CATRO: Channel Pruning via Class-Aware Trace Ratio Optimization

Wenzheng Hu, Zhengping Che, Member, IEEE, Ning Liu, Member, IEEE, Mingyang Li, Fellow, IEEE, Changshui Zhang, Fellow, IEEE, and Jianqiang Wang

Abstract—Deep convolutional neural networks are shown to be overkill with high parametric and computational redundancy in many application scenarios, and an increasing number of works have explored model pruning to obtain lightweight and efficient networks. However, most existing pruning approaches are driven by empirical heuristics and rarely consider the joint impact of channels, leading to unguaranteed and suboptimal performance. In this article, we propose a novel channel pruning method via class-aware trace ratio optimization (CATRO) to reduce the computational burden and accelerate the model inference. Utilizing class information from a few samples, CATRO measures the joint impact of multiple channels by feature space discriminations and consolidates the layerwise impact of preserved channels. By formulating channel pruning as a submodular set function maximization problem, CATRO solves it efficiently via a two-stage greedy iterative optimization procedure. More importantly, we present theoretical justifications on convergence of CATRO and performance of pruned networks. Experimental results demonstrate that CATRO achieves higher accuracy with similar computation cost or lower computation cost with similar accuracy than other state-of-the-art channel pruning algorithms. In addition, because of its class-aware property, CATRO is suitable to prune efficient networks adaptively for various classification subtasks, enhancing handy deployment and usage of deep networks in real-world applications.

Index Terms—Compression, deep model, pruning, subtask, trace ratio.

I. INTRODUCTION

In the past few years, deep convolutional neural networks (CNNs) have achieved impressive performance in computer vision tasks, especially image classification [1], [2]. However, deep models often have an enormous number of parameters, which require colossal memory and massive amounts of computation. These requirements not only increase infrastructure costs, but also impose a great challenge to deploy models in resource-constrained environments, including mobile devices, embedded systems, and autonomous robots. Therefore, it is significant to obtain deep models with high accuracy but relatively low computation used in various scenarios. Pruning is an effective way to accelerate and compress deep networks by removing less important connections in the network. Channel pruning [3], [4], [5], as a hardware-friendly method, directly reduces redundancy computation by removing channels in convolutional layers and is widely used in practice.

Among recent advances in channel pruning, most approaches are driven by empirical heuristics and rarely consider channel dependency. Many pruning methods directly measure the importance of individual channels by the weights of filters [3], [4], [6], and others introduce some intuitive losses [5], [7], [8] about reconstruction, discrimination, sparsity, and so on. However, neglecting the dependency among channels leads to suboptimal pruning results. For example, even if multiple channels are important when examined independently, keeping them all in a pruned network may still be redundancy because of similar extracted features from different channels. On the contrary, two or more individually unimportant channels may have boosted impact when working together. Such a joint impact has been moderately studied in machine-learning methods [9], [10], [11] but is rarely exploited in channel pruning. CCP [12] moves one step ahead with only pairwise channel correlations in pruning, and it remains demanding and challenging for effective utilizations and rigorous investigations of multiple channel joint impact.

In this article, we develop a novel channel pruning method. As shown in Fig. 1, our method benefits from taking the joint impact of channels into account and is superior to most existing methods pruning channels independently. The proposed method extends network discriminative power [8], [13] for multichannel setting and finds channels by solving a combinatorial optimization problem. To be more specific, we first transfer the channel combination optimization into a graph trace optimization and then we solve the trace optimization in a global view to obtain a result with the consideration of the joint impact. By formulating the pruning objective into
a submodular set function maximization problem, we further show a theoretical lower bound guarantee on accuracy and convergence under an assumption about the relationship between discrimination and accuracy.

In addition, different from most approaches measuring the channel importance by the filter weights, the discrimination in class-aware trace ratio optimization (CATRO) is based on input samples and class-aware in the optimization steps. This enables CATRO to adaptively mine subsets of data samples and handle various pruning tasks with the same original network. For example, suppose we have a large traffic sign classification model that works perfectly in the whole world, and we need to deploy a tiny model on an autonomous vehicle operated in a small town, where only subset types of signs are on the road. This kind of customization and compression combination tasks are ubiquitous in the real world and require efficient solutions. We name such application scenarios classification subtask compression and showcase that CATRO is quite suitable in these scenarios.

To the best of our knowledge, CATRO is the first to exploit the multichannel joint impact to find the optimal combination of preserved channels in pruning and the first to use submodularity for effective and theoretically guaranteed pruning performance. We summarize our contributions as follows.

1) We introduce graph-based trace ratio as discrimination criteria to consider the multichannel joint impact for channel pruning.
2) We formulate the pruning to a submodular maximization problem, provide theoretical guarantees on convergence of the pruning step and performance of pruned networks, and solve it with an efficient two-stage algorithm.
3) We demonstrate in extensive experiments that our method achieves superior performance compared to others. Variational CNN pruning [27] provides a Bayesian model compression technique, which approximates batch-norm scaling parameters to a factorized normal distribution using stochastic variational inference. GBN [28] introduces a gate for each channel and performs a backward method to optimize the gates. Then it independently prunes channels by the rank of gates. Similarly, DCPH [23] designs a channelwise gate to enable or disable each channel. The greedy algorithm is also applied to identify the subnetwork inside the original network [29]. Recently, MFP [22] introduces a new set of criteria to consider the geometric distance of filters. These works evaluate and prune the importance of channels individually. Another work [13] leverages Fisher’s linear discriminant analysis (LDA) to prune the last layer of the DNN, which, however, is an empirical heuristic and suffers from the expensive cost of cross-layer tracing. CCP [12] exploits the interchannel dependency but stops with only pairwise correlations. It prunes channels without global consideration either. SCOP [30] utilizes samples to set up a scientific control and then prunes the neural network in the group by introducing generated input features. Recent prevailing AutoML-based pruning approaches [31], [32], [33], [34], [35], [36] can achieve a high FLOP reduction of DNNs. For instance, AMC [31] leverages deep reinforcement learning (DRL) to let the agent optimizes the policy from extensive experiences by repeatedly pruning and evaluating for DNNs. DNC [21] uses a set of learnable architecture parameters and calculates the probability of each channel being retained based on these parameters. However, they usually require tremendous computation budgets due to the large search space and trial-and-error manner.

II. RELATED WORK

A. Channel Pruning

The recent advances in weight pruning can be mainly categorized into unstructured pruning and channel pruning. Unstructured pruning can effectively prune the over-parameterized deep neural networks (DNNs) but results in irregular sparse matrices, which require customized hardware and libraries to support the practical speedup, leading to a limited inference acceleration in most practical cases [14], [15]. Recent work concentrates on channel pruning [4], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], which prunes the whole channel and results in structured sparsity. The compacted DNN after channel pruning can be directly deployed on the existing prevalent CNN acceleration frameworks without dedicated hardware/libraries implementation. Early work [26] proposes a straightforward pruning heuristic, penalizing weights with $l_1$-norm group lasso regularization and then eliminating the least $l_1$-norm of channels. Network Slimming [16] introduces a factor in the batch normalization layer with respect to each channel and evaluates the importance of channels by the magnitude of the factor. Then, many criteria are proposed to evaluate the importance of neurons. SCP [6] measures the relative importance of a filter in each layer by calculating the $l_1$-norm and $l_2$-norm, respectively. FPGM [4] removes the filters minimizing the sum of distances to others. Variational CNN pruning [27] provides a Bayesian model compression technique, which approximates batch-norm scaling parameters to a factorized normal distribution using stochastic variational inference. GBN [28] introduces a gate for each channel and performs a backward method to optimize the gates. Then it independently prunes channels by the rank of gates. Similarly, DCPH [23] designs a channelwise gate to enable or disable each channel. The greedy algorithm is also applied to identify the subnetwork inside the original network [29]. Recently, MFP [22] introduces a new set of criteria to consider the geometric distance of filters. These works evaluate and prune the importance of channels individually. Another work [13] leverages Fisher’s linear discriminant analysis (LDA) to prune the last layer of the DNN, which, however, is an empirical heuristic and suffers from the expensive cost of cross-layer tracing. CCP [12] exploits the interchannel dependency but stops with only pairwise correlations. It prunes channels without global consideration either. SCOP [30] utilizes samples to set up a scientific control and then prunes the neural network in the group by introducing generated input features. Recent prevailing AutoML-based pruning approaches [31], [32], [33], [34], [35], [36] can achieve a high FLOP reduction of DNNs. For instance, AMC [31] leverages deep reinforcement learning (DRL) to let the agent optimizes the policy from extensive experiences by repeatedly pruning and evaluating for DNNs. DNC [21] uses a set of learnable architecture parameters and calculates the probability of each channel being retained based on these parameters. However, they usually require tremendous computation budgets due to the large search space and trial-and-error manner.

