Neural Computing and Applications (2021) 33:16197–16212
https://doi.org/10.1007/s00521-021-06222-0

ORIGINAL ARTICLE

CondenseNet with exclusive lasso regularization

Lizhen Ji[1] · Jiangshe Zhang[1] · Chunxia Zhang[1] · Cong Ma[1] · Shuang Xu[1] · Kai Sun[1]

Received: 18 December 2020 / Accepted: 8 June 2021 / Published online: 27 June 2021

Abstract
Group convolution has been widely used in deep learning community to achieve computation efficiency. In this paper, we develop CondenseNet-elasso to eliminate feature correlation among different convolution groups and alleviate neural network's overfitting problem. It applies exclusive lasso regularization on CondenseNet. The exclusive lasso regularizer encourages different convolution groups to use different subsets of input channels therefore learn more diversified features. Our experiment results on CIFAR10, CIFAR100 and Tiny ImageNet show that CondenseNets-elasso are more efficient than CondenseNets and other DenseNet' variants.

Keywords CondenseNet · Exclusive lasso · Group convolution · Neural network regularization

1 Introduction

In the past decade, deep learning has achieved remarkable breakthroughs in a large variety of applications, such as image classification [25, 40], object detection [37, 38] and semantic segmentation [20, 29]. Meanwhile, the architecture of deep convolutional neural networks (CNNs) has evolved for years. ResNet [12] is a milestone in the development of neural network architectures by introducing shortcut connections to ease optimization in training very deep networks; however it utilizes a large number of parameters. To alleviate the huge computational burden from ResNet, a more computationally efficient DenseNet is proposed [17]. Different from ResNet of combining features through summation, DenseNet combines feature from concatenation, therefore it can be trained more efficiently. However, some redundancy still lies in there, especially in bottleneck layers. Visualization of connectivity pattern of DenseNet shows that later layers tend to prefer more recently learned features and some features from very early layers [17]. Deep roots also claims it is unlikely that every filter in a neural network relies on the output of all the filters in the previous layer [18]. These observations lead to sparsely connected architectures like LogDenseNet [14] and SparseNet [57], which use a sparsely "log-offset" connectivity pattern to improve DenseNet's efficiency in terms of parameters. However, this pre-defined connectivity pattern is not flexible. To overcome this shortcoming, Huang et al. propose CondenseNet [16], a novel network architecture to gradually remove less important connections in bottleneck layers. Based on the observation that different convolutional groups in CondenseNet prunes filters independently and inspired by the fact that exclusive lasso regularization brings sparsity at intra-group level [22], we insert exclusive lasso penalty into CondenseNet to learn more diversified features. Our proposed model is denoted as **CondenseNet-elasso**. Our method applies to scenarios whose network backbone composed of stacks of dense blocks. It is also promising to study to combine our proposed method to applications like medical images [21], graph convolution networks [7] and body pose prediction [46].

Our Contribution In this paper, we propose an exclusive lasso regularizer on the learned group convolutional layer in CondenseNet to decorrelate filters between different convolutional groups, therefore alleviating the neural network's overfitting problem. This method can reduce filter co-dependence between different groups. We validate CondenseNet-elasso of varying depths on our proposed

✉ Jiangshe Zhang
jszhang@mail.xjtu.edu.cn

Lizhen Ji
jlz_stat@stu.xjtu.edu.cn

1 School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, People's Republic of China

method on three public datasets and medical images (Appendix B). The experimental results demonstrate that it achieves better performance under the same computation budget compared with CondenseNet and achieves much better performance compared with other DenseNet's variants.

Outline of the paper Section 2 gives a brief review of related works. Section 3 describes DenseNet and CondenseNet which our method is based on. Section 4 devotes to introduce our proposed CondenseNet-elasso. Next, in Sect. 5, a large set of experiments are carried out to examine the performance of CondenseNet-elasso. To be specific, Sect. 5.3 shows classification results on CIFAR and Tiny ImageNet on models of different scales. Section 5.4 validates our assumption on why the exclusive lasso regularization works. Section 5.6 compares our model with other group convolution variants. Finally, we conclude this paper in Sect. 6.

2 Related work

In this section, we review related works on network pruning methods, group convolutions, neural network regularization and exclusive lasso.

Network pruning Deep networks often have a large number of redundant weighs that can be pruned without sacrificing accuracy [6]. Han et al. propose an iterative process to remove unimportant connections based on l_1-norm and l_2-norm, followed by fine-tuning to recover model accuracy [10]. Hu et al. claim that neurons with high APoZ (average percentage of zeros) are redundant and can be pruned without affecting the overall performance of the network [15]. Structured Sparsity Learning (SSL) is proposed to regularize the structures (filter, filter shapes and layer depth) of DNNs [45]. ThiNet uses the reconstruction error of the of the next layer to measure the importance of filters in the current layer [31]. Li et al. propose a data free filter selection criterion to use the l_1-norm as the importance measurement [26]. This method can prune multiple layers at once based on a layer level sensitivity analysis. He et al. propose the soft filter pruning method which allows pruned filters to recover to nonzero through backpropagation [13]. Yu et al. measure neuron importance to minimize the reconstruction error in the pre-softmax layer. The paper gives out a closed form solution to calculate the "importance score" in earlier layers by propagating back from the "final response layer" [51].

Neural network regularization Many works on neural network regularization have been proposed to benefit generalization. Dropout [41] and dropconnect [42] introduce regularization of deep networks through randomly setting a subset of activations or weights to zero during training. Cogswell et al. propose a regularizer that encourages non-redundant or diverse representations in DNNs through minimizing the cross-covariance of hidden activations [4]. Changpinyo et al. propose to train models in an incrementally manner by starting with a network only contains a small fraction of connections and add connections over time [2].

Group lasso regularization has been introduced into deep neural networks to obtain highly compact networks [39]. However, there exists a strong filter correlation among the convolutional filters trained by group lasso constraints. Based on this observation, a decorrelation regularization method was proposed to weaken the correlation between different filters and achieve a more compact model [58]. In the meantime, a sparsity-inducing regularizer called GrOWL (group ordered weighted l_1) was introduced, which not only eliminates unimportant neurons but also identifies heavily correlated neurons by setting the corresponding weights to a universal value [52].

Group convolution Group convolution has been widely used in designing efficient networks. It was first introduced in AlexNet due to a lack of GPU memory by partitioning inputs into G mutually exclusive groups, each producing its own output [25]. As a result, the number of parameters and FLOPs reduce to $1/G$ of standard convolution. Meanwhile, ResNeXt investigates the trade-off between depth, width and the number of groups [47]. The paper suggests that a larger number of groups leads to better accuracy under similar computational costs. Ioannou et al. make a further study on group convolution and propose Deep roots, which uses filters groups to force the network to learn filers with only limited dependence on previous layers [18].

Some researchers also propose learnable group convolutions to make the group convolution more flexible. Peng et al. use the low-rank decomposition to approximate the weight matrix W into the product of two matrices D and P, in which D is a block diagonal matrix which can be turned into a group convolution and P is a 1×1 convolution [35]. Later on, fully learnable group convolution (FLGC) is proposed to learn a binary selection matrix for input channels and groups, including input channel-group connectivity and filter-group connectivity [44]. One disadvantage of this method is that the final group convolution does not have the same number of input channels in each group and therefore it is hard to implement in some deep learning frameworks (like PyTorch). Besides, FLGC forces each input channel to be selected into the group with the maximum probability. However, it may not be desirable since some important input channels can be shared among different convolutional groups. Meanwhile, Zhang et al.

propose a dynamic grouping method, which can learn group number and channel connections simultaneously through a relationship matrix [55]. To reduce the number of learnable parameters in the relationship matrix, the authors decompose it to a series of kronecker product of smaller matrices. Dynamic grouping convolution can learn the number of groups in each layer in an end-to-end manner; however it requires the number of convolutional filters to be the power of two, besides, some group connectivity patterns cannot be represented by kronecker product of small matrices. Recently, Guo et al. propose a self-grouping convolutional neural networks (SG-CNN) who generate groups based on a clustering method, resulting in similar filters within groups and diverse filters between groups [9]. However, the resulting convolution groups have an uneven number of filters inside each group which leads to inefficient implementation.

Exclusive lasso Exclusive lasso penalty ($l_{1,2}$-norm) [56] was first introduced in a multi-task feature selection problem. This regularizer introduces competition among different tasks for the same feature. An empirical way to analyze the behavior of a penalty is to visualize its corresponding isosurface [39]. An illustration example is shown in Fig. 1c. The figure shows two special cases of exclusive lasso: if each parameter is in its own group, the penalty is equivalent to l_2 penalty while if all parameters are in the same group, the penalty is equivalent to the square of l_1 norm.

Later on, Kong et al. propose the exclusive group lasso leading to sparsity at an intra-group level [22]. In deep learning context, Yoon and Hwang first use exclusive lasso penalty as a regularization method in neural networks [50]. The paper employs group lasso to promote inter-group sparsity and exclusive lasso to enforce intra-group sparsity. They use a weighted version of these two regularizers to achieve feature sharing and feature competition simultaneously. the resulting model is denoted as CGES (combined group and exclusive sparsity). Exclusive sparsity helps the network converge faster, learn less redundant features, and make each group be as different as possible. Different from CGES, our model achieves a group-level sparsity while CGES achieves a filter-level sparsity. This is because our model prunes filters based on the *condensation criterion* while CGES prunes filters based on group lasso and exclusive lasso regularizer.

3 DenseNet and CondenseNet

In this section, we describe DenseNet [17] and CondenseNet [16] on which our method is based.

