OICSR: Out-In-Channel Sparsity Regularization for Compact Deep Neural Networks

Jiashi Li[1,2,*], Qi Qi[1,2,*], Jingyu Wang[1,2,†], Ce Ge[1,2], Yujian Li[1,2], Zhangzhang Yue[1], and Haifeng Sun[1,2]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, P.R. China
[2]EBUPT Information Technology Co., Ltd., Beijing 100191, P.R. China

Abstract

Channel pruning can significantly accelerate and compress deep neural networks. Many channel pruning works utilize structured sparsity regularization to zero out all the weights in some channels and automatically obtain structure-sparse network in training stage. However, these methods apply structured sparsity regularization on each layer separately where the correlations between consecutive layers are omitted. In this paper, we first combine one out-channel in current layer and the corresponding in-channel in next layer as a regularization group, namely out-in-channel. Our proposed Out-In-Channel Sparsity Regularization (OICSR) considers correlations between successive layers to further retain predictive power of the compact network. Training with OICSR thoroughly transfers discriminative features into a fraction of out-in-channels. Correspondingly, OICSR measures channel importance based on statistics computed from two consecutive layers, not individual layer. Finally, a global greedy pruning algorithm is designed to remove redundant out-in-channels in an iterative way. Our method is comprehensively evaluated with various CNN architectures including CifarNet, AlexNet, ResNet, DenseNet and PreActSeNet on CIFAR-10, CIFAR-100 and ImageNet-1K datasets. Notably, on ImageNet-1K, we reduce 37.2% FLOPs on ResNet-50 while outperforming the original model by 0.22% top-1 accuracy.

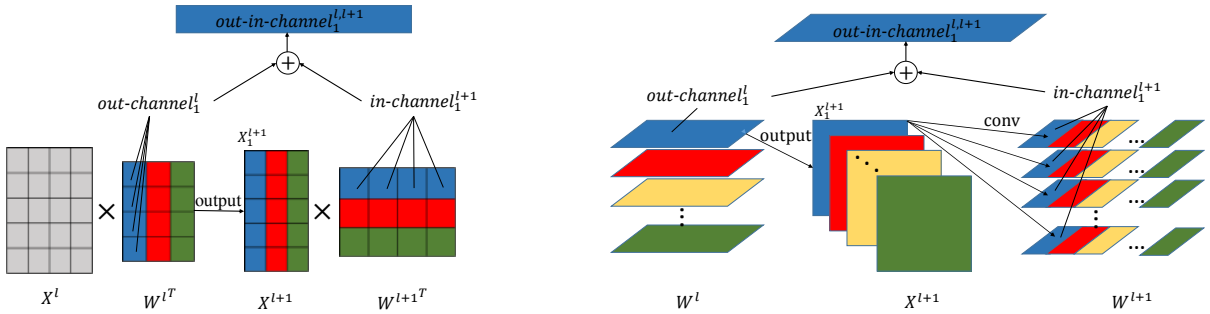
1. Introduction

Convolutional neural networks (CNNs) have achieved significant successes in visual tasks, including image classification [10, 21, 33], object detection [5, 31], semantic segmentation [2, 25], *etc*. However, large CNNs suffer from massive computational and storage overhead. For instance, deep residual network ResNet-50 [10] takes up about 190MB storage space, and needs more than 4 billion float point operations (FLOPs) to classify a single image. High demand for computation and storage resources severely hinders the deployment of large-scale CNNs in resource constrained devices such as mobile devices, wearable devices and Internet of Things (IoT) equipment.

Pruning [9, 34] is an important family of methods to slim neural network by removing redundant connections, channels and layers. Connection pruning gains high compression ratio but leads to non-structured sparsity of CNNs [34]. The practical acceleration of non-structured sparsity is limited due to irregular memory access. Therefore, structured sparsity pruning [34, 36] becomes growing popular.

Regularization-based channel pruning [24, 34] is a popular direction of structured sparsity pruning. These works introduce structured sparsity regularization (structured regularization) into optimization objective of model training. Training with structured regularization transfers important features into a small quantity of channels and automatically obtains structure-sparse model. Pruning structure-sparse models keeps more features/accuracy compared with directly pruning non-sparse models [34]. For channel-level pruning, existing regularization-based works apply structured regularization on each layer separately, and only enforce channel-level sparsity in out-channels. However, the corresponding in-channels in next layer are neglected and non-sparse. We call them as the separated structured regularization. Pruning one out-channel in current layer results in a dummy zero output feature map that in turn prunes a corresponding in-channel in next layer together. Without structure sparsity, useful features in in-channels of next layer are falsely discarded, which severely impair the representational capacity of the network.

* Authors contributed equally
† Corresponding author: wangjingyu@bupt.edu.cn
‡ Please contact: lijiashi@bupt.edu.cn



(a) Consecutive fully-connected layers

(b) Consecutive convolutional layers

Figure 1: Correlations between two consecutive layers and the definition of out-in-channel. $out\text{-}channel_i^l$ is the weight vector of i^{th} out-channel of W^l. X^l is the input of l^{th} layer and X_i^{l+1} is the output of X^l multiplied/convoluted by $out\text{-}channel_i^l$. Next, X_i^{l+1} is multiplied/convoluted by $in\text{-}channel_i^{l+1}$ to obtain the input of next layer. The corresponding channels (marked by the same color) $out\text{-}channel_i^l$ and $in\text{-}channel_i^{l+1}$ of two consecutive layers tend to work cooperatively and are simultaneously pruned/saved. Therefore, $out\text{-}channel_i^l$ and $in\text{-}channel_i^{l+1}$ are regarded as one regularization group ($out\text{-}in\text{-}channel_i^{l,l+1}$) and are regularized together. Above '$\oplus$' denotes concatenation of $out\text{-}channel_i^l$ and $in\text{-}channel_i^{l+1}$.

