Channel Pruning Guided by Classification Loss and Feature Importance

Jinyang Guo,[1] **Wanli Ouyang,**[2] **Dong Xu**[1]
[1]School of Electrical and Information Engineering, The University of Sydney
[2]The University of Sydney, SenseTime Computer Vision Research Group, Australia
{jinyang.guo, wanli.ouyang, dong.xu}@sydney.edu.au

Abstract

In this work, we propose a new layer-by-layer channel pruning method called Channel Pruning guided by classification Loss and feature Importance (CPLI). In contrast to the existing layer-by-layer channel pruning approaches that only consider how to reconstruct the features from the next layer, our approach additionally take the classification loss into account in the channel pruning process. We also observe that some reconstructed features will be removed at the next pruning stage. So it is unnecessary to reconstruct these features. To this end, we propose a new strategy to suppress the influence of unimportant features (*i.e.*, the features will be removed at the next pruning stage). Our comprehensive experiments on three benchmark datasets, *i.e.*, CIFAR-10, ImageNet, and UCF-101, demonstrate the effectiveness of our CPLI method.

1 Introduction

While deep learning methods have achieved remarkable success in many computer vision tasks, it is still a challenging task to deploy convolutional neural networks (CNNs) on mobile devicies due to tight computational resources. Several model compression technologies (*e.g.*, quantization and tensor factorization) were recently developed to improve the efficiency of the CNNs, among which channel pruning is one of the most popular techniques.

The channel pruning approaches can be roughly categorized as loss minimization methods (Molchanov et al. 2017; LeCun, Denker, and Solla 1990; Molchanov et al. 2019) and layer-by-layer approaches (He, Zhang, and Sun 2017; Luo, Wu, and Lin 2017). The loss minimization methods iteratively remove the channels with least effect on the final loss across the entire network. However, in order to evaluate the effect of channels on the final loss, the fine-tuning process needs to be performed frequently in these methods, which makes the channel pruning process slow. On the other hand, the layer-by-layer methods select informative channels and adjust model parameters by minimizing the reconstruction error of the features in the next layer. These methods are much faster because they prune the channels within a layer in one step and the fine-tuning process is performed only once. However, the layer-by-layer methods do not consider the final loss. The channels selected in these methods may have little effect on the final loss, leading to a sub-optimal solution for channel selection. In addition, the existing layer-by-layer approaches treat the reconstruction of each feature equally important, which leads to the *next-layer feature removal* problem. Specifically, at the current pruning stage, the layer-by-layer approaches will minimize the reconstruction error of a feature from the next layer under the assumption that this feature will be kept in the compressed model. However, this assumption will be violated if this feature will be removed at the next pruning stage, and reconstruction of this feature will thus become unnecessary.

To solve the aforementioned problems, we propose a new channel pruning approach called Channel Pruning guided by classification Loss and feature Importance (CPLI), in which we follow the layer-by-layer approaches for faster speed and take the final loss into account to solve the sub-optimal problem for channel selection. In order to address the *next-layer feature removal* problem, we also propose a new strategy to pay more attention to important features (*i.e.*, the features to be kept in the compressed model) and ignore unimportant features (*i.e.*, the features to be removed at the next pruning stage) . Specifically, our newly proposed method selects the channels in each layer by solving a LASSO optimization problem based on two aspects: 1) the cross-entropy loss and 2) the importance of features. In order to avoid frequently performing the fine-tuning process, we adjust the weights by solving a least square optimization problem based on the features from the next layer, which has a close-form solution. Moreover, due to the close-form solution of the least square optimization problem, the adjusted weights in this process can converge better than the alternative approaches, where the weights are adjusted by using simple fine-tuning techniques. Therefore, the channel selection process in our method is based on well-learned weights, which is more accurate than the existing loss minimization approaches.

The main contributions of this work are as follows: First, our newly proposed method takes advantage of efficiency from the layer-by-layer channel pruning approaches while also taking the final loss into account, which can partially solve the sub-optimal problem for channel selection in the layer-by-layer channel pruning methods. Second, our approach can ignore reconstruction of unimportant features to address the *next-layer feature removal* problem. As a result, our method can select more informative channels when compared with the existing layer-by-layer channel pruning approaches. Comprehensive experiments on three benchmark datasets demonstrate the effectiveness of our newly proposed approach for model compression.

2 Related Work

Model compression technologies can be roughly classified as four categories: quantization (Rastegari et al. 2016), tensor factorization (Gong et al. 2014; Jaderberg, Vedaldi, and Zisserman 2014; Kim et al. 2015; Lebedev et al. 2014; Xue, Li, and Gong 2013), compact network design (Howard et al. 2017; Zhang et al. 2018a; Figurnov et al. 2016), and channel pruning (He et al. 2018; Hu et al. 2016; Li et al. 2016; Yu et al. 2018; Lemaire, Achkar, and Jodoin 2019; Zhao et al. 2019; Ding et al. 2019). The **quantization** approaches directly represent float points by using smaller number of bits. For example, the work in (Rastegari et al. 2016) quantizes the network parameters into +1/-1 to accelerate computation. The **tensor factorization** methods decompose the weights into several low-rank matrices. In (Jaderberg, Vedaldi, and Zisserman 2014), one 3×3 filter is decomposed into one 1×3 and one 3×1 matrix, while in (Zhang et al. 2016), each layer is decomposed into a 3×3 and a 1×1 matrix. The **compact network design** approaches accelerate deep models by designing an efficient network architecture. The work in (Howard et al. 2017) introduced the depth-wise convolution operation to accelerate the inference speed. In (Zhang et al. 2018a), group-wise convolution is used to reduce computation of conventional convolution operations.