3.1 DenseNet

The distinguishing property of DenseNet is that each layer receives a concatenation of all feature maps that are generated by all preceding layers within the same dense block

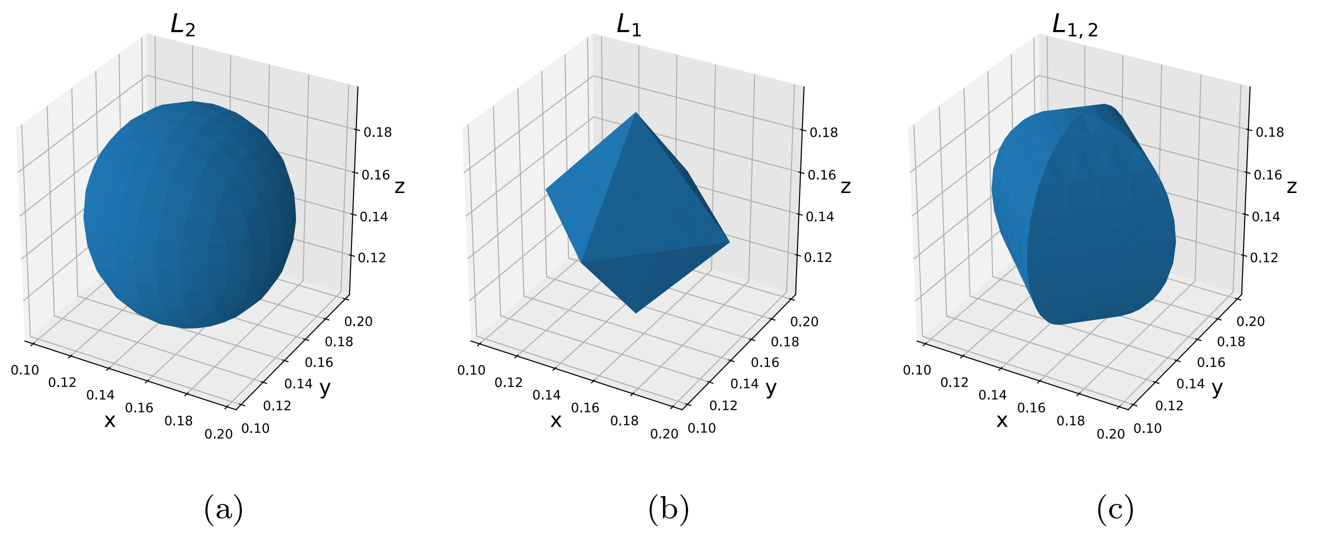

Fig. 1 Unit balls for different regularization terms. let $W = (w_1^1, w_2^1, w_1^2)$ where superscript denotes group number and subscript denotes index within a group. In this case, w_1^1, w_2^1, w_1^2 are variables along x-, y- and z-axes, respectively. The penalty is defined as $\Omega = (|w_1^1| + |w_2^1|)^2 + |w_1^2|^2$. **a** Considering variables in different groups (w_1^1 and w_1^2) by setting $w_2^1 = 0$, this yields the ball generated by l_2-norm. The l_2 unit ball is a sphere, which does not favor any of the variables. **b** Considering variables in the same group (w_1^1 and w_2^1) by setting $w_1^2 = 0$, this yields a unit ball generated by l_1-norm. The l_1 unit ball is a regular octahedron surface, enforcing sparsity between different variables. **c** Unit ball for $l_{1,2}$ penalty. $l_{1,2}$ unit ball enforcing sparsity inside group and diversity between groups

as its input. In particular, the lth layer receives the feature maps of all preceding layers,namely,

$$x_l = H_l([x_0, x_1, \ldots, x_{l-1}]) \tag{1}$$

where $[x_0, x_1, \ldots, x_{l-1}]$ refers to the concatenation of the feature maps produced in the previous l layers. There are two architectures in DenseNet [17]. One is DenseNet whose basic building block is a 3×3 convolutional layer (Fig. 2a). The other one is DenseNet-bottleneck (DenseNet-BC) whose basic building block is composed of one 1×1 bottleneck layer followed by one 3×3 convolutional layer (Fig. 2b). In this paper, we mainly focus on pruning bottleneck layers, therefore we use DenseNet as an abbreviation for DenseNet-BC in later discussions.

3.2 CondenseNet

To learn a good connectivity pattern automatically, CondenseNet was proposed on the basis of DenseNet [16]. Specifically, the authors design a new basic building block, *Learned Group Convolution*, by splitting the filters of bottleneck layer into multiple groups and gradually remove less important features during training. CondenseNet improves DenseNet's efficiency in terms of number of parameters and floating-point operations (FLOPs). The most prominent characteristic of CondenseNet is that the final pre-trained model can be converted into standard group convolutions which brings in actual acceleration at deployment.

Here, we first introduce some notations to facilitate the discussions in this section. Standard convolutional layer generates O output feature maps by applying O convolutional filters over R input feature maps. For bottleneck layers with kernel size one, a 4D weight tensor is simplified to a 2D matrix. For each convolutional layer, the kernel is divided to G groups, denoted as $\boldsymbol{W}^1, \boldsymbol{W}^2, \ldots, \boldsymbol{W}^G$ where G is a pre-defined number, in which $\boldsymbol{W}^g$ is of size $\frac{O}{G} \times R$. We use the symbol $W_{i,j}^{g,l}$ to represent the weight of the jth input for the ith output within group g in layer l. In what follows, we will introduce some key components in CondenseNet.

Network DenseNet's basic building layer is composed of one 1×1 convolutional layer followed by one 3×3 convolutional layer. CondenseNet replaces its first layer with learned group convolution and replaces its second layer with group convolution. DenseNet adds k new feature maps at each block, which is referred as *growth rate*. Meanwhile, CondenseNet uses an "exponentially increasing growth rate" schedule: the growth rate doubles when the feature map size downsamples. To encourage feature reuse, CondenseNet removes 1×1 convolutional layers for channel reduction in transition blocks between stages in DenseNet. If not, the resulting model is called **CondenseNet-light**.

Condensation criterion Condensation criterion measures the importance of each input channel to each convolutional group. It gives the each convolutional group the flexibility to select the most relevant input features and guarantees filters in each convolutional group select the same subset of input channels. To be specific, the importance of jth input channel for the filter group g is evaluated by the averaged absolute value of weights between them across all outputs within the group. This importance score is denoted as $S_j^{g,l}$, namely, $S_j^{g,l} = \sum_{i=1}^{O/G} \left| W_{i,j}^{g,l} \right|$. The pruning procedure used in CondenseNet can be summarized as follows: For a given group g, calculate importance score $S_j^{g,l}$ for each input channel j; Sort $S_j^{g,l}$ in an ascending order, select the smaller $1/C$ filters, and denote the corresponding filters as $\boldsymbol{W}_{j,lower}^{g,l}$; Zero out the filters in $\boldsymbol{W}_{j,lower}^{g,l}$. This condensation criterion forces each layer to have limited dependency on previous layers and this sparse connectivity pattern reduces computational complexity and model size.

Training Suppose M denote the total number of training epochs, the first $M/2$ training epochs called *condensing stage* is used for pruning while the second $M/2$ training epochs called *optimization stage* is used for fine-tuning. Each condensing stage screens out $1/C$ of filters for each convolutional group, there are $C-1$ condensing stage in total and therefore only $1/C$ filters left in the final model. One thing to note is that the pruned weights are not removed during training, the Filter $\mathcal{F}$ is masked by a binary tensor $\mathcal{M}$ of the same size using the element-wise product. When training is finished, the learned group convolutions are rearranged to standard group convolution through $\mathcal{M}$, the final model is called the *converted model*. The FLOPs and parameters for learned group convolutions reduce to $1/C$ of standard convolution in the final converted model. To facilitate the understanding of the training process of CondenseNet, we illustrate it in Fig. 3. Suppose that the total training epoch is 300 and the condensation factor C is 4, each learned group convolution is pruned before epoch 50, 100 and 150 with 75%, 50% and 25% filters remained,

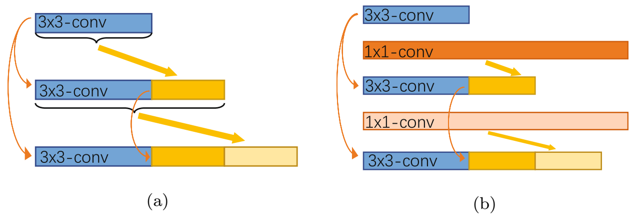

Fig. 2 **a** DenseNet basic building block. **b** DenseNet-bottleneck basic building block

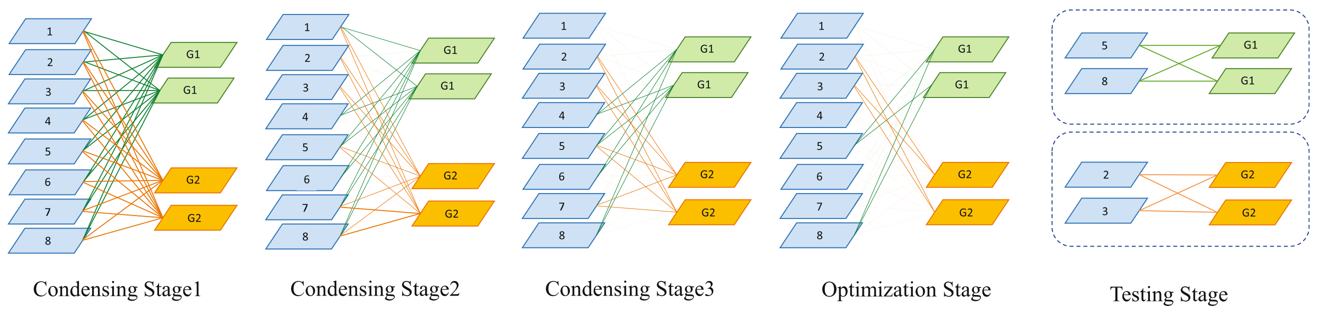

Fig. 3 CondenseNet training process illustration with group number G=2 and input channel number R=8. Condensation factor C is set to 4. Group 1 prunes out $2, 3 \rightarrow 1, 7 \rightarrow 4, 6$ while Group 2 prunes out $4, 6 \rightarrow 1, 8 \rightarrow 5, 7$ in three condensing stages. In the testing stage, $\{5, 8\}$ and $\{2, 3\}$ are kept for Group 1 and Group 2 in the final converted model. Pruned filters are marked by dashed light lines (Best viewed in color)

respectively. Next, the model is fine-tuned for another 150 epochs. A detailed training process is illustrated in Algorithm 1.