B. Submodular
of interest in lots of computer tasks, such as visual recognition [37], segmentation [38], clustering [39], [40], active learning [41], and user recommendation [42]. The submodular set function maximization [43], which is to maximize a submodular function, is one of the most basic and important problems. There exist several algorithms for solving non-negative and monotone submodular function maximization. Under the uniform matroid constraint [43], a standard greedy algorithm gives an $1 - e^{-1}$ approximation [44], [45], [46] in polynomial time, and $1 - e^{-1}$ is shown to be the best possible approximation unless $P = NP$ [44]. There are also many works investigating more effective algorithms [47], [48] and many other works investigating the maximization of a submodular set function under different constraints [49], [50], [51]. Although submodularity has been widely used in many tasks and widely studied, it has not been explored in model compression as far as we know.

III. METHODOLOGY

A. Problem Reformulation

For a CNN $F(\cdot; \Theta)$ with a stack of $L$ convolution (CONV) layers and weight parameters $\Theta$, given a set of $N$ image samples $\{x^{[i]}\}_{i=1}^{N}$ from $K$ classes and the corresponding labels $\{y^{[i]} \in \{1, \ldots, K\}\}_{i=1}^{N}$, channel pruning aims to find binary channel masks $\mathbf{m}_l$ with the following objective:

$$
\min_{\Theta, \mathbf{m}_l} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(x^{[i]}, y^{[i]}, \Theta, \mathbf{m}_l^{i})
$$

s.t. $T(F(\cdot; \Theta, \mathbf{m}_l^{i})) \leq T_B$, $\mathbf{m}_l \in \{0, 1\}^c_l$

(1)

where $\mathcal{L}(\cdot)$ is the loss function (e.g., cross-entropy for classification), $c_l$ is the channel number of the $l$th layer, $T(\cdot)$ is the FLOPs function, and $T_B$ is a given FLOP constraint.

Most methods prune channels with some surrogate layer-wise criteria $\mathcal{C}(\cdot)$ instead of directly solving (1). Following this setting and with the goal of keeping $d_l$ channels in layer $l$, we can express the pruning objective as follows:

$$
\text{Optimize } \frac{1}{N} \sum_{i=1}^{N} \mathcal{C}(o^{[i]}_l \odot \mathbf{m}_l)
$$

s.t. $\mathbf{m}_l \in \{0, 1\}^c_l$, $\|\mathbf{m}_l\|_0 = d_l$

(2)

where $o^{[i]}_l \in \mathbb{R}^{d_l \times w_l \times h_l}$ is the feature map of input $x^{[i]}$ in the $l$th layer, and $\odot$ is elementwise multiplication with broadcasting. A variety of $\mathcal{C}(\cdot)$ have been proposed [4], [5], [7], [8], which may not be completely equivalent to solving (1) but effective in practice.

Alternatively, we can formulate (2) as a matrix projection problem. For clarity, we denote the $i$th element in a vector (or an ordered set) as $a(i)$. We build an ordered set of the channel index $I_l = \{i | \mathbf{m}_l(i) = 1\}$ with $|I_l| = d_l$ and an indicator matrix $V_{I_l} = [e^{[i]}_{l(1)}, \ldots, e^{[i]}_{l(d_l)}] \in \mathbb{R}^{c_l \times d_l}$, where $e^{[i]}_{l} \in \{0, 1\}^c_l$ denotes the one-hot vector with its $i$th element as one. Therefore, (2) can be expressed as follows:

$$
\text{Optimize } \frac{1}{N} \sum_{i=1}^{N} \mathcal{C}(V_{I_l}^T o^{[i]}_l)
$$

s.t. $\text{rank}(V_{I_l}) = d_l$.

(3)

B. Trace Ratio Criterion for Channel Pruning

To obtain effective criteria for model pruning, inspired by the classical Fisher LDA [52], we dig into the feature discrimination for measuring multichannel joint impact. As a simple example, Fig. 2 visualizes two feature spaces from the 18th layer in a well-trained ResNet-20 model trained on CIFAR-10. The red and blue points represent features of images containing ships and airplanes, respectively. We have an intuitive feeling that the features produced by channels in the right figure are more discriminative, and a classifier built on that space is more likely to outperform that on the left space. For deep CNNs with more layers, we hypothesize that the discrimination affects accumulate and the classifier performance gap enlarges. We describe the relation between space discrimination and classifier accuracy in Assumption 1.

Assumption 1: For a classification task, if samples are more discriminatory in feature space $S_1 \in \mathbb{R}^d$ than in feature space $S_2 \in \mathbb{R}^d$, it holds that the accuracy of a classifier built on $S_1$, namely $A(S_1)$, is no less than that of the classifier built on $S_2$ with large probability, that is, $A(S_1) \geq_r A(S_2)$, where $\geq_r$ means “no less with probability $p$.”

The above assumption enables us to concretely measure and optimize the discriminations. Suppose $X = [x^{[1]}, \ldots, x^{[N]}] \in \mathbb{R}^{d \times N}$ denotes $N$ data samples from the training set with $K$ classes and $y = [y^{[1]}, \ldots, y^{[N]}]$ denotes the corresponding class label. We build two graph weight matrices $G^{(w)} \in \mathbb{R}^{N \times N}$ and $G^{(b)} \in \mathbb{R}^{N \times N}$ to depict the prior knowledge about data sample relationship. $G^{(w)}$ reflects within-class relationship, where each element $G^{(w)}_{i,j}$ has a larger value if samples $x^{[i]}$ and $x^{[j]}$ belong to the same class, and smaller otherwise. Analogically, $G^{(b)}$ represents the between-class relationship, where $G^{(b)}_{i,j}$ is larger if samples $x^{[i]}$ and $x^{[j]}$ belong to different classes. Without any other priors, we directly use the Fisher score [52] as the graph metric for its simplicity and efficiency

$$
G^{(w)}_{i,j} = \begin{cases} 
\frac{1}{n_k}, & y^{[i]} = y^{[j]} = k \\
0, & y^{[i]} \neq y^{[j]} 
\end{cases}
$$

$$
G^{(b)}_{i,j} = \frac{1}{N} - G^{(w)}_{i,j} 
$$

(4)

(5)

where $n_k$ is the number of samples in the $k$th class, and $\sum_{k=1}^{K} n_k = N$.\[\]
Given Assumption 1 and the prior matrices in (4) and (5), a natural criterion $C(\cdot)$ for channel pruning is to keep channels that reduce the discrimination of samples within the same class ($\sum_{ij} \|\tilde{O}_{i,j}^{[l]} - \hat{O}_{i,j}^{[l]}\|_{G_{i,j}^{[l]}}$) and enlarge that of samples from different classes ($\sum_{ij} \|\hat{O}_{i,j}^{[l]} - \hat{O}_{i,j}^{[l]}\|_{G_{i,j}^{[l]}}$). Therefore, we can replace (3) by

$$\max_{I_l} \frac{\sum_{ij} \|V_{i,j}^{[l]} - V_{i,j}^{[l]}\|_{G_{i,j}^{[l]}}}{\sum_{ij} \|V_{i,j}^{[l]} - V_{i,j}^{[l]}\|_{G_{i,j}^{[l]}}} \quad \text{s.t. } |I_l| = d_l.$$  