In this paper, we propose a novel structured regularization form, namely Out-In-Channel Sparsity Regularization (OICSR), to learn more compact deep neural networks. Different from separated structured regularization, correlations between two consecutive layers are taken into account for channel pruning. An out-channel in current layer and the corresponding in-channel in next layer are combined to be a regularization group, namely out-in-channel. In training stage, features in one out-in-channel are simultaneously redistributed by OICSR. After training, features in redundant out-in-channels are thoroughly transferred to automatically selected important out-in-channels. Specially, the channel importance is measured based on statistics of two consecutive layers, not individual layer. To minimize accuracy loss induced by incorrect channel pruning, a greedy algorithm is proposed to globally prune redundant out-in-channels in an iterative way. As a result, pruning redundant out-in-channels induces negligible accuracy loss and it can be greatly compensated by the fine-tuning procedure. Our method achieves higher speedup ratio and compression compared with existing regularization-based methods. For ResNet-18 [10] on CIFAR-10 [20] dataset, OICSR achieves 7.4× speedup and 11× parameters compression with tiny (0.19%) top-1 accuracy drop.

The key advantages and major contributions of this paper can be summarized as follows:

- We propose a novel structured regularization form, namely OICSR, which takes account correlations between two consecutive layers to further retain predictive power of the compact network.
- To minimize accuracy loss induced by incorrect channel pruning, OICSR measures channel importance based on statistics computed from two consecutive layers. A global greedy pruning algorithm is proposed to remove out-in-channels in an iterative way.
- To the best of our knowledge, this paper is the first attempt to present the evaluation of regularization-based channel pruning methods for very deep neural networks (ResNet-50 in this paper) on ImageNet [6] dataset.

2. Related Work

Obtaining compact deep neural networks for speeding up inference and reducing storage overhead has been a long-studied project in both academia and industry.

Recently, much attention has been focused on structured sparsity pruning to reduce network complexity. He *et al*. [13] pruned channels by a LASSO regression based channel selection and least square reconstruction. Channel pruning was regarded as an optimization problem by Luo *et al*. [26] and redundant channels were pruned by statistics of its next layer. Yu *et al*. [36] conducted feature ranking to obtain neuron/channel importance score and propagated it throughout the network. The neurons/channels with smaller importance scores were removed with negligible accuracy loss. Chin *et al*. [3] considered channel pruning as a global ranking problem and compensated the layer-wise approximation error that improved the performance for various heuristic metrics. To reduce accuracy loss caused by incorrect channel pruning, redundant channels were pruned in a dynamic way in [11, 23]. Furthermore, Huang *et al*. [17] and Huang & Wang [18] trained pruning agents and removed redundant structure in a data-driven way. These methods directly pruned insignificant channels on non-structured sparse models, which may falsely abandon useful features and induce obvious accuracy decline.

More recent developments adopted structured regularization to learn structured sparsity in training stage. Zhang *et al*. [39] incorporated sparse constraints into objective function to decimate the number of channels in CNNs. Similarly, Wen *et al*. [34] utilized Group Lasso to automatically obtain channel, filter shape and layer level sparsity in CNNs during network training. In [35], group and exclusive sparsity regularization are combined to exploit both positive and negative correlations among features, while enforcing the network to be structure-sparse. Moreover, Liu *et al*. [24] proposed Network-Sliming which applied L1-norm on channel scaling factors. After training, channels with small-magnitude scaling factors are pruned. Zhang *et al*. [38] adopted GrOWL regularization for simultaneous parameter sparsity and tying in CNNs learning. For channel pruning, these works automatically obtain channel-level sparse networks in training stage. Therefore, redundant channels are pruned with less accuracy decline. However, the above methods only apply separated structured regularization on out-channel in current layer but in-channels in next layer are neglected. Besides, these methods have not been accessed with very deep neural networks on ImageNet dataset.

Low rank approximation [7, 29], network quantization [4, 19, 30], knowledge distillation [14] and reinforcement learning [1, 12] are popular techniques to speedup and compress CNNs. These techniques can be combined with our channel pruning method for further improvement.

3. Approach

3.1. Motivation

We start by analyzing the drawbacks of separated structured regularization. The optimization objective of CNNs with separated structured regularization is formulized as:

$$J(W) = Loss(W, D) + \lambda R(W) + \lambda_s \sum_{l=1}^{L} R_{ss}(W^l) \quad (1)$$

where W is trainable weights across all the L layers in CNNs and $D = \{(x_i, y_i)\}_{i=1}^{N}$ is a training dataset. $Loss(W, D)$ denotes the normal training loss on the dataset D. And $R(W)$ represents the non-structured regularization, *e.g*., L1 regularization and L2 regularization. The function $R_{ss}(\cdot)$ denotes the separated structured regularization applied on L layers separately. λ and λ_s are the hyper-parameters of non-structured regularization and structured regularization.

The correlations between two consecutive layers are illustrated in Fig. 1. Features in the i^{th} out-channel of layer l and the i^{th} in-channel of layer $l+1$ are interdependent and tend to work cooperatively [26, 36]. Accordingly they should be regularized and redistributed together during training. However, $R_{ss}(W^l)$ regularizes layer l separately. Suppose the l^{th} layer is a fully connected layer with $W^l \in \mathbb{R}^{OC_l \times IC_l}$, where OC_l and IC_l are the dimensions of W^l along the axes of out-channels and in-channels respectively. The separated structured regularization $R_{ss}(\cdot)$ applied on out-channels of W^l for channel-level sparsity is:

$$R_{ss}(W^l) = \sum_{i=1}^{OC_l} \|W^l_{i,:}\|_c \quad (2)$$

where $W^l_{i,:}$ is the weight vector of i^{th} out-channel of W^l, $\sum_i \|\cdot\|_c$ is a specific structured regularization term which can effectively zero out all weights in some out-channels, such as Group Lasso [37], CGER [35] and GrOWL [8]. The separated Group Lasso for channel-level sparsity is:

$$R^*_{ss}(W^l) = \sum_{i=1}^{OC_l} \sqrt{\sum_j (W^l_{i,j})^2} \quad (3)$$

And the derivative of $R^*_{ss}(W^l)$ with respect to $W^l_{i,j}$ is:

$$\frac{\partial R^*_{ss}(W^l)}{\partial W^l_{i,j}} = \frac{W^l_{i,j}}{\sqrt{\sum_j (W^l_{i,j})^2}} \quad (4)$$

