

# DAIS: Automatic Channel Pruning via Differentiable Annealing Indicator Search

Yushuo Guan, Ning Liu, *Member, IEEE*, Pengyu Zhao, Zhengping Che, *Member, IEEE*,  
Kaigui Bian, *Senior Member, IEEE*, Yanzhi Wang, *Member, IEEE*, and Jian Tang, *Fellow, IEEE*

**Abstract**—The convolutional neural network has achieved great success in fulfilling computer vision tasks despite large computation overhead against efficient deployment. Channel pruning is usually applied to reduce the model redundancy while preserving the network structure, such that the pruned network can be easily deployed in practice. However, existing channel pruning methods require hand-crafted rules, which can result in a degraded model performance with respect to the tremendous potential pruning space given large neural networks. In this paper, we introduce Differentiable Annealing Indicator Search (DAIS) that leverages the strength of neural architecture search in the channel pruning and automatically searches for the effective pruned model with given constraints on computation overhead. Specifically, DAIS relaxes the binarized channel indicators to be continuous and then jointly learns both indicators and model parameters via bi-level optimization. To bridge the non-negligible discrepancy between the continuous model and the target binarized model, DAIS proposes an annealing-based procedure to steer the indicator convergence towards binarized states. Moreover, DAIS designs various regularizations based on *a priori* structural knowledge to control the pruning sparsity and to improve model performance. Experimental results show that DAIS outperforms state-of-the-art pruning methods on CIFAR-10, CIFAR-100, and ImageNet.

## I. INTRODUCTION

In recent years, the community has achieved great success on various computer vision tasks [1], [2], [3], [4] by designing deeper and wider convolutional neural networks (CNNs). Despite the superior performance of these networks, the computation cost is burdensome. Hence, it is difficult to directly deploy such “expensive” networks over computation-limited applications such as robotics, self-driving vehicles, and most of the mobile devices. A straightforward solution to tackle the problem is *channel pruning*. Channel pruning methods target at compressing the networks in terms of parameters, computation cost, and inference latency, while maintaining the performance of the unpruned networks [5], [6], [7]. Channel pruning methods could reduce the computation cost of the network by eliminating redundant channels in CNNs, which

require no extra hardware support and is perfectly compatible with representative deep learning acceleration frameworks, such as TVM [8], TFLite [9], and Alibaba MNN [10].

Earlier channel pruning methods leverage hand-crafted criteria, such as lasso regression [6] and geometric median-based criterion [11], to filter out unimportant channels. Recent research on pruning [12] implies that the pruning process is equivalent to searching a compact network structure, suggesting the possibility of automatic channel pruning. Since the pruning search space increases exponentially with original networks being deeper and wider, a potential approach to address this problem is through the idea of automatic channel pruning [13], [7]. These methods search for the appropriate pruned models automatically, but the search efficiency might be low [13] and there might be architecture discrepancies between the search process and the final pruned model [7].

To address these problems in the automatic pruning approaches, we propose DAIS, a Differentiable Annealing Indicator Search method for channel pruning, taking advantage of differentiable neural architecture search [14]. In general, DAIS first searches for a pruned model in an automatic manner with computational constraints and then fine-tunes the derived model. Specifically, DAIS introduces binarized channel indicators and utilizes continuous auxiliary parameters to relax the indicators for optimization. Then it jointly learns model parameters and auxiliary parameters via bi-level optimization, to simultaneously obtain the accurate model and identify the importance of each channel. An annealing-relaxed function is incorporated into the channel indicators to mitigate the discrepancy between the derived pruned model and the pretrained supernet for the search procedure. The indicators are initialized to be gently continuous at high temperatures, and gradually converge to binarized states as the training proceeds and temperature anneals. Furthermore, to provide the structural constraints on the pruned model, we design dedicated regularizers in DAIS: 1) A continuous FLOPs estimator regularizer that controls the computational cost of the pruned model, and 2) a symmetry regularizer which optimizes the gradient propagation on the pruned ultra-deep neural networks with residual connections. Additionally, DAIS is more efficient, since it is a one-shot solution that does not require multi-round pruning compared with most existing methods [6], [15], [16]. Experimental results show that DAIS outperforms state-of-the-art pruning methods on representative datasets such as CIFAR-10 [17], CIFAR-100 [17], and ImageNet [18], as well as having the great transfer ability to other vision tasks such as semantic segmentation and scene text recognition.