(6)

As $A^{(w)}$ is a diagonal matrix with $A_{i,j}^{(w)} = \sum_j G_{i,j}^{(w)}$ and $A^{(b)}$ is a diagonal matrix with $A_{i,j}^{(b)} = \sum_j G_{i,j}^{(b)}$, $A^{(w)} - G^{(w)}$ and $A^{(b)} - G^{(b)}$ are two Laplace matrices. Based on the Laplace matrix properties, (6) is mathematically equivalent to the following trace optimization:

$$\max_{I_l} \frac{\text{tr}(V_{I_l}^\top O_{I_l}(A^{(b)} - G^{(b)})O_{I_l}^\top V_{I_l})}{\text{tr}(V_{I_l}^\top O_{I_l}(A^{(w)} - G^{(w)})O_{I_l}^\top V_{I_l})} \quad \text{s.t. } |I_l| = d_l.$$  

(7)

where $O_{I_l} = [o^{[w]}_1, \ldots, o^{[w]}_{|l|}] \in \mathbb{R}^{|l| \times |l| \times h \times N}$ is the aggregated feature map for $X$ in layer $l$, and $\text{tr}(\cdot)$ denotes the trace. Equivalently, our goal is to find the optimal channel index set $I_l^*$ with the maximum trace ratio criterion for discrimination

$$\lambda_{l}^* = \frac{\text{tr}(V_{I_l}^\top G_{I_l}^{(b)} V_{I_l})}{\text{tr}(V_{I_l}^\top G_{I_l}^{(w)} V_{I_l})},$$  

(8)

where $G_{I_l}^{(w)} = O_{I_l}(A^{(w)} - G^{(w)})O_{I_l}^\top$ and $G_{I_l}^{(b)} = O_{I_l}(A^{(b)} - G^{(b)})O_{I_l}^\top$. Since $\text{tr}(V_{I_l}^\top (G_{I_l}^{(b)} - \lambda_{l}^* G_{I_l}^{(w)}) V_{I_l}) = 0$, for all $I_l$ satisfying $|I_l| = d_l$, we have

$$\frac{\text{tr}(V_{I_l}^\top G_{I_l}^{(b)} V_{I_l})}{\text{tr}(V_{I_l}^\top G_{I_l}^{(w)} V_{I_l})} \leq \lambda_{l}^*.$$  

(9)

$$\Rightarrow \text{tr}(V_{I_l}^\top (G_{I_l}^{(b)} - \lambda_{l}^* G_{I_l}^{(w)}) V_{I_l}) \leq 0.$$  

(10)

Therefore, to optimize (7) is further equivalent to optimize the following optimization:

$$\max_{I_l} \text{tr}(V_{I_l}^\top (G_{I_l}^{(b)} - \lambda_{l}^* G_{I_l}^{(w)}) V_{I_l}) \quad \text{s.t. } |I_l| = d_l.$$  

(11)

To be more clear, the objective in (7) is the left-hand side terms of the equation in (9), and the optimal solution $I_l^*$ of (7) makes the equality of the equation in (9) hold (by the definition of $\lambda_{l}^*$). And similarly, given $\lambda_{l}^*$, the objective in (11) is the left-hand side terms of the equation in (10), and the solution $I_l$ of (11) makes the equality of the equation in (10) hold. Lastly, the equality of the equation in (9) holds if and only if the equality of the equation in (10) holds. Thus, the optimal solution of (7) makes equality of the equation in (9) hold, makes equality of the equation in (10) hold, and is also the optimal solution of (11). In contrast, the optimal solution of (11) makes equality of the equation in (10) hold, makes equality of the equation in (9) hold, and is also the optimal solution of (7).

Noting that solving (11) naturally considers the multichannel joint impacts inherited in $V$.  

C. Iterative Layerwise Optimization

To solve (11), we introduce a submodular set optimization, provide theoretical guarantees on the pruned model accuracy, and propose a layerwise pruning procedure.

First, we introduce $s_{l,i} = \exp(e_{i}^\top (G_{I_l}^{(b)} - \lambda_{l}^* G_{I_l}^{(w)}) e_{i})$ as the discrimination score for channel $i$ in layer $l$ given a specified ratio $\lambda_{l}$, and $H_l(I) = \log(\sum_{i \in I_l} s_{l,i})$ as the index set function given an index set $I \subseteq \{1, \ldots, c_l\}$. When $\lambda_{l} = \lambda_{l}$, we notice that (11) is equivalent to the following one:

$$\max_{I_l} H_l(I; \lambda_{l}) \quad \text{s.t. } |I_l| = d_l.$$  

(12)

We introduce two nice properties of set functions and show $H_l(\cdot)$ satisfies them.

Definition 1 (Submodularity): A set function $f(S) : 2^d \rightarrow \mathbb{R}$ is submodular if for any subset $S_1 \subseteq S_2 \subseteq U$ and any element $i \in U \setminus S_2$, it has $f(S_1 \cup \{i\}) - f(S_1) \geq f(S_2 \cup \{i\}) - f(S_2)$.

Definition 2 (Monotonicity): A set function $f(S) : 2^d \rightarrow \mathbb{R}$ is monotone if for any subset $S_1 \subseteq S_2 \subseteq U$ and any element $i \in U \setminus S_2$, it has $f(S_1) \leq f(S_2)$.

Lemma 1: Given a $\lambda_{l}$, $H_l(\cdot)$ is a set function with monotonicity and submodularity.

Proof: Suppose $I \subseteq I' \subseteq U_l = \{1, \ldots, c_l\}$, where $c_l$ is the number of channels in layer $l$. For any $i \in U \setminus I'$, we have

$$H_l(I') - H_l(I) = \log \left(1 + \frac{\sum_{i \in I' \setminus S_{l,i}} s_{l,i}}{\sum_{i \in I \setminus S_{l,i}} s_{l,i}}\right) > 0$$

and

$$H_l(I \cup \{i\}) - H_l(I) = \log \left(1 + \frac{S_{l,i}}{\sum_{i \in I' \setminus S_{l,i}} s_{l,i}}\right) > \log \left(1 + \frac{S_{l,i}}{\sum_{i \in I \setminus S_{l,i}} s_{l,i}}\right) = H_l(I \cup \{i\}) - H_l(I).$$

Therefore, $H_l(\cdot)$ is a set function with monotonicity and submodularity.

Second, to maximize $H_l(I)$ with a fixed $\lambda_{l}$, we show that sequentially selecting the indexes $i$ with the largest score $s_{l,i}$ obtains a model with a guaranteed accuracy according to Theorem 1.

Theorem 1: Given a $\lambda_{l}$, maximizing $H_l(\cdot)$ by sequentially selecting $d_l$ elements with the highest scores from $\{s_{l,i}|1 \leq i \leq c_l\}$, with $\tilde{I}_l$ representing the selected index set, results in a model that, with a large probability $p$, has a lower boundary on the accuracy related to the optimal index set $I_l^*$, that is,

$$A(\tilde{I}_l; \lambda_{l}) \geq p \cdot \Phi \left\{ \left(1 - \frac{1}{e}\right) H_l(I_l^*; \lambda_{l}) \right\}$$

where $\Phi(\cdot)$ is the monotone nondecreasing mapping from $H_l(\cdot)$ to accuracy $A(\cdot)$ under Assumption 1.