4 CondenseNet with exclusive lasso regularizer

At present, various regularization strategies have been developed and they are indeed beneficial to enhance the generalization capacity of DNNs. A simple and effective way is to add a proper regularization term to the loss function. To the best of our knowledge, there is no literature investigating how the exclusive lasso regularizer can be applied in CondenseNet. In order to address this issue, we would like to improve CondenseNet's performance by introducing an exclusive lasso penalty on learned group convolutions. In later discussions, the proposed model is abbreviated as CondenseNet-elasso for ease of exhibition. First, we introduce some notations for the rest of the paper. Suppose $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$ denote the training dataset with N instances. $\boldsymbol{x}_i$ is ith input, y_i is the class label with k classes. Let $\boldsymbol{W}^{(l)}$ denote the weights of the lth layer, and L denote the total number of layers. $\{\boldsymbol{W}^{(l)}\}$ includes the weights across all L layers. $\mathcal{L}(\boldsymbol{W})$ is the cross-entropy loss of the classification task parameterized by $\boldsymbol{W}$. Suppose R and O denote the number of input and output channels, respectively. G indicates the number of group convolutions. $W_{i,j}^{g,l}$ refers to the jth input for the ith output for group g in layer l.

Regularizer Exclusive lasso was first introduced in a multi-task learning framework to enforce the model parameters in different tasks to compete for features. The training procedure of CondenseNet guarantees that different groups are pruned independently. Based on this observation, we expect different convolutional groups to compete for input channels, therefore achieve diversified feature representations. As a result, we regard filters connected to each input channel as a group. Specifically, we define the regularizer in layer l as:

$$\Omega(\boldsymbol{W}^{(l)}) = \sum_{j=1}^{R}\left(\sum_{g=1}^{G} S_j^{g,l}\right)^2 \tag{2}$$

$$= \sum_{j=1}^{R}\left(\sum_{g=1}^{G}\sum_{i=1}^{O/G}\left|W_{i,j}^{g,l}\right|\right)^2 \tag{3}$$

$$= \sum_{j=1}^{R}\left(\sum_{i=1}^{O}\left|W_{i,j}^{g,l}\right|\right)^2. \tag{4}$$

Experiment results in Sect. 5.4.1 show that our proposed regularization term actually brings in less overlap of incoming channels among different convolutional groups. Moreover, Sect. 5.4.2 shows that the proposed regularizer does help different groups to learn more diversified features. In preliminary experiments, we also tried different grouping strategies, for example, we regard filters from the convolutional group g connected with input channel j as a group, which is $\Omega_1(\boldsymbol{W}^{(l)}) = \sum_{j=1}^{R}\sum_{g=1}^{G}\left(\sum_{i=1}^{O/G}\left|W_{i,j}^{g,l}\right|\right)^2$. However, this regularizer does not lead to a consistent model performance improvement over baseline models.

One thing to note is that Eq. (2) can be reformulated to Eq. (4) while the format of Eq. (4) is equivalent to the exclusive sparsity regularization in [50]. However, our method is different from [50] in the following two aspects. First, CondenseNets-elasso prune filters through condensation criterion, not from this sparsity inducing regularization term. The exclusive lasso penalty here is used to **encourage** less redundancy between groups, not **strictly forcing** each filter being included in exactly one convolutional group. Second, condensation criterion guarantees that filters in the same group take the same subset of incoming channels. In other words, a given input channel can be selected by *all* filters in a convolutional group or *none* filter in this convolutional group. This characteristic results in group-level filter pruning while [50] results in kernel-level filter pruning.

Loss We use the following loss function in training:

$$\mathcal{L}(\{\boldsymbol{W}^{(l)}\}, \mathcal{D}) + \lambda \sum_{l=1}^{L} \Omega(\boldsymbol{W}^{l}). \tag{5}$$

Here, λ represents the hyperparameter for the regularization term $\Omega(\boldsymbol{W}^{(l)})$ and details on the hyperparameter setting are described in Sect. 5.5. One thing to note is that we don't include $\Omega(\boldsymbol{W}^{l})$ at optimization stage since it is only used for selecting promising filters. The detailed training process is shown in Algorithm 1.

Algorithm 1 Training Process of CondenseNet and CondenseNet-elasso

```
Parameter: Condense factor C, Total training epoch M
for i in [1,2,..,C-1] do                                          ▷ Condensing Stage i
    for j in [1,2,...,$\frac{M}{2(C-1)}$] do
        Train the model, Loss = $\mathcal{L}(\{\boldsymbol{W}^{(l)}\}, \mathcal{D}) + \lambda \sum_{l=1}^{L} \Omega(\boldsymbol{W}^{l})$   ▷ CondenseNet-elasso
        Loss = $\mathcal{L}(\{\boldsymbol{W}^{(l)}\}, \mathcal{D})$                     ▷ CondenseNet
    end for
    Pruning: Mask $\frac{1}{C}$ of filters
end for
for j in [1,2,...,$\frac{M}{2}$] do                                ▷ Optimization Stage
    Train the model, Loss = $\mathcal{L}(\{\boldsymbol{W}^{(l)}\}, \mathcal{D})$
end for
pre-trained model $\rightarrow$ converted-model                    ▷ Convert the model
```

Optimization Campbell et al. give a coordinate descent algorithm for optimization in the context of predictive regression modeling with exclusive lasso regularization [1]. Meanwhile, CGES use a proximal gradient descent method for optimization. Concretely, the learned group convolution kernels can be updated in an iterative manner by first updating the selected variables with the loss-based gradients, then applying the proximal operator for them. Suppose $\boldsymbol{W}_j^l \in \mathcal{R}^{O \times 1 \times k \times k}$ represent the weights connected with jth input channel in layer l, the proximal operator for the exclusive lasso regularizer is defined as:

$$prox_{EL}(\boldsymbol{W}_j^l) = sign(W_{j,i}^l)\left(\left|W_{j,i}^l\right| - \lambda \|\boldsymbol{W}_j^l\|_1\right)_+. \tag{6}$$

In our case, we prune filters based on condensation criterion therefore we do not follow Eq. (6) and stick to the conventional stochastic gradient descent algorithm for optimization in our experiments.

5 Experiments

5.1 Datasets

CIFAR Both CIFAR-10(C10) and CIFAR-100(C100) [24] consist of colored images with 32×32 pixels. C10 and C100 have 10 and 100 classes, respectively. The training and testing sets contain 50,000 and 10,000 images, respectively. Following [16], we adopt a standard data augmentation schedule, including mirroring, shifting and normalizing data using the channel means and standard deviations.

Tiny ImageNet The Tiny ImageNet dataset is a subset of ImageNet [5]. There are 200 categories sampled from 1000 classes of ImageNet, each class consists of 500 training images, 50 validation images and 50 test images. All images are downsampled to a fixed resolution of 64×64. For preprocessing, 4 pixels are padded on every side and a 64×64 crop is randomly sampled from the padded image or its horizontal flip. We normalize the data using the channel means and standard deviations. For testing, we only evaluate the original 64×64 images.

5.2 Training

We use the following default settings in all our experiments unless otherwise specified. The default growth rate is {8,16,32} on CIFAR and is {12,24,48} on Tiny ImageNet. Default condensation factor C and group number G is 4. We call the blocks within the same **stage** if they have the same feature map size. All our networks have three stages and each stage has the same number of blocks. We choose models with different depths to test the effect of the exclusive lasso regularization on different model scales. Specifically, models with 50, 86, 122 and 182 layers have {8-8-8}, {14-14-14}, {20-20-20} and {30-30-30} blocks in three stages, respectively.

Following the training schedule in [16], all networks are trained using stochastic gradient descent (SGD). Specifically, we adopt Nesterov momentum with a weight decay of 1e-4 and a momentum of 0.9. We train the models using a batch size of 64 for 300 epochs on all datasets by default. We adopt the weight initialization introduced by [11] and batch normalization [19]. For CondenseNet-182 on CIFAR, we train the model for 600 epochs with a dropout [41] rate of 0.1. CondenseNet-182 on Tiny ImageNet is trained for 300 epochs with a dropout rate of 0.1. We use a cosine learning rate [30] starting from 0.1 and gradually reduces to 0. Following the implementation of CondenseNet, we apply a dropout layer after batch normalization layer, which is suggested by [27] to avoid the "variance shift" phenomenon when dropout layers are placed before batch normalization layers. We zero out gradients of the pruned filters during backward propagation. To ensure a fair comparison between our proposed method and the original model, we report the re-implemented results of CondenseNets following https://github.com/ShichenLiu/CondenseNet. We use the same random seed for weight initialization when comparing CondenseNets and CondenseNets-elasso. To save GPU storage and fit large models on one GTX 1080ti, we follow the implementation in memory-efficient DenseNet [36]. To be more specific, we checkpoint the learned group convolution part during training by discarding the intermediate feature maps during the forward pass and recompute them for the backward pass at the expense of additional training time.