The **channel pruning** methods target at pruning a predefined number of channels and the corresponding weights related to these channels. For example, in (Liu et al. 2017), the authors used the batch normalization layer to scale each channel and prune the channels with small scaling factors. In the width-multiplier method (Howard et al. 2017), the network is shrank uniformly and trained from scratch. More recently, the work in (Zhuang et al. 2018) proposed to select the channels according to the discriminative power of each channel, while the work in (Lin et al. 2019) used adversarial training to prune the channels automatically. The other channel pruning methods can be roughly categorized as two groups: loss minimization methods (Molchanov et al. 2017; 2019) and layer-by-layer approaches (Luo, Wu, and Lin 2017; He, Zhang, and Sun 2017). On one hand, for loss minimization methods, Molchanov et al. (Molchanov et al. 2017) aims at minimizing the loss change by iteratively removing the least important channels. When compared with the loss minimization methods, we do not need to frequently perform the time-consuming fine-tuning process, which makes our approach more efficient. In addition, in order to save the computational costs, the loss minimization approaches can only fine-tune the network under compression for a limitted number of iterations between two pruning stages, which cannot guarantee convergence. This leads to inaccurate evaluation of channel importance in the next pruning stage, which will degrade the channel selection performance. Our method selects the channels based on well-learned weights, which makes the channel selection in our approach more accurate. On the other hand, for the layer-by-layer approaches, He et al. (He, Zhang, and Sun 2017) proposed a channel pruning method to select the channels by solving a LASSO optimization problem and update the corresponding weights by solving a least square optimization problem in a layer-by-layer fashion. Our work also uses a similar approach to select the channels and update the weights, which makes our approach fast. However, our approach additionally considers the final loss in the channel pruning process and solves the *next-layer feature removal* problem by paying more attention to the important features, which are important factors that influence the channel selection process but not considered in (He, Zhang, and Sun 2017).

3 Channel Pruning Guided by Classification Loss and Feature Importance

In this section, we firstly introduce the overview of our CPLI approach. Then, we present our method in the case of pruning a single layer in details. Finally, we present the pseudo code of the channel pruning process of the proposed approach.

3.1 Overview of Our CPLI Framework

Given a pre-trained model, our goal is to compress this model to achieve the highest accuracy for a given compression ratio. The pre-trained model is called the uncompressed model and the model after model compression is called the compressed model.

Firstly, we extract the features of the uncompressed model based on the training dataset. Then, we prune the given uncompressed model in a layer-by-layer fashion from shallow layers (closer to the image) to deep layers (closer to the output) by using the objective function introduced in Eq. (5). After the channel pruning process, we perform the fine-tuning process on the compressed model to recover from the accuracy drop.

3.2 CPLI in Each Layer

For better presentation, we introduce our CPLI approach in each layer in this section. We firstly introduce the objective function of our CPLI method. Then, we will explain each part of our objective function in details.

(a) Formulation

Suppose we have a network with L layers. When we have an input image $\mathcal{I}$, we can obtain the features at each layer in the network. Let us denote $\mathbf{X}^{0,(l)} \in \mathbb{R}^{c_{in}^{(l)} \times h_{in}^{(l)} \times w_{in}^{(l)}}$ as the input feature at the l-th layer of the uncompressed model, where $c_{in}^{(l)}$ is the number of input channels at this layer. $h_{in}^{(l)}$ and $w_{in}^{(l)}$ are the height and the width of the input feature $\mathbf{X}^{0,(l)}$, respectively. Denote $\mathbf{Y}^{0,(l)} \in \mathbb{R}^{c_{out}^{(l)} \times h_{out}^{(l)} \times w_{out}^{(l)}}$ as the output feature at the l-th layer, where $c_{out}^{(l)}$ is the number of output channels at this layer. $h_{out}^{(l)}$ and $w_{out}^{(l)}$ are the height and the width of the output feature $\mathbf{Y}^{0,(l)}$, respectively. Denote $\mathbf{W}^{0,(l)} \in \mathbb{R}^{c_{out}^{(l)} \times c_{in}^{(l)} \times h_k^{(l)} \times w_k^{(l)}}$ as the weights at the l-th layer of the uncompressed model. The l-th convolutional layer connects the features $\mathbf{X}^{0,(l)}$ and $\mathbf{Y}^{0,(l)}$. The output feature $\mathbf{Y}^{0,(l)}$ can be calculated as:

$$\mathbf{Y}_{i,:,:}^{0,(l)} = \sum_{j=1}^{c_{in}^{(l)}} \beta_j^{0,(l)} \mathbf{X}_{j,:,:}^{0,(l)} * \mathbf{W}_{i,j,:,:}^{0,(l)}, \tag{1}$$

where $\mathbf{X}_{j,:,:}^{0,(l)}$ is the j-th channel of the input feature $\mathbf{X}^{0,(l)}$. $\mathbf{Y}_{i,:,:}^{0,(l)}$ is the i-th channel of the output feature $\mathbf{Y}^{0,(l)}$. $\mathbf{W}_{i,j,:,:}^{0,(l)}$ is the j-th channel of the i-th convolutional filter from $\mathbf{W}^{0,(l)}$. $*$ is the convolution operation. $\boldsymbol{\beta}^{0,(l)} \in \{0,1\}^{c_{in}^{(l)}}$ is the vector containing a set of binary channel selection indicators $\beta_j^{0,(l)}, j = 1, \ldots, c_{in}^{(l)}$. For the uncompressed model, $\beta_j^{0,(l)} = 1$ for $j = 1, \ldots, c_{in}^{(l)}$, namely, all the channels are preserved. We omit the activation function and the bias term for better representation.

