Automatic Pruning via Structured Lasso with Class-wise Information

Xiang Liu[1,†], **Mingchen Li**[2,†], **Xia Li**[3,†], **Leigang Qu**[1], **Zifan Peng**[4],
Yijun Song[5], **Zemin Liu**[6], **Linshan Jiang**[1,†] and **Jialin Li**[1]

[1]National University of Singapore, [2] University of North Texas,
[3]Department of Information Technology and Electrical Engineering, ETH Zürich
[4] Hong Kong University of Science and Technology (Guangzhou),
[5]Dongfang College, Zhejiang University of Finance & Economics
[6]College of Computer Science and Technology, Zhejiang University
† Equal Contribution, † Corresponding Authors
{liuxiang,lijl}@comp.nus.edu.sg, mingchenli@my.unt.edu, xiaxiali@ethz.ch,
leigangqu@gmail.com, zpengao@connect.hkust-gz.edu.cn, yijunsong.0377@gmail.com,
liu.zemin@hotmail.com, linshan@nus.edu.sg

Abstract

Most pruning methods concentrate on unimportant filters of neural networks. However, they face the loss of statistical information due to a lack of consideration for class-wise data. In this paper, from the perspective of leveraging precise class-wise information for model pruning, we utilize structured lasso with guidance from Information Bottleneck theory. Our approach ensures that statistical information is retained during the pruning process. With these techniques, we introduce two innovative adaptive network pruning schemes: sparse graph-structured lasso pruning with Information Bottleneck (**sGLP-IB**) and sparse tree-guided lasso pruning with Information Bottleneck (**sTLP-IB**). The key aspect is pruning model filters using sGLP-IB and sTLP-IB to better capture class-wise relatedness. Compared to multiple state-of-the-art methods, our approaches demonstrate superior performance across three datasets and six model architectures in extensive experiments. For instance, using the VGG16 model on the CIFAR-10 dataset, we achieve a parameter reduction of 85%, a decrease in FLOPs by 61%, and maintain an accuracy of 94.10% (0.14% higher than the original model); we reduce the parameters by 55% with the accuracy at 76.12% using the ResNet architecture on ImageNet (only drops 0.03%). In summary, we successfully reduce model size and computational resource usage while maintaining accuracy. Our codes are at https://anonymous.4open.science/r/IJCAI-8104.

1 Introduction

Deep Learning [Krizhevsky *et al.*, 2012], especially CNN-based methods [Simonyan and Zisserman, 2014; Szegedy *et al.*, 2015; He *et al.*, 2016], has proven its effectiveness in tasks like object detection and classification. Recently, researchers have been attempting to deploy neural network models on resource-constrained edge devices for real-time analysis [Liu *et al.*, 2024c; Yu *et al.*, 2024]. However, these network parameters are often too large, making deployment difficult. For instance, the VGGNet-16 model has parameters exceeding 500MB [Simonyan and Zisserman, 2014], which introduces a heavy memory burden and computation overhead.

Fortunately, scientists have discovered that the parameters are usually redundant and the key weights of neural networks exhibit sparsity characteristics [He *et al.*, 2017]. As a result, a significant amount of work has focused on pruning unimportant weights to reduce the size of the model, which reduces the model's FLOPs and lowers computational overhead as well. The common methods can be primarily divided into two categories: unstructured pruning and structured pruning [Shao and Shin, 2022]. Unstructured pruning mainly focuses on weight pruning [Ding *et al.*, 2019] and N:M sparsity [Zhang *et al.*, 2022]. However, both methods require hardware support, making them less scalable. Therefore, this paper primarily focuses on structured pruning, which benefits from removing unimportant filters within the CNN.

To avoid the limitations of hand-crafted approaches, most structured pruning methods utilize statistical information [Zhang *et al.*, 2021; Barisin and Horenko, 2024] to analyze filters, with the aim of minimizing the loss before and after pruning and achieving better accuracy results. However, these methods all overlook the class-wise information [Hu *et al.*, 2016], resulting in a less precise pruning scheme. Furthermore, an ideal pruning method should also include an adaptive approach [Guo *et al.*, 2023; Xie *et al.*, 2024] to control the level of sparsity based on specific requirements.

Based on these, we employ the precise Information Bottleneck (IB) [Tishby *et al.*, 2000] theory perspective to model statistical information of feature maps of CNN for pruning, aiming to minimize the loss from pruning. We also utilize structured lasso techniques that account for sparsity, such as graph-structured lasso [Liu *et al.*, 2024b] and tree-guided lasso [Kim and Xing, 2009; Liu *et al.*, 2024a], which can consider the relatedness in feature maps and form corresponding
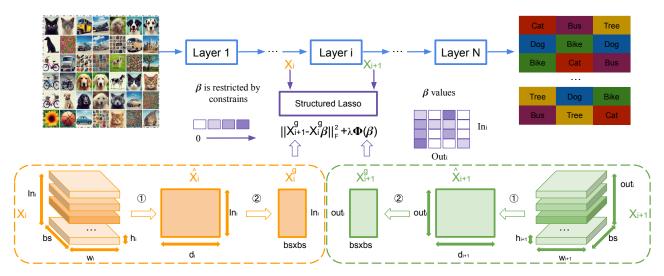
Figure 1: The overview of our methods. To utilize information bottleneck theory, ① reshape the input and output feature maps. ② transforms the reshaped feature maps with Gram matrix. Then, we utilize the structured lasso to account for the class-wise information to get the linkage matrix and prune the filters.

sub-groups and sub-trees. This approach helps to incorporate class-wise information. Consequently, we propose two novel methods: Sparse Graph-structured Lasso Pruning with Information Bottleneck (**sGLP-IB**) and Sparse Tree-guided Lasso Pruning with Information Bottleneck (**sTLP-IB**). Our methods perform pruning at the layer level rather than end-to-end, ensuring more precise pruning filters and preventing cumulative errors layer by layer [Sakamoto and Sato, 2024]. Based on this, our achievements have already been surprising as we implement structured lasso into pruning. Further, we (1) combine Gram matrix [Lanckriet *et al.*, 2002] to reduce the dimensions and accelerate computation without compromising accuracy, (2) propose an adaptive method to ensure that our method can be automatically suitable for varying sparsity requirements rather than setting hyperparameters based on experiences, (3) use novel optimization method which has polynomial time complexity with guaranteed convergence rate. Our main contributions are summarized as follows:

- We are the first to discover pruning with structured lasso from the information bottleneck perspective via precise class-wise information. By enhancing techniques such as graph-structured lasso and tree-guided lasso with information bottleneck, we propose two novel methods: sGLP-IB and sTLP-IB.

- We have optimized our proposed methods, sGLP-IB and sTLP-IB, to ensure faster computation using Gram matrix and adaptively address different model sizes or FLOPs requirements with our proposed algorithm. These optimizations still guarantee convergence and polynomial computation time complexity using the proximal gradient descent method.