5.3 Classification results

Results on CIFAR In Table 1, we perform experiments on CIFAR datasets to validate the effectiveness of our proposed method. Concretely, we compare CondenseNet-elasso with DenseNet, CondenseNet, interleaved group convolution [53] and variants of ResNet and DenseNet [3, 43, 47, 48]. We train CondenseNets and CondenseNets-elasso 3 times and report the mean errors. First, compared with CondenseNets, our proposed method drops classification error rate by 0.03%, 0.08%, 0.18% and 0.12% on CIFAR-10 and 0.52%, 0.38%, 0.25% and 0.34% on CIFAR-100 on models of 50, 86, 122 and 182 layers, respectively. The performance gain becomes larger on CIFAR-10 as the model goes deeper in most cases. While on CIFAR-100, our proposed method achieves a noticeable 0.3725% performance boost on average. Compared with DenseNet-40-60 on CIFAR100, CondenseNet-182-elasso achieve a 1.79% lower error rate with only 0.28x FLOPs and 1.03x parameters.

Moreover, we apply the two recently proposed LAP [34] and Hinge [28] methods on DenseNet for comparison. When comparing our CondenseNet-122-elasso with LAP-DenseNet-122-{8,16,32}, our model achieves 3.04% and

Table 1 CIFAR: Model performance comparison between our proposed method and models from ResNet's and DenseNet's family

Model	MParams	MFLOPs	C-10 (%)	C-100 (%)
DPN-28-10 [3]	47.80	–	3.65	20.23
ResNeXt-29(8x64d) [47]	34.40	–	3.65	17.77
MixNet-100-12 [43]	1.50	–	4.19	21.12
MixNet-250-24 [43]	29.00	–	3.32	17.06
CliqueNet-36-12 [48]	0.98	901	5.80	26.41
CliqueNet-150-30 [48]	10.02	8490	5.06	21.83
LGC-L16M32 [53]	17.70	–	3.37	19.31
LGC-L32M26 [53]	24.10	–	3.31	18.75
LAP-DenseNet-50-{8,16,32} [34]	0.26	29	7.08	29.04
LAP-DenseNet-86-{8,16,32} [34]	0.59	66	5.49	25.74
LAP-DenseNet-122-{8,16,32} [34]	1.05	117	4.98	24.57
Hinge-DenseNet-28 [28]	1.03	134	5.96	27.41
Hinge-DenseNet-28-60%pruned [28]	0.55	53.8	7.18	28.58
Hinge-DenseNet-58 [28]	4.98	637	4.75	23.64
Hinge-DenseNet-58-78%pruned [28]	1.10	140	6.08	25.39
DenseNet-100-12 [17]	0.80	293	4.51	22.27
DenseNet-40-36 [17]	1.58	651	4.58	22.30
DenseNet-40-48 [17]	2.78	1154	3.99	20.29
DenseNet-40-60 [17]	4.32	1801	4.01	19.99
CondenseNet-50*	0.26	29	5.95	27.67
CondenseNet-86*	0.59	66	5.02	23.62
CondenseNet-122*	1.05	117	4.59	21.78
CondenseNet-182*	4.45	513	3.70	18.54
CondenseNet-50-elasso*	0.26	29	5.92 (↑0.03)	27.15 (↑0.52)
CondenseNet-86-elasso*	0.59	66	4.94 (↑0.08)	23.24 (↑0.38)
CondenseNet-122-elasso*	1.05	117	4.41 (↑0.18)	21.53 (↑0.25)
CondenseNet-182-elasso*	4.45	513	3.58 (↑0.12)	18.20 (↑0.34)

MParams and MFLOPs represent parameters and FLOPs in megabytes measured on final converted models on CIFAR-100. In model name, the first number after "-" represents the model depth while the second number(if exists) after "–" represents the growth rate. For example, DenseNet-100-12 represents the DenseNet with 100 layers in total and a growth rate of 12. Models with superscript * are self-implemented and "–" denotes unavailable data

0.57% higher classification accuracy on CIFAR-100 and CIFAR-10, respectively, under the same computation settings. Comparing CondenseNet-122-elasso with Hinge-DenseNet-58-78%pruned, our model achieves 3.86% and 1.67% error rate reduction on CIFAR-100 and CIFAR-10 with 84% and 95% of FLOPs and parameters. One thing to note is that Hinge-DenseNet-58 means un-pruned DenseNet-58 trained by the original implementation and 78% pruned represents 78% of FLOPs being pruned. More detailed training settings on Hinge and LAP are described in Appendix A. Experiment results in Table 1 show that CondenseNets-elasso is more compute efficient.

Next, to validate the regularization effect of this exclusive lasso term, we plot the training and validation loss during the training process in Fig. 4. Our proposed model converges faster and yields lower training accuracy but higher test accuracy. This observation suggests that the performance gain comes from less overfitting induced from exclusive lasso regularization.

Results on tiny imageNet In Table 2, we conduct experiments on a more complicated Tiny ImageNet dataset to validate the effectiveness of our proposed method. The results in the table show that CondenseNets-elasso drop top-1 error rate by 0.57%, 0.45%, 0.27% and 0.51% while drop top-5 error rate by 0.58%, 0.63%, 0.57% and 0.36% on networks of 50, 86, 122 and 182 layers compared with CondenseNets. Moreover, compared with DenseNet-100, CondenseNet-122-elasso attains 1.62% lower top-1 error with 0.9x FLOPs and 1.16x parameters. As for DenseNet-192, CondenseNet-182-elasso achieves 0.66% lower top-1 error using 0.8x FLOPs and 1.04x parameters. When comparing our CondenseNet-50-elasso with LAP-DenseNet-52-{8,16,32}, our model achieves 3.3% top-1 and 1.1% top-5 higher classification accuracy with only 62% FLOPs and 82% parameters. Comparing CondenseNet-122-elasso with Hinge-DenseNet-58-60%pruned, our model achieves 4.46% top-1 and 2.2% top-5 error rate reduction with 1.1x and 83% of FLOPs and parameters.

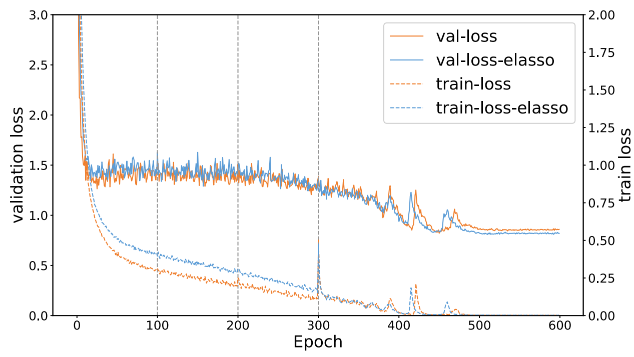

Fig. 4 CIFAR100: Training and validation loss comparison of CondenseNet-182 and CondenseNet-182-elasso. Vertical dashed lines show the pruning epochs. The bump at epoch 300 is caused by pruning half of the filters, model performance gets recovered soon

Next, a more detailed efficiency comparison on parameters and FLOPs among DenseNet, CondenseNet and CondenseNet-elasso is shown in Fig. 5. From the figure we can see that CondenseNet-elasso is more efficient in FLOPs and parameters compared with DenseNet and CondenseNet. To sum up, our model achieves 0.1%, 0.37% and 0.54% error rate reduction on average on CIFAR10, CIFAR100 and Tiny ImageNet, this observation shows that the exclusive lasso regularization term performs better on complicated datasets, which validates the regularization effect.

5.4 Channel reuse

5.4.1 Overlap statistic

In this section, we validate our assumption that exclusive lasso penalty encourages different groups to use different subsets of incoming channels. The results are shown in Fig. 6. Suppose that the total number of incoming channels in lth layer is C_{in}^l and the group number is G. The set of selected channels for gth group in layer l is denoted as $Set_{in}^{l,g}$. OS_{g_m,g_k}^l denotes the overlap percentage between group g_m and group g_k in lth layer and $\#$ denotes the number of the set, namely,

$$OS_{g_m,g_k}^l = \frac{\#(Set_{in}^{l,g_m} \bigcap Set_{in}^{l,g_k})}{C_{in}^l/G} \tag{7}$$

As a result, $OS_{g_m,g_k}^l \in [0,1]$. There are $\binom{G}{2}$ pair of groups between G groups, therefore the overlap statistic between two groups is defined as:

$$OS_2^l = \frac{2}{G(G-1)} \sum_{m=1}^{G} \sum_{k>m}^{G} OS_{g_m,g_k}^l \tag{8}$$

Similarly, we can define overlap statistic between 3 groups as OS_3^l and 4 groups as OS_4^l.