After pruning the previous layers, the l-th layer will have an input feature $\mathbf{X}^{(l)}$. Then we prune some channels of the input feature in the l-th layer and the compressed model will have an output feature $\mathbf{Y}^{(l)}$ at the l-th layer. Denote $\mathbf{W}^{(l)} \in \mathbb{R}^{c_{out}^{(l)} \times c_{in}^{(l)} \times h_k^{(l)} \times w_k^{(l)}}$ as the convolutional filters at the l-th layer in the compressed model, the output feature $\mathbf{Y}^{(l)}$ can be calculated as:

$$\mathbf{Y}_{i,:,:}^{(l)} = \sum_{j=1}^{c_{in}^{(l)}} \beta_j^{(l)} \mathbf{X}_{j,:,:}^{(l)} * \mathbf{W}_{i,j,:,:}^{(l)}, \tag{2}$$

where $\mathbf{X}_{j,:,:}^{(l)}$ is the j-th channel of the input feature $\mathbf{X}^{(l)}$. $\mathbf{Y}_{i,:,:}^{(l)}$ is the i-th channel of the output feature $\mathbf{Y}^{(l)}$. $\mathbf{W}_{i,j,:,:}^{(l)}$ is the j-th channel of the i-th convolutional filter from $\mathbf{W}^{(l)}$. The other notations are the same as those in Eq. (1). $\boldsymbol{\beta}^{(l)} \in \{0,1\}^{c_{in}^{(l)}}$ is the vector containing a set of binary channel selection indicators $\beta_j^{(l)}, j = 1, \ldots, c_{in}^{(l)}$. If $\beta_j^{(l)} = 0$, the j-th channel of the input feature at layer l will be pruned.

The layer-by-layer channel pruning approach in the work (He, Zhang, and Sun 2017) prunes the channels by solving the following optimization problem:

$$\begin{aligned} & \underset{\boldsymbol{\beta}^{(l)}, \mathbf{W}^{(l)}}{\arg\min} \|\mathbf{Y}^{0,(l)} - \mathbf{Y}^{(l)}\|_F^2, \\ = & \underset{\boldsymbol{\beta}^{(l)}, \mathbf{W}^{(l)}}{\arg\min} \sum_{i=1}^{c_{out}^{(l)}} \sum_{m=1}^{M^{(l)}} \left[y_{i,m}^{0,(l)} - y_{i,m}^{(l)} \right]^2, \\ & \text{subject to } \|\boldsymbol{\beta}^{(l)}\|_0 \leq B^{(l)}, \end{aligned} \tag{3}$$

where $y_{i,m}^{0,(l)}$ and $y_{i,m}^{(l)}$ are the output features of the i-th channel at location m for the uncompressed model and the compressed model, respectively. $y_{i,m}^{0,(l)}$ is an element in $\mathbf{Y}^{0,(l)}$ and $y_{i,m}^{(l)}$ is an element in $\mathbf{Y}^{(l)}$. $M^{(l)}$ is the length of the vectorized feature map at the l-th layer. $B^{(l)}$ is the pre-defined number of remained channels for the l-th layer. $\| \cdot \|_0$ is l_0 norm and $\| \cdot \|_F$ is the Frobenius norm.

The objective function in Eq. (3) only minimizes the reconstruction error between $y_{i,m}^{0,(l)}$ and $y_{i,m}^{(l)}$. It does not take the final classification loss or the feature importance into account. Therefore, we propose our objective function by additionally considering the final classification loss and feature importance.

Notation change. Since we focus on how to prune the l-th layer of the network in this section, we drop the layer index l thereafter except in Algorithm 1 for better presentation.

Classification loss. Let us denote $\mathcal{C}(\mathbf{Y}, g; \mathcal{W})$ as the classification loss function of the compressed network when the output feature at the l-th layer is $\mathbf{Y}$. The classification loss function of the compressed model can be defined as follows:

$$\mathcal{C} = \mathcal{L}_c \left[\mathcal{N}(\mathbf{Y}; \mathcal{W}), g \right], \tag{4}$$

where $\mathcal{L}_c$ is the cross-entropy loss function. g is the ground truth label for the image $\mathcal{I}$. When pruning the l-th layer, $\mathcal{N}$ and $\mathcal{W}$ are the sub-network and its parameters from the $(l+1)$-th layer to the L-th layer in the compressed model, respectively.

Our overall objective function. Denote $y_{i,m}$ for $i = 1, \ldots, c_{out}, m = 1, \ldots, M$ as an element in $\mathbf{Y}$. In our CPLI approach, we prune the channels by considering the final classification loss and the feature importance. To this end, we propose the following objective function for each layer:

$$\begin{aligned} & \underset{\boldsymbol{\beta}, \mathbf{W}}{\arg\min} \sum_{i=1}^{c_{out}} \sum_{m=1}^{M} \left[\frac{\partial \mathcal{C}}{\partial y_{i,m}} \cdot (y_{i,m}^0 - \gamma y_{i,m}^* \cdot y_{i,m}) \right]^2, \\ & \text{subject to } \|\boldsymbol{\beta}\|_0 \leq B, \end{aligned} \tag{5}$$

where $\frac{\partial \mathcal{C}}{\partial y_{i,m}}$ is the partial derivative of the classification loss with respect to $y_{i,m}$, and the loss function $\mathcal{C}$ is defined in Eq. (4). $y_{i,m}^*$ is the i-th channel at location m from the output features in the compressed model after the pruning previous layers. γ is a constant, which is empirically set as 1. The term $\frac{\partial \mathcal{C}}{\partial y_{i,m}}$ corresponds to the guidance from the classification loss and the term $\gamma y_{i,m}^*$ corresponds to the guidance from the feature importance. We will introduce the motivation of these two terms in the following section.