- We conduct comparative experiments against numerous baselines using two real-world datasets, CIFAR-10, CIFAR-100 and ImageNet. We tested multiple variants of VGGNet, ResNet, and GoogleNet. The experimental results demonstrate the effectiveness of our meth-

ods across three metrics: accuracy, memory size, and FLOPs. They both achieve state-of-the-art results, while sTLP-IB demonstrates a stronger ability to learn class information compared to sGLP-IB. Extensive ablation studies further confirm the robustness of our methods and the potential of utilizing class-wise information.

2 Related Works

Structured Pruning

MPF [He *et al.*, 2019] considers the geometric distance between filters and neighboring filters to guide the pruning process. CPMC [Yan *et al.*, 2021] leverages relationships between filters in the current layer and subsequent layers for pruning decisions. HRank [Lin *et al.*, 2020] compresses models by constructing low-rank matrices using information from the same batch. Scop [Tang *et al.*, 2020] reduces filters based on their deviation from the expected network behavior. GDP [Guo *et al.*, 2021] employs gates with differentiable polarization, learning whether gates are zero during training for pruning purposes. EEMC [Zhang *et al.*, 2021] uses a multivariate Bernoulli distribution along with stochastic gradient Hamiltonian Monte Carlo for pruning. SRR-GR [Wang *et al.*, 2021] identifies redundant structures within CNNs. FTWT [Elkerdawy *et al.*, 2022] predicts pruning strategies using a self-supervised mask. DECORE [Alwani *et al.*, 2022] assigns an agent to each filter and uses lightweight learning to decide whether to keep or discard each filter. EPruner [Lin *et al.*, 2021] employs Affinity Propagation for efficient pruning. DPFPS [Ruan *et al.*, 2021] directly learns the network with structural sparsity for pruning. CC [Li *et al.*, 2021] combines tensor decomposition for filter pruning. NPPM [Gao *et al.*, 2021] trains the network to predict the performance of the pruned model, guiding the pruning process. LRF-60 [Joo *et al.*, 2021] uses Linearly Replaceable Filters to aid pruning decisions. AutoBot [Castells and Yeom, 2021] tests each filter one-by-one to ensure pruning precision. PGMPF [Cai

et al., 2022] utilizes prior masks as the basis for pruning masks. DLRFC [He *et al.*, 2022] uses Receptive Field Criterion to measure filter importance. FSM [Duan *et al.*, 2022] aggregates feature and filter information to evaluate their relevance for shifting. Rrandom [Li *et al.*, 2022] employs random search strategies for pruning. The study [Hussien *et al.*, 2024] employs activation statistics, foundations in information theory, and statistical analysis with designed regularizations for pruning. CSD [Xie *et al.*, 2024] uses channel spatial dependability metric and leverages feature characteristics for pruning. However, the statistical learning information considered by these methods may not be precise enough.

Information Bottleneck and Lasso Principle

The Information Bottleneck (IB) [Tishby *et al.*, 2000] method extracts the most relevant output information by considering the input. Aside from the previous mentioned methods, the works [Tishby and Zaslavsky, 2015; Dai *et al.*, 2018] were the first to use IB for compressing CNNs. FPGM [He *et al.*, 2018] employs the geometric median for pruning. L1 [Li *et al.*, 2016] illustrates the correspondence between Lasso and IB for pruning. The approach in [He *et al.*, 2017] utilizes a two-step optimization process incorporating Lasso for pruning. Papers [Scardapane *et al.*, 2016; Scardapane *et al.*, 2016] leverage the l_2 lasso (e.g., group lasso) for pruning, with analyzed complexity [Rajaraman *et al.*, 2023]. APIB [Guo *et al.*, 2023] integrates IB with the Hilbert-Schmidt Independence Criterion Lasso for exploration. The paper [Sakamoto and Sato, 2024] also shows through analysis that layer-level pruning, based on the Hilbert-Schmidt independence criterion, is preferable to end-to-end pruning.

However, none of these methods account for class-wise information. To address this, from the information bottleneck perspective, we employ state-of-the-art techniques like graph-structured lasso [Liu *et al.*, 2024b] and tree-guided lasso [Liu *et al.*, 2024a] to aggregate class-wise information effectively to propose our novel methods and prune filters inside CNNs.

3 Our Method

3.1 Design Overview

The overview of our methods is shown in Figure 1. To avoid the imprecision that end-to-end methods might introduce, we perform layer-wise pruning independently [Sakamoto and Sato, 2024]. We reshape each layer's input and output feature map tensors into two-dimensional matrices, enabling us to apply the information bottleneck theory to determine which filters can be pruned. To further reduce computational complexity, capture internal geometric structures, and utilize more flexible kernel functions, we employ Gram matrix [Lanckriet *et al.*, 2002]. Since lasso aligns well with the information bottleneck theory [Tishby and Zaslavsky, 2015; Dai *et al.*, 2018], we use structured lasso for solving this problem. This approach also takes into account class-wise information in the feature maps (as well as filters), resulting in a sparse linkage map between layers' feature maps for pruning.

Math Notations: In this paper, given a CNN model M with N layers, we let C_i denote the i-th layer of M. The input feature map of C_i is X_i with the shape (bs, h_i, w_i, in_i);

the output feature map of C_i is X_{i+1} with the shape $(bs, h_{i+1}, w_{i+1}, out_i)$, where $i \in \{1, 2, ..., N\}$, X_1 is the input sample batch from the datasets, X_{N+1} is the final output results, X_{i+1} is the input of C_{i+1}; for layer i, bs is the batch size, h_i and w_i are the height and width of each feature map, in_i is the input channel, out_i is the output channel (filter number of C_i). We convert X_i from the original shape to $\hat{X}_i$, $(bs \times h_i \times w_i, in_i)$, i.e., (d_i, in_i). Thus, the shape of $\hat{X}_{i+1}$ is $(bs \times h_{i+1} \times w_{i+1}, out_i)$, i.e., (d_{i+1}, out_i). Therefore, we use β to denote the linkage matrix of C_i among X_i and X_{i+1}, β_j denotes the j-th column β.