In Fig. 6, three subplots represent overlap statistics in stage1, stage2 and stage3 from left to right. Lines in different colors show average overlap statistics between 2, 3 and 4 groups. In this experiment, hyperparameter λ is set to 1e-7, 5e-7, 1e-6 in three stages, respectively. The plot in the figure first shows that overlap statistics in CondenseNets-elasso are lower than their corresponding CondenseNets' counterparts in most cases. This result confirms our assumption that the model with exclusive lasso term tends to use more diverse input channels. Second, the gap between the overlap statistic of CondenseNet and

Table 2 Tiny ImageNet: Model performance of DenseNets, CondenseNets and CondenseNets-elasso

Model	MParams	MFLOPs	Top1 (%)	Top5 (%)
DenseNet-52-{8,16,32}	0.74	102	41.59	18.96
DenseNet-76-{8,16,32}	0.74	102	37.68	17.45
DenseNet-88-{8,16,32}	1.68	239	39.47	18.15
DenseNet-100-{8,16,32}	2.07	285	35.74	16.19
DenseNet-192-{12,24,48}	4.54	639	32.95	14.54
LAP-DenseNet-52-{8,16,32}	0.74	102	44.47	20.32
LAP-DenseNet-88-{8,16,32}	1.68	239	41.37	18.2
Hinge-DenseNet-28	1.07	134	41.47	18.44
Hinge-DenseNet-28-53%-pruned	0.62	63	42.02	18.68
Hinge-DenseNet-58	5.08	637	38.06	17.32
Hinge-DenseNet-58-60%-pruned	2.90	255	38.58	16.97
CondenseNet-50	0.61	63	41.74	19.79
CondenseNet-86	1.37	145	36.80	16.88
CondenseNet-122	2.40	258	34.39	15.34
CondenseNet-182	4.71	513	32.80	14.16
CondenseNet-50-elasso	0.61	63	41.17(↑0.57)	19.21(↑0.58)
CondenseNet-86-elasso	1.37	145	36.35(↑0.45)	16.25(↑0.63)
CondenseNet-122-elasso	2.40	258	34.12(↑0.27)	14.77(↑0.57)
CondenseNet-182-elasso	4.71	513	32.29(↑0.51)	13.80(↑0.36)

All results are self-implemented under the same cosine learning rate schedule and data augmentation. Results with ± are average of three runs

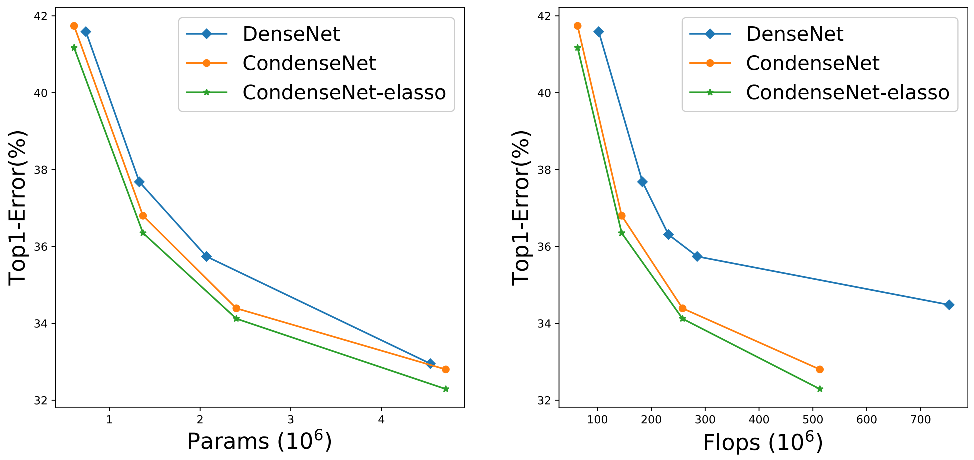

Fig. 5 Tiny ImageNet: Parameters and FLOPs comparison of DenseNets, CondenseNets and CondenseNets-elasso

CondenseNet-elasso grows larger as the model goes deeper. One possible explanation is that we are using an increasing λ in this experiment, besides, higher-level features are more diverse and may have an intrinsic grouping structure.

5.4.2 Hilbert-Schmidt independence criterion

In this section, we validate our assumption that exclusive lasso penalty encourage different convolutional groups to learn different features. We use Hilbert-Schmidt Independence Criterion (HSIC) [8, 23, 32] as a measurement of similarity. HSIC was originally proposed as a test statistics for determining whether two sets of variables are independent. Suppose X and Y are two random variables, $HSIC(X, Y) = 0$ implies $p(x, y) = p(x)p(y)$. The empirical estimator of HSIC is defined as

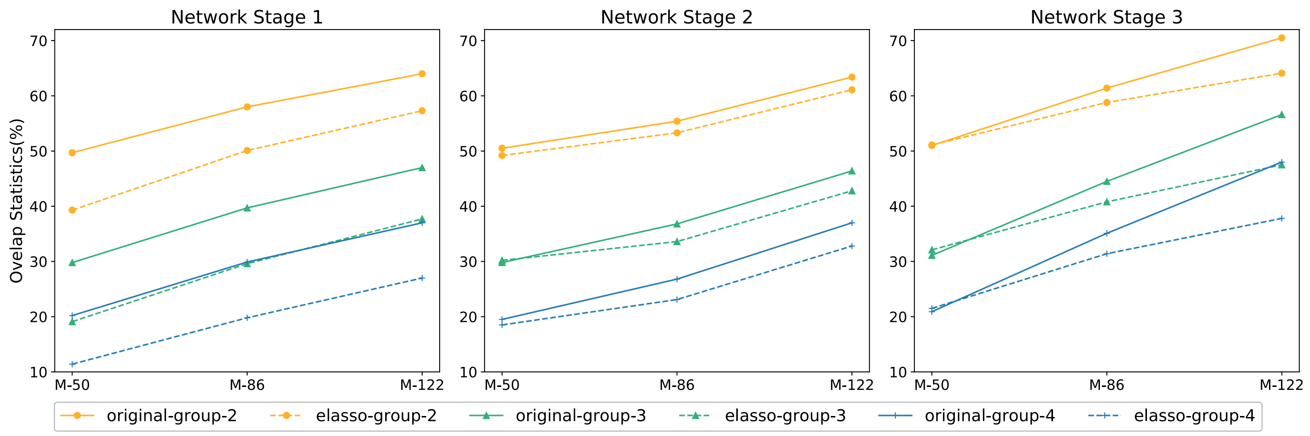

Fig. 6 CIFAR100: Overlap statistics comparison on different models. "M-50" represents models with 50 layers. "elasso-group-2" denotes average overlap statistics between 2 groups in CondenseNets-elasso

$$HSIC(K,L) = \frac{1}{(n-1)^2} tr(KHLH) \tag{9}$$

and a normalized version of HSIC is defined as

$$CKA(K,L) = \frac{HSIC(K,L)}{\sqrt{HSIC(K,K)HSIC(L,L)}} \tag{10}$$

where $H_n = I_n - \frac{1}{n}11^T$ represents the centering matrix. $K_{ij} = k(x_i, x_j)$ and $L_{ij} = l(y_i, y_j)$ represent the kernel matrices. In this experiment, we use the Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = exp(-\|\mathbf{x}_i - \mathbf{x}_j\|)_2^2/2\sigma^2$ where the bandwidth σ is set to the median distance between examples [23].

In this experiment, each output convolutional filter represents each sample, filters in different convolutional groups are in different sets. Therefore there are O samples in total, O/G samples in each set and each feature is of size R. We calculate the average HSIC statistics between different groups on all datasets. Figure 7 shows HSIC statistics between CondenseNets and CondenseNets-elasso. For example, in CIFAR-100, HSIC statistics in CondenseNet are lower than in CondenseNet-elasso, especially in the third stage of the network. This may due to the reason that λ is set in an increasing manner in three stages. Similar results can be found in CIFAR-10 and Tiny ImageNet. This observation validates our assumption that exclusive lasso encourage different groups to learn different features.

5.5 Hyperparameter

In this section, we describe how to choose hyperparameter λ. The training dataset can be split into two parts, 80% is used for training while the other 20% is used for validation. We train the model with candidate λ and choose the one with the lowest validation error. The main results (Table 1, Table 2) are trained under the entire training dataset with the chosen λ. One thing to note is that all models are trained under the same "training-validation" dataset split for a fair comparison. We choose λ in an increasing manner in three stages since features in lower layers will be quite generic while features in higher layers are more discriminative [49, 50]. This design is in line with the "increasing growth rate" in three stages. Figure 8 shows different settings of λ for CondenseNet-50-elasso on CIFAR-100. *CondenseNet-50 error* in Fig. 8 shows that our model has lower classification validation error rates even under different λ, which validates the effectiveness of our proposed method.

The chosen λ for all experiments are set as follows. On CIFAR dataset, λ is set as {1e-7,5e-7,1e-6} on all models except CondenseNet-182-elasso. Meanwhile, on Tiny ImageNet dataset, λ is set as {1e-7,5e-7,1e-6} on CondenseNet-50-elasso and is {1e-8,5e-8,1e-7} on CondenseNet-86-elasso and CondenseNet-122-elasso. λ takes value of {1e-8,5e-8,1e-7} on CondenseNet-182-elasso for both datasets. Additionally, since we only have one GPU for training, our experiment results could be improved with further fine-tuning of λ.