Yushuo Guan and Pengyu Zhao are with the School of Computer Science, Peking University, Beijing, China 100871. E-mail: {david.guan, pengyuzhao}@pku.edu.cn

Ning Liu, Zhengping Che, and Jian Tang are with Midea Group, Beijing, China 100102. E-mail: {liuning22, chezp, tangjian22}@midea.com

Kaigui Bian is with the School of Computer Science, National Engineering Laboratory for Big Data Analysis and Applications, Peking University, Beijing, China 100871. E-mail: bkg@pku.edu.cn

Yanzhi Wang is with the Department of Electrical & Computer Engineering, Northeastern University, Boston, MA, USA 02115. E-mail: yanz.wang@northeastern.edu

Corresponding author: Kaigui Bian.

The main contributions of this paper are as follows. 1) We propose DAIS, a differentiable annealing indicator search framework for channel pruning, which leverages the gradient-based bi-level optimization to search for appropriate pruned models with sparsity requirements. 2) We design an annealing-relaxed channel indicator function for the differentiable search process. This function is initialized to be continuous to relax the search process to be differentiable and then gradually converges into binarization to produce the pruned model. 3) We design different regularizers to constrain the computational cost of the pruned model. 4) DAIS achieves state-of-the-art performance on different datasets and models, and extensive experiments and ablation studies demonstrate its effectiveness.

## II. RELATED WORKS

**Channel pruning.** The recent advances of network pruning mainly fall into two categories: unstructured pruning and channel pruning. Unstructured pruning [19], [20] removes weights at arbitrary locations to reduce the storage and computation. However, the resulted sparse matrix and indexing scheme derived by unstructured pruning methods requires special sparse matrix operation libraries and/or hardware, which limit the practical acceleration in general CNN acceleration frameworks. To avoid these limitations, recent works [15], [21], [22], [23], [11], [24] focus on channel pruning, which filter out redundant channels of the convolutional networks. The pruned model would maintain the network structure and take full advantage of Basic Linear Algebra Subprograms (BLAS) operations [25]. Thus it can be perfectly supported by the prevalent CNN acceleration frameworks such as TVM [8], TensorFlow-Lite (TFLite) [9], and Alibaba Mobile Neural Network (MNN) [10].

Early works [26], [27], [28] heuristically evaluate the importance by the magnitude of the weights. Wen *et al.* [26] adopt group lasso to force groups of weights to be smaller and filter out the channels with zero weights. PFEC [27] identifies the importance of each channel by L1 norm, and prunes the channels which are identified to have small impact on the network performance. SFP [28] first selects the filters based on L2 norm and then prunes them softly. All these works use simple criteria like L1 and L2 norm to determine the significance of channels and filter out channels with smaller norms.

Recent methods [11], [29], [16], [30], [31], [32], [33] design various criteria or utilize additional optimization tools for channel pruning. FPGM [11] filters out redundant channels with a criterion of the geometric median. It proves that pruning the channels near the geometric median has a substantial impact on the network accuracy. CNN-FCF [29] learns binary scalars associated with filters to determine the target filters to prune. The binary scalars are not differentiable, and additional optimization tools (such as ADMM) are needed for addressing the binary constraints. GBN [16] introduces the gate decorator into the network and estimates the performance of candidate pruned models based on Taylor expansion. LFPC [30] learns different criteria for each convolutional layer, considering that different layers have various distributions. HRank [31]

discovers that the low-rank feature maps contain less information, therefore it prunes filters with low-rank feature maps. ABCPruner [32] leverages the artificial bee colony (ABC) algorithm to search for appropriate pruned models. SCP [33] jointly considers the effect of batch normalization (BN) [34] and ReLU activation, and filters out the channels where the BN and ReLU operations are likely to deactivate each feature map.

Besides, some channel pruning approaches [35], [36], [37] target at the efficiency of the pruning process. Dong *et al.* [35] propose the low-cost collaborative layer (LCCL) to speed up the inference of CNNs. SSS [36] prunes the networks in one training pass with the help of a modified stochastic Accelerated Proximal Gradient (APG) method. Lin *et al.* [37] optimizes the channel pruning by generative adversarial learning (GAL), which replaces the traditional multi-stage pruning into the end-to-end optimization.