Proof: Suppose the optimal solution is $I_l^* = \{i_1, \ldots, i_{d_l}\}$, the index set obtained by sequentially selected top $d_l$ elements is $\tilde{I}_l = \{i_1, \ldots, i_{d_l}\}$. Let $I_{t}^* = \{i_1, \ldots, i_t\}$ represent the index set at step $t$ of the sequential selection. Since $H_l(\cdot)$ is
monotonic and submodular, following the similar proof [43], we have the inequalities:
\[
H_t(I_t^r) \leq H_t(I_t^r \cup I_t^{r+1}) = H_t(I_t^r) + (H_t(I_t^r \cup \{i_t^r\}) - H_t(I_t^r)) + (H_t(I_t^r \cup \{i_t^r, i_t^{r+1}\}) - H_t(I_t^r \cup \{i_t^r\})) + \ldots + (H_t(I_t^r \cup I_t^r) - H_t(I_t^r \cup \{i_t^r, \ldots, i_t^n\})) \\
\leq H_t(I_t^r) + \delta_t(I_t^r \cup I_t^{r+1}) - H_t(I_t^r))
\]
and then we have
\[
H_t(I_t^r) - H_t(I_t^{r+1}) \leq \left(1 - \frac{1}{d_t}\right)H_t(I_t^r) - H_t(I_t^r) \\
\Rightarrow H_t(\hat{I}_t) \geq \left(1 - \frac{1}{d_t}\right)H_t(I_t^r) \Rightarrow H_t(\hat{I}_t) \geq \left(1 - \frac{1}{d_t}\right)H_t(I_t^r).
\]
When \(\hat{I}_t\) is obtained, we can calculate a discrimination
\[
\hat{\lambda}_t = \frac{\mathbf{tr}(\mathbf{V}_{\hat{I}_t}^T \mathbf{G}_t^{(w)} \mathbf{V}_{\hat{I}_t})}{\mathbf{tr}(\mathbf{V}_{\hat{I}_t}^T \mathbf{G}_t^{(b)} \mathbf{V}_{\hat{I}_t})}.
\]
Thus, \(H_t(\hat{I}_t)\) can be regarded as a description of discrimination. Under the assumption 1, let \(\Phi(\cdot)\) is the monotone non-decreasing mapping from \(H_t(\cdot)\) to the accuracy \(A(\cdot)\), we have such an accuracy lower boundary with a large probability \(p\)
\[
A(\hat{I}_t) \geq_p \Phi \left(1 - \frac{1}{d_t}\right)H_t(I_t^r).
\]
We can further express the boundary as
\[
A(I_t; \lambda_t) \geq_p \Phi \left(1 - \frac{1}{d_t}\right)H_t(I_t^r; \lambda_t).
\]
Finally, we consider iteratively optimizing the discrimination ratio. Suppose the current index set is \(I_t\) and the discrimination ratio is \(\lambda_t\). After maximizing \(H_t(I_t; \lambda_t)\) with the index selection procedure, we get a new index set \(I_t^{r+1}\) and a new ratio \(\lambda_t^{r+1}\). From Theorem 2, we have \(\lambda_t^{r+1} \geq \lambda_t\).

Theorem 2: The \(\lambda_t\) obtained by iteratively maximizing \(H_t(\cdot)\) is monotonic increasing and converging to the optimal value.

Proof: Under the specific indicator matrix \(\mathbf{V}_{I_t} = [e_{I_t(1)}, \ldots, e_{I_t(d)}] \in \mathbb{R}^{c_t \times d_t}\), where \(e_{I_t(i)} \in \{0, 1\}\) denotes the one-hot vector with its \(i\)th element as one, we have the following equivalent optimizations:
\[
\max_{I_t} \exp \left(\mathbf{tr}(\mathbf{V}_{I_t}^T (\mathbf{G}_t^{(b)} - \lambda_t \mathbf{G}_t^{(w)}) \mathbf{V}_{I_t})\right) \\
\max_{I_t} \sum_{i=1}^{d_t} (e_{I_t(i)}^T (\mathbf{G}_t^{(b)} - \lambda_t \mathbf{G}_t^{(w)}) e_{I_t(i)})^{(i)} \\
\max_{I_t} \log \left(\sum_{i \in I_t} s_{I_t(i)}\right) \\
\max_{I_t} H_t(I_t; \lambda_t).
\]
The first equivalence relation comes from the definition of trace and \(\mathbf{V}_t\), the second equivalence relation comes from the monotonicity of exponential function and logarithmic function, and the last comes from the definition of \(H_t(\cdot)\).

Suppose the current index set (of the iteration round \(r\)) is \(I_t^r\) and the discrimination is \(\lambda_t^r\), and after maximizing \(H_t(I_t; \lambda_t^r)\), we get the next index set \(I_t^{r+1}\). Let
\[
f(\lambda_t^r) = \max_{I_t} \exp \left(\mathbf{tr}(\mathbf{V}_{I_t}^T (\mathbf{G}_t^{(b)} - \lambda_t^r \mathbf{G}_t^{(w)}) \mathbf{V}_{I_t})\right) \\
= \exp \left(\mathbf{tr}(\mathbf{V}_{I_t}^T (\mathbf{G}_t^{(b)} - \lambda_t^r \mathbf{G}_t^{(w)}) \mathbf{V}_{I_t})\right) > 0
\]
then
\[
f(\lambda_t^r) = f(\lambda_t^r)(\mathbf{tr}(\mathbf{G}_t^{(b)} - \lambda_t^r \mathbf{G}_t^{(w)}) \mathbf{V}_{I_t}) \leq 0
\]
which means that \(f(\lambda_t^r)\) is a monotonic nonincreasing function. In addition, \(f(\lambda_t^r) = 1\) because of the definition of \(\lambda_t^r\).

Let \(h(\lambda_t)\) be a piecewise logarithmic linear approximation of \(f(\lambda_t)\) at point \(\lambda_t^r\), namely
\[
h(\lambda_t) = \exp \left(\frac{f(\lambda_t^r)}{f(\lambda_t^r)} (\lambda_t - \lambda_t^r) + \log f(\lambda_t^r)\right) > 0
\]
then we know \(h(\lambda_t^r) = f(\lambda_t^r)\), and \(h(\lambda_t) = h(\lambda_t^r) \cdot (f(\lambda_t^r))/\left(f(\lambda_t^r)\right) \leq 0\), which means \(h(\lambda_t)\) is also monotonic nonincreasing.

Let \(h(\lambda_t^r+1) = 1\), we have
\[
\lambda_t^{r+1} = \frac{\mathbf{tr}(\mathbf{V}_{I_t}^T (\mathbf{G}_t^{(b)} \mathbf{V}_{I_t}^{r+1})}{\mathbf{tr}(\mathbf{V}_{I_t}^T (\mathbf{G}_t^{(b)} \mathbf{V}_{I_t}^{r+1})}
\]
which is the same value as the discrimination score calculated with the next index set \(I_t^{r+1}\).

Since \(f(\lambda_t)\) is a monotonic nonincreasing function with optimal (i.e., the largest) \(\lambda_t^*\) satisfying \(f(\lambda_t^*) = 1\), we have
\[
h(\lambda_t^*) = f(\lambda_t^*) > f(\lambda_t^r) = 1 = h(\lambda_t^{r+1})
\]
for \(\forall \lambda_t^* \neq \lambda_t^r\). This shows \(\lambda_t^{r+1} > \lambda_t^*\) when \(\lambda_t^* 
\neq \lambda_t^r\).

In addition, the definition of \(\lambda_t\) guarantees that \(\lambda_t\) can only take a finite number of different values during the iterative optimization process.

Thus, the discrimination \(\lambda_t\) obtained by iteratively maximizing \(H_t(I_t; \lambda_t)\) is monotonic increasing until it converges to the optimal value.

The whole layerwise pruning procedure, given the channel budgets for each layer, is shown in Algorithm 1.

Theorems 2 and 1 show that, given a well-trained unpruned network and channel numbers \(d_t\), the pruning procedure in Algorithm 1 leads to monotonically increasing feature discrimination value until reaching the optimal solution (by Theorem 2) and obtains the pruned network with an accuracy lower bound proportional to the best-pruned network with large probability (by Theorem 1).