5.6 Group convolution comparison

In this section, we compare our model with other group convolution variants, including DenseNet with group convolution and shuffle operation, DenseNet-aggregated and Dynamic grouping convolution [55] to measure the effectiveness of our proposed method. These networks are specially designed to match parameters and FLOPs as the baseline models. Results are shown in Table 3. All experiments in this section are conducted on CIFAR100 with growth rate {8-16-32}. Model-52, Model-88 and Model-124 has {8-8-8},{14-14-14} and {20-20-20} blocks

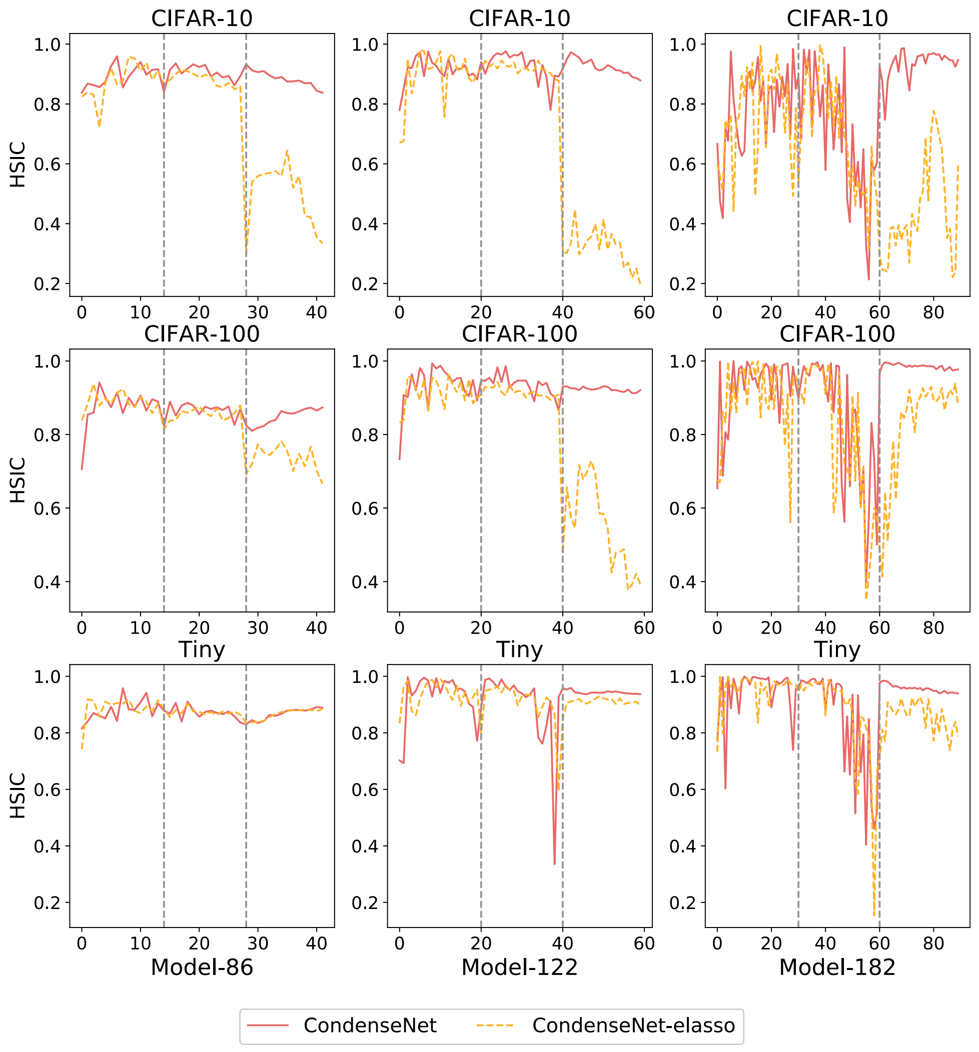

Fig. 7 HSIC statistics between CondenseNet and CondenseNet-elasso on CIFAR-10, CIFAR-100 and Tiny ImageNet

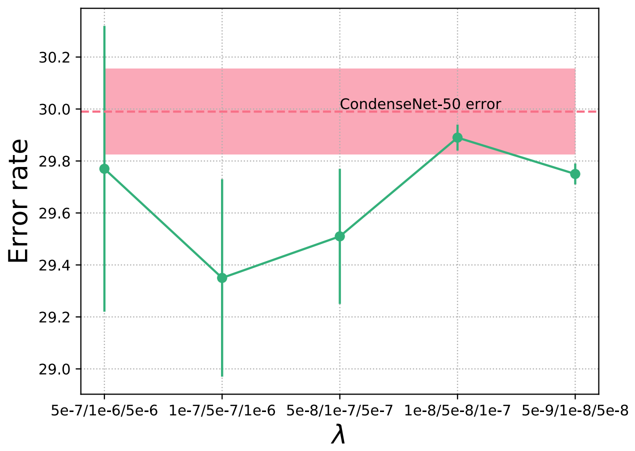

Fig. 8 CIFAR100:Choosing hyperparameter λ on CondenseNet-50-elasso. The line chart shows mean error rate of three runs with standard deviations. “CondenseNet-50 error” means CondenseNet-50 without exclusive lasso term where the translucent pink area shows its standard deviation

in three stages. For a fair comparison, all models are trained under the same training schedule and the cosine learning rate.

CondenseNet-light We use CondenseNet-light instead of CondenseNet as the baseline since it has a channel reduction layer in accordance with other listed models. λ is set as $\{1e\text{-}7,5e\text{-}7,1e\text{-}6\}$ as previously defined. Table 3 shows that CondenseNets-light-elasso drop classification error rate by 0.85%, 0.51% and 0.1%, respectively, compared with CondenseNets-light. We can see model with 122 layers does not drop the top-1 error rate as its shallower counterparts. One possible reason is that the reduction layer, which prunes out half of the filters in transition blocks, can be seen as a regularization method and weaken the effect of our proposed regularization term.

DenseNet-G-shuffle DenseNet-G replaces each 1×1 convolutional layer and the following 3×3 convolutional layer in DenseNet’s block with group convolution. The

Table 3 CIFAR100: Performance comparison of CondenseNets-light-elasso with other DenseNet group convolution counterparts

Model	MParams	MFLOPs	top1 (%)	Error (%) ↓
DenseNet-52-G	0.22	32.11	29.14	2.07
DenseNet-52-G-shuffle	0.22	32.11	28.22	1.15
DenseNet-52-aggregated	0.25	41.64	27.56	0.49
DenseNet-52-DGConv-b-4	0.51	77.85	30.71	3.64
CondenseNet-light-52	0.22	32.11	27.92	0.85
CondenseNet-light-52-elasso	0.22	32.11	27.07	–
DenseNet-88-G	0.51	74.88	25.19	2.05
DenseNet-88-G-shuffle	0.51	74.88	24.98	1.84
DenseNet-88-aggregated	0.62	97.20	24.21	1.07
DenseNet-88-DGConv-b-16	0.57	89.23	25.33	2.19
CondenseNet-light-88	0.51	74.88	23.65	0.51
CondenseNet-light-88-elasso	0.51	74.88	23.14	–
DenseNet-124-G	0.89	134.07	23.32	1.55
DenseNet-124-G-shuffle	0.89	134.07	23.28	1.51
DenseNet-124-aggregated	1.16	174.67	23.04	1.27
DenseNet-124-DGConv-b-32	1.41	238.96	22.79	1.02
CondenseNet-light-124	0.89	134.07	21.87	0.10
CondenseNet-light-124-elasso	0.89	134.07	21.77	–

The last column shows the classification error rate reduction in our proposed method with other models. In model name, the first number after "-" represents the model depth. Number of convolutional group in DenseNet-G and DenseNet-G-shuffle is set to 4. *b-16* in DenseNet-88-DGConv-b-16 means complexity hyperparameter *b* in DGConv is set to 16. Results of CondenseNets-light and CondenseNets-light-elasso are average of three runs

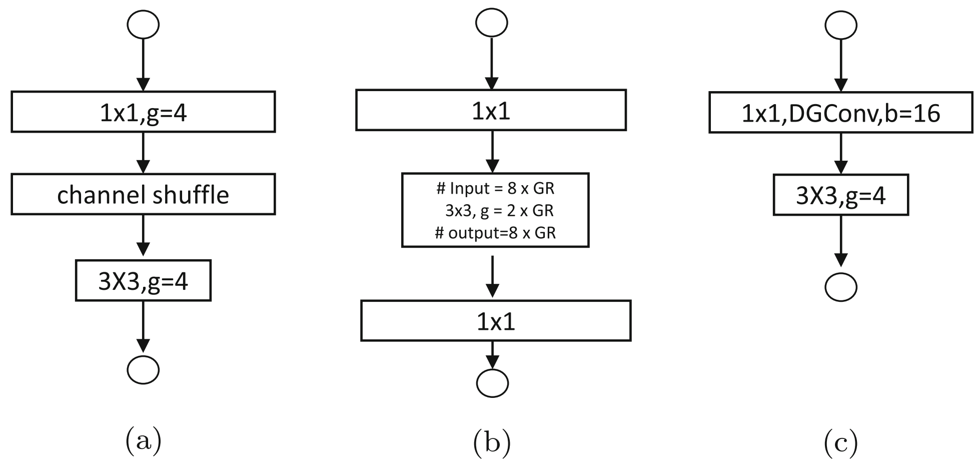

Fig. 9 The Basic building blocks of different network architectures in Sect. 5.6. **a** DenseNet-G-shuffle, g means group number. **b** DenseNet-aggregated, GR means growth rate. **c** DenseNet-DGConv, b controls model complexity

group number is set to 4 following CondenseNet. Based on DenseNet-G, we add a channel shuffle operation [54] after bottleneck layers to help information flow across different groups, the resulting model is denoted as DenseNet-G-shuffle. Figure 9a shows its basic building block. DenseNet-G-shuffle has the same architecture with the converted CondenseNet-elasso and therefore it can be seen as CondenseNet-elasso with a *pre-defined group structure* trained from scratch.

First, we find the effectiveness of shuffle operation gets decreased as model goes deeper. Results in Table 3 shows that DenseNets-G-shuffle achieves 0.92%, 0.21% and 0.04% error rate reductions compared with DenseNets-G in three models. One possible explanation is that deeper DenseNets have more fused features therefore information flow between convolutional groups becomes less effective. Second, our model achieves noticeable 1.15%, 1.84% and 1.51% classification error rate reductions at networks with

52, 88 and 124 layers, which confirms the validity of our proposed method.