3.2 Structured Lasso

Information Bottleneck Function and Gram matrix

For each layer C_i, we optimize on the Information Bottleneck (IB) formular [Tishby *et al.*, 2000] as:

$$\min_{\tilde{X_{i+1}}} -I(\tilde{X_{i+1}}; X_i) + \gamma I(X_{i+1}; \tilde{X_{i+1}}) \qquad (1)$$

where $\tilde{X_{i+1}}$ denotes the pruned ouput features and γ is a positive Largrange multiplier. Output feature maps are derived from input feature maps through filters and corresponding convolution operations. As such, important filters result in output feature maps that contain more information from the corresponding input feature maps. To better utilize the IB theory, we first reshaped X_i and X_{i+1} into two-dimensional matrices, denoted as $\hat{X}_i$ and $\hat{X_{i+1}}$. Therefore, the problem of pruning filters is transformed into the task of exploring to preserve output feature channels that contain the most statistical information with respect to the input feature maps using the Information Bottleneck approach.

$$\min_{\tilde{X_{i+1}}} -I(\tilde{X_{i+1}}; \hat{X}_i) + \gamma I(\hat{X_{i+1}}; \tilde{X_{i+1}}) \qquad (2)$$

Gram matrix allows for the use of more flexible kernel functions, enabling linear operations in high-dimensional spaces, thereby reducing computational complexity and better considering the geometric structure between feature maps. To account for class-wise information expansion between feature maps, we employ Gram matrix. We transform $\hat{X}_i$ and $\hat{X_{i+1}}$ into centered Gram matrices X_i^g and X_{i+1}^g. Therefore, we have the optimization formula as:

$$\min_{\tilde{X_{i+1}}} -I(\tilde{X_{i+1}}; X_i^g) + \gamma I(X_{i+1}^g; \tilde{X_{i+1}}) \qquad (3)$$

where $X_i^g = \Psi X_i^k \Psi$, $\Psi = I_{bs} - \frac{1}{bs} 1_{bs} 1_{bs}^T$, I_{bs} is a bs-dimensional identity matrix, 1_{bs} is a bs-dimensional vector consisting entirely of ones. The element in row a and column b of X_i^k is equal to the inner product of a-th feature map and b-th feature map of X_i. Therefore, we have $X_i^g \in R^{bs \times bs, in_i}, X_{i+1}^g \in R^{bs \times bs, out_i}, \beta \in R^{in_i, out_i}$.

The first step of reshaping is intended to capture the relationship between input features and output features from the channel perspective. The second step is to utilize Gram matrix. In practice, we can directly transform X_i to X_i^g.

Graph-structured Lasso and sGLP-IB

The computation of the IB method is often considered computationally infeasible due to the significant overhead introduced by density estimation in high-dimensional spaces [Ma *et al.*, 2020]. We are able to use the Lasso-based methods to get the nonlinear dependencies between the input and output of each layer [Li *et al.*, 2016; Guo *et al.*, 2023].

$$\beta = \min_{\beta} \frac{1}{2}||X_{i+1}^g - X_i^g\beta||_F^2 + \Phi(\beta) \qquad (4)$$

where $|| \cdot ||_F$ denotes the Frobenius norm, and Φ is determined by structured lasso. Due to the fact that filters generally contain class-wise information [Hu *et al.*, 2016]. Filters are tailored to different classes, meaning they also possess distinct statistical properties. As a result, it is possible to form corresponding subgroups based on class information. Thus, the output feature maps also conceal class-wise information due to the filters. This results in a complex relationship between output feature maps, especially in terms of statistical subgroups. Therefore, we introduce graph-structured lasso from structured lasso to better model this relationship.

Given β for C_i, we have the node sets V, $\{\beta_1, \beta_2, ...\beta_{out_i}\}$ that forms the graph, where β_j^r denotes element in j-th column and r-th row of β. We have weight edge set E of the graph by computing pairwise Pearson correlations and linking pair of nodes when their correlation exceeds a threshold th, where the weight of each edge indicates correlation degree. Let f_{lm} represent the weight of edge $e = (l,m) \in E$, which quantifies the correlation between element l and m. With these, we define graph-structured lasso as:

$$\Phi(\beta) = \lambda||\beta||_1 + \mu \sum_{e=(l,m)\in E} |f_{lm}| \sum_{j=1}^{in_i} |\beta_l^j - sign(f_{lm})\beta_m^j| \qquad (5)$$

where λ is the hyperparameter for the sparsity and μ regulates inner class-wise dependency. Substituting (5) into (4) yields our sparse graph-structured lasso pruning with Information Bottleneck (sGLP-IB).

As shown in the β matrix in Figure 1, each column of β represents the relatedness between all feature maps of this batch for a specific output channel and the current layer's input feature maps. Our method can consider output feature maps and β, forming subgroups according to the classes. Within each subgroup, we identify and select the filters that contribute the most. The larger the value of elements in β, the stronger the correlation. Therefore, we can determine whether to retain a filter based on the number of non-zero values in its corresponding column.

Tree-Guided Lasso and sTLP-IB

We also apply the state-of-the-art tree-guided lasso penalty to model the more closely related class-wise information and then form the tree structures. The tree numbers and the overlap among groups are determined by the hierarchical clustering tree due to class information. We have:

$$\Phi(\beta) = \lambda \sum_j W_j(v_{root}) = \lambda \sum_j \sum_{v\in V} w_{ve}||\beta_j^{G_v}||_2 \qquad (6)$$

Where each group of j-th subtree regression factors $\beta_j^{G_v}$ is weighted by w_{ve}, as defined in (7), where A is the ancestor set. $\beta_j^{G_v}$ is a vector and represents regression factors $\{\beta_j^k \mid k \in G_v\}$. Each node $v \in V$ of one tree is associated with a group G_v, whose members are the columns of β within the nodes of the same subtree.

$$w_{ve} = \begin{cases} (1-h_v)\prod_{q\in A(v)} h_q & v \in \text{internal node} \\ \prod_{q\in A(v)} h_q & v \in \text{leaf node} \end{cases} \qquad (7)$$

Generally, h_v represents the weight for selecting covariates relevant specifically to the column of β linked to each child of node v, while $1-h_v$ is the weight for jointly selecting covariates relevant to all children of node v, $h_v \in (0,1)$. Assuming $|V|$ represents the number of nodes in a tree, since a tree associated with out_i columns of β for C_i has $2out_i - 1$ nodes, $|V|$ in the tree-lasso penalty is upper-bounded by $2out_i$.

Substituting (6) into (4) yields our sparse tree-guided lasso pruning with Information Bottleneck (sTLP-IB). Similarly, within each subtree formed by classes, we select the filters that could convey more information.

3.3 Optimization with Proximal Gradient Descent

With smoothing proximal gradient descent method [Chen *et al.*, 2012; Yuan, 2024] we implemented, we could easily apply into (4) and yield the β results for each layer, which also guarantee the convergence rate of $O(\frac{1}{\epsilon})$ with polynomial time complexity, where ϵ is the differences between the yielded β and the optimal β. We then prune the model M based on the β obtained from sGLP-IB and sTLP-IB.