(b) Guidance from the Classification Loss

Channel pruning with guidance from the classification loss. If we only consider the classification loss without considering the feature importance, the objective function in Eq. (5) is redefined as follows:

$$\begin{aligned} & \underset{\boldsymbol{\beta}, \mathbf{W}}{\arg\min} \sum_{i=1}^{c_{out}} \sum_{m=1}^{M} \left[\frac{\partial \mathcal{C}}{\partial y_{i,m}} \cdot (y_{i,m}^0 - y_{i,m}) \right]^2, \\ & \text{subject to } \|\boldsymbol{\beta}\|_0 \leq B, \end{aligned} \tag{6}$$

where $\frac{\partial \mathcal{C}}{\partial y_{i,m}}$ is the partial derivative of the classification loss with respect to $y_{i,m}$. $y^0_{i,m}$ and $y_{i,m}$ are the output features from the i-th channel at location m for the uncompressed model and the compressed model, respectively. M is the length of the vectorized feature map. B is the pre-defined number of remained channels for this layer. The other notations are the same as those in Eq. (5).

Analysis. The term $(y^0_{i,m} - y_{i,m})$ in Eq. (6) can be considered as the reconstruction error, which is the error by using $y_{i,m}$ to reconstruct $y^0_{i,m}$. At the spatial location m, the term $\frac{\partial \mathcal{C}}{\partial y_{i,m}}$ can be treated as the "importance" of the i-th channel, which indicates the impact by changing $y_{i,m}$ on the classification loss $\mathcal{C}$. For the same reconstruction error, a large "importance" value indicates that even a small change of this channel will cause a large change of the classification loss. In this situation, this feature is important and we should pay more attention to how to minimize the reconstruction error of this channel. As another extreme, if $\frac{\partial \mathcal{C}}{\partial y_{i,m}} = 0$, the i-th channel is not important and the reconstruction error does not contribute to the objective function for channel pruning.

When compared with the objective function in the work (He, Zhang, and Sun 2017), we additionally consider the term $\frac{\partial \mathcal{C}}{\partial y_{i,m}}$, which can weight each channel by considering the impact of this channel on the final classification loss. In this way, our approach can be guided by the classification loss.

(c) Guidance from the Feature Importance

The *next-layer feature removal* problem. At the current pruning stage, we prune the input features at layer l, in which the output feature $y_{i,m}$ is used in the objective function to reconstruct $y^0_{i,m}$. In the next pruning stage, we will prune layer $l+1$, in which the feature $y_{i,m}$ will be treated as the input features and may be pruned, which is not taken into consideration in the previous section. If the i-th channel of the output features is removed when pruning the next layer $l+1$ at the next pruning stage, then $y_{i,m} = 0$. In this case, reconstruction of $y_{i,m}$ is unnecessary and may lead to inaccurate channel selection. This problems is called as the *next-layer feature removal* problem.

It is worth mentioning that the *next-layer feature removal* problem often occurs when the compression ratio is large. For example, more than half of channels in each layer in a ResNet-50 model under 2.25× compression ratio is removed, which indicates that the probability of the *next-layer feature removal* problem occurs is high.

Channel pruning with guidance from feature importance. To handle the *next-layer feature removal* problem, we propose to introduce the term $\gamma y^*_{i,m}$ into the objective function in Eq. (5). If we only consider the feature importance without considering the classification loss, the objective function can be written as follows:

$$\underset{\boldsymbol{\beta},\mathbf{W}}{\arg\min} \sum_{i=1}^{c_{out}} \sum_{m=1}^{M} (y^0_{i,m} - \gamma y^*_{i,m} \cdot y_{i,m})^2, \quad \text{subject to } \|\boldsymbol{\beta}\|_0 \le B. \tag{7}$$

This objective function can be equivalently rewritten as follows:

$$\begin{aligned} \underset{\boldsymbol{\beta},\mathbf{W}}{\arg\min} \sum_{i=1}^{c_{out}} \sum_{m=1}^{M} \big[&\gamma y^*_{i,m} \cdot (y^0_{i,m} - y_{i,m}) + \\ &(1 - \gamma y^*_{i,m}) \cdot (y^0_{i,m} - 0) \big]^2, \end{aligned} \tag{8}$$
$$\text{subject to } \|\boldsymbol{\beta}\|_0 \le B,$$

where $y^*_{i,m}$ is the output features of the i-th channel at location m after pruning the previous layers and γ is a constant, which is empirically set as 1 in our experiments. The other notations are the same as those in Eq. (6).