DenseNet-aggregated Following ResNeXt's [47] idea, we design a "ResNeXt-like" DenseNet by setting cardinality as $2 \times growthrate$ while the input and output channel number is set to $8 \times growthrate$. This design aims to generate models with comparable parameters and flops. The basic building block is shown in Fig. 9b. Results in Table 3 show that our model achieves 0.49%, 1.07% and 1.27% lower error rates compared with DenseNet-aggregated with 0.88x, 0.82x and 0.76x parameters and about 0.77x FLOPS. The error rate gaps between these two sets of models become larger as the model goes deeper.

DenseNet-DGConv Dynamic grouping convolution (DGConv) [55] can learn optimal grouping strategy and group number in each layer automatically through learnable binary relationship matrices. To compare the performance of learned group convolution and dynamic grouping convolution, we replace bottleneck layer in DenseNet with DGConv. The resulting model is DenseNet-DGConv and the basic building block is shown in Fig. 9c.

Original DGConv is inserted into ResNeXt. Here, we make a small modification to apply DGConv to CondenseNet. Suppose the input channel number is C^l_{in} and the output channel number is C^l_{out} in lth layer. Relationship matrix U is a Kronecker product of a 2×2 matrices, therefore C^l_{in} and C^l_{out} are the power of 2 by default. In DenseNet, C^l_{out} equals $\{32, 64, 128\}$ in three stages, which meets this prerequisite. In ResNeXt, C^l_{in} equals C^l_{out} by default, while in DenseNet, $C^l_{in} \geq C^l_{out}$ in most blocks. This case is implemented by a variation of *GroupDown* method in the appendix of the paper. Specifically, suppose $C^l_{in} = r * C^l_{out} + m$, where r is an integer, we first construct $\tilde{\boldsymbol{U}}^l \in \{0,1\}^{C^l_{out} \times C^l_{out}}$. The relationship matrix $\boldsymbol{U}^l$ can be computed as:

$$\boldsymbol{U}^l = \tilde{\boldsymbol{I}}^l_d \tilde{\boldsymbol{U}}^l, \quad \tilde{\boldsymbol{I}}^l_d = [\boldsymbol{I}^l_{out}, \ldots, \boldsymbol{I}^l_{out}, \boldsymbol{I}^l_m] \tag{11}$$

where $\tilde{\boldsymbol{I}}^l_d \in \{0,1\}^{C^l_{in} \times C^l_{out}}$ is a matrix concatenated by identity matrices $\boldsymbol{I}^l_{out} \in \{0,1\}^{C^l_{out} \times C^l_{out}}$ and the remainder $\boldsymbol{I}^l_m = \boldsymbol{I}^l_{out}[: m, :, :, :]$, which truncates the first m filters in $\boldsymbol{I}^l_{out}$. When $C^l_{in} < C^l_{out}$, we use standard group convolutions with a group number of 4. Parameters and FLOPs of DenseNet-DGConv are calculated through the "gate" parameters in the final model, which is equivalent to a pruning ratio taking values from $1/2^n$ (n is a positive integer). The hyperparameter for measuring model complexity in this paper is denoted as b. We tried different model complexity settings of "b" from 2, 4, 8, 16 and 32, and pick the one with comparable parameters and FLOPs with CondenseNet-elasso. Results in Table 3 shows that CondenseNets-elasso achieves 3.64%, 2.19% and 1.02% smaller error rate compared with its DenseNet-DGConv counterparts with 0.43x, 0.89x and 0.63x parameters and 0.41x, 0.84x and 0.56x FLOPs.

5.7 Discussion

In this section, we conduct experiments to validate the effectiveness of our proposed method. First, Sect. 5.3 presents our main results on CIFAR-100, CIFAR-10 and Tiny ImageNet, as long as a Param/FLOPs comparison between different models. Section 5.5 analysis how we choose hyperparameter λ. Second, in Sect. 5.4, we validate our assumption that exclusive lasso encourage different convolutional groups to use different subsets of input channels through *overlap statistic* and learn more diversified features through *HSIC statistic*. Third, we compare our proposed method with group convolution variants in Sect. 5.6, such as the effect of the shuffle operation, increasing cardinality and dynamic grouping convolutional, which are designed to learn more efficient group convolutions. All the evaluated methods are not as efficient as CondenseNet-elasso under similar computation settings, which validates the effectiveness of our proposed method.

Our method applies to scenarios where the network backbone is DenseNet or stacks of Dense blocks, especially deep convolutional networks. Still, there may be some possible limitations in this study: our proposed approach assumes that diversified features help to boost the performance. There are some other works [4, 58] on decorrelating features in neural networks. Experiment results show that our model outperforms other reported methods with more diversified features in Figs. 7 and 6; however, this assumption needs to be further validated and explored. If this assumption holds, designing new methods to decorrelate features in neural networks can help to build compact models and saves computation.

6 Conclusion

In this paper, we insert the exclusive lasso penalty to CondenseNet to encourage different convolution groups to learn less correlated features. In our experiments, CondenseNets-elasso achieves a noticeable performance boost compared with CondenseNets and other group convolution variants under similar computation budget on three public datasets. Experiment results validate our assumption that the regularizer helps different groups to use different subsets of incoming channels and to learn more diversified features.

Appendix A: Experiment settings for compared models

LAP Lookahead pruning (LAP) [34] prunes redundant neurons or filters through a lookahead distortion by considering its neighboring layers. In this set of experiments, we use DenseNet as the baseline model and the pruned version is denoted as LAP-DenseNet. For CondenseNet-elasso, the pruning ratios for all 1×1 convolutional layers are 75% and the final fully connected layer is 50%. All 3×3 convolutional layers are group convolutional with group number 4 (except the initial convolutional layer), which is equivalent to prune 75% of filters. Therefore, we prune LAP-DenseNets under the same setting. Besides, we do not include convolutional layers inside transition blocks, same as that in CondenseNets-elasso. Training steps for pre-training and retraining are [120 k,40 k] for CIFAR and [80 k, 26 k] for Tiny ImageNet. Preliminary experiment results show that the cosine learning rate performs better than the fixed learning rate schedule, therefore, we use a cosine learning rate schedule starting from 0.1 and gradually reduces to 0. We test LAP-DenseNet with 50, 86 and 122 layers on CIFAR and LAP-DenseNet with 52 and 88 layers on Tiny ImageNet. Experiment results in Tables 1 and 2 show that our model performs better than lookahead pruning on DenseNet.

Hinge Hinge (Hinge) [28] provides a way to compress the whole network together at a given target FLOPs compression ratio. The original heavyweight convolution is converted to a lightweight convolution and a linear projection. For example, in basic building block in DenseNet (shown in Fig. 2a), one 1×1 convolution is added after each 3x3 convolution to select important output filters. Group sparsity is added by introducing a sparsity-inducing matrix, such as L_1 norm or $L_{1/2}$ norm and the matrix is optimized through proximal gradient descent. In this set of experiments, we follow the original implementation of Hinge https://github.com/ofsoundof/group_sparsity, the resulting unpruned model is denoted as Hinge-DenseNet while the pruned model is denoted as Hinge-DenseNet-pruned. The target FLOPs pruning ratio is selected to be comparable to our baseline models. The model configuration is set as follows: Hinge-DenseNet-28 has $\{8,8,8\}$ dense blocks in each stage, Hinge-DenseNet-58 has $\{18,18,18\}$ dense blocks in each stage, and the growth rate is set to $\{8,16,32\}$. The searching epochs and converging epochs are set to 200 and 300, respectively. Experiment results in Table 1 and Table 2 show that our model performs better than Hinge-DenseNet under similar computation settings.

Appendix B: Covid-19 X-ray images classification

In this section, we evaluate our proposed model on Covid-19 X-Ray images. The dataset COVID-Xray-5k is constructed from paper DeepCovid [33] whose training dataset is composed of 2000 non-covid examples and 84 covid examples while the validation dataset is composed of 3000 non-covid examples and 100 non-covid examples. [33] use the pre-trained model on ImageNet2012 and use transfer learning to fine-tune the neural networks on the training images of the COVID-Xray-5k dataset. The models predict a probability score for each image and a threshold is selected and any sample with probability higher than the threshold is considered as COVID-19. The paper uses the following two metrics to report the model performance:

$$\text{Sensitivity} = \frac{\text{Number of Images correctly predicted as COVID-19}}{\text{Number of Total COVID Images}}$$

$$\text{Specificity} = \frac{\text{Number of Images correctly predicte as Non-COVID}}{\text{Number Total Non-COVID Images}}$$

In this example, following the training schedule from DeepCovid, we first train the models on ImageNet2012 and use transfer learning to retrain the last fully connected layers from 1000 classes to classes(covid and non-covid). We train three models for comparison: DenseNet-G, CondenseNet and CondenseNet-elasso, all models have $\{4,6,8,10,8\}$ dense blocks in each stage and the growth rate is set to $\{8,16,32,64,128\}$. The pre-trained models are validated on the whole ImageNet2012 validation dataset, the result is shown in Table 4. Our model achieves 4.23% and 0.42% higher top-1 error rate compared with DenseNet-G and CondenseNet-elasso under the same computation settings. At transferring stage, we use a learning rate of 0.0005 for CondenseNet and DenseNet-G and 0.001 for DenseNet-G. The sensitivity and specificity under different threshold levels are shown in Table 5. Table 5 shows that our proposed CondeseNet-elasso achieves much higher sensitivity and comparable specificity compared with DenseNet-G and CondenseNet. Both experiment results on ImageNet2012 classification task and Covid-19 X-ray

Table 4 10%-ImageNet Classification Error rate

Model	Top-1 (%)	Top-5 (%)
DenseNet-74-G	51.13	27.12
CondenseNet-74	47.32	23.83
CondenseNet-elasso	46.90	23.54

Table 5 Sensitivity and Specificity on Covid-19 X-Ray Images

Model	Threshold	Sensitivity	Specificity
DenseNet-74-G	0.2	0.85	0.998
CondenseNet-74	0.2	0.81	1.000
CondenseNet-elasso	0.2	0.92	0.978
DenseNet-74-G	0.3	0.79	0.998
CondenseNet-74	0.3	0.77	1.000
CondenseNet-elasso	0.3	0.90	0.990
DenseNet-74-G	0.4	0.75	0.999
CondenseNet-74	0.4	0.73	1.000
CondenseNet-elasso	0.4	0.86	0.995
DenseNet-74-G	0.5	0.72	0.999
CondenseNet-74	0.5	0.67	1.000
CondenseNet-elasso	0.5	0.80	0.999

image classification show that our model performs better than its baseline CondenseNet and DenseNet-G.