Algorithm 1 Adaptive Method for Pruning

Input: Training Dataset Batch, Pretrained Model M with N layers, compressed memory size/FLOPs range $\{E_l, E_u\}$
Parameter: λ_l, λ_u
Output: Pruned Model M^*

1: **for** i in N **do**
2:    $\lambda_{lt} = \lambda_l, \lambda_{ut} = \lambda_u, C_i \leftarrow M$.
3:    $flag = True$.
4:    **while** $flag$ **do**
5:      $\lambda = log(\frac{1}{2}(exp(\lambda_{lt}) + exp(\lambda_{ut})))$.
6:      $C_i^* =$ sGLP-IB(C_i, λ) or sTLP-IB(C_i, λ).
7:      **if** $size(C_i^*) \leq E_l$ **then**
8:        $\lambda_{ut} = \lambda$.
9:      **else if** $size(C_i^*) \geq E_u$ **then**
10:     $\lambda_{lt} = \lambda$.
11:     **else**
12:       $flag = False$.
13:     **end if**
14:    **end while**
15:    $M^* \leftarrow C_i*$.
16: **end for**
17: **return** M^*.

Method	Top-1(%)	FLOPs↓	Params ↓	Δ Acc(%)
VGGNet-16	93.96	0(314.57M)	0(14.98M)	0
CPMC	93.40	66%	-	-0.56
L1	93.41	34%	64%	-0.55
HRank	93.42	54%	83%	-0.54
FPGM	93.50	36%	-	-0.46
PGMPF	93.60	66%	-	-0.36
EEMC	93.63	56%	-	-0.33
CSD	93.69	45%	48%	-0.27
sGLP-IB	**93.98**	**65%**	**91%**	**+0.02**
GDP	93.99	31%	-	+0.03
APIB	94.00	66%	78%	+0.04
sTLP-IB	**94.01**	**64%**	**91%**	**+0.05**
Decore	94.02	35%	63%	+0.06
NORTON	94.08	60%	83%	+0.12
AutoBot	94.10	54%	50%	+0.14
sGLP-IB	**94.05**	**61%**	**85%**	**+0.09**
sTLP-IB	**94.10**	**61%**	**85%**	**+0.14**
ResNet-56	93.28	0(0.853M)	0(127.62M)	0
DECORE	90.85	81%	85%	-2.43
FTWT	92.63	66%	-	-0.65
FSM	92.76	68%	68%	-0.52
Hrank	93.17	50%	42%	-0.11
DLRFC	93.57	53%	55%	+0.29
SRR-GR	93.75	54%	-	+0.47
CSD	93.89	45%	40%	+0.61
APIB	93.92	47%	51%	+0.64
AutoBot	93.94	50%	44%	+0.66
NORTON	94.00	42%	48%	+0.72
sGLP-IB	**93.93**	**53%**	**52%**	**+0.65**
sTLP-IB	**93.96**	**53%**	**52%**	**+0.68**
ResNet-110	93.50	0(256.04M)	0(1.73M)	0
DECORE	92.71	77%	80%	-0.79
Hrank	93.36	58%	59%	-0.14
APIB	93.37	77%	82%	-0.13
MPF	93.38	63%	-	-0.12
DECORE	93.50	61%	64%	0
EPruner	93.62	65%	76%	+0.12
NORTON	93.68	66%	68%	+0.18
sGLP-IB	**93.69**	**64%**	**68%**	**+0.19**
sTLP-IB	**93.80**	**62%**	**68%**	**+0.30**
GoogleNet	95.05	0(1531.98M)	0(6.17M)	0
Hrank	94.53	55%	55%	-0.52
FSM	94.72	63%	56%	-0.33
EPruner	94.90	63%	66%	-0.15
sGLP-IB	**94.75**	**69%**	**63%**	**-0.30**
sTLP-IB	**94.78**	**69%**	**63%**	**-0.27**

Table 1: Pruning results of VGG-16, ResNet-56, ResNet-110 and GoogleNet on CIFAR-10 dataset.

3.4 Adaptive Algorithm

We illustrate how our approach utilizes sGLP-IB and sTLP-IB to prune networks. We observe that they include a hyperparameter λ, which greatly influences the sparsity of β. If λ is too small, each element of β is a non-zero value, making the pruning ineffective; if λ is too large, β may contain too few non-zero values, making the pruning impossible as they will prune all the filters in one layer. Given that models have many layers, setting λ for each layer can be very time-consuming. Therefore, we propose an adaptive method to selectively prune based on memory size and FLOPs requirements as shown in Algorithm 1.

In the adaptive method, we use a binary search approach. We set an upper and lower limit for the value of λ and use exponential mapping to quickly find an appropriate λ that fits the sparsity level of the structured Lasso. Given a range of the proportion of parameters to retain or a range of FLOPs to retain, our algorithm can adaptively find suitable λ values for

each layer to ensure that only a specific number of columns has non-zero values in β and meet the requirements. This method ensures a relatively balanced number of parameters across layers, preventing any layer from having too many or too few parameters. It also allows for the setting of different parameter numbers and FLOP retention ratios for each layer.

4 Experiments

To demonstrate the effectiveness of our sGLP-IB and sTLP-IB, we conduct extensive experiments based on many representative CNNs, including various variants of VGGNet, ResNet, and GoogleNet, using three widely used datasets: CIFAR-10, CIFAR-100, and ImageNet. We also perform numerous ablation studies to analyze the robustness and characteristics of our methods.

Implementation Details: All codes are implemented using Pytorch. We use the SGD optimizer with a decaying learning rate initialized to $1e$-4, and we set the batch size to 128. After pruning, we finetune VGGNet, ResNet and GoogleNet for 400 to 500 epochs. On ImageNet datasets, we train VGGNet and ResNet for 150 to 200 epochs. We choose the Laplacian kernel as our base kernel function for Gram matrix. We prune without the first four layers.

Evaluation Metric: We evaluate the pruned model based on its Top-1 test accuracy, the pruned ratio of parameters, the reduced FLOPs ratio with the original pretrained model.

Baselines: Our extensive baselines are primarily based on the aforementioned structured pruning methods in Section 2. Besides, we include NORTON [Pham *et al.*, 2024a; Pham *et al.*, 2024b], which represents state-of-the-art unstructured pruning methods through the use of tensor decomposition.

4.1 Experiments on CIFAR-10

VGG-16. Table 1 presents our experiments on CIFAR-10 using VGGNet-16. Our methods, sGLP-IB and sTLP-IB, demonstrate strong performance, achieving Top-1 accuracy rates of 94.05% and 94.10%, and pruning ratios of 61% for FLOPs and 85% for parameters. sTLP-IB achieves the highest Top-1 accuracy among all previous baselines but with lower computational overhead and memory requirements, which is 0.14% higher than the pretrained model and demonstrates the effectiveness of utilizing class-wise information. Although sGLP-IB's Top-1 accuracy is slightly lower than AutoBot by 0.05%, it achieves greater reductions in both FLOPs and parameters. We also present additional results based on varying pruning ratios, showing that our methods remain superior to most baselines even at higher pruning levels. This proves the effectiveness of using graph-structured lasso and tree-guided lasso, which utilize class-wise information, in pruning. Notably, sTLP-IB performs better than sGLP-IB, as the tree-structured approach, which constructs subtrees, captures class information more precisely than graphs.