At the spatial location m, we use $\gamma y^*_{i,m}$ as the guidance of whether the i-th channel will be removed or not in the next pruning stage. When $\gamma y^*_{i,m} = 0$ for most of the spatial locations m, it is more likely that the i-th channel will be removed at the next pruning stage. In this case, we will use the reconstruction error of $(y^0_{i,m} - 0)$. On the other hand, if the i-th channel is not pruned at the next stage, we will use the reconstruction error of $(y^0_{i,m} - y_{i,m})$. Since the removal of the channels at the next pruning stage is determined by many factors, including the spatial location m, the input image, and the learned parameters, it is hard to predict whether this channel will be removed at the next pruning stage. In this work, we empirically use $\gamma y^*_{i,m}$ as the guidance, which is based on the following two observations. 1) In the ideal case, if we have $\gamma y^*_{i,m} = 0$ for all positions m in the i-th channel, the i-th channel can be readily removed at the next pruning stage. 2) The magnitude of $\gamma y^*_{i,m}$ is also used as the guidance for preserving/pruning the neuron in (Han et al. 2015). Although the aforementioned ideal case will rarely happen in real applications, our channel pruning method using $\gamma y^*_{i,m}$ as the guidance effectively avoids minimizing the reconstruction error from uninformative features that will be removed at the next pruning stage and thus can achieve reasonable results.

(d) Solution

Notation change. For better presentation, thereafter, we omit the summation over spatial locations M and the index for location m. The formulation for multiple locations can be readily obtained.

Solution. It is a NP-hard problem to solve the objective function in Eq. (5). Therefore, we relax the l_0 regularization to l_1 regularization and arrive at the following objective function:

$$\underset{\boldsymbol{\beta},\mathbf{W}}{\arg\min} \sum_{i=1}^{c_{out}} \left[\frac{\partial \mathcal{C}}{\partial y_i} \cdot (y^0_i - \gamma y^*_i \cdot y_i) \right]^2 + \lambda \|\boldsymbol{\beta}\|_1, \quad \text{subject to } \|\boldsymbol{\beta}\|_0 \le B, \tag{9}$$

where λ is the coefficient to balance different terms. Following (He, Zhang, and Sun 2017), we solve the optimization problem in Eq. (9) by using two steps. First, we fix $\mathbf{W}$ and solve a LASSO optimization problem:

$$\begin{aligned} &\arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{c_{out}} \left[\frac{\partial \mathcal{C}}{\partial y_i} \cdot (y_i^0 - \gamma y_i^* \cdot y_i)\right]^2 + \lambda \|\boldsymbol{\beta}\|_1, \\ &\text{subject to } \|\boldsymbol{\beta}\|_0 \leq B. \end{aligned} \tag{10}$$

We gradually increase λ until the constraint $\|\boldsymbol{\beta}\|_0 \leq B$ is satisfied. After the channel selection process is finished, we treat each selected channel equally important because the remaining channels have large impact on the output of the classification loss (*i.e.*, the channels with small impact are already pruned). Therefore, we consider $\gamma y_i^* = 1$, drop the term $\frac{\partial \mathcal{C}}{\partial y_i}$, and minimize the reconstruction error by solving a least square optimization problem with fixed $\boldsymbol{\beta}$:

$$\arg\min_{\mathbf{W}} \sum_{i=1}^{c_{out}} (y_i^0 - y_i)^2. \tag{11}$$

In this way, we minimize the objective function for each layer. We iteratively perform the pruning process in a layer-by-layer fashion to compress the pre-trained model and obtain the compressed model before the fine-tuning process.

3.3 Pseudo Code

Algorithm 1 presents the pseudo code of our CPLI approach for pruning a pre-trained model. Given a pre-trained model, we can use Algorithm 1 to prune this model, and obtain the compressed model M_c before the fine-tuning process. Again, for better representation, we omit the summation over spatial locations M and the index for location m when introducing our algorithm. In practice, we do not prune the first layer of the uncompressed model because the input features of the first layer are raw images. Then, we fine-tune the compressed model to recover from the accuracy drop.

4 Experiments

In this section, we first compare our CPLI approach with several state-of-the-art channel pruning methods on two benchmark datasets: CIFAR-10 (Krizhevsky 2009) and ImageNet (Russakovsky et al. 2015) for the image classification task. We further conduct the experiments to prune 3D convolutional network for the action recognition task on the UCF-101 dataset (Soomro, Zamir, and Shah 2012) to demonstrate the generalization ability of our method. We finally investigate each component of our method in details.

The compression ratio (CR) refers to the ratio of the floating point operations (FLOPs) from the uncompressed model over that from the compressed model, which is a commonly used criterion for computational complexity measurement. Since the accuracies of the pre-trained model vary a lot for different baseline methods, we follow the work in (Zhuang et al. 2018) to report the accuracy drop after the fine-tuning process.