Note that the objective in each layer is not a simply linear sum of each channel. Determining the value of \(\lambda_t\) in \(s_{I_t(i)}\) incorporates the joint-channel impact and depends on other channels. With the update of \(\lambda_t\), channels selected in the
Algorithm 1 Layerwise Pruning With Trace Ratio Criterion

Input: $N$ sampled data $X \in \mathbb{R}^{C \times w \times h \times N}$ and their labels $y \in \{1, \ldots, K\}^N$; Original CNN model $F(\Theta)$ with $L$ CONV layers; Original and pruned channel number $c = [c_1, \ldots, c_L]$ and $d = [d_1, \ldots, d_L]$ for all layers; Stop criterion $\epsilon$.

Output: Channel masks $\{m_l\}_{l=1}^L$ for all layers.

1: Set $\hat{O}_0 \leftarrow X$.
2: for $l \leftarrow 1$ to $L$ do
3:   Calculate feature maps $O_l$ by $\hat{O}_{l-1}$ and CNN $F(\Theta)$.
4:   Randomly set channel index subset $I_l$ with $|I_l| = d_l$.
5:   Calculate $\lambda'_l$ with $I_l$.
6:   repeat
7:      Update $(I_l, \lambda'_l) \leftarrow (I_l', \lambda'_l)$.
8:      Calculate $\{s_{l,j}\}_{j=1}^{c_l}$ for all channels with $\lambda_l$.
9:      Set $I_l'$ by selecting $d_l$ indexes from $\{1, \ldots, c_l\}$ with the largest $s_{l,1}$.
10:    Calculate $\lambda'_l$ with $I_l'$.
11:   until $\lambda'_l - \lambda_l \leq \epsilon$.
12:   Set mask $m_l$ for layer $l$ based on $I_l$.
13:   Calculate pruned feature maps $\tilde{O}_l$ by $m_l$ and $\hat{O}_{l-1}$.
14: end for
15: return $\{m_l\}_{l=1}^L$.

Algorithm 2 Greedy Selection for Channel Numbers

Input: $N$ sampled data $X \in \mathbb{R}^{C \times w \times h \times N}$ and their labels $y \in \{1, \ldots, K\}^N$; Original CNN model $F(\Theta)$ with $L$ CONV layers; Initial channel number $d_{\text{min}}$; Step size $\eta$; FLOPs constraint $T_B$.

Output: Channel numbers $d = [d_1, \ldots, d_L]$ for all layers.

1: Calculate feature maps $\{O_l\}_{l=1}^L$ for all layers with $X$.
2: Set all $L$ values in $d$ by $d_{\text{min}}$.
3: Randomly select $d'_l$ channels in each layer and calculate $\{\lambda_l\}_{l=1}^L$ accordingly.
4: Calculate $\{\Gamma_l(d_l)\}_{l=1}^L$ by (15) with $\{O_l\}_{l=1}^L$ and $y$.
5: Set $l_{\text{max}} \leftarrow \arg \max_l \Gamma_l(d_l)$.
6: Set $d' \leftarrow d_{\text{max}} + \eta$.
7: while $\sum_{l_{\text{max}}} T_l(d_l) + T_{\text{max}}(d') \leq T_B$ do
8:   Update $d_{\text{max}} \leftarrow d'$.
9:   $\tilde{l}_{\text{max}} \leftarrow \arg \max_l \Gamma_l(d_l)$.
10: Update $\Gamma_{\text{max}}(d'_{\text{max}})$ and other changed terms in $\Gamma_l(d_l)$.
11: Set $l_{\text{max}} \leftarrow \arg \max_l \Gamma_l(d_l)$.
12: Set $d' \leftarrow d_{\text{max}} + \eta$.
13: end while
14: return $d = [d_1, \ldots, d_L]$.

Fig. 3. Pipeline of the overall pruning procedure in CATRO.

On the other hand, to measure the network computational efficiency, we define the FLOP function of layer $l$ as $T_l(d) = d_l - d_l w_l h_l q_l^2$, where $q_l \times q_l$, $w_l$, and $h_l$ are the $l$th layer’s kernel size, feature map width, and feature map height, respectively. The FLOP function for the entire network is $T(d) = \sum_{l=1}^L T_l(d_l)$, and its derivative for $d_l$ is as follows:

$$\frac{\partial T(d)}{\partial d_l} = d_l - w_l h_l q_l^2 + d_{l+1} w_{l+1} h_{l+1} q_{l+1}^2.$$  (14)

As a tradeoff to jointly considering computations and discriminations, we define the following function to evaluate the effect of changing each channel number $d_l$:

$$\Gamma(d_l) = \frac{\partial D}{\partial T}.$$  (15)

For the global architecture optimization, we manipulate the channel numbers without selecting the channels. Therefore, we take a greedy coordinate ascent strategy to find the optimal channel numbers, which is described in Algorithm 2. It is worth noting that, similar to the layerwise pruning procedure, this algorithm only performs inference once without backpropagation. After that, it only needs to recalculate a small portion of variables in each iteration.

E. Overall Pruning Procedure

We take the two stages together to build CATRO, the proposed channel pruning method. As illustrated in Fig. 3,
CATRO first employs Algorithm 2 (Greedy Selection for Channel Numbers) to select the channel numbers for each layer and then uses Algorithm 1 (Layerwise Pruning with Trace Ratio Criterion) to prune each layer. It fine-tunes the pruned network as the last step on all available training data, similar to other pruning methods.

IV. EXPERIMENTS

A. Datasets and Implementation Details

1) Datasets: We conducted our experiments on four datasets: CIFAR-10, CIFAR-100 [60], ImageNet [61], and GTSRB [62]. Both CIFAR-10 and CIFAR-100 contain 50 K training images and 10 K test images, and they have 10 and 100 classes, respectively. ImageNet contains 1.2 M training images and 50 K test images for 1000 classes. GTSRB, which is used to evaluate pruning performance for classification subtasks, contains 39 209 training images and 12 630 test images of 43 types of traffic signs.

2) Implementation Details: For CIFAR-10/100 and GTSRB, we trained the original model 300 epochs using SGD with momentum 0.9, weight decay parameter 0.0005, batch size 256, and the initial learning rate of 0.1 dropped the factor of 0.1 after the 80th, 150th and 250th epochs. The step size is 1. We fine-tuned 200 epochs for pruned models with an initial learning rate of 0.01 dropped by the factor of 0.1 after half epochs for fair comparisons. The models are trained on GPU Tesla P40 using Pytorch. For quantitative performance comparisons, all our models do not prune the first convolution layer, and the minimal number of channels is three the same as the input images. The number of channels is directly set to the minimum values. For the CIFAR-100 classification subtask compression, we moved about 3–5 epochs from fine-tune phrase to the pruning phrase after pruning each block while keeping the total number of backward the same. To reduce overfitting, we limited the maximal number of channels to half of the original model. For the GTSRB classification subtask compression, we further limited the maximal number of channels with (3/8) of the original model for the ultrahigh compression ratios. For all subtask compression experiments, to get the high compression ratio, the first convolution layer was pruned in the same way as the other layers. For the ImageNet, we trained 90 epochs, and the learning rate decayed by a factor of 0.1 after the 30th and 60th epochs.

3) Hyperparameters: The key hyperparameters are $d_{\text{min}}$ and the number of data samples. We studied different numbers of samples on CIFAR-10/100. Results showed that CATRO is robust within a large range of sample numbers. We chose the value of $d_{\text{min}}$ with the following considerations. First, $d_{\text{min}}$ should be a positive value to avoid cutting off the network or layer-collapse issues. Second, $d_{\text{min}}$ should be small enough to have a large search space. We empirically set $d_{\text{min}}$ to be 3, and we set all layers’ as the same value for ease of use. We found out that it may influence some layers’ pruning ratios, but the overall compression ratios are similar with almost the same accuracy performance. In practice, the minimum initial channel number $d_{\text{min}}$ is not sensitive to the overall channel number selection algorithm since most pruned layers are larger than the $d_{\text{min}}$.