ResNet-56. Table 1 also shows that our methods, sGLP-IB and sTLP-IB, outperform most baselines using ResNet-56 with Top-1 accuracy rates of 93.93% and 93.96%, respectively, while maintaining the same pruning ratios of 53% for FLOPs and 52% for parameters. sTLP-IB outperforms sGLP-IB, consistent with the results observed with VGGNet-16. Although NORTON slightly surpasses our methods by 0.04%,

Method	Top-1(%)	FLOPs↓	Params ↓	Δ Acc(%)
ResNet-50	76.15	0(4133.74M)	0(25.56M)	0
Hrank	71.98	62%	62%	-4.17
DECORE	72.06	61%	-	-4.09
MFP	74.86	54%	-	-1.29
Random	75.13	49%	54%	-1.02
SCOP	75.26	55%	-	-0.89
APIB	75.37	62%	58%	-0.78
DPFPS	75.55	46%	-	-0.60
CC	75.59	53%	-	-0.56
LRF-60	75.71	56%	-	-0.44
NORTON	75.95	64%	59%	-0.20
NPPM	75.96	56%	-	-0.19
CSD	76.11	54%	41%	-0.04
sGLP-IB	**76.10**	**57%**	**55%**	**-0.05**
sTLP-IB	**76.12**	**57%**	**55%**	**-0.03**

Table 2: Pruning results of ResNet-50 on ImageNet dataset.

its pruning ratios were lower, with only 42% for FLOPs and 48% for parameters. Other methods like DECORE, FTWT, and FSW exhibit higher pruning ratios than SGLP-IB and STLP-IB, but they compromise Top-1 accuracy. Therefore, our methods achieve a balance between accuracy and pruning ratios, improving upon the pretrained model by 0.68%.

Method	64	128	256	512	1024
sGLP-IB	93.77	94.05	94.07	94.08	94.08
sTLP-IB	93.76	94.10	94.11	94.10	94.11

Table 3: Top-1 Accuracy of our methods using VGGNet-16 on CIFAR-10 dataset when varying the batch sizes.

ResNet-110. Both sGLPIB and sTLP-IB achieve state-of-the-art performance on ResNet-110. sTLP-IB behaves better with a Top-1 accuracy of 93.80%, surpassing the best baseline by 0.12%, while reducing 62% of FLOPs and 68% of parameters as Table 1, showing an improvement of 0.30% over the unpruned model. Notably, even though we choose the same pruning ratios, sGLP-IB eliminates more FLOPs and also surpasses all baselines with an accuracy of 93.69%, albeit 0.11% lower than sTLP-IB. This further underscores the effectiveness of the tree-guided lasso, which retains more critical filters, explaining the FLOPs difference compared to sGLP-IB. Our experiments on ResNet-56 and ResNet-110 validate the efficacy of our two pruning methods on ResNet architectures.

GoogleNet. On GoogleNet, our methods also demonstrate effectiveness, with sGLP-IB reaching 94.75% and sTLP-IB achieving a test accuracy of 94.78%. Despite removing 69% of FLOPs and 63% of parameters, we significantly reduce overhead and memory size. However, due to the Inception structure of GoogleNet, more fine-grained pruning is necessary. EPruner, which employs Affinity Propagation for fine-grained clustering, achieves better results but with a larger computational overhead compared to ours. Our methods surpass the remaining baselines in terms of accuracy, FLOPs pruning ratio, and parameter pruning ratio. The integration of class-wise information with our methods and other approaches holds substantial promise for future advancements.

4.2 Experiments on ImageNet

Compared to the CIFAR dataset, as shown in Table 2, we conduct experiments on a larger dataset, ImageNet, with numerous baselines using ResNet-50. The results indicate that our sTLP-IB achieves state-of-the-art performance, outperforming all baselines with a Top-1 accuracy of 76.12%, a 57% reduction in FLOPs, and a 55% reduction in parameters, with a mere 0.03% drop compared to the pretrained model. Our sGLP-IB also outperforms the majority of baselines and was comparable to CSD. Although sGLP-IB achieves a higher pruning ratio than CSD, its accuracy is only 0.01% lower, which may be attributed to CSD benefiting from its Discrete Wavelet Transform on large datasets, which shows the effectiveness of sGLP-IB. Combined with the results on CIFAR-10 from Section 4.1, our methods are applicable not only to simple datasets but also to complex, real-world datasets, showcasing their significant potential. We can leverage class-wise information and structured lasso for pruning across datasets.

4.3 Ablation Studies

Influence of batch sizes. As we keep the same pruning ratio as Table 1, the accuracy of sGLP-IB and sTLP-IB increases with the batch size but essentially converges once the batch size reaches 128 in Table 3. This observation forms the basis for setting our experimental batch size to 128. With a batch size of 64, the randomness due to smaller data volume results in sGLP-IB slightly outperforming sTLP-IB by 0.01% in Top-1 accuracy. As mentioned in work [Lin *et al.*, 2020], the average rank of feature maps generated by a single filter remains consistently the same and the pruned model's accuracy is not significantly affected by the batch size during the pruning process. Nonetheless, sTLP-IB consistently outperforms sGLP-IB across varying batch sizes, aligning with our previous findings that tree-guided lasso is more effective than graph-structured lasso when the batch size is larger. The experimental results demonstrate that our methods are robust to batch size variations, provided that the random selection of samples within batches is maintained.

Influence of Gram matrix. Following previous settings, we conduct experiments on the influence of Gram matrix. With the same pruning ratio for VGGNet-16 on CIFAR-10 as Table 1, sGLP-IB and sTLP-IB achieve Top-1 accuracies of 94.06% and 94.10%, respectively. This demonstrates that our methods can be effectively combined with Gram matrix. The integration of Gram matrix with structured lasso accelerates computation without sacrificing class-wise statistical information and accuracy, as evidenced by the comparable results of 94.05% and 94.10% obtained with Gram matrix. Notably, the time required for pruning the pretrained model without Gram matrix is approximately 25.7 times greater than when it is used, indicating that the introduction of Gram matrix significantly enhances pruning efficiency.

Influence of kernel selections. With the same pruning ratio, we conduct experiments by selecting different kernel functions, as shown in Table 4. Our methods yield similar results across Linear, Gaussian, and Laplacian kernels, maintaining high Top-1 accuracy shown in Table 5. By utilizing Gram matrix, our methods achieve faster computation and obtain

Details of the kernel functions	
Linear	$K(x,y) = x^T y + c$
Gaussian	$K(x,y) = exp(\frac{\|x-y\|^2}{2\sigma^2})$
Sigmoid	$K(x,y) = tanh(ax^T + c)$
Laplacian	$K(x,y) = exp(\frac{\|x-y\|}{\sigma})$

Table 4: Kernel functions of Gram matrix.