Algorithm 1: Our CPLI approach for pruning the pre-trained model.

```
Input: Pre-trained model $\mathrm{M}_u$,
       $\mathrm{M}_u = \{\mathbf{W}^{0,(1)}, \mathbf{W}^{0,(2)}, \ldots, \mathbf{W}^{0,(L)}, \boldsymbol{\Theta}\}$,
       where $\mathbf{W}^{0,(l)}$ for $l \in L$ is the parameters for the
       $l$-th layer and $\boldsymbol{\Theta}$ is the parameters for other
       layers (e.g., the fully connected layers) that will
       not be pruned.
Output: Compressed model $\mathrm{M}_c$, which is then used for
        the fine-tuning process.
Extract the output features in the pre-trained model for
  each layer $y_i^{0,(1)}, \ldots, y_i^{0,(L)}$ for all channels.
Set $\mathrm{M}_c = \mathrm{M}_u = \{\mathbf{W}^{0,(1)}, \mathbf{W}^{0,(2)}, \ldots, \mathbf{W}^{0,(L)}, \boldsymbol{\Theta}\}$.
for $l$ = 2 to $L$ do
    Based on the current compressed model $\mathrm{M}_c$, use
      forward-propagation to calculate $y_i^{*(l)}$ in Eq. (10),
      where the superscript $\cdot^{(l)}$ denotes the $l$-th layer.
    Use back-propagation to calculate $\frac{\partial \mathcal{C}}{\partial y_i^{(l)}}$ in Eq. (10),
      where $y_i^{(l)}$ is a feature of the current compressed
      model at layer $l$ and channel $i$.
    Solve the LASSO optimization problem in Eq. (10),
      and obtain the channel selection vector $\boldsymbol{\beta}^{(l)}$ for the
      $l$-th layer.
    Solve the the least square optimization problem in
      Eq. (11) with fixed $\boldsymbol{\beta}^{(l)}$, and obtain the adjusted
      weights $\tilde{\mathbf{W}}^{(l)}$ for the $l$-th layer.
    Prune the filters in the $(l-1)$-th layer by removing
      the $k$-th filter in $\tilde{\mathbf{W}}^{(l-1)}$ where $k$ are the indices
      for all $\beta_k^{(l)} = 0$ in $\boldsymbol{\beta}^{(l)}$ and obtain $\hat{\mathbf{W}}^{(l-1)}$.
    Prune the channels by setting
      $\mathrm{M}_c = \{\hat{\mathbf{W}}^{(1)}, \hat{\mathbf{W}}^{(2)}, \ldots, \hat{\mathbf{W}}^{(l-1)},$
      $\tilde{\mathbf{W}}^{(l)}, \mathbf{W}^{0,(l+1)}, \ldots, \mathbf{W}^{0,(L)}, \boldsymbol{\Theta}\}$
end
Obtain the compressed model $\mathrm{M}_c$ before the fine-tuning
  process.
```

4.1 Results on CIFAR-10

We take three popular models VGGNet (Simonyan and Zisserman 2014), ResNet-56 (He et al. 2016), and MobileNet-V2 (Sandler et al. 2018) on the CIFAR-10 dataset to demonstrate the effectiveness of the proposed approach. The CIFAR-10 dataset (Krizhevsky 2009) consists of 50k training samples and 10k testing images from 10 classes. Based on the pre-trained models, we apply our proposed CPLI method to prune the channels and fine-tune the compressed model. In the pruning process, we randomly choose 10 locations from each feature map instead of using all the spatial locations to accelerate the pruning process. At the fine-tuning stage, similar to (Zhuang et al. 2018), we use SGD with nesterov for optimization. The momentum, the weight decay, and the mini-batch size are set to 0.9 and 0.0001, and 256, respectively. The initial learning rate is set to 0.1 and step learning rate decay is used.

In Table 1, we compare our method with several state-of-the-art methods including ThiNet (Luo, Wu, and Lin 2017), Channel Pruning (CP) (He, Zhang, and Sun 2017), Slimming (Liu et al. 2017), Width-multiplier (WM) (Howard et al. 2017), and DCP (Zhuang et al. 2018) for compressing

Table 1: Comparison of different channel pruning methods for compressing VGGNet, ResNet-56, and MobileNet-V2 on the CIFAR-10 dataset. For reference, in our implementation, the accuracies of the uncompressed VGGNet, ResNet-56, and MobileNet-V2 models are 93.99%, 93.74%, and 95.02%, respectively. "+" denotes that the accuracy increases, while "-" denotes that the accuracy decreases after channel pruning. We directly quote the results of the existing works from (Zhuang et al. 2018).

Model		ThiNet	CP	Slimming	WM	DCP	**Ours**
VGGNet	CR	2.00×	2.00×	2.04×	2.00×	2.86×	**2.86**×
	Top-1 Acc. drop (%)	-0.14	-0.32	-0.19	-0.38	+0.58	**+0.96**
ResNet-56	CR	1.99×	2×	-	1.99×	1.99×	**1.99**×
	Top-1 Acc. drop (%)	-0.82	-1.0	-	-0.56	-0.31	**+0.07**
MobileNet-V2	CR	-	-	-	1.36×	1.36×	**1.36**×
	Top-1 Acc. drop (%)	-	-	-	-0.45	+0.22	**+0.35**

Table 2: Comparison of Top-5 accuracies when compressing ResNet-50 and Top-1 accuracies when compressing MobileNet-V2 by using different channel pruning methods on the ImageNet dataset. For reference, in our implementation, the Top-5 accuracy of the uncompressed ResNet-50 model and Top-1 accuracy of the uncompressed MobileNet-V2 model on the ImageNet dataset are 92.87% and 72.19%, respectively. For the existing works, we directly quote the results from (Zhuang et al. 2018).

Model		ThiNet	CP	WM	DCP	GAL	**Ours**
ResNet-50	CR	2.25×	2×	2.25×	2.25×	2.22×	**2.25**×
	Top-5 Acc. drop (%)	-1.12	-1.40	-1.62	-0.61	-2.05	**-0.38**
MobileNet-V2	CR	1.81×	-	1.81×	1.81×	-	**1.81**×
	Top-1 Acc. drop (%)	-6.36	-	-6.40	-5.89	-	**-4.84**

VGGNet, ResNet-56, and MobileNet-V2 on the CIFAR-10 dataset.