**Neural architecture search (NAS).** Recently, there has been a growing interest in neural architecture search (NAS), which automatically designs neural network architecture with no human effort. The neural architecture search approaches could be classified according to the *search space* and *search strategy*.

The search space of NAS is the network topology, and could be categorized into the macro search space and micro search space. The methods [38], [39] search for the entire CNN architecture in the macro search space, including kernel sizes, skip connections and number of filters. Since the cost of the macro space search is too large, more methods focus on the micro space search, where they only search for a certain module structure in the network [14], [40]. The derived module will be stacked to form a complete network. Early works leverage reinforcement learning [38], [41], [40], [42] or the evolutionary algorithm [43], [44] as the search strategy. These works mainly sample a large number of networks from search space, then train them from scratch to obtain a supervision signal, and optimize the sampling agent with different search strategies. PGNAS [45] models the NAS task from Bayesian perspective and needs a posterior-guided sampling process. It reduces the computational complexity compared with earlier approaches [38], [40], but still needs 11 GPU days to search for appropriate networks. Recent attempts [46], [47], [48], [49], [50], [51] introduce weight-sharing paradigm in NAS to boost search efficiency, where all candidate sub-networks share the weights in a single one-shot model that contains every possible architecture in the search space. Among weight-sharing methods, DARTS [14] has attracted much attention. It relaxes the search space to be continuous with architecture parameters and then efficiently optimizes model parameters and architecture parameters together via gradient descent.

**Automatic channel pruning.** The channel pruning could be thought of searching for appropriate pruned models as suggested by Liu *et al.* [12], which implies the possibility of automatic channel pruning. The automatic channel pruning [13], [7] leverages the idea of NAS, which spends less search cost compared with the complete NAS. NetSlimming [6] automatically selects filters by associating scaling factors from BN layers and prunes filters with smaller scaling

factors. AMC [13] leverages deep reinforcement learning to determine the pruning rate of each layer. It uses a sampling, estimating, and learning process, which is time-consuming, especially for deep networks. Besides, AMC only prunes the middle channels of the blocks with shortcut (like ResNet), which limits the optimal upper bound of the pruning ratio. TAS [7] directly searches the width and depth of a network with a novel transformable architecture search procedure, but there are architecture discrepancies between the supernet in the search procedure and the pruned model, which may result in performance degradation in terms of pruning ratio and accuracy.

**Comparisons to related works.** DAIS leverages the automatic differentiable search procedure to replace heuristic rules in traditional pruning methods [15], [21], [22], [23], [11], [24]. DAIS is one-shot, which is more efficient compared with multi-round pruning approaches [6], [15], [16]. The heuristic annealing idea of DAIS could efficiently support the convergence of the bi-level pruning process. DAIS has different optimization targets compared with general NAS works like PGNAS [45]. General NAS works focus on neural architectures with high accuracy, while DAIS targets at automatic channel pruning and searches for a small and fast pruned model for the original model. DAIS also breaks the limitations of automatic channel pruning methods like AMC [13] and TAS [7]. DAIS is built on top of the differentiable indicator search procedure for faster search efficiency, and leverages the annealing-relaxed channel indicator to fill the architecture discrepancies. DAIS also uses different regularizers to satisfy structural restrictions.

### III. METHODOLOGY

In this section, we firstly formulate the channel pruning task and then introduce the Differentiable Annealing Indicator Search (DAIS) method for channel pruning. Specifically, we demonstrate the differentiable channel indicator search procedure of DAIS, propose the annealing-relaxed channel indicator, and introduce three regularizers for structural restrictions. The overview of DAIS is illustrated in Fig. 1.

#### A. Problem Definition

A general CNN model is built with a stack of convolutional layers. Suppose the model has  $L$  layers, and we define  $O_l$  as the output feature tensor of the  $l$ -th convolutional layer. The computation process of  $O_l$  can be commonly depicted as:

$$O_l = F_l(W_l, O_{l-1}), \quad (1)$$

where  $W_l \in \mathcal{R}^{k_l \times k_l \times c_{l-1} \times c_l}$  denotes the convolutional kernel of the  $l$ -th layer with input channel of  $c_{l-1}$  and output channel of  $c_l$ , and  $F_l(\cdot)$  represents the convolution function. The optimization target of model pruning is to derive the network architecture with the minimum number of convolutional filters from the original model while maintaining accuracy. The channel pruning further preserves the structural information of CNN and only reduces the number of channels, i.e.,  $c_l$  for each layer.