Method	Linear	Gaussian	Sigmoid	Laplacian
sGLP-IB	94.05	94.06	93.71	94.05
sTLP-IB	94.07	94.09	93.86	94.10

Table 5: Top-1 Accuracy of our methods using VGGNet-16 on CIFAR-10 dataset when varying the kernel functions.

(a) Linear (T) (b) Linear (T) (c) APIB

(d) Gaussian (G) (e) Sigmoid (G) (f) Laplacian (G)
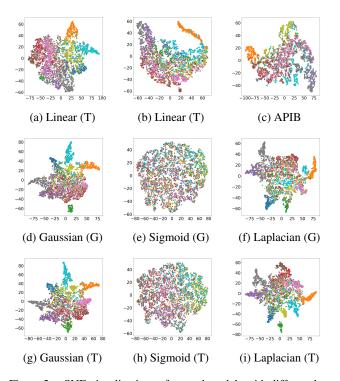
(g) Gaussian (T) (h) Sigmoid (T) (i) Laplacian (T)

Figure 2: t-SNE visualizations of pruned models with different kernel functions before finetuning using VGGNet-16 on CIFAR-10 dataset, where T denotes sTLP-IB and G denotes sGLP-IB.

good performance. In all cases, sTLP-IB outperforms sGLP-IB. It is worth noting that both sGLP-IB and sTLP-IB perform slightly worse with the Sigmoid kernel, due to the strong non-linear properties introduced by the tanh function. But sGLP-IB and sTLP-IB still behave better than most baselines with Sigmoid kernel. Overall, our methods exhibit robustness to different kernel functions.

t-SNE of the pruned model before finetuning. We further analyze the effectiveness of our method using t-SNE visualization on the pruned model before fine-tuning, following the previous settings. As illustrated in Figure 2, we also compare our approach with the most similar state-of-the-art

Method	Linear	Gaussian	Sigmoid	Laplacian
sGLP-IB	24.85	37.90	10.00	42.51
sTLP-IB	25.37	42.80	11.13	43.68

Table 6: Top-1 Accuracy of our methods when varying the kernel functions without finetuning.

method, APIB. Under the linear function, both sGLP-IB and sTLP-IB from our method perform better than APIB (Row 1), with more distinct clusters between classes, indicating that our approach better leverages class-wise information. We also present t-SNE visualizations for the Gaussian, Sigmoid, and Laplacian kernels (Row 2 and 3). We find that both the Gaussian and Laplacian kernels aid in learning class information, while the Sigmoid kernel function results in almost random distributions, consistent with the results in Table 5. We also evaluate the pruned model's Top-1 accuracy before finetuning, as shown in Table 6. We observe that the Gaussian and Laplacian kernels facilitate learning more information in high-dimensional spaces, while the linear kernel demonstrates the inherent capability of our structured lasso. The results with the Sigmoid kernel mirror the t-SNE visualizations. Notably, both in Table 6 and Figure 2, sTLP-IB consistently outperformed sGLP-IB, aligning with the results in Table 1.

Ratio	10	30	50	70	90
sGLP-IB	94.10(8%)	94.13(32%)	94.10(54%)	94.08(63%)	93.97(64%)
sTLP-IB	94.10(8%)	94.13(32%)	94.11(54%)	94.10(63%)	94.01(64%)

Table 7: Results of our methods when varying the parameter pruning ratios using VGGNet-16 on CIFAR-10 (The pruning ratio of pruned FLOPs is indicated in parentheses.).

Influence of pruning ratio. By adjusting the pruning ratio, we find that when the ratio is below 50%, the performance of sGLP-IB and sTLP-IB is similar due to the low pruning rate, resulting in nearly identical retained information and thus similar accuracy. When the ratio is below 70%, the variations in accuracy are minimal, indicating that our methods effectively capture and utilize statistical information for pruning. When the ratio falls below 10%, the accuracy has a slight drop, due to the excessive number of redundant filters.

Influence of layer pruning. We also conduct experiments to assess the effect of retaining the initial few layers without pruning. To maintain the same pruning ratio, we compensate by pruning more in the later layers for a similar number of parameters. In this scenario, the accuracies for sGLP-IB and sTLP-IB are 93.76% and 93.88% on CIFAR-10 with VGGNet-16, respectively. If the first two layers were not pruned, the accuracies are 93.79% and 93.98%, respectively. These results are lower than those in Table 1, which can be attributed to the fact that the early layers focus on low-level features in the images, while the later layers contain more class-wise information.

Supplementary Experiments. Experiments on CIFAR-100 dataset, experiments with FLOPs pruning ratio and time complexity discussions can be found in Appendix.

5 Conclusion

We propose two novel automatic pruning methods that both achieve state-of-the-art, sGLP-IB and sTLP-IB, which leverage information bottleneck theory and structured lasso to perform pruning using penalties that construct subgroup and subtree structures based on class-wise statistical information. Additionally, we present an adaptive approach that utilizes Gram matrix to accelerate the process and employs proximal gradient descent to further ensure convergence and polynomial time complexity. Through extensive experiments and ablation studies, we demonstrate the potential of our methods, both theoretically and experimentally, to effectively utilize class-wise information to aid pruning tasks, compared with numerous baselines. Exploring the better method to capture the class-wise information is our future direction.

References

[Alwani *et al.*, 2022] Manoj Alwani, Vashisht Madhavan, and Yang Wang. Decore: Deep compression with reinforcement learning. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[Barisin and Horenko, 2024] Tin Barisin and Illia Horenko. Towards generalized entropic sparsification for convolutional neural networks. *ArXiv*, abs/2404.04734, 2024.

[Cai *et al.*, 2022] Linhang Cai, Zhulin An, Chuanguang Yang, Yangchun Yan, and Yongjun Xu. Prior gradient mask guided pruning-aware fine-tuning. In *AAAI*, 2022.

[Castells and Yeom, 2021] Thibault Castells and Seul-Ki Yeom. Automatic neural network pruning that efficiently preserves the model accuracy. *ArXiv*, abs/2111.09635, 2021.

[Chen *et al.*, 2010] X. Chen, Seyoung Kim, Qihang Lin, Jaime G. Carbonell, and Eric P. Xing. Graph-structured multi-task regression and an efficient optimization method for general fused lasso. *ArXiv*, abs/1005.3579, 2010.

[Chen *et al.*, 2012] Xi Chen, Qihang Lin, Seyoung Kim, Jaime G Carbonell, and Eric P Xing. Smoothing proximal gradient method for general structured sparse regression. *The Annals of Applied Statistics*, pages 719–752, 2012.