According to the gradient descent algorithm, the update of $W^l_{i,j}$ is influenced by all the weights in the i^{th} out-channel of layer l. And the decrease of $\sum_j (W^l_{i,j})^2$ will boosts the decrease of $W^l_{i,j}$. The smaller value of $\sum_j (W^l_{i,j})^2$ is, the faster $W^l_{i,j}$ decreases. Therefore, $R^*_{ss}(\cdot)$ can effectively zero out all the weights in some out-channels in training stage.

The critical issue of separated structured regularization is that the correlations between two consecutive layers of CNNs are disregarded. It separately regularizes and enforces out-channels of each layer to be sparse. After training with separated structured regularization, features in the i^{th} out-channel of layer l may be squeezed to the i^{th} in-channel of layer $l+1$, instead of the rest out-channels of layer l. Pruning the i^{th} out-channel of layer l results in pruning the i^{th} in-channel of layer $l+1$ together. Important features in the i^{th} in-channel of layer $l+1$ may be falsely discarded that losses massive accuracy. Moreover, the separated structured regularization fails to maximally prune redundant channels and utilize the representational capacity of CNNs.

3.2. Out-In-Channel Sparsity Regularization

We propose out-in-channel sparsity regularization to tackle the drawbacks of separated structured regularization. The definition of out-in-channel is demonstrated in Fig. 1. The corresponding channels of two consecutive layers work cooperatively and are simultaneously pruned/saved. Therefore, OICSR concatenates $out\text{-}channel^l_i$ with $in\text{-}channel^{l+1}_i$ as one regularization group

$out\text{-}in\text{-}channel_i^{l,l+1}$. The optimization objective with OICSR of CNNs can be given as follows:

$$J(W) = Loss(W, D) + \lambda R(W) + \lambda_s \sum_{l=1}^{L-1} R_{oic}(W^l, W^{l+1}) \tag{5}$$

where $R_{oic}(W^l, W^{l+1})$ is the out-in-channel sparisity regularization which regularizes out-in-channels of layer l and layer $l+1$ together. OICSR of two consecutive fully-connected layers is given as:

$$R_{oic}(W^l, W^{l+1}) = \sum_{i=1}^{OC_l} \|W_{i,:}^l \oplus W_{:,i}^{l+1}\|_{oic} \tag{6}$$

where $\sum_i \|\cdot\|_{oic}$ is a specific structured regularization term in OICSR form which can simultaneously zero out all weights in some out-in-channels. The symbol $\oplus$ denotes concatenation of $W_{i,:}^l$ and $W_{:,i}^{l+1}$. For two consecutive convolutional layers, with $W^l \in \mathbb{R}^{OC_l \times IC_l \times H_l \times W_l}$ and W^{l+1}, where H_l and W_l denote the height and width respectively, we first reshape W^l and W^{l+1} to 2D matrices, *i.e.* $W^l \in \mathbb{R}^{OC_l \times (IC_l H_l W_l)}$ and $W^{l+1} \in \mathbb{R}^{(OC_{l+1} H_{l+1} W_{l+1}) \times IC_{l+1}}$. Then, OICSR of two consecutive convolutional layers can be similarly formulated as Eq. 6.

The other structured regularization terms can be extended into OICSR form in a similar way. The derivative of $R_{oic}^*(W^l, W^{l+1})$ with respect to $W_{i,j}^l$ is:

$$\frac{\partial R_{oic}^*(W^l, W^{l+1})}{\partial W_{i,j}^l} = \frac{W_{i,j}^l}{\sqrt{\sum_j (W_{i,j}^l)^2 + \sum_j (W_{j,i}^{l+1})^2}} \tag{7}$$

By incorporating $R_{oic}^*(\cdot)$ into objective function, the update of $W_{i,j}^l$ is synchronously influenced by weights in the i^{th} out-channel of layer l and weights in the i^{th} in-channel of layer $l+1$. The decrease of some weights will boost the decrease of the remaining weights in the same out-in-channel. In training stage, the out-in-channels with more redundant (nearly zero) weights will be gradully turned to insignificant out-in-channels by $R_{oic}^*(\cdot)$. Compared with $R_{ss}^*(\cdot)$ in Eq. 3 and Eq. 4, $R_{oic}^*(\cdot)$ can synchronously zero out all the weights in some less important out-in-channels and automatically obtain out-in-channel level sparse model in training stage.

OICSR regards one out-in-channel as a regularization group in which features are simultaneously redistributed during network training. After training, features in redundant out-in-channels are thoroughly transferred to important out-in-channels. As a result, OICSR is able to prune more redundant out-in-channels in large networks with less accuracy loss. Actually, the correlations between layer l and layer $l+n$ ($n \geqslant 2$) are too complex to be formulated in structured regularization form. It is a trade-off between practicability and effectiveness to consider the correlation of two consecutive layers.

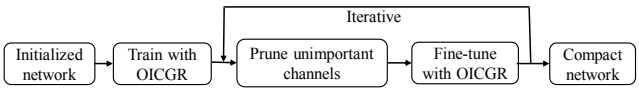


Figure 2: Iterative channel pruning procedure with OICSR.