A popular channel pruning approach [6] introduces channel indicators, which are binarized vectors generating sparsity for each convolutional layer, i.e.:

$$O'_l = F_l(W_l, O'_{l-1}) \otimes I_l, \quad (2)$$

where  $I_l \in \{0, 1\}^{c_l}$  denotes the indicator vector,  $\otimes$  represents the tensor product, and  $O'_l$  denotes the masked (pruned) output tensor by  $I_l$ . With channel indicator, the optimization target of channel pruning can be represented by:

$$\min_{\mathcal{W}, \mathcal{I}} \mathcal{L}(\mathcal{W}, \mathcal{I}) + \lambda \mathcal{R}(\mathcal{I}), \quad (3)$$

where  $\mathcal{W} = [W_1, W_2, \dots, W_L]$  and  $\mathcal{I} = [I_1, I_2, \dots, I_L]$  denote respectively the parameter and indicator sets,  $\mathcal{L}(\cdot)$  denotes the loss function, and  $\mathcal{R}(\cdot)$  denotes the regularizer that induces the structural restrictions. The details of regularizers will be discussed in Sec. III-D.

As the channel indicator is binarized and non-differentiable, it could not be learned directly by conventional gradient descent methods. Therefore, some channel pruning approaches [6], [16] associate the channel indicator with differentiable metrics, e.g., leverage the weights in BN as an indirect indicator, and zero out the channels with small BN weights [6]. Instead of building up these indirect associations, we deal with the channel indicator from another perspective which is more effective and flexible, i.e., *relax the channel indicators to be continuous, and then jointly learn the model parameters and relaxed channel indicators via a differentiable search procedure.*

#### B. Differentiable Indicator Search

The differentiable search procedure is widely used in NAS [52], [14], [50] but seldom explored towards solving the channel pruning problem. In this section, we incorporate the idea of differentiable search into channel pruning and introduce the differentiable indicator search.

To make the search space continuous, we relax each binarized channel indicator  $I_l^i$ ,  $i \in [1, c_l]$ , into a relaxed channel indicator  $\tilde{I}_l^i$  parameterized by an auxiliary parameter  $\alpha_l^i$  (discussed in Sec. III-C). After the relaxation, the goal of the indicator search is to jointly learn model parameters  $\mathcal{W}$  and auxiliary parameters  $\alpha$ . An intuitive option of optimization is to update  $\mathcal{W}$  and  $\alpha$  simultaneously on the training set, but the simultaneous optimization will cause  $\alpha$  to overfit on the training set, which might derive the pruned result with poor generalization. Therefore, we leverage the bi-level optimization in the differentiable indicator search procedure, with  $\alpha$  as the upper-level variable and  $\mathcal{W}$  as the lower-level variable. Specifically, it searches for  $\alpha^*$  that minimizes the combination of validation loss  $\mathcal{L}_{\text{val}}(\mathcal{W}^*(\alpha), \alpha)$  and regularizers  $\mathcal{R}(\alpha)$ , where the values of model parameters  $\mathcal{W}^*$  are obtained by minimizing the training loss  $\mathcal{L}_{\text{train}}(\mathcal{W}, \alpha)$ :

$$\min_{\alpha} \mathcal{L}_{\text{val}}(\mathcal{W}^*(\alpha), \alpha) + \lambda \mathcal{R}(\alpha), \quad (4)$$

$$\text{s.t. } \mathcal{W}^*(\alpha) = \arg \min_{\mathcal{W}} \mathcal{L}_{\text{train}}(\mathcal{W}, \alpha). \quad (5)$$

To solve the bi-level optimization problem,  $\mathcal{W}$  and  $\alpha$  are updated in a multi-step scheme by gradient descent on training