[Dai *et al.*, 2018] Bin Dai, Chen Zhu, and David Paul Wipf. Compressing neural networks using the variational information bottleneck. In *International Conference on Machine Learning*, 2018.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[Ding *et al.*, 2019] Xiaohan Ding, Guiguang Ding, Xiangxin Zhou, Yuchen Guo, Ji Liu, and Jungong Han. Global sparse momentum sgd for pruning very deep neural networks. *Advances in Neural Information Processing Systems*, 2019.

[Duan *et al.*, 2022] Yuanzhi Duan, Xiaofang Hu, Yue Zhou, Peng He, Qi Liu, and Shukai Duan. Network pruning via feature shift minimization. In *Asian Conference on Computer Vision*, 2022.

[Elkerdawy *et al.*, 2022] Sara Elkerdawy, Mostafa Elhoushi, Hong Zhang, and Nilanjan Ray. Fire together wire together: A dynamic pruning approach with self-supervised mask prediction. *CVPR*, 2022.

[Gao *et al.*, 2021] Shangqian Gao, Feihu Huang, Weidong (Tom) Cai, and Heng Huang. Network pruning via performance maximization. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[Guo *et al.*, 2021] Yi Guo, Huan Yuan, Jianchao Tan, Zhangyang Wang, Sen Yang, and Ji Liu. Gdp: Stabilized neural network pruning via gates with differentiable polarization. *IEEE/CVF International Conference on Computer Vision*, 2021.

[Guo *et al.*, 2023] Song Guo, Lei Zhang, Xiawu Zheng, Yan Wang, Yuchao Li, Fei Chao, Chenglin Wu, Shengchuan Zhang, and Rongrong Ji. Automatic network pruning via hilbert-schmidt independence criterion lasso under information bottleneck principle. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.

[He *et al.*, 2016] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[He *et al.*, 2017] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. *ICCV*, 2017.

[He *et al.*, 2018] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4335–4344, 2018.

[He *et al.*, 2019] Yang He, Ping Liu, Linchao Zhu, and Yi Yang. Filter pruning by switching to neighboring cnns with good attributes. *IEEE Transactions on Neural Networks and Learning Systems*, 34:8044–8056, 2019.

[He *et al.*, 2022] Zhiqiang He, Yaguan Qian, Yuqi Wang, Bin Wang, Xiaohui Guan, Zhaoquan Gu, Xiang Ling, Shaoning Zeng, Haijiang Wang, and Wujie Zhou. Filter pruning via feature discrimination in deep neural networks. In *European Conference on Computer Vision*, 2022.

[Hu *et al.*, 2016] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *ArXiv*, abs/1607.03250, 2016.

[Hussien *et al.*, 2024] Mostafa Hussien, Mahmoud Afifi, Kim Khoa Nguyen, and Mohamed Cheriet. Small contributions, small networks: Efficient neural network pruning based on relative importance. *ArXiv*, abs/2410.16151, 2024.

[Joo *et al.*, 2021] Donggyu Joo, Eojindl Yi, Sunghyun Baek, and Junmo Kim. Linearly replaceable filters for deep network channel pruning. In *AAAI*, 2021.

[Kim and Xing, 2009] Seyoung Kim and Eric P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *International Conference on Machine Learning*, 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[Lanckriet *et al.*, 2002] Gert R. G. Lanckriet, Nello Cristianini, Peter L. Bartlett, Laurent El Ghaoui, and Michael I. Jordan. Learning the kernel matrix with semidefinite programming. In *Journal of machine learning research*, 2002.

[Lee *et al.*, 2020] Min Kyu Lee, Seunghyun Lee, Sang Hyuk Lee, and Byung Cheol Song. Channel pruning via gradient of mutual information for light-weight convolutional neural networks. *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1751–1755, 2020.

[Li *et al.*, 2016] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ArXiv*, abs/1608.08710, 2016.

[Li *et al.*, 2021] Yuchao Li, Shaohui Lin, Jianzhuang Liu, Qixiang Ye, Mengdi Wang, Fei Chao, Fan Yang, Jincheng Ma, Qi Tian, and Rongrong Ji. Towards compact cnns via collaborative compression. *CVPR*, 2021.

[Li *et al.*, 2022] Yawei Li, Kamil Adamczewski, Wen Li, Shuhang Gu, Radu Timofte, and Luc Van Gool. Revisiting random channel pruning for neural network compression. *CVPR*, 2022.

[Lin *et al.*, 2020] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. *CVPR*, 2020.

[Lin *et al.*, 2021] Mingbao Lin, Rongrong Ji, Shaojie Li, Yan Wang, Yongjian Wu, Feiyue Huang, and Qixiang Ye. Network pruning using adaptive exemplar filters. *IEEE Transactions on Neural Networks and Learning Systems*, 33:7357–7366, 2021.

[Liu *et al.*, 2024a] Xiang Liu, Hui Liu, Jing Diao, Wenting Ye, Xueling Liu, and Dehui Wei. Sparse variable selection on high dimensional heterogeneous data with tree structured responses. *IEEE Access*, 2024.

[Liu *et al.*, 2024b] Xiang Liu, Xueling Liu, Jing Diao, Mengyao Zheng, Jihe Li, Yongyi Xie, Kang Lai, Xiao Geng, Yijun Song, and Linshan Jiang. Novel truncated-rank graph-structured and tree-guided sparse linear mixed models for variable selection on genome-wide association studies. *BIBM*, 2024.

[Liu *et al.*, 2024c] Xiang Liu, Yijun Song, Xia Li, Yifei Sun, Huiying Lan, Zemin Liu, Linshan Jiang, and Jialin Li. Ed-vit: Splitting vision transformer for distributed inference on edge devices. *ArXiv*, abs/2410.11650, 2024.

[Ma *et al.*, 2020] Kurt Wan-Duo Ma, J. P. Lewis, and W. Kleijn. The hsic bottleneck: Deep learning without back-propagation. *AAAI*, 2020.

[Pham *et al.*, 2024a] Van Tien Pham, Yassine Zniyed, and Thanh Phuong Nguyen. Efficient tensor decomposition-based filter pruning. *Neural networks : the official journal of the International Neural Network Society*, 178:106393, 2024.

[Pham *et al.*, 2024b] Van Tien Pham, Yassine Zniyed, and Thanh Phuong Nguyen. Enhanced network compression through tensor decompositions and pruning. *IEEE transactions on neural networks and learning systems*, 2024.

[Rajaraman *et al.*, 2023] Nived Rajaraman, Devvrit, Aryan Mokhtari, and Kannan Ramchandran. Greedy pruning with group lasso provably generalizes for matrix sensing and neural networks with quadratic activations. *NeurIPS*, 2023.

[Ruan *et al.*, 2021] Xiaofeng Ruan, Yufan Liu, Bing Li, Chunfen Yuan, and Weiming Hu. Dpfps: Dynamic and progressive filter pruning for compressing convolutional neural networks from scratch. In *AAAI*, 2021.