Acknowledgements The authors are supported by the National Natural Science Foundation of China (Nos. 61976174, 11671317).

Declarations

Conflict of interest The authors declare that they have no conflict of interest. This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Campbell F, Allen GI et al (2017) Within group variable selection through the exclusive lasso. Electron J Statist 11(2):4220–4257
2. Changpinyo S, Sandler M, Zhmoginov A (2017) The power of sparsity in convolutional neural networks. arXiv preprint arXiv:1702.06257
3. Chen Y, Li J, Xiao H, Jin X, Yan S, Feng J (2017) Dual path networks. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems, vol 30. Curran Associates Inc, pp. 4467–4475. https://proceedings.neurips.cc/paper/2017/file/f7e0b956540676a129760a3eae309294-Paper.pdf
4. Cogswell M, Ahmed F, Girshick R, Zitnick L, Batra D (2015) Reducing overfitting in deep networks by decorrelating representations. arXiv preprint arXiv:1511.06068
5. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp. 248–255. Ieee
6. Denil M, Shakibi B, Dinh L, Ranzato M, De Freitas N (2013) Predicting parameters in deep learning. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds) Advances in neural information processing systems, vol 26, Curran Associates, Inc., pp. 2148–2156. https://proceedings.neurips.cc/paper/2013/file/7fec306d1e665bc9c748b5d2b99a6e97-Paper.pdf
7. Dong W, Wu J, Bai Z, Hu Y, Li W, Qiao W, Woźniak M (2021) Mobilegcn applied to low-dimensional node feature learning. Pattern Recognit 112:107788
8. Gretton A, Bousquet O, Smola A, Schölkopf B (2005) Measuring statistical dependence with hilbert-schmidt norms. In: International conference on algorithmic learning theory, pp. 63–77. Springer
9. Guo Q, Wu XJ, Kittler J, Feng Z (2020) Self-grouping convolutional neural networks. Neural Netw 132:491–505
10. Han S, Pool J, Tran J, Dally W (2015) Learning both weights and connections for efficient neural network. In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R (eds) Advances in neural information processing systems, vol 28. Curran Associates, Inc., pp. 1135–1143. https://proceedings.neurips.cc/paper/2015/file/ae0eb3eed39d2bcef4622b2499a05fe6-Paper.pdf
11. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp. 1026–1034
12. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778
13. He Y, Kang G, Dong X, Fu Y, Yang Y (2018) Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint arXiv:1808.06866
14. Hu H, Dey D, Del Giorno A, Hebert M, Bagnell JA (2017) Log-densenet: How to sparsify a densenet. arXiv preprint arXiv:1711.00002
15. Hu H, Peng R, Tai YW, Tang CK (2016) Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint arXiv:1607.03250
16. Huang G, Liu S, Van der Maaten L, Weinberger KQ (2018) Condensenet: An efficient densenet using learned group convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2752–2761
17. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700–4708
18. Ioannou Y, Robertson D, Cipolla R, Criminisi A (2017) Deep roots: Improving cnn efficiency with hierarchical filter groups. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1231–1240
19. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167
20. Jiang F, Grigorev A, Rho S, Tian Z, Fu Y, Jifara W, Adil K, Liu S (2018) Medical image semantic segmentation based on deep learning. Neural Comput Appl 29(5):1257–1265
21. Ke Q, Zhang J, Wei W, Połap D, Woźniak M, Kośmider L, Damaševičius R (2019) A neuro-heuristic approach for recognition of lung diseases from x-ray images. Expert Syst Appl 126:218–232
22. Kong D, Fujimaki R, Liu J, Nie F, Ding C (2014) Exclusive feature learning on arbitrary structures via $\ell_{1,2}$ -norm. Adv Neural Inf Process Syst 27:1655–1663
23. Kornblith S, Norouzi M, Lee H, Hinton G (2019) Similarity of neural network representations revisited. arXiv preprint arXiv:1905.00414
24. Krizhevsky A, Hinton G et al (2009) Learning multiple layers of features from tiny images. Tech. rep, Citeseer
25. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ (eds) Advances in neural information processing systems, vol 25. Curran Associates, Inc.,

pp. 1097–1105. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
26. Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2016) Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710
27. Li X, Chen S, Hu X, Yang J (2019) Understanding the disharmony between dropout and batch normalization by variance shift. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2682–2690
28. Li Y, Gu S, Mayer C, Gool LV, Timofte R (2020) Group sparsity: The hinge between filter pruning and decomposition for network compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8018–8027
29. Long J, Shelhamer E, Darrell T(2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440
30. Loshchilov I, Hutter F(2016) Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983
31. Luo JH, Wu J, Lin W (2017) Thinet: A filter level pruning method for deep neural network compression. In: Proceedings of the IEEE international conference on computer vision, pp. 5058–5066
32. Ma KWD, Lewis J, Kleijn WB (2020) The hsic bottleneck: Deep learning without back-propagation. Proc AAAI Conf Artif Intell 34(4):5085–5092. https://doi.org/10.1609/aaai.v34i04.5950
33. Minaee S, Kafieh R, Sonka M, Yazdani S, Soufi GJ (2020) Deep-covid: Predicting covid-19 from chest x-ray images using deep transfer learning. Med Image Anal 65:101794
34. Park S, Lee J, Mo S, Shin J (2020) Lookahead: A far-sighted alternative of magnitude-based pruning. In: International Conference on Learning Representations. https://openreview.net/forum?id=ryl3ygHYDB
35. Peng, B., Tan, W., Li, Z., Zhang, S., Xie, D., Pu, S.: Extreme network compression via filter group approximation. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 300–316 (2018)
36. Pleiss G, Chen D, Huang G, Li T, van der Maaten L, Weinberger KQ (2017) Memory-efficient implementation of densenets. arXiv preprint arXiv:1707.06990
37. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788
38. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. In: Cortes C, Lawrence N, Lee D, Sugiyama M, Garnett R (eds) Advances in neural information processing systems, vol 28. Curran Associates, Inc., pp. 91–99. https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf
39. Scardapane S, Comminiello D, Hussain A, Uncini A (2017) Group sparse regularization for deep neural networks. Neurocomputing 241:81–89
40. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
41. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
42. Wan L, Zeiler M, Zhang S, Le Cun Y, Fergus R (2013) Regularization of neural networks using dropconnect. In: International Conference on Machine Learning, pp. 1058–1066
43. Wang W, Li X, Yang J, Lu T (2018) Mixed link networks. arXiv preprint arXiv:1802.01808
44. Wang X, Kan M, Shan S, Chen X (2019) Fully learnable group convolution for acceleration of deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 9049–9058
45. Wen W, Wu C, Wang Y, Chen Y, Li H (2016) Learning structured sparsity in deep neural networks. In: Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R (eds) Advances in neural information processing systems, vol 29. Curran Associates, Inc., pp. 2074–2082. https://proceedings.neurips.cc/paper/2016/file/41bfd20a38bb1b0bec75acf0845530a7-Paper.pdf
46. Woźniak M, Wieczorek M, Siłka J, Połap D (2020) Body pose prediction based on motion sensor data and recurrent neural network. IEEE Trans Ind Inf 17(3):2101–2111
47. Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1492–1500
48. Yang Y, Zhong Z, Shen T, Lin Z (2018) Convolutional neural networks with alternately updated clique. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2413–2422
49. Ye J, Lu X, Lin Z, Wang JZ (2018) Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. arXiv preprint arXiv:1802.00124
50. Yoon J, Hwang SJ (2017) Combined group and exclusive sparsity for deep neural networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70, pp. 3958–3966. JMLR. org
51. Yu R, Li A, Chen CF, Lai JH, Morariu VI, Han X, Gao M, Lin CY, Davis LS (2018) Nisp: Pruning networks using neuron importance score propagation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 9194–9203
52. Zhang D, Wang H, Figueiredo M, Balzano L (2018) Learning to share: Simultaneous parameter tying and sparsification in deep learning. In: International Conference on Learning Representations. https://openreview.net/forum?id=rypT3fb0b
53. Zhang T, Qi GJ, Xiao B, Wang J (2017) Interleaved group convolutions. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4373–4382
54. Zhang X, Zhou X, Lin M, Sun J (2018) Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6848–6856
55. Zhang Z, Li J, Shao W, Peng Z, Zhang R, Wang X, Luo P (2019) Differentiable learning-to-group channels via groupable convolutional neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3542–3551
56. Zhou Y, Jin R, Hoi SCH (2010) Exclusive lasso for multi-task feature selection. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pp. 988–995
57. Zhu L, Deng R, Maire M, Deng Z, Mori G, Tan P (2018) Sparsely aggregated convolutional networks. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 186–201
58. Zhu X, Zhou W, Li H (2018) Improving deep neural network sparsity through decorrelation regularization. In: International Joint Conference on Artificial Intelligence, pp 3264–3270