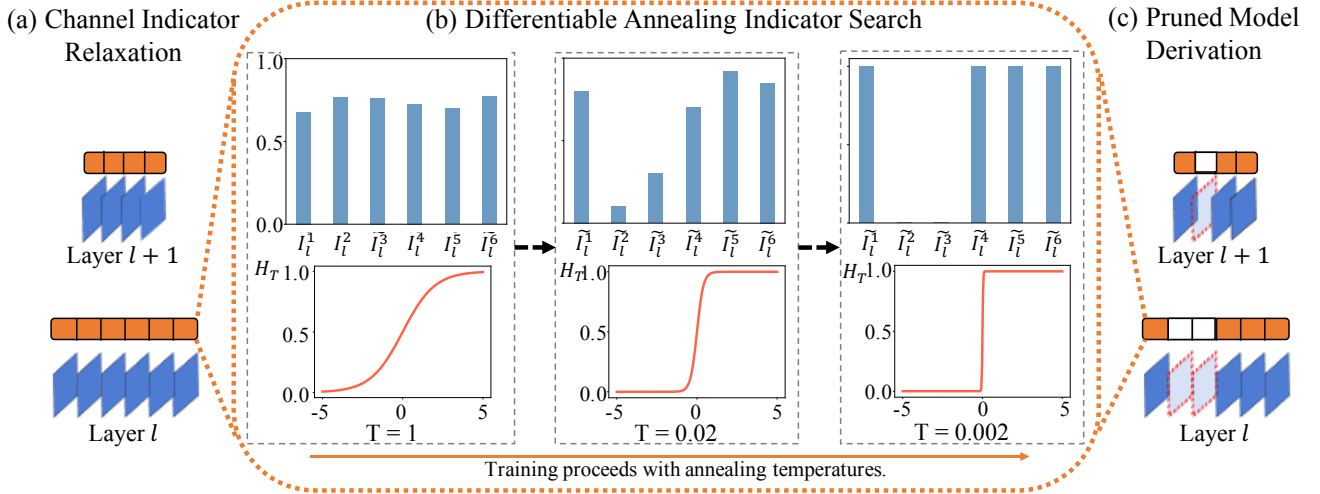


Fig. 1. Overview of the Differentiable Annealing Indicator Search (DAIS) framework. (a) Channel indicator relaxation. We build annealing-relaxed channel indicators for each convolutional layer. (b) Differentiable annealing indicator search. As the training proceeds, the annealing-relaxed indicator  $\tilde{I}_l^i$  will converge to binarization with annealing temperatures, as illustrated in the histograms.  $H_T(\cdot)$  is an annealing function, which approximates the binarized state when the temperature anneals. (c) Pruned model derivation. After the differentiable annealing indicator search, the pruned model will be derived by filtering out the output channels with  $\tilde{I}_l^i = 0$  from the original model.

and validation sets, and finally reach the local minima. The alternating gradient updating process resembles [14] with more details provided in the reference paper.

### C. Annealing-Relaxed Channel Indicator

To make the relaxed channel indicator a better approximation of binarized channel indicator, the value range for each entry  $\tilde{I}_l^i$  should be limited between 0 and 1. Hence, a straightforward solution is to use a sigmoid function over auxiliary parameters, formally:

$$\tilde{I}_l^i = \frac{1}{1 + e^{-\alpha_l^i}}. \quad (6)$$

However, the above approximation has two inevitable problems. (1) The resulting  $\tilde{I}_l^i$  might not converge to sparse values, i.e., close to 0 or 1, as there is no guarantee for sparsity based on the continuous auxiliary parameters. Hence, a hand-crafted threshold is still required to binarize the final pruning result, which limits the robustness of this method and violates the idea of automatic channel pruning. (2) Moreover, the discretization brings the non-negligible discrepancy between the continuous search result and binarized model [53], [52], [14]. This could deteriorate the accuracy of the pruned model when joint parameters of  $W$  and  $\alpha$  are stuck in sharp local minima where a small perturbation can lead to large performance degradation.

We propose an annealing-relaxed channel indicator to fill the discrepancy. Concretely, we add a temperature variable  $T$  on the sigmoid function, which is set to be high at the beginning and gradually anneals to zero at the end of training. As a result, the indicator is initially continuous to support the gradient update of the auxiliary parameters, and finally converges to the binarized state that leads to the pruned model:

$$\tilde{I}_l^i = H_T(\alpha_l^i) = \frac{1}{1 + e^{-\alpha_l^i/T}}, \quad I_l^i = \lim_{T \rightarrow 0} H_T(\alpha_l^i), \quad (7)$$

where  $\alpha_l^i$  represents the auxiliary parameter. The annealing-relaxed function  $H_T(\cdot)$  is initialized with a high temperature  $T = T_0$ . When the training proceeds into  $n$ -th epoch, the temperature of  $H_T$  anneals to  $T_0/\psi(n)$ , where  $\psi(\cdot)$  denotes the temperature annealing scheme. Finally, the discrete  $I_l^i$  can be approached with  $T \rightarrow 0$  at the end of the search.