3.3. Criterion of Channel Importance

For computational efficiency, the channel energy is chosen as the channel importance metric. Existing regularization-based methods [22, 24, 34, 35] only utilize statistics of individual layer to guide the channel pruning. In this paper, the channel importance of separated Group Lasso and non-structured regularization is defined as:

$$E_i^l = \|W_{i,:}^l\|_2^2 = \sum_j (W_{i,j}^l)^2 \tag{8}$$

where E_i^l is the energy of the i^{th} out-channel of layer l. The statistical imformation of next layer are abandoned that may cause incorrect selection of redundant channels. In particular, OICSR measures channel importance based on statistical imformation of two consecutive layers. The channel importance of Group Lasso in OICSR form is given as:

$$E_i^{l,l+1} = \|W_{i,:}^l \oplus W_{:,i}^{l+1}\|_2^2 = \sum_j (W_{i,j}^l)^2 + \sum_j (W_{j,i}^{l+1})^2 \tag{9}$$

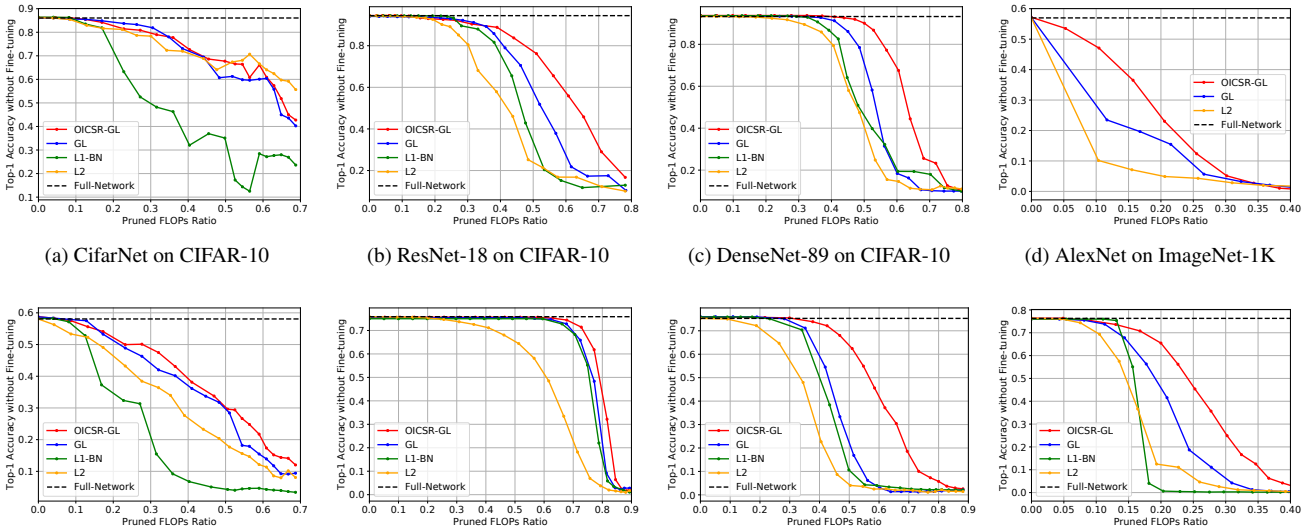
where $E_i^{l,l+1}$ is the energy of the i^{th} out-in-channel of layer l and layer $l+1$. The higher the energy, the more important the out-in-channel is.

3.4. Channel Pruning Framework

With initialized deep neural networks, our iterative channel pruning procedures are illustrated in Fig. 2. In fact, it is puzzling to manually determine the redundancy and channel pruning ratio for each layer. Therefore, a global greedy pruning algorithm is proposed to minimize the accuracy loss caused by incorrect channel pruning. As shown in Algorithm 1, in each iteration, redundant out-in-channels across all layers are globally selected and greedily removed until reaching the preset FLOPs pruning ratio.

Compared with single pass pruning, iterative pruning leads to smoother pruning process with less accuracy drop. Pruning a whole layer is detrimental to the network [3, 24]. Accordingly, we set a constraint that no more than 50% of out-in-channels in two consecutive layers are pruned in one channel pruning iteration.

Fine-tuning is an important process after channel pruning. To the best of our knowledge, we are the first to fine-tune the pruned network with structured regularization.



(a) CifarNet on CIFAR-10 (b) ResNet-18 on CIFAR-10 (c) DenseNet-89 on CIFAR-10 (d) AlexNet on ImageNet-1K

(e) CifarNet on CIFAR-100 (f) ResNet-56 on CIFAR-100 (g) PreActSeNet-18 on CIFAR-100 (h) ResNet-50 on ImageNet-1K

Figure 3: Comparison between OICSR-GL and baselines of the trade-off between top-1 accuracy (without fine-tuning) and pruned FLOPs ratio. L1-BN [24] can not be appied on the variant AlexNet[3] which has no batch normalization layers. Obviously, OICSR-GL generally has less accuracy drop compared with baselines under the same pruned FLOPs.

