ON THE IMAGENET DATASET

Model	#P (M)	Acc. (%) Top-1	Acc. (%) Top-5	FLOPs (M)
ResNet-50-BN [40]	25.56	77.60	93.70	4151
ResNet-50-GN [51]	25.56	76.00	92.80	4155
ResNet-50-SN [41]	25.56	76.90	93.20	4225
ResNet-50-SSN [42]	25.56	77.20	93.10	4186
ResNet-50-EN [43]	25.91	78.10	93.60	4325
G-ResNeXt-50 [11]	25.00	78.40	94.00	**4090**
ResNeXt-50-B (g=32)	**23.63**	**78.86**	**94.54**	4200

We compare our method with the state-of-the-art efficient model architecture methods [11], [40]–[43], [51] to examine model accuracy, parameter volume, and FLOPs in Table VII. It is shown that our proposed Bayesian sharing framework beats these state-of-the-art efficient model architecture methods in terms of the model accuracy and the parameter volume. In particular, some advanced normalization layers are developed to enhance the generalization ability of the model, but they make the model cumbersome while bringing more FLOPs [40]–[43], [51]. Besides, compared with the dynamic grouping convolution [11], the sharing grouped convolution can reduce parameter volume since model parameters are shared among different groups. For the sharing ResNeXt-50-B on ImageNet, we can reduce the parameters in group convolutional layers by 96.875%, i.e., $100\% - 3.125\% = 96.875\%$. The reuse model parameters structure with the adaptive importance learning strategy can improve model accuracy to 78.86% and 94.54% for top-1 and top-5, respectively.

D. Model Compatibility

The experimental results also verify that our method has strong compatibility while tackling various grouped convolutional models, from 3×3 convolutions to 1×1 pointwise convolutions in ResNeXt, ShuffleNet and G-DenseNet, for different numbers of groups (different cardinalities), on different datasets, as shown in Tables III and IV. Overall, our experiments listed above have covered all of these diverse grouped convolution structures.

E. Intergroup Importance

Intergroup importance is proposed in our framework. γ_i reflects the intergroup importance of group i. In our model, the stable values of all the γ_i values are mutually distinct. Fig. 7 is taken as an example to illustrate this. For a layer in ResNeXt-50 with eight groups, each γ_i is initialized as

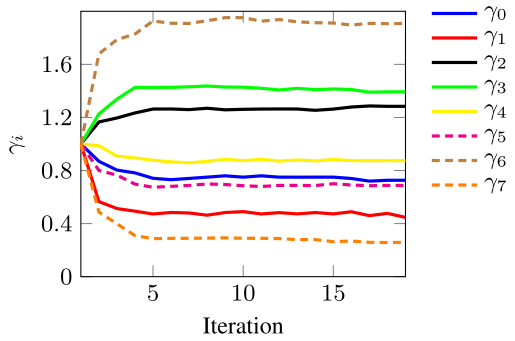


Fig. 7. Optimization iteration versus $\boldsymbol{\gamma}$ values of ResNeXt-50 ($g = 8$).

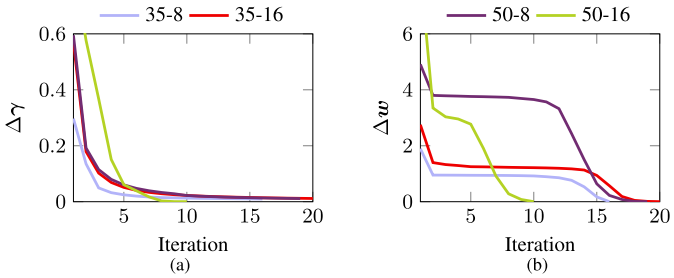


Fig. 8. Optimization iteration versus $\Delta\boldsymbol{\gamma}$ and $\Delta\boldsymbol{w}$. Four models are listed as follows: (a) ResNeXt-35 ($g = 8$ and $g = 16$) and (b) ResNeXt-50 ($g = 8$ and $g = 16$).

1 before training. In other words, in the beginning, these groups are equally important. After about five optimization iterations, these γ values reach different stable statuses. The group with higher γ_i is more important. If we use mean sharing, γ_i can always be regarded as 1, i.e., $\boldsymbol{\gamma} \equiv \mathbf{1}_g$ in (27). In comparison, our Bayesian sharing framework can better characterize the differences of intergroup importance reasonably and effectively.