### D. Structural Restrictions by Regularizers

The vanilla cross-entropy loss itself is infeasible to induce *a priori* structural restrictions, e.g., the number of floating point operations (FLOPs), which play a critical role in pruning. Therefore, we introduce three regularizers into the search procedure when updating the auxiliary parameters.

**Lasso regularizer.** A naturally used sparsity regularizer is the lasso regularizer:

$$\mathcal{R}_{\text{lasso}} = \sum_{l=1}^L \sum_{i=1}^{c_l} \|H_T(\alpha_l^i)\|_1. \quad (8)$$

$\mathcal{R}_{\text{lasso}}$  uses the lasso/L1 regularization and can effectively zero out some channel indicators, but the pruning rate resulted from  $\mathcal{R}_{\text{lasso}}$  is not controllable and heavily dependent on its weight on the whole loss function.

**Continuous FLOPs estimator regularizer.** The value of the annealing-relaxed channel indicator  $H_T(\alpha_l^i)$  can be viewed as the probability of preserving the corresponding channel in the final pruned model. With the annealing-relaxed indicator of all output channels, we can estimate the expectation of overall FLOPs of the pruned model by accumulating FLOPs of each channel. In the  $l$ -th convolutional layer, the continuous FLOPs estimator could be represented as:

$$E_{\text{FLOPs}}(\alpha) = \sum_{l=1}^L \left( p_l \cdot \left( \sum_{i=1}^{c_{l-1}} H_T(\alpha_{l-1}^i) \right) \cdot \left( \sum_{i=1}^{c_l} H_T(\alpha_l^i) \right) \right) \quad (9)$$

where  $p_l = h_l \times w_l \times k_l^2$ , and  $k_l$  denotes the kernel size,  $h_l$  and  $w_l$  denote the spatial size of the output feature maps.

Inspired by Dong *et al.* [7], we build up the regularizer given the continuous FLOPs estimator:

$$\mathcal{R}_{\text{FLOPs}}(\alpha) = \begin{cases} \log(E_{\text{FLOPs}}(\alpha)), & \frac{E_{\text{FLOPs}}(\alpha)}{F} > 1 \\ -\log(E_{\text{FLOPs}}(\alpha)), & \frac{E_{\text{FLOPs}}(\alpha)}{F} < 1 - \epsilon \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

where  $F$  denotes the expected FLOPs and  $\epsilon \ll 1$ . The continuous FLOPs estimator regularizer guides DAIS to search for the optimized pruned model with FLOPs in range of  $[(1 - \epsilon) * F, F]$ .

**Symmetry regularizer.** Residual blocks are widely integrated in recent network designs, which greatly improve the capability of gradient propagation across multiple layers. The residual block is realized by adding a shortcut connection of non-parameterized identity mapping from input to output when the numbers of input and output channels are equal. However, the existing channel pruning methods [50], [7] cannot guarantee the equality between the numbers of input and output channels of a residual block, and they either drop the shortcut [50] or replace the identity function with a  $1 \times 1$  convolutional layer [7] for dimension matching. These replacements might increase the difficulty of gradient propagation across multiple layers and lead to the problem of vanishing and exploding gradients [54].

Therefore, we propose a symmetry regularizer for channel pruning on networks with residual connections, which keeps the consistency between the input and output channels within a residual block and reserves the identity mapping. The symmetry regularizer is defined as:

$$\mathcal{R}_{\text{sym}} = \sum_{(l,l')} \left| \left( \sum_{i=1}^{c_l} H_T(\alpha_l^i) \right) - \left( \sum_{i=1}^{c_{l'}} H_T(\alpha_{l'}^i) \right) \right|, \quad (11)$$

where  $(l, l')$  denotes a residual block with  $c_l$  input channels and  $c_{l'}$  output channels. We also explore the importance of the identity mapping on deep residual networks in Sec. IV-C.