B. Quantitative Performance Comparisons

1) ResNet on CIFAR: We compared the classification performance of our proposed method with several state-of-the-art channel pruning approaches on CIFAR-10 and CIFAR-100 with ResNet-20/32/56/110 [2]. As different baselines use different training efforts or strategies, the original accuracy values can be different. We reported both the pruned accuracy and the accuracy drop from all baselines’ original articles to draw fair comparisons. Results in Table I clearly demonstrate the superiority of our CATRO over other channel pruning approaches. For CIFAR-10, CATRO consistently outperformed other channel pruning algorithms while achieving or approaching the highest FLOPs reduction ratios in ResNet-20/32/56/110. In the case of ResNet-110, CATRO reduced 60.8% FLOPs with a notably high accuracy of 94.41% and reduced 56.1% FLOPs with the highest accuracy of 74.04% on CIFAR-10 and CIFAR-100, respectively.

2) MobileNet on CIFAR: We further validated CATRO on the compact MobileNet-v2 models. Results are presented in Table II, and CATRO reduced 41.6% FLOPs with a notably high accuracy of 94.27%.

3) ResNet on ImageNet: Results on ImageNet supported our findings on CIFAR, and we show the results of ResNet-34/50 in Table III, where CATRO achieved the best accuracies with better or similar FLOP drops. For ResNet-34, with fewer FLOPs (42.9% FLOP reduction), CATRO achieved better classification accuracies compared to others. For ResNet-50, CATRO achieved better or more competitive classification accuracies compared to many state-of-the-art methods.

C. Investigations on Determining Channel Numbers

Fig. 4 showcases the procedure of greedy architecture optimization (described in Section III-D) for ResNet-110 on CIFAR-10 for better investigations of CATRO. A green point at $(\text{iter}, l, 0)$ represents that at the $l$th iteration in Algorithm 2 our method decided to increase the retaining number of the $l$th layer by one. The channel numbers of the
pruned and original network are visualized on the $c_{rec}$ plane in green and gray, respectively. As shown in the figure, the proposed optimization method focused more on deeper layers (i.e., layers close to the output) at the beginning (e.g., the first 300 iterations) and gradually expanded other layers, which is consistent with the state that features in neural networks have a tendency to transfer from general to special [63], [64].

After 2126 iterations, the optimization procedure ended up with a config of network architecture with the desired FLOPs (1.0E8). The resulting network had 21 narrow bottleneck layers (with $\leq 5$ channels) in the first half of the network and 37 fully recovered layers.

### D. Results on Classification Subtask Compression

We evaluated image classification subtask compressions to demonstrate the efficacy of CATRO for both general purposes (CIFAR-100) and practical applications (GTSRB).

| Depth | Method | Accuracy | CIFAR-10 | CIFAR-100 | FLOPs (Drop ↓) | Accuracy | CIFAR-10 | CIFAR-100 | FLOPs (Drop ↓) |
|-------|--------|----------|----------|-----------|---------------|----------|----------|-----------|---------------|
| 20    | Ours   | 91.76%   | 0.98%    | 1.95E7 (52.7%) | 68.61%   | 0.99%    | 2.02E7 (51.0%) |
|      | LCCL [53] | 90.74%   | 1.59%    | 4.76E7 (31.2%) | 67.39%   | 2.69%    | 4.32E7 (37.5%) |
|      | SPP [6] | 90.85%   | 0.55%    | 4.03E7 (41.5%) | 68.37%   | 1.40%    | 4.03E7 (41.5%) |
|      | TAS [32] | 93.16%   | 0.73%    | 5.00E7 (49.4%) | 72.41%   | 1.00%    | 4.25E7 (38.5%) |
|      | FPGM [4] | 92.31%   | 0.32%    | 4.03E7 (41.5%) | 69.52%   | 1.25%    | 4.03E7 (41.5%) |
|      | CNN-FCF [54] | 92.18%   | 0.25%    | 3.99E7 (42.2%) | -        | -        | -         |
|      | LFPC [19] | 92.12%   | 0.51%    | 3.27E7 (49.4%) | -        | -        | -         |
|      | SCOP [30] | 92.13%   | 0.53%    | 3.00E7 (55.8%) | -        | -        | -         |
|      | LFPC [19] | 92.12%   | 0.51%    | 3.27E7 (49.4%) | -        | -        | -         |
|      | DCERP [53] | 92.85%   | -0.49%   | 4.82E7 (30.0%) | 69.51%   | -0.63%   | 4.82E7 (30.1%) |
|      | MFF [22] | 91.85%   | 0.78%    | 3.23E7 (53.2%) | -        | -        | -         |
| 32    | Ours   | 93.17%   | 0.77%    | 2.98E7 (56.1%) | 71.71%   | 0.24%    | 3.81E7 (43.8%) |
|      | LCCL [53] | 92.81%   | 1.54%    | 7.81E7 (37.9%) | 68.37%   | 2.96%    | 7.63E7 (39.3%) |
|      | SPP [6] | 93.35%   | 0.24%    | 5.94E7 (52.6%) | 68.79%   | 2.61%    | 5.94E7 (52.6%) |
|      | AMC [31] | 91.90%   | 0.90%    | 6.29E7 (50.0%) | -        | -        | -         |
|      | DCF [8] | 93.49%   | 0.31%    | 6.27E7 (50.0%) | -        | -        | -         |
|      | TAS [32] | 93.59%   | 0.77%    | 5.95E7 (52.7%) | 72.25%   | 0.93%    | 6.12E7 (51.3%) |
|      | FPGM [4] | 93.49%   | 0.10%    | 5.94E7 (52.6%) | 69.66%   | 1.75%    | 5.94E7 (52.6%) |
|      | CNN-FCF [54] | 93.38%   | -0.24%   | 7.20E7 (42.8%) | -        | -        | -         |
|      | GAL [58] | 93.88%   | 0.12%    | 7.83E7 (37.6%) | -        | -        | -         |
|      | CCP [12] | 93.40%   | 0.12%    | 7.83E7 (47.0%) | -        | -        | -         |
|      | LFPC [19] | 93.24%   | 0.35%    | 5.91E7 (52.9%) | 70.38%   | 0.58%    | 6.08E7 (51.6%) |
|      | SCP [5] | 93.23%   | 0.46%    | 6.10E7 (51.5%) | -        | -        | -         |
| 56    | HRank [18] | 93.17%   | 0.09%    | 6.27E7 (50.0%) | -        | -        | -         |
|      | DHP [57] | 93.58%   | -0.63%   | 6.47E7 (49.0%) | -        | -        | -         |
|      | PScratch [56] | 93.05%   | 0.18%    | 6.35E7 (50.0%) | -        | -        | -         |
|      | LeGR [59] | 93.70%   | 0.20%    | 5.89E7 (53.0%) | -        | -        | -         |
|      | DCERP [23] | 93.26%   | -0.60%   | 8.76E7 (30.2%) | 71.31%   | -0.41%   | 8.78E7 (30.0%) |
|      | MFF [22] | 93.56%   | 0.03%    | 5.94E7 (52.6%) | -        | -        | -         |
| 110   | Ours   | 93.87%   | 0.03%    | 5.83E7 (54.0%) | 72.13%   | 0.67%    | 5.64E7 (55.5%) |
|      | LCCL [53] | 93.44%   | 0.19%    | 1.68E8 (34.2%) | 70.78%   | 2.01%    | 1.73E8 (31.3%) |
|      | SPP [6] | 93.56%   | -0.18%   | 1.21E8 (52.3%) | 71.28%   | 2.86%    | 1.21E8 (52.3%) |
|      | TAS [32] | 94.33%   | 0.64%    | 1.39E8 (53.0%) | 73.16%   | 1.93%    | 1.28E8 (52.6%) |
|      | FPGM [4] | 93.85%   | -0.17%   | 1.21E8 (52.3%) | 72.55%   | 1.59%    | 1.21E8 (52.3%) |
|      | CNN-FCF [54] | 93.67%   | -0.09%   | 1.44E8 (43.1%) | -        | -        | -         |
|      | GAL [58] | 92.74%   | 0.76%    | 1.30E8 (48.5%) | -        | -        | -         |
|      | LFPC [19] | 93.07%   | 0.61%    | 1.01E8 (60.0%) | -        | -        | -         |
|      | HRank [18] | 93.36%   | 0.87%    | 1.06E8 (58.2%) | -        | -        | -         |
|      | DHP [57] | 93.39%   | -0.06%   | 1.06E8 (36.3%) | -        | -        | -         |
|      | PScratch [56] | 93.69%   | -0.20%   | 1.53E8 (40.0%) | -        | -        | -         |
|      | DCERP [23] | 94.11%   | -0.73%   | 1.77E8 (30.1%) | 72.79%   | -0.26%   | 1.77E8 (30.0%) |
|      | MFF [22] | 93.31%   | 0.37%    | 1.21E8 (52.3%) | -        | -        | -         |
| 110   | Ours   | 94.41%   | 0.41%    | 1.00E8 (60.8%) | 74.04%   | 0.45%    | 1.12E8 (56.1%) |
We compared CATRO with a data-independent channel pruning method (FPGM [4]). As shown in Table VI, under the same compression ratio strategy, our method consistently outperformed others with a distinct gap in all settings. For example, the proposed feature discrimination criteria demonstrated the effectiveness of the proposed criterion. CATRO (even without the global architecture optimization) clearly outperformed the baselines on both accuracies and accuracy drops. On the other hand, incorporating the architecture optimization further boosted the effects of CATRO and led to a more compact but powerful network. For example, 0.04% accuracy improvement with 1.1E6 FLOP reduction for ResNet-56 and 0.08% accuracy improvement with 4.8E6 FLOP reduction for ResNet-20.