[Sakamoto and Sato, 2024] Keitaro Sakamoto and Issei Sato. End-to-end training induces information bottleneck through layer-role differentiation: A comparative analysis with layer-wise training. *ArXiv*, abs/2402.09050, 2024.

[Scardapane *et al.*, 2016] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2016.

[Shao and Shin, 2022] Tuanjie Shao and Dongkun Shin. Structured pruning for deep convolutional neural networks via adaptive sparsity regularization. *IEEE 46th Annual Computers, Software, and Applications Conference*, 2022.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, D. Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[Tang *et al.*, 2020] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *ArXiv*, abs/2010.10732, 2020.

[Tishby and Zaslavsky, 2015] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. *IEEE Information Theory Workshop*, 2015.

[Tishby *et al.*, 2000] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *ArXiv*, physics/0004057, 2000.

[Wang *et al.*, 2021] Z. Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning

with structural redundancy reduction. *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[Xie *et al.*, 2024] Weiying Xie, Mei Yuan, Jitao Ma, and Yunsong Li. Adaptive pruning of channel spatial dependability in convolutional neural networks. In *ACM Multimedia*, 2024.

[Yan *et al.*, 2021] Yangchun Yan, Chao Li, Rongzuo Guo, Kang Yang, and Yongjun Xu. Channel pruning via multi-criteria based on weight dependency. *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

[Yu *et al.*, 2024] Di Yu, Xin Du, Linshan Jiang, Wentao Tong, and Shuiguang Deng. Ec-snn: Splitting deep spiking neural networks for edge devices. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, 2024.

[Yuan, 2024] Ganzhao Yuan. Smoothing proximal gradient methods for nonsmooth sparsity constrained optimization: Optimality conditions and global convergence. In *International Conference on Machine Learning*, 2024.

[Zhang *et al.*, 2021] Yanfu Zhang, Shangqian Gao, and Heng Huang. Exploration and estimation for model compression. *ICCV*, 2021.

[Zhang *et al.*, 2022] Yuxin Zhang, Mingbao Lin, Zhihang Lin, Yiting Luo, Ke Li, Fei Chao, Yongjian Wu, and Rongrong Ji. Learning best combination for efficient n: M sparsity. *ArXiv*, abs/2206.06662, 2022.

A Experiments

A.1 Dataset Information

We used three widely adopted datasets to evaluate our proposed methods against baselines: CIFAR10 [Krizhevsky, 2009], CIFAR100 [Krizhevsky, 2009] and Tiny ImageNet [Deng *et al.*, 2009].

- CIFAR-10 consists of 60,000 color images with 32×32 pixels, evenly distributed across 10 classes.

- CIFAR-100 dataset consists of 60,000 32×32 color images, evenly distributed across 100 classes.

- ImageNet dataset contains 1.2 million 224×224 color images across 1,000 classes, with each class having around 1,000 images.

A.2 Time Complexity Discussion

For sGLP-IB: the time complexity of graph-structured lasso could be found in the paper [Chen *et al.*, 2010].

For sTLP-IB: since a tree associated with out_i responses can have at most $2out_i - 1$ nodes, which constrains the $|G_v|$. It is computationally efficient and spatially economical to run sTLP-IB. We employ a smoothing proximal gradient method that is originally developed for structured-sparsity-inducing penalties. By using the efficient method, the convergence rate of the algorithm is $O(\frac{1}{\epsilon})$, given the desired accuracy ϵ and the time complexity per iteration of the smoothing proximal gradient is $O(in_i^2 out_i + in_i \sum v \in V|G_v|)$. Thus the overall complexity for our method is $O(\frac{1}{\epsilon} \times (in_i^2 out_i + in_i \sum v \in V|G_v|))$.

A.3 Details of Pruning on ResNet and GoogleNet

For ResNet: due to ResNet's architecture featuring shortcuts, when pruning ResNet variants, we initially ignore the shortcuts. We perform pruning layer by layer, and then prune this shortcut layer corresponding to the pruned channels of its related layers inside the residual block when we approach the end of this residual block. This pruning approach ensures the integrity and functionality of the shortcut connections while effectively reducing the network's complexity.

For GoogleNet: we perform pruning on each branch of Inception individually using sGLP-IB and sTLP-IB.

Method	Top-1(%)	FLOPs↓	Params ↓	△ Acc(%)
VGGNet-16	73.80	0(314.57M)	0(14.98M)	0
CPMC	73.01	48%	-	-0.79
PGMPF	73.45	48%	-	-0.35
CPGMI	73.53	37%	-	-0.27
PGMPF-SFP	73.66	48%	-	-0.14
APIB	73.89	48%	57	+0.09
sGLP-IB	**73.96**	**55%**	**71%**	**+0.16**
sTLP-IB	**73.99**	**55%**	**71%**	**+0.19**

Table 8: Pruning results of VGG-16 on CIFAR-100 dataset.

A.4 Experiments with CIFAR-100 Dataset

We compare sGLP-IB and sTLP-IB with CPMC [Yan *et al.*, 2021], PGMPF [Cai *et al.*, 2022], CPGMI [Lee *et al.*, 2020], PGMPF-SFP [Cai *et al.*, 2022], and APIB [Guo *et al.*, 2023]

on the CIFAR-100 dataset using VGGNet-16. Our methods, sGLP-IB and sTLP-IB, achieve the highest accuracy with the largest pruning ratio (55% for FLOPs and 71% for parameters), reaching 73.96% and 73.99%, respectively, which are 0.16% and 0.19% higher than the base model. This comparison with these state-of-the-art methods highlights the effectiveness of our approach, considering Top-1 accuracy, FLOPs pruned ratio, and parameter reduction ratio metrics. Combined with results from experiments on CIFAR-10 and ImageNet, it demonstrates that our methods can be extended to various real-world datasets.

Ratio	40	50	60	70	80
sGLP-IB	94.12(45%)	94.09(54%)	94.04(84%)	93.74(93%)	91.00(94%)
sTLP-IB	94.12(47%)	94.11(55%)	94.09(84%)	93.80(93%)	91.75(94%)

Table 9: Results of our methods when varying the FLOPs pruning ratios using VGGNet-16 on CIFAR-10 (The parameter ratio of pruned FLOPs is indicated in parentheses.).

A.5 Experiments with Pruning Ratios for FLOPs

Before the pruning ratio exceeds 50%, we observe that the parameter pruned ratio is also below 60%, and the performance of sGLP-IB and sTLP-IB is nearly identical. It is noteworthy, however, that sTLP-IB prunes more parameters, indicating that it retains fewer parameters while maintaining the same FLOPs. This suggests that sTLP-IB is more effective at focusing on important filters due to its superior ability to model class-wise information. When the pruning ratio exceeds 70%, the accuracy of both methods begins to drop. At a ratio of 80%, the accuracy of our methods drops by 2-3% due to excessive parameter reduction (94%), though sTLP-IB still remains more robust than sGLP-IB.