## IV. EXPERIMENTS

### A. Experiment Setup

**Datasets.** We evaluated the effectiveness of DAIS on CIFAR-10 [17], CIFAR-100 [17], and ImageNet ILSVRC-12 [18]. CIFAR-10 and CIFAR-100 both consist of 50K training images and 10K test images, and they have 10 and 100 classes, respectively. ImageNet ILSVRC-12 contains 1280K training images and 50K test images for 1000 classes.

**Channel indicator setting.** We implemented the annealing-relaxed channel indicator to each convolutional layer except for the first layer, conforming to baselines [35]. The implementation details of the annealing-relaxed indicator are as follows. On ResNet, there are two kinds of residual blocks: normal block and reduction block. Most residual blocks in the network are normal blocks, and the reduction block is employed to down-sample the spatial size of the feature map while doubling the number of the output channel. We adopted the same implementation for these two types of blocks: the

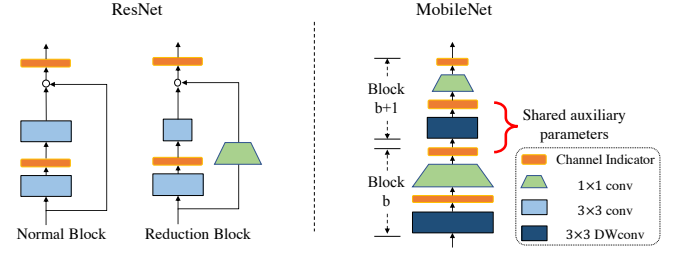


Fig. 2. The channel indicator setup for ResNet and MobileNet networks.

first annealing channel indicator is used right behind the first  $3 \times 3$  convolution, and the second indicator is implemented after the addition of two paths. On MobileNet, we built up the annealing-relaxed channel indicator for each  $1 \times 1$  convolution and  $3 \times 3$  depth-wise convolution (DWconv). As the DWconv requires the same number of input and output channels, the annealing-relaxed indicator is shared before and after the DWconv. The implementation details are shown in Fig. 2.

**Differentiable annealing indicator search.** In the search process, we used  $\mathcal{R}_{\text{FLOPs}}$  and  $\mathcal{R}_{\text{sym}}$  as the default regularizers. For the update of the auxiliary parameters, DAIS adopted Adam [59] as the optimizer with the momentum of (0.5, 0.999), where the learning rate and weight decay rate were both set to  $1\text{E-}3$ . For the update of the network parameters, DAIS leveraged the SGD optimizer with 0.9 momentum, and the learning rate was initialized with 0.1 and reduced by the cosine scheduler [60]. On the CIFAR experiments, we implemented the differentiable annealing indicator search for 100 epochs. The original training images were divided into the training set and validation set with the 7 : 3 ratio. On the ImageNet experiments, we searched 7508 iterations with batch size 256.

For all experiments, the auxiliary parameters were initialized with a normal distribution  $\alpha \in \mathcal{N}(1, 0.1^2)$ . The initialization temperature was  $T_0 = 1$ , and the temperature annealing scheme was  $\psi(n) = 49 \times n / N_{\text{max}} + 1$ , where  $N_{\text{max}}$  denotes the total number of training epochs. The batch size of the search procedure was 256. The weights of  $\mathcal{R}_{\text{FLOPs}}$  was 2 and  $\epsilon = 0.05$ . The weights of  $\mathcal{R}_{\text{sym}}$  was 0.01 for ResNet-56/110 and 0 otherwise. The weight decay of the SGD optimizer on the CIFAR and ImageNet experiments were  $5\text{E-}5$  and  $1\text{E-}5$  respectively.