2) Investigations on Different Layer Groups: To further explore in future work.

E. Ablation Studies

1) Investigations With a Fixed Compression Ratio: To evaluate the effectiveness of the proposed trace ratio criterion in layerwise pruning, we provided ablation studies with a fixed compressions ratio (either uniform pruning or by given structures). This ablation study followed the settings of the baselines, and we replaced Algorithm 2 (searching the pruning ratios of each layer) in CATRO by directly using the same pruning ratios as baselines. The results in Table VII clearly demonstrated the effectiveness of the proposed criterion.

2) Investigations on Different Layer Groups: To further figure out to which extent the feature discriminations and the proposed criteria benefit channel pruning in CATRO, we conducted additional ablation studies on the layerwise pruning method in different parts of the network. To be more specific, for each ResNet model, we grouped all layers into the Shallow-Group (the first one-third conv layers), the Middle-Group (the second one-third conv layers), and the Deep-Group (the last one-third conv layers). With the same channel numbers obtained by Algorithm 2, we replaced the layerwise pruning (Algorithm 1) in CATRO by the common $l_1$ pruner in different layer groups. As shown in Table VIII, we can find that removing the trace criterion in either shallow or deep stages leads to an accuracy drop, and removing the trace criterion in the Deep-Group leads to more accuracy drop compared to the Shallow-Group. Therefore, we can conclude that using discrimination in either shallow or deep stages both help, and the proposed feature discrimination criteria has its superiority over common baselines. This also leads to an open and exciting topic on better-utilizing discrimination or combining it with other pruning strategies, which we will explore in future work.

F. Discussions

1) Studies on Number of Data Samples Used in CATRO: The number of data samples used in the trace ratio calculation is a key factor. To investigate the effects of the number of data

| Method   | Accuracy | Acc. Drop | FLOPs |
|----------|----------|-----------|-------|
| DCP [8]  | 94.02%   | 0.45%     | 26.4% |
| SCOP [30]| 94.24%   | 0.24%     | 40.3% |
| DNCP [21]| 93.71%   | 0.44%     | 41.6% |
| Ours     | 94.27%   | 0.20%     | 41.6% |

We expected the model pruned and fine-tuned on the same task would perform better, and the results shown in Table VI confirm it. Comparing “LSS-DS” and “DS-DS,” we find the accuracy of Direct-Sign with task-exchanging channels drops 0.54%. Similarly, we can find the accuracy drop by comparing “LSS-LSS” and “DS-LSS.” The results demonstrate that CATRO is a task-oriented pruning approach and its selected/pruned channels are based on the specific subtask.

Fig. 5. Channel visualizations of the pruned ResNet-20 for limit-speed-sign and direction-sign classification subtasks on GTSRB. The red and blue triangles indicate the preserved channels for the limit-speed-sign and direction-sign classification, respectively.

1) CIFAR-100: CIFAR-100 is a benchmark dataset for general classification with 100 classes. Without losing generality, we directly selected the top 5/10/20 classes as subtasks of interest and pruned tiny networks for each specific subtask. We compared CATRO with a data-independent channel pruning method (“Direct,” which uses that $l_2$-norm [3]) and a training-based channel pruning method (FPGM [4]). As shown in Table IV, with a high compression ratio (84% FLOPs drop), our method consistently outperformed others with a distinct gap in all settings.

2) GTSRB: We used GTSRB to demonstrate the efficacy of CATRO for practical usages, for example, in intelligent traffic and driver-assistance systems. To mimic the real-world application scenarios, we chose two subtasks from GTSRB (limit-speed-sign recognition and direction-sign recognition) and evaluated the pruning methods with two ultrahigh compression ratios (90% and 97%). Table V shows the results of pruned ResNet-20 and clearly indicates the better performance of CATRO in terms of both the accuracy and the model size. Fig. 5 visualizes the pruned ResNet-20 models from CATRO for different tasks. As shown in the figure, we can find out that different channels contributed to different tasks, and CATRO obtained specific pruned models for each task by selecting different channels. To be more specific, some of the selected/preserved channel indexes in Fig. 5 are the same (the squares filled with both red and blue) while each subtask has its own specific channels (the squares filled with only red or only blue).

3) Study With Task Exchange on GTSRB: To investigate the subtask-oriented channel selection, we further explored a task-exchange experiment. We compared the original CATRO model on one target task (e.g., Direction Signs) with a task-exchange CATRO model, which is pruned (i.e., channel selection) on another task (e.g., Limited Speed Signs) and fine-tuned on the target task (Direction Signs). These two models are named “DS-DS” and “LSS-DS,” respectively.
samples, we studied different numbers of samples on CIFAR-10 and CIFAR-100. Results are shown in Fig. 6, and we found that more data provided limited gains with enlarging samples. Furthermore, we found that CATRO needed fewer samples for each class with more classes, which may be due to the help of more negative samples from other classes. Since the time complexity of CATRO is independent of the number of training samples or batches, a reasonable number of data is a number that is small but represents the data distribution. In our experiments, we empirically randomly sampled 5120/2560 samples for CIFAR-10/CIFAR-100, which is marked with a star in Fig. 6.

2) Pruning Time Complexity of CATRO: The main difference in pruning time lies in the trace ratio computation steps. The time complexity is linear to the number of sampled data which is usually much smaller than the original training data.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
We provided a detailed time analysis on different phases of the training schedule on a machine with a Tesla P40 GPU and E5-2630 CPU. We conducted the experiment for ResNet-110 on CIFAR-10 and used 5120 out of the 50 K samples. The one-time forward propagation took about 22.17 s (including feature map transfer from GPU to CPU), and the average trace optimization for one layer on CPU took about 0.82 s. The time of Algorithm 2 is linear to the time of updating $\lambda$. Note that $\lambda$ in Algorithm 2 can be calculated in parallel in advance, Algorithm 2 only took about 1.0 s to perform the greedy search on the CPU, while common training forward and backward propagation on GPU took about 63 s per epoch on average. Overall, the pruning time for CATRO was about $22.17 + 1.0 + 0.82 \times 110 = 113.37$ s. Therefore, CATRO needs much less time for the pruning step than other training-based methods, for example, GBN [28], which takes 10 epochs for each tock phase with $10 \times 63 = 630$ s and many rounds of tock phases. As CATRO has similar fine-tuning time to other pruning methods and much less pruning time, it is a much more efficient pruning method.