Algorithm 1 Global greedy pruning algorithm

Input: Training dataset $\mathcal{D}$, initialzed model $\mathcal{W}$, number of pruning iteration $\mathcal{T}$, FLOPs pruning ratio $\mathcal{P} \in \mathbb{R}^{\mathcal{T}}$

$\mathcal{W}^{(0)} \leftarrow \texttt{train}(\mathcal{W}, \mathcal{D})$ with OICSR from scratch
for $t = 1$ to $\mathcal{T}$ **do**
  $\tilde{E} \leftarrow \varnothing$
  // global channel selection
  for $l = 1$ to $L - 1$ **do**
    for $i = 1$ to OC_l **do**
      $\tilde{E} \leftarrow \tilde{E} \cup \{E_i^{l,l+1}\}$ as Eq. 9
  $\tilde{E} = \texttt{sort}(\tilde{E})$
  // greedy channel pruning
  repeat
    // remove the corresponding channel with $\tilde{E}(0)$
    $\mathcal{W}^{(t-1)} \leftarrow \texttt{prune}(\mathcal{W}^{(t-1)}, \tilde{E}(0))$
    $\tilde{E} \leftarrow \tilde{E} \backslash \tilde{E}(0)$
  until $\texttt{flops}(\mathcal{W}^{(t-1)}) < (1 - \mathcal{P}_t) \cdot \texttt{flops}(\mathcal{W}^{(0)})$
  $\mathcal{W}^{(t)} \leftarrow \mathcal{W}^{(t-1)}$
  $\mathcal{W}^{(t)} \leftarrow \texttt{fine-tune}(\mathcal{W}^{(t)}, \mathcal{D})$ with OICSR

Output: The compact model $\mathcal{W}^{(\mathcal{T})}$

Fine-tuning with OICSR simultaneously recovers the diminished accuracy of channel pruning in last step and enforces channel-level sparsity on the pruned model. Therefore, the next iteration of channel pruning is smoothly conducted after fine-tuning.

4. Experiments

In this section, we evaluate the effectiveness of OICSR on CIFAR-10 [20], CIFAR-100 [20], ImageNet-1K [6] datasets using popular CNNs architectures: CifarNet [20], AlexNet [21], ResNet [10], DenseNet [16] and SeNet [15]. OICSR is mainly compared with non-structured regularization and separated structured regularization to demonstrate its superiority. Moreover, we also compare OICSR with other state-of-the-art channel pruning methods [3, 11, 13, 17, 18, 23, 26, 27, 36]. All the experiments are implemented using PyTorch [28] on four NVIDIA P100 GPUs.

4.1. Experimental Setting

For CIFAR-10/100 datasets, OICSR is evaluated with CifarNet[1], ResNet-18[2], ResNet-56[2], DenseNet-89[2] and PreActSeNet-18[2]. OICSR is also accessed with AlexNet[3] and ResNet-50[3] on ImageNet-1K dataset. All the initialized networks are trained from scratch using SGD optimizer with a weight decay 10^{-4} and Nesterov momentum [32] of 0.9. On CIFAR-10/100 datasets, we train networks using mini-batch size 100 for 160 epochs. On ImageNet-1K dataset, we train AlexNet and ResNet-50 with mini-batch size 256 for 90 and 120 epochs, respectively. All the accuracies on ImageNet-1K dataset are tested on the validation dataset using the single view center crop.

[1] https://github.com/tensorflow/models/blob/master/research/slim/nets
[2] https://github.com/kuangliu/pytorch-cifar/tree/master/models
[3] https://github.com/pytorch/vision/tree/master/torchvision/models

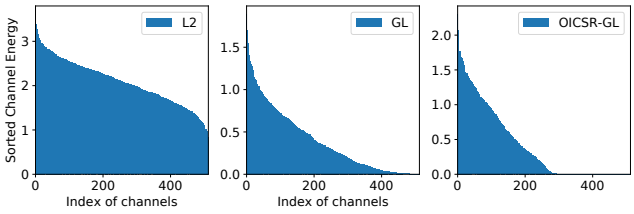


Figure 4: The distribution of energy of out-in-channels (layer4.2.conv1 and layer4.2.conv2 in ResNet-56 on CIFAR-100 dataset) after training with L2, GL and OICSR-GL respectively.

The hyper-parameter λ_s balances the normal training loss and the structured sparsity. We empirically recommend choosing relatively large λ_s for simple task but small λ_s for complex task. The hyper-parameter λ_s is set to 10^{-4} for networks (except for DenseNet-89 with 5×10^{-5}) on CIFAR-10/100 dataset and 10^{-5} for ImageNet-1K dataset.

Considering that fully-connected layers are much important in CifarNet and AlexNet and the majority of FLOPs is contributed by convolutional layers, only convolutional layers in these networks are regularized and pruned.

4.2. Comparison with Non-structured Regularization and Seperated structured Regularization

In this section, OICSR is compared with non-structured regularization and separated structured regularization from multiple aspects. The most classic structured regularization, Group Lasso [37], is chosen as the specific regularization term to demonstrate the effectiveness of OICSR. OICSR with Group Lasso (**OICSR-GL**) is used in all experiments. OICSR-GL is compared with three baselines: **(1) L2.** The network is only trained with non-structured regularization L2 regularization. The structure regularization is not used.**(2) GL (Separated Group Lasso).** Group Lasso is separately applied on layers of the network as described in Eq. 1. **(3) L1-BN.** L1-BN [24] is another form of separated structured regularization which applies L1-norm on the scaling factors of batch normalization layers. After training with L1-BN, we obtain a network in which the vector of scaling factors is sparse. Scaling factors with small magnitudes and their corresponding channels are pruned.

For all the above methods, the global greedy pruning algorithm (Algorithm 1) is uniformly adopted to prune the redundant channels. All the experimental settings are the same except for the regularization and the corresponding criterion of channel importance.

4.2.1 Accuracy without Fine-tuning

We first validate whether OICSR-GL retains more important features and accuracy after channel pruning. The substantial remaining feature/accuracy works as a great initializer that leads to higher accuracy after fine-tuning [3]. OICSR-GL is compared with relevant baselines by measuring top-1 accuracy (without fine-tuning) over pruned FLOPs. For fair comparison, results of all the methods are reported after channel pruning of the first iteration.