From the results in Table 1, our CPLI approach consistently outperforms other baseline methods, which demonstrates the effectiveness of our method on small-scale datasets. It is also worth mentioning that our CPLI approach outperforms the pre-trained VGGNet, ResNet-56, and MobileNet-V2 by 0.96%, 0.07%, and 0.35%, respectively. Similar results are also reported in the DCP work (Zhuang et al. 2018). We hypothesize that the overfitting problem on small-scale datasets like CIFAR-10 can be partially solved by pruning redundant channels.

4.2 Results on ImageNet

To evaluate the effectiveness of our CPLI approach on large-scale datasets, we further conduct the experiments to prune ResNet-50 (He et al. 2016) and MobileNet-V2 (Sandler et al. 2018) on the ILSVRC-12 dataset (Russakovsky et al. 2015). The ILSVRC-12 dataset is a large-scale dataset, which contains 1.28 million training images and 50k testing images from 1000 categories. Due to the shortcut design in the residual block, we cannot directly prune the first layer of the residual block. Therefore, we follow the method in (He, Zhang, and Sun 2017) to prune the multi-branch networks like ResNet. For ResNet-50, we follow the setting in (Lin et al. 2019) to fine-tune the pruned model with hint (Romero et al. 2015) from the last layer. The initial learning rate is set to 0.01 and the batch size is set to 128. The other settings are the same as those on the CIFAR-10 dataset. For MobileNet-V2, we use the same setting as that for ResNet-50 except that the batch size is set to 256.

Similar to the experiments on CIFAR-10, we compare our proposed approach with ThiNet (Luo, Wu, and Lin 2017), Channel Pruning (CP) (He, Zhang, and Sun 2017), Width-multiplier (WM) (Howard et al. 2017), GAL (Lin et al. 2019) and DCP (Zhuang et al. 2018) in Table 2. Consistent with the experiments on the CIFAR-10 dataset, for ResNet-50, our CPLI approach outperforms DCP and GAL by 0.23% and **1.67%**, respectively.

Since the work in (Zhuang et al. 2018) only reports the Top-1 accuracy of the uncompressed model for MobileNet-V2, we follow the setting in (Zhuang et al. 2018) and report the Top-1 accuracy drop after the model compression process. Again, our proposed approach surpasses the baseline methods ThiNet, WM, and DCP by **1.52%**, **1.56%**, and **1.05%**, respectively, which are **significant improvements** on the ImageNet dataset. The results on the ImageNet dataset clearly demonstrate that it is beneficial to prune channel by using our approach on the large-scale datasets.

4.3 Results on UCF-101

In order to demonstrate the generalization ability of our CPLI approach, we further compress the C3D model (Tran et al. 2015) on the UCF-101 dataset (Soomro, Zamir, and Shah 2012) for the action recognition task. The UCF-101 dataset consists of 13320 videos with the resolution of 320 × 240. We uniformally extract a set of frames from the videos at the rate of 25 fps. In the fine-tuning process, we resize the frames into 128 × 171 and randomly crop the frames to the resolution of 112 × 112. For testing, all frames are center cropped to the resolution of 112 × 112. We also split the frames of each video into several non-overlapped clips,

Table 3: Clip-level accuracy drops after compressing C3D on UCF-101 by using different channel pruning approaches. The clip-level accuracy of the pre-trained model is 79.93%. We directly quote the results from (Zhang et al. 2018b).

Model		TP	FP	RBP	**Ours**
C3D	CR	2×	2×	2×	**2×**
	Clip-level Acc. drop (%)	-11.50	-4.92	-3.56	**-2.18**

Table 4: Accuracy drops after pruning VGGNet under 2.86× compression ratio on CIFAR-10 without considering the final loss (FL) and the feature importance (FI). For reference, in our implementation, the accuracy of the uncompressed VGGNet model is 93.99%. "+" denotes that the accuracy increases.

Methods	Acc. drop (%)
CPLI (Baseline)	+0.96
CPLI w/o FL	+0.04
CPLI w/o FI	+0.47

where each clip contains 16 frames. These clips are used as the inputs of the C3D network.

Following (Zhang et al. 2018b), we compare our method with Taylor Pruning (TP) (Molchanov et al. 2017), Filter Pruning (FP) (Li et al. 2016), and Regularization-based pruning (RBP) (Zhang et al. 2018b) in terms of the clip-level accuracy drop. The work in FP selects the channels based on the magnitude of the features. In RBP, the authors add the regularization terms to the loss function and remove the channels based on the l_1 norm of the weights. The results are shown in Table 3. Again, our CPLI method outperforms the baseline methods TP, FP, and RBP, which demonstrates the generalization ability of our proposed method for compressing the C3D model.

4.4 Discussion

The results on CIFAR-10, ImageNet and UCF-101 clearly demonstrate the effectiveness of our CPLI approach for different datasets, network structures, and tasks. Our method performs better than the loss minimization methods, such as TP, because the channel selection process is based on the weights learned by solving the least square optimization problem. It converges better than the alternative approaches where the weights are learned by several rounds of fine-tuning. Compared with the layer-by-layer methods (He, Zhang, and Sun 2017; Luo, Wu, and Lin 2017), the performance improvement comes from the consideration of the final loss and the feature importance.

4.5 Ablation Study

In this section, we take pruning VGGNet under 2.86× compression ratio on the CIFAR-10 dataset as an example to investigate each component in our CPLI approach.