## V. Conclusion

In this article, we present CATRO, a novel channel pruning method via CATRO. By investigating and preserving the joint impact of channels, which is rarely considered in existing pruning methods, CATRO can coherently determine the network architecture and select channels in each layer, both in an efficient nontraining-based manner. Both empirical studies and theoretical justifications have been presented to demonstrate the effectiveness of our proposed CATRO and its superiority over other state-of-the-art channel pruning algorithms.

## REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2012, pp. 1097–1105.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.

[3] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient ConvNets,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2017, pp. 1–13.

[4] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, “Filter pruning via geometric median for deep convolutional neural networks acceleration,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4335–4344.

[5] M. Kang and B. Han, “Operation-aware soft channel pruning using differentiable masks,” in Proc. Int. Conf. Mach. Learn. (ICML), 2020, pp. 5122–5131.

[6] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, “Soft filter pruning for accelerating deep convolutional neural networks,” in Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI), Jul. 2018.

[7] J.-H. Luo, J. Wu, and W. Lin, “ThiNet: A filter level pruning method for deep neural network compression,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 5058–5066.

[8] Z. Zhang et al., “Discrimination-aware channel pruning for deep neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 875–886.
F. Nie, S. Xiang, Y. Jia, C. Zhang, and S. Yan, “Trace ratio criterion,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 9, pp. 1105–1111, Sep. 2004.

F. Nie, S. Xiang, Y. Jia, C. Zhang, and S. Yan, “Trace ratio criterion for feature selection,” in Proc. Conf. Artif. Intell. (AAAI), vol. 2, 2008, pp. 671–676.

K. Wang, R. He, L. Wang, W. Tang, and T. Tan, “Joint feature selection and subspace learning for cross-modal retrieval,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 10, pp. 2010–2023, Oct. 2016.

H. Peng, J. Wu, S. Chen, and J. Huang, “Collaborative channel pruning for deep networks,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 5113–5122.

Q. Tian, T. Arbel, and J. J. Clark, “Task dependent deep LDA pruning of neural networks,” Comput. Vis. Image Understand., vol. 203, Feb. 2021, Art. no. 103154.

S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2015, pp. 1135–1143.

S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2016, pp. 1–14.

Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, “Learning efficient convolutional networks through network slimming,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2755–2763.

X. Dai, H. Yin, and N. K. Jha, “NeSt: A neural network synthesis tool based on a Grow-and-Prune paradigm,” IEEE Trans. Comput., vol. 68, no. 10, pp. 1487–1497, Oct. 2019.

M. Lin et al., “HRank: Filter pruning using high-rank feature map,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 1526–1535.

Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, and Y. Yang, “Learning filter pruning criteria for deep convolutional neural networks acceleration,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 2009–2018.

Y. Guan et al., “DAIS: Automatic channel pruning via differentiable annealing indicator search,” IEEE Trans. Neural Netw. Learn. Syst., early access, Apr. 5, 2022, doi: 10.1109/TNNLS.2022.3161284.

Y.-J. Zheng, S.-B. Chen, C. H. Q. Ding, and B. Luo, “Model compression based on differentiable network channel pruning,” IEEE Trans. Neural Netw. Learn. Syst., early access, Apr. 15, 2022, doi: 10.1109/TNNLS.2022.3165123.

Y. He, P. Liu, L. Zhu, and Y. Yang, “Filter pruning by switching to neighborhood good attributes,” IEEE Trans. Neural Netw. Learn. Syst., early access, Feb. 18, 2022, doi: 10.1109/TNNLS.2021.3143932.

Z. Chen, T.-B. Xu, C. Du, C.-L. Liu, and H. He, “Dynamical channel pruning by conditional accuracy change for deep neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 2, pp. 799–813, Feb. 2021.

Z. Tang et al., “Automatic sparse connectivity learning for neural networks,” IEEE Trans. Neural Netw. Learn. Syst., early access, Jan. 25, 2022, doi: 10.1109/TNNLS.2021.3146655.

J. Peng et al., “Overcoming long-term catastrophic forgetting through adversarial neural pruning and synaptic consolidation,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 9, pp. 4243–4256, Sep. 2022.

W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, “Learning structured sparsity in deep neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2016, pp. 2074–2082.

C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, “Variational convolutional neural network pruning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 2780–2789.

Z. You, K. Yan, J. Ye, M. Ma, and P. Wang, “Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2019, pp. 1–12.

M. Ye, C. Gong, L. Nie, D. Zhou, A. Klivans, and Q. Liu, “Good subnetworks provably exist: Pruning via greedy forward selection,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 10820–10830.

Y. Tang et al., “SCOP: Scientific control for reliable neural network pruning,” in Proc. Neural Inf. Process. Syst., 2020, pp. 10936–10947.

Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, “AMC: AutoML for model compression and acceleration on mobile devices,” in Proc. Euro. Conf. Comput. Vis. (ECCV), 2018, pp. 1–17.

X. Dong and Y. Yang, “Network pruning via transformable architecture search,” in Proc. Neural Inf. Process. Syst., 2019, pp. 1–11.
Wenzhong Hu received the B.S. degree from the Department of Automation, Beihang University, Beijing, China, in 2013, and the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, in 2019. He was a postdoc with the School of Vehicle and Mobility, Tsinghua University. He is currently a Researcher with the Department of Automation, Tsinghua University, Beijing. His research interests include machine learning, deep learning, and intelligent transportation systems.

Zhengping Che (Member, IEEE) received the B.Eng. degree in computer science from Tsinghua University, Beijing, China, in 2013, and the Ph.D. degree in computer science from the University of Southern California, Los Angeles, CA, USA, in 2018. He is currently with Midea Group, Beijing. His research interests lie in the areas of deep learning, computer vision, and time-series analysis with applications to robot learning.

Ning Liu (Member, IEEE) received the Ph.D. degree in computer engineering from Northeastern University, Boston, MA, USA, in 2019. He is a Researcher with Midea Group, Beijing, China. His current research interests lie in deep learning, deep model compression and acceleration, deep reinforcement learning, and edge computing.

Mingyang Li received the master’s degree from the Department of Automation, Tsinghua University, Beijing, China, in 2020. He is currently an Engineer with KE Holdings Inc., Beijing. His research interests include machine learning and computer vision.

Jian Tang (Fellow, IEEE) received the Ph.D. degree in computer science from Arizona State University, Tempe, AZ, USA, in 2006. He is currently with Midea Group, Beijing, China. He has published over 180 papers in premier journals and conferences. His research interests lie in the areas of AI, the IoT, wireless networking, and mobile computing.

Changshui Zhang (Fellow, IEEE) received the B.S. degree in mathematics from Peking University, Beijing, China, in 1986, and the M.S. and Ph.D. degrees in control science and engineering from Tsinghua University, Beijing, in 1989 and 1992, respectively. In 1992, he joined the Department of Automation, Tsinghua University, where he is currently a Professor. He has authored more than 200 articles. His current research interests include pattern recognition and machine learning.

Jianqiang Wang received the B.Tech. and M.S. degrees in automotive application engineering from the Jilin University of Technology, Changchun, China, in 1994 and 1997, respectively, and the Ph.D. degree in vehicle operation engineering from Jilin University, Changchun, in 2002. He is currently a Professor with the School of Vehicle and Mobility, Tsinghua University, Beijing, China. He has authored more than 40 journal articles and is currently a Co-Holder of 30 patent applications. His research focuses on intelligent vehicles, driving assistance systems, and driver behavior.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.