F. Convergence

To verify the optimization and convergence of our method more clearly, in ResNeXt-35 ($g = 8$ and $g = 16$) and ResNeXt-50 ($g = 8$ and $g = 16$), one layer is sampled from each model and its $\Delta\boldsymbol{\gamma}$ and $\Delta\boldsymbol{w}$ are shown in Fig. 8(a) and (b), respectively. Each layer has several groups, and $\Delta\boldsymbol{\gamma}$ is computed according to the following equation:

$$\Delta\boldsymbol{\gamma} = \sum_{i=1}^{g} \left| \frac{\gamma_i^{(t+1)} - \gamma_i^{(t)}}{\gamma_i^{(t+1)}} \right|. \tag{28}$$

$\Delta\boldsymbol{w}$ is for $\boldsymbol{w}$ and is the summation of all the element-wise changes between two continuous training steps. Obviously, by using Algorithm 2, parameter $\boldsymbol{w}$ converges quickly, and γ for each group also reaches a stable status quickly. Another trend is that the models with more parameters (e.g., ResNeXt-50 with $g = 16$) can have a faster convergence rate.

VI. Conclusion

In this article, we propose a sharing grouped convolution structure with the Bayesian sharing framework to efficiently eliminate parameter redundancy and boost the model performance. Intragroup correlation and intergroup importance are introduced into the prior of the parameters. We handle the Maximum Type II likelihood estimation problem of the intragroup correlation and intergroup importance by a group LASSO-type algorithm. Experiments demonstrate that the proposed sharing grouped convolution structure with the Bayesian sharing framework can reduce parameters and improve prediction accuracy. The proposed sharing framework can reduce parameters up to 64.17%. For ResNeXt-50 with the sharing grouped convolution on the ImageNet dataset, network parameters can be reduced by 96.875% in grouped convolutional layers, and accuracies are improved to 78.86% and 94.54% for top-1 and top-5, respectively.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[2] J. Liu, B. Ni, Y. Yan, P. Zhou, S. Cheng, and J. Hu, "Pose transferrable person re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4099–4108.
[3] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[4] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 21–37.
[5] H. Geng *et al.*, "Hotspot detection via attention-based deep layout metric learning," in *Proc. 39th Int. Conf. Comput.-Aided Design*, Nov. 2020, pp. 1–8.
[6] Y. Yan, B. Ni, and X. Yang, "Predicting human interaction via relative attention model," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3245–3251.
[7] Q. Sun, T. Chen, J. Miao, and B. Yu, "Energy-driven DNN dataflow optimization on FPGA," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, pp. 1–7.
[8] Y. Yang *et al.*, "Synetgy: Algorithm-hardware co-design for ConvNet accelerators on embedded FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, Feb. 2019, pp. 23–32.
[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.
[10] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1492–1500.
[11] Z. Zhang *et al.*, "Differentiable learning-to-group channels via groupable convolutional neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3542–3551.
[12] X. Wang, M. Kan, S. Shan, and X. Chen, "Fully learnable group convolution for acceleration of deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9049–9058.
[13] C. Robert, *Machine Learning: A Probabilistic Perspective*. London, U.K.: Taylor & Francis, 2014.
[14] *The CIFAR-10 and CIFAR-100 Datasets*. Accessed: Sep. 2019. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html
[15] *ImageNet Dataset*. Accessed: Sep. 2019. [Online]. Available: http://www.image-net.org.
[16] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.
[17] G. Huang, S. Liu, L. V. D. Maaten, and K. Q. Weinberger, "CondenseNet: An efficient DenseNet using learned group convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2752–2761.
[18] S. Kundu, M. Nazemi, M. Pedram, K. M. Chugg, and P. A. Beerel, "Pre-defined sparsity for low-complexity convolutional neural networks," *IEEE Trans. Comput.*, vol. 69, no. 7, pp. 1045–1058, Jul. 2020.
[19] X. Wei, Y. Liang, and J. Cong, "Overcoming data transfer bottlenecks in FPGA-based DNN accelerators via layer conscious memory management," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.
[20] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, Jan. 2019.