Effect of the classification loss. To investigate the effectiveness after taking the classification loss into account in the channel pruning process, we prune the channels without considering the final loss by setting the term $\frac{\partial \mathcal{C}}{\partial y_i}$ in Eq. (9) to 1, which is referred as CPLI w/o FL in Table 4. In this case, we only address the *next-layer feature removal* problem during the channel pruning process but ignore the final loss. In Table 4, the accuracy drops 0.92% when compared with our proposed method, which shows that it is important to take the final loss into account in the channel pruning process.

Table 5: Comparison of accuracy drops after pruning VGGNet with 2.86× compression ratio on CIFAR-10 when using different number of sampled spatial locations. "+" denotes that the accuracy increases.

Number of points	Acc. drop (%)
1	+0.12
5	+0.77
10	+0.96
20	+0.97

Effect of feature importance. To investigate the effectiveness after considering the feature importance in the channel pruning process, we set the term γy_i^* in Eq. (9) as 1, which is denoted by CPLI w/o FI in Table 4. In this case, we only consider the classification loss but ignore the *next-layer feature removal* problem in the channel pruning process. In Table 4, the accuracy drops 0.49% when compared with the proposed method, which suggests that it is beneficial to address the *next-layer feature removal* issue.

Results using different number of sampled locations. We conduct more experiments to investigate the performance when using different number of sampled locations. The results are shown in Table 5. From Table 5, we observe that the accuracies increase when the number of spatial locations increases from 1 to 10, but the accuracies are almost the same when the number of spatial locations increases from 10 to 20. The results indicate that it is sufficient to use 10 spatial locations in the pruning process. Therefore, we choose 10 spatial locations to prune the original model for the trade-off between accuracy and speed.

5 Conclusion

In this work, we have proposed a new channel pruning approach called Channel Pruning guided by classification Loss and feature Importance (CPLI) to compress CNNs. Our method take the classification loss into account without frequently performing the fine-tuning process. We have also proposed an effective approach to address the *next-layer feature removal* problem, in which the uninformative features can be ignored in the pruning process. Comprehensive experiments on three benchmark datasets clearly demonstrate the effectiveness of our newly proposed CPLI approach for model compression.

References

Ding, X.; Ding, G.; Guo, Y.; and Han, J. 2019. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *CVPR*.

Figurnov, M.; Ibraimova, A.; Vetrov, D. P.; and Kohli, P. 2016. Perforatedcnns: Acceleration through elimination of redundant convolutions. In *NeurIPS*.

Gong, Y.; Liu, L.; Yang, M.; and Bourdev, L. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *NeurIPS*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

He, Y.; Lin, J.; Liu, Z.; Wang, H.; Li, L.-J.; and Han, S. 2018. AMC: Automl for model compression and acceleration on mobile devices. In *ECCV*.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *ICCV*.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Hu, H.; Peng, R.; Tai, Y.-W.; and Tang, C.-K. 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*.

Jaderberg, M.; Vedaldi, A.; and Zisserman, A. 2014. Speeding up convolutional neural networks with low rank expansions. In *BMVC*.

Kim, Y.-D.; Park, E.; Yoo, S.; Choi, T.; Yang, L.; and Shin, D. 2015. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Lebedev, V.; Ganin, Y.; Rakhuba, M.; Oseledets, I.; and Lempitsky, V. 2014. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*.

LeCun, Y.; Denker, J. S.; and Solla, S. A. 1990. Optimal brain damage. In *NeurIPS*.

Lemaire, C.; Achkar, A.; and Jodoin, P.-M. 2019. Structured pruning of neural networks with budget-aware regularization. In *CVPR*.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2016. Pruning filters for efficient convnets. *ICLR*.

Lin, S.; Ji, R.; Yan, C.; Zhang, B.; Cao, L.; Ye, Q.; Huang, F.; and Doermann, D. 2019. Towards optimal structured cnn pruning via generative adversarial learning. In *CVPR*.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning efficient convolutional networks through network slimming. In *CVPR*.

Luo, J.-H.; Wu, J.; and Lin, W. 2017. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2017. Pruning convolutional neural networks for resource efficient inference. *ICLR*.

Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; and Kautz, J. 2019. Importance estimation for neural network pruning. In *CVPR*.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2015. Fitnets: Hints for thin deep nets. *ICLR*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV* 115(3):211–252.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv*.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. Ucf101: A dataset of 101 human actions classes from videos in the wild.

Tran, D.; Bourdev, L.; Fergus, R.; Torresani, L.; and Paluri, M. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*.

Xue, J.; Li, J.; and Gong, Y. 2013. Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*.

Yu, R.; Li, A.; Chen, C.-F.; Lai, J.-H.; Morariu, V. I.; Han, X.; Gao, M.; Lin, C.-Y.; and Davis, L. S. 2018. NISP: Pruning networks using neuron importance score propagation. In *CVPR*.

Zhang, X.; Zou, J.; He, K.; and Sun, J. 2016. Accelerating very deep convolutional networks for classification and detection. *T-PAMI* 38(10):1943–1955.

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018a. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*.

Zhang, Y.; Wang, H.; Luo, Y.; and Hu, R. 2018b. Three dimensional convolutional neural network pruning with regularization-based method. *NeurIPS Workshop*.

Zhao, C.; Ni, B.; Zhang, J.; Zhao, Q.; Zhang, W.; and Tian, Q. 2019. Variational convolutional neural network pruning. In *CVPR*.

Zhuang, Z.; Tan, M.; Zhuang, B.; Liu, J.; Guo, Y.; Wu, Q.; Huang, J.; and Zhu, J. 2018. Discrimination-aware channel pruning for deep neural networks. In *NeurIPS*.