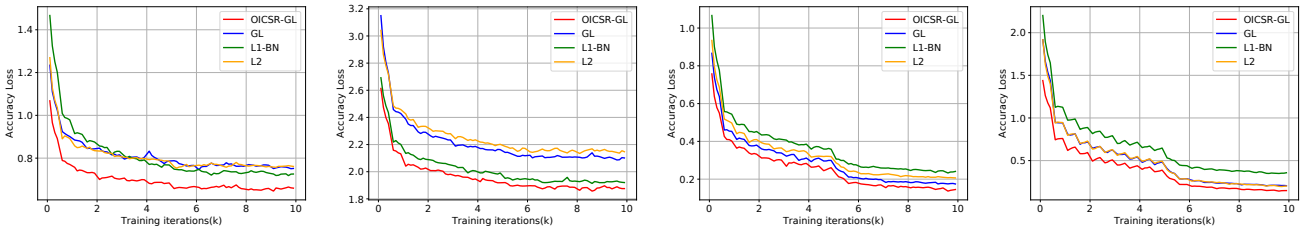
Fig. 3 shows the top-1 accuracy without fine-tuning of different classification tasks over pruned FLOPs, obtained by differentiating the structured regularization. As expected, training and pruning with structured regularization GL, L1-BN and OICSR-GL reserve more prediction accuracy compared with non-structured regularization L2 in most of cases. Separated Group Lasso is an efficient structured regularization which in general performs better than L1-BN and non-structured regularization L2. Specifically, L2, GL and OICSR-GL achieve similar performance (L1-BN performs worst) with CifarNet on CIFAR-10 dataset (Fig. 3(a)) due to simplicity of both CIFAR-10 dataset and CifarNet architecture. Finally, OICSR-GL achieves the best trade-off between pruned FLOPs and prediction accuracy without fine-tuning owing to the fact that OICSR enforces channel-level sparsity on out-in-channels and prunes redundant out-in-channels based on statistical information computed from two consecutive layers.

For channel pruning, feature/energy in one out-in-channel is pruned/saved together. The distribution of energy of out-in-channels (Eq. 9) after training with different regularization are visualized (Fig. 4) to show the effect of OICSR. After training with OICSR-GL, energy distributes in less out-in-channels and more redundant channels are automatically selected. Here, OICSR-GL regularizes out-in-channels and transfers important features in much less out-in-channels compared with separated GL and non-structured regularization L2. Therefore, important features and accuracy can be maximally preserved by OICSR-GL after pruning the redundant out-in-channels.

4.2.2 Convergence Speed

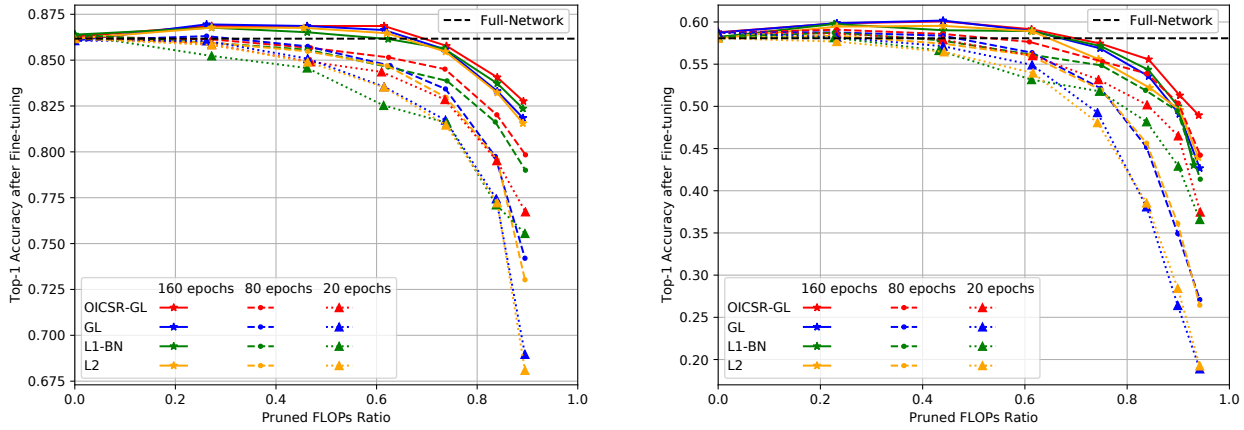
To further study advantages of OICSR-GL over baselines, we next analyze learning curves of different methods shown in Fig. 5. Interestingly, we find that L1-BN outperforms separated Group Lasso and L2 with CifarNet but performs worst with multi-branches architecture ResNet on CIFAR-10/100 datasets. For CifarNet on CIFAR-100 and ResNet-18 on CIFAR-10, separated Group Lasso converges faster and has lower accuracy loss compared with non-structured regularization L2.

OICSR-GL outperforms all the relevant baselines, which is reflected in three aspects. Firstly, OICSR-GL has less accuracy loss compared with baselines at fine-tuning iteration 0. This implies that OICSR-GL keeps higher accuracy after channel pruning, which agrees with the conclusion in



(a) CifarNet on CIFAR-10 (84% FLOPs pruned) (b) CifarNet on CIFAR-100 (74% FLOPs pruned) (c) ResNet-18 on CIFAR-10 (94% FLOPs pruned) (d) ResNet-56 on CIFAR-100 (97% FLOPs pruned)

Figure 5: Accuracy loss of pruning/fine-tuning with OICSR and relevant baselines using CifarNet and ResNet on CIFAR-10 and CIFAR-100 datasets. Pruning/fine-tuning with OICSR-GL converges fastest with the lowest accuracy loss.



(a) CifarNet on CIFAR-10

(b) CifarNet on CIFAR-100

Figure 6: Comparison between OICSR-GL and relevant baselines of the trade-off between pruned FLOPs ratio and top-1 accuracy with various fine-tuning epochs. As expected, all the methods achieve higher accuracy with more fine-tuning epochs. With the same fine-tuning epochs, OICSR-GL consistently outperforms all the baselines under different pruned FLOPs ratio.

section 4.2.1. Secondly, OICSR-GL converges faster and achieves the same accuracy loss using much fewer fine-tuning iterations. Thirdly, OICSR-GL has lowest accuracy loss after fine-tuning.