[21] T. Zhang *et al.*, "A systematic DNN weight pruning framework using alternating direction method of multipliers," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 184–199.
[22] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 2074–2082.
[23] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1389–1397.
[24] Y. Hu, S. Sun, J. Li, X. Wang, and Q. Gu, "A novel channel pruning method for deep neural network compression," 2018, *arXiv:1805.11394*. [Online]. Available: http://arxiv.org/abs/1805.11394
[25] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2736–2744.
[26] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–13.
[27] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 1135–1143.
[28] H. Wang, Q. Zhang, Y. Wang, L. Yu, and H. Hu, "Structured pruning for efficient ConvNets via incremental regularization," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.
[29] H. Wang, X. Hu, Q. Zhang, Y. Wang, L. Yu, and H. Hu, "Structured pruning for efficient convolutional neural networks via incremental regularization," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 775–788, May 2020.
[30] K. Yamamoto and K. Maeno, "PCAS: Pruning channels with attention statistics for deep network compression," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2019, pp. 1–13.
[31] Y. Ma *et al.*, "A unified approximation framework for compressing and accelerating deep neural networks," in *Proc. IEEE 31st Int. Conf. Tools with Artif. Intell. (ICTAI)*, Nov. 2019, pp. 376–383.
[32] J. Ou and Y. Li, "Vector-kernel convolutional neural networks," *Neurocomputing*, vol. 330, pp. 253–258, Feb. 2019.
[33] P. Chen, S. Si, Y. Li, C. Chelba, and C.-J. Hsieh, "GroupReduce: Block-wise low-rank approximation for neural language model shrinking," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 10988–10998.
[34] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 742–751.
[35] B. Minnehan and A. Savakis, "Cascaded projection: End-to-end network compression and acceleration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10715–10724.
[36] G. Li, X. Shen, J. Li, and J. Wang, "Diagonal-kernel convolutional neural networks for image classification," *Digit. Signal Process.*, vol. 108, Jan. 2021, Art. no. 102898.
[37] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2016, pp. 1317–1327.
[38] S. Wiedemann *et al.*, "DeepCABAC: Context-adaptive binary arithmetic coding for deep neural network compression," in *Proc. ICML Workshop*, 2019, pp. 1–4.
[39] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016, pp. 1–14.
[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn. (ICML)*, PMLR, 2015, pp. 448–456.
[41] P. Luo, J. Ren, Z. Peng, R. Zhang, and J. Li, "Differentiable learning-to-normalize via switchable normalization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–19.
[42] W. Shao *et al.*, "SSN: Learning sparse switchable normalization via sparsestmax," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 443–451.
[43] R. Zhang, Z. Peng, L. Wu, Z. Li, and P. Luo, "Exemplar normalization for learning deep representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12726–12735.
[44] W. Shao, S. Tang, X. Pan, P. Tan, X. Wang, and P. Luo, "Channel equilibrium networks for learning deep representation," in *Proc. Int. Conf. Mach. Learn. (ICML)*, PMLR, 2020, pp. 8645–8654.
[45] M. Javaheripi, M. Samragh, T. Javidi, and F. Koushanfar, "AdaNS: Adaptive non-uniform sampling for automated design of compact DNNs," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 750–764, May 2020.
[46] G. Li, M. Zhang, J. Li, F. Lv, and G. Tong, "Efficient densely connected convolutional neural networks," *Pattern Recognit.*, vol. 109, Jan. 2021, Art. no. 107610.
[47] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[48] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861
[49] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
[50] T. Zhang, G.-J. Qi, B. Xiao, and J. Wang, "Interleaved group convolutions," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 4373–4382.
[51] Y. Wu and K. He, "Group normalization," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.
[52] Z. Su, L. Fang, W. Kang, D. Hu, M. Pietikäinen, and L. Liu, "Dynamic group convolution for accelerating convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2020, pp. 138–155.
[53] R. Zhao and W. Luk, "Efficient structured pruning and architecture searching for group convolution," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1961–1970.
[54] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 3288–3298.
[55] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 2575–2583.
[56] J. Wang, H. Bai, J. Wu, and J. Cheng, "Bayesian automatic model compression," *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 727–736, May 2020.
[57] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114
[58] T. Chen, B. Lin, H. Geng, S. Hu, and B. Yu, "Leveraging spatial correlation for sensor drift calibration in smart building," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Aug. 12, 2020, doi: 10.1109/TCAD.2020.3015438.
[59] T. Chen, B. Lin, H. Geng, and B. Yu, "Sensor drift calibration via spatial correlation model in smart building," in *Proc. 56th Annu. Design Autom. Conf.*, Jun. 2019, pp. 1–6.
[60] Z. Zhang and B. D. Rao, "Extension of SBL algorithms for the recovery of block sparse signals with intra-block correlation," *IEEE Trans. Signal Process.*, vol. 61, no. 8, pp. 2009–2015, Apr. 2013.
[61] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2007.
[62] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[63] *Group LASSO Solver*. Accessed: Sep. 2019. [Online]. Available: https://github.com/fabianp/group_lasso
[64] J. Huang, T. Zhang, and D. Metaxas, "Learning with structured sparsity," *J. Mach. Learn. Res.*, vol. 12, pp. 3371–3412, Jan. 2011.
[65] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

Tinghuan Chen (Graduate Student Member, IEEE) received the B.Eng. and M.Eng. degrees in electronics engineering from Southeast University, Nanjing, China, in 2014 and 2017, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His research interests include machine learning in analog/mixed-signal very large-scale integration (VLSI) design-for-reliability and cyber-physical systems.

Bin Duan received the B.Eng. degree in electronics engineering from Changzhou University, Changzhou, China, in 2018. He is currently pursuing the master's degree with the School of Microelectronics, Southeast University, Wuxi, China.

His research interests include neural network model compression and optimization.

Qi Sun (Graduate Student Member, IEEE) received the B.Eng. degree in computer science from Xidian University, Xi'an, China, in 2018. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His current research interests include deep neural network hardware acceleration, high-level synthesis, and design space exploration.

Meng Zhang received the B.S. degree in electrical engineering from the China University of Mining and Technology, Xuzhou, China, in 1986, and the M.S. degree in bioelectronics and the Ph.D. degree in microelectronic engineering, as an on-the-job postgraduate student, from Southeast University, Nanjing, China, in 1993 and 2014, respectively.

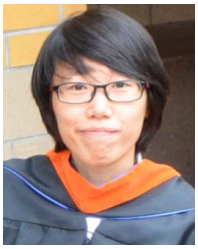
He is currently a Professor and a Faculty Adviser of Ph.D. graduates at the National ASIC System Research Center, School of Electronic Science and Engineering, Southeast University. He has published more than 40 refereed journal articles and international conference papers. He holds more than 90 patents, including some PCT and U.S. patents. His research interests include deep learning, machine learning, digital signal processing, digital communication systems, and digital integrated circuit design.

Guoqing Li (Graduate Student Member, IEEE) received the B.S. degree from Qingdao University, Qingdao, China, in 2014, and the M.S. degree from South China Normal University, Guangzhou, China, in 2017. He is currently pursuing the Ph.D. degree with the National ASIC Engineering Technology Research Center, School of Electronics Science and Engineering, Southeast University, Nanjing, China.

His current research interests include computer vision, convolutional neural networks, and deep learning hardware accelerator.

Hao Geng received the M.E. degree from the Department of Electronic Engineering and Information Sciences, University of Science and Technology of China, Hefei, China, in 2015, and the M.Sc. degree in machine learning from the Department of Computing, Imperial College London, London, U.K., in 2016. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His research interests include machine learning, deep learning, and optimization methods with applications in very large-scale integration (VLSI) computer-aided design (CAD).



Qianru Zhang received the M.S. degree in electrical and computer engineering from the University of California at Irvine, Irvine, CA, USA, in 2016. She is currently pursuing the Ph.D. degree with the National ASIC Center, School of Electronic Science and Engineering, Southeast University, Nanjing, China.

Her research interests include digital signal processing, big data analysis, and deep learning techniques.

Bei Yu (Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu has served as the TPC Chair for the ACM/IEEE Workshop on Machine Learning for Computer-Aided Design (CAD) and in many journal editorial boards and conference committees. He is also an Editor of the IEEE Technical Committee on Cyber-Physical Systems Newsletter. He received seven Best Paper Awards from the Asia and South Pacific Design Automation Conference (ASPDAC) 2021, the International Conference on Tools with Artificial Intelligence (ICTAI) 2019, Integration, the IEEE TRANSACTIONS ON VERY LARGE-SCALE INTEGRATION (VLSI) SYSTEMS journal in 2018, ISPD 2017, the SPIE Advanced Lithography Conference 2016, ICCAD 2013, ASPDAC 2012, and six ICCAD/ISPD Contest Awards.