4.2.3 Accuracy after Fine-tuning

OICSR-GL is also evaluated over relevant baselines in terms of prediction accuracy after fine-tuning. The results with various fine-tuning epochs for CifarNet on CIFAR-10/100 datasets are reported in Fig 6. As expected, under the same pruned FLOPs ratio, all the methods achieve higher accuracy with more fine-tuning epochs. The loss feature/accuracy induced by channel pruning can be well retrieved by fine-tuning. The fewer fine-tuning epochs, the more obvious the advantage of OICSR-GL is. With the same fine-tuning epochs, OICSR-GL consistently performs better than all the baselines under different pruned FLOPs ratios. Moreover, OICSR-GL fine-tuned with 20 epochs achieves higher accuracy compared with baselines fine-tuned with 80 epochs in certain sparsity ranges.

Results of the other network architectures are reported in Table 1. To maintain channel-level sparsity during pruning, the networks on CIFAR-10/100 dataset and ImageNet-1K are fine-tuned with 160 epochs and 60 epochs respectively for each channel pruning iteration. Shown in Table 1, the superiority of OICSR-GL gradually emerges with the increase of pruned FLOPs ratio. Pruning channels and fine-tuning with OICSR-GL lead to higher generalization accuracy. For ResNet-18 on CIFAR-10 dataset, OICSR-GL obtains 0.64% accuracy improvement using 39.2% less FLOPs and achieves 7.4× FLOPs reduction with only 0.19% top-1 accuracy drop. OICSR-GL improves 1.09%

Acc \ Regu / FLOPs	L2	L1-BN	GL	OICSR-GL
39.2%↓	95.04%	94.95%	94.96%	**95.10%**
86.5%↓	94.10%	93.77%	94.10%	**94.27%**
94.5%↓	92.15%	91.02%	92.15%	**92.44%**

(a) ResNet-18 on CIFAR-10 (Original Acc is 94.46%)

	L2	L1-BN	GL	OICSR-GL
34.5%↓	94.09%	94.09%	94.08%	**94.21%**
81.7%↓	92.69%	92.37%	92.69%	**92.95%**
86.0%↓	90.87%	91.21%	90.37%	**91.50%**

(b) DenseNet-89 on CIFAR-10 (Original Acc is 93.25%)

	L2	L1-BN	GL	OICSR-GL
38.5%↓	76.13%	75.28%	76.04%	**76.23%**
86.2%↓	74.60%	75.13%	75.30%	**75.75%**
97.2%↓	71.98%	72.36%	72.29%	**73.10%**

(c) ResNet-56 on CIFAR-100 (Original Acc is 75.87%)

	L2	L1-BN	GL	OICSR-GL
48.0%↓	75.65%	75.76%	75.80%	**76.38%**
83.0%↓	73.26%	72.66%	72.79%	**73.91%**
95.1%↓	67.43%	65.52%	67.69%	**68.30%**

(d) PreActSeNet-18 on CIFAR-100 (Original Acc is 75.29%)

	L2	L1-BN	GL	OICSR-GL
23.4%↓	57.62%	——	57.35%	**57.87%**
54.0%↓	55.14%	——	55.02%	**56.83%**
68.3%↓	52.55%	——	49.65%	**53.78%**

(e) AlexNet on ImageNet-1K (Original Acc is 56.98%)

	L2	L1-BN	GL	OICSR-GL
37.3%↓	76.39%	76.04%	76.23%	**76.53%**
44.4%↓	76.03%	75.98%	76.02%	**76.30%**
50.0%↓	75.80%	75.53%	75.76%	**75.95%**

(f) ResNet-50 on ImageNet-1K (Original Acc is 76.31%)

Table 1: Summary of the trade-off between top-1 accuray after fine-tuning and pruned FLOPs ratio with various CNNs on three benchmark datasets. [xx.x%↓] denotes the percentage of pruned FLOPs.

accuracy while using 48.0% less FLOPs for PreActSeNet-18 on CIFAR-100 dataset. OICSR speeds up ResNet-56 by 7.2× with only 0.12% top-1 accuracy loss on CIFAR-100 dataset. For AlexNet and ResNet-50 on ImageNet dataset, OICSR gains 0.89% and 0.32% accuracy improvement while using 23.4% and 37.3% less FLOPs; OICSR-GL also achieves 2.2× and 2.0× speedup with only 0.19% and 0.36% top-1 accuracy decline respectively. Moreover, as shown in Table 1(b) and Table 1(d), OICSR-GL outperforms relevant baselines on both popular networks DenseNet-89 and PreActSeNet-18.

Methods	FLOPs↓	Params↓	Top-1 Acc↓
Huang *et al*. [17]	35.30%	—	1.00%
OICSR-GL	**39.20%**	**59.09%**	**-0.64%**
Huang *et al*. [17]	76.00%	—	2.90%
OICSR-GL	**86.50%**	**90.89%**	**0.19%**

(a) ResNet-18 on CIFAR-10

Methods	FLOPs↓	Params↓	Top-1 Acc↓
FMP [27] ([23] impl.)	37.62%	—	1.87%
GDP [23]	52.30%	—	0.77%
NISP [36]	53.70%	2.91%	0.54%
OICSR-GL	**54.00%**	**3.06%**	**0.15%**

(b) AlexNet on ImageNet-1K

Methods	FLOPs↓	Top-1 Acc↓	Top-5 Acc↓
LcP [3]	25.00%	-0.09%	-0.19%
NISP [36]	27.31%	0.21%	—
SSS [18]	31.08%	1.94%	0.95%
ThiNet [26]	36.79%	0.84%	0.47%
OICSR-GL	**37.30%**	**-0.22%**	**-0.16%**
He *et al*. [11]	41.80%	1.54%	0.81%
GDP [23]	42.00%	2.52%	1.25%
LcP [3]	42.00%	0.85%	0.26%
NISP [36]	44.41%	0.89%	—
OICSR-GL	**44.43%**	**0.01%**	**0.08%**
He *et al*. [13]	50.00%	—	1.40%
LcP [3]	50.00%	0.96%	0.42%
OICSR-GL	50.00%	**0.37%**	**0.34%**

(c) ResNet-50 on ImageNet-1K

Table 2: Comparison with existing methods. FLOPs↓ and Params↓ denote the reduction of FLOPs and parameters. Top-k Acc↓ denotes the decline of top-k accuracy and a negative value indicates an improvement of model accuracy.

4.3. Comparison with Other Methods

We compare our method with other state-of-the-art channel pruning techniques (not regularization-based) using AlexNet and ResNet on CIFAR-10 and ImageNet-1K datasets. As shown in Table 2, for ResNet-18 on CIFAR-10, OICSR-GL significantly reduces more parameters (90.89%) and FLOPs(86.50% vs. 35.30% [17]), while achieving less accuracy decline (0.19% vs. 1.00% [17]).

Our method also shows superior performance on ImageNet-1K dataset. For AlexNet, with less accuracy loss, OICSR-GL prunes more FLOPs(54.00%) compared with FMP [27] (37.62%); OICSR reduces similar FLOPs but achieves much less accuracy loss (0.15%) compared with GDP [23] (0.77%) and NISP [36] (0.54%). We are the first to exploit regularization-based channel pruning methods for very deep residual network ResNet-50 on ImageNet-1K dataset. Under various pruned FLOPs ratios, our method consistently achieves state-of-the-art result compared with

prior arts [3, 11, 13, 18, 23, 26, 27, 36], which strongly aligns with our pervious analysis and observation.

5. Conclusion

Current deep neural networks are effective with high inference costs. In this paper, we propose a novel structured regularization form, namely OICSR, which takes account correlations between successive layers to learn more compact CNNs. OICSR regularizes out-in-channels and measures channel importance based on statistical information of two consecutive layers. To minimize accuracy loss caused by incorrect channel pruning, we investigate a global greedy pruning algorithm to select and remove redundant out-in-channel in an iterative way. As a result, important features and accuracy are greatly preserved by OICSR after channel pruning. Experiments demonstrated the superiority of OICSR against non-structured regularization and separated structured regularization. Furthermore, our method achieves better results compared with existing state-of-the-art channel pruning techniques.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under Grant 61671079, Grant 61771068, and Grant 61471063, and in part by the Beijing Municipal Natural Science Foundation under Grant 4182041.

References

[1] Anubhav Ashok, Nicholas Rhinehart, Fares Beainy, and Kris M Kitani. N2n learning: Network to network compression via policy gradient reinforcement learning. *arXiv:1709.06030*, 2017.

[2] Liangchieh Chen, George Papandreou, Iasonas Kokkinos, Kevin P Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *International Conference on Learning Representations*, 2015.

[3] Ting-Wu Chin, Cha Zhang, and Diana Marculescu. Layer-compensated pruning for resource-constrained convolutional neural networks. *arXiv:1810.00518*, 2018.

[4] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv:1602.02830*, 2016.

[5] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, 2016.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, 2009.

[7] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, 2014.

[8] Mario Figueiredo and Robert Nowak. Ordered weighted l1 regularized regression with strongly correlated covariates: theoretical aspects. In *Artificial Intelligence and Statistics*, 2016.

[9] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations*, 2016.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Computer Vision and Pattern Recognition*, 2016.

[11] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *International Joint Conferences on Artificial Intelligence*, 2018.

[12] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision*, 2018.

[13] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision*, 2017.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

[15] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *arXiv:1709.01507*, 2017.

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition*, 2017.

[17] Qiangui Huang, Kevin Zhou, Suya You, and Ulrich Neumann. Learning to prune filters in convolutional neural networks. *Workshop on Applications of Computer Vision*, 2018.

[18] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *European Conference on Computer Vision*, 2018.

[19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Computer Vision and Pattern Recognition*, 2018.

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[22] Vadim Lebedev and Victor Lempitsky. Fast convnets using group-wise brain damage. In *Computer Vision and Pattern Recognition*, 2016.

[23] Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *International Joint Conferences on Artificial Intelligence*, 2018.

[24] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *International Conference on Computer Vision*, 2017.

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition*, 2015.

[26] Jianhao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. *International Conference on Computer Vision*, 2017.

[27] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *International Conference on Learning Representations*, 2017.

[28] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[29] Bo Peng, Wenming Tan, Zheyang Li, Shun Zhang, Di Xie, and Shiliang Pu. Extreme network compression via filter group approximation. In *European Conference on Computer Vision*, 2018.

[30] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, 2016.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[32] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and ,momentum in deep learning. In *International Conference on Machine Learning*, 2013.

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition*, 2015.

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

[35] Jaehong Yoon and Sung Ju Hwang. Combined group and exclusive sparsity for deep neural networks. In *International Conference on Machine Learning*, 2017.

[36] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I. Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S. Davis. Nisp: Pruning networks using neuron importance score propagation. In *Computer Vision and Pattern Recognition*, 2018.

[37] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2006.

[38] Dejiao Zhang, Haozhu Wang, Mario Figueiredo, and Laura Balzano. Learning to share: Simultaneous parameter typing and sparsification in deep learning. *International Conference on Learning Representations*, 2018.

[39] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*. Springer, 2016.