**Fine-tuning.** For all experiments, we used SGD as the optimizer with a momentum of 0.9. The batch size was 256, and the learning rate was initialized with 0.1. For the CIFAR experiments, we trained each model for 300 epochs. The learning rate was warmed up with 5 epochs and then reduced with the cosine scheduler. We adopted random crop, random horizontal flipping and random erasing [61] as default augmentations. We trained all models with a single NVIDIA P40 GPU in the CIFAR experiments. For the ImageNet experiments, we trained 120 epochs for each model, and the learning rate was decayed by 10 every 30 epochs. We used the random resized crop and random horizontal flipping for data augmentation. Besides, conforming to the methods [11], [7] that use original models to implicitly-or-explicitly transfer the knowledge to

TABLE I

COMPARISON ON CHANNEL PRUNING APPROACHES USING RESNET ON THE CIFAR-10 AND CIFAR-100 DATASETS. “PRUNING ACC” = ACCURACY OF THE PRUNED MODEL, “ACC DROP” = ACCURACY DROP, “FLOPS” = FLOPS (PRUNING RATIO). WE DISPLAY TWO PRUNING RESULT OF DAIS FOR EACH MODEL, WHERE THE FIRST RESULT IS FOR BEST ACCURACY AND THE LATTER ONE IS FOR BEST PRUNING RATE.

Depth	Method	CIFAR-10			CIFAR-100		
		Pruning Acc	Acc Drop	FLOPs	Pruning Acc	Acc Drop	FLOPs
32	LCCL [35]	90.74%	1.59%	4.76E7 (31.2%)	67.39%	2.69%	4.32E7 (37.5%)
	SFP [28]	92.08%	0.55%	4.03E7 (41.5%)	68.37%	1.40%	4.03E7 (41.5%)
	FPGM [11]	92.31%	<b>0.32%</b>	4.03E7 (41.5%)	68.52%	1.25%	4.03E7 (41.5%)
	CNN-FCF [29]	92.18%	1.07%	3.99E7 (42.2%)	-	-	-
	TAS [7]	93.16%	0.73%	3.50E7 (49.4%)	<b>72.41%</b>	<b>-1.80%</b>	4.25E7 (38.5%)
	LFPC [30]	92.12%	0.51%	3.27E7 (49.4%)	-	-	-
	DAIS	<b>93.49%</b>	0.57%	3.19E7 ( <b>53.9%</b> )	72.20%	-1.04%	3.94E7 ( <b>42.9%</b> )
56	LCCL [35]	92.81%	1.54%	7.81E7 (37.9%)	68.37%	2.96%	7.63E7 (39.3%)
	AMC [13]	91.90%	0.90%	6.29E7 (50.0%)	-	-	-
	SFP [28]	93.35%	0.56%	5.94E7 (52.6%)	68.79%	2.61%	5.94E7 (52.6%)
	FPGM [11]	93.49%	0.42%	5.94E7 (52.6%)	69.66%	1.75%	5.94E7 (52.6%)
	CNN-FCF [29]	93.38%	<b>-0.24%</b>	7.20E7 (42.8%)	-	-	-
	TAS [7]	<b>93.69%</b>	0.77%	5.95E7 (52.7%)	72.25%	0.93%	6.12E7 (51.3%)
	GBN [16]	93.07%	0.03%	3.72E7 (70.3%)	-	-	-
	GAL [37]	93.38%	0.12%	7.83E7 (37.6%)	-	-	-
	LFPC [30]	93.24%	0.35%	5.91E7 (52.9%)	70.83%	<b>0.58%</b>	6.08E7 (51.6%)
	HRank [31]	93.17%	0.35%	6.27E7 (50.0%)	-	-	-
	ABCPruner [32]	93.23%	0.03%	5.84E7 (54.1%)	-	-	-
	SCP [33]	93.23%	0.46%	6.10E7 (51.5%)	-	-	-
	DAIS	93.53%	1.00%	3.64E7 ( <b>70.9%</b> )	<b>72.57%</b>	0.81%	5.84E7 ( <b>53.6%</b> )
110	LCCL [35]	93.44%	0.19%	1.68E8 (34.2%)	70.78%	2.01%	1.73E8 (31.3%)
	SFP [28]	92.97%	0.70%	1.21E8 (52.3%)	71.28%	2.86%	1.21E8 (52.3%)
	FPGM [11]	93.85%	-0.17%	1.21E8 (52.3%)	72.55%	1.59%	1.21E8 (52.3%)
	CNN-FCF [29]	93.67%	-0.09%	1.44E8 (43.1%)	-	-	-
	TAS [7]	94.33%	0.64%	1.19E8 (53.0%)	73.16%	1.90%	1.20E8 (52.6%)
	GAL [37]	92.74%	0.76%	1.30E8 (48.5%)	-	-	-
	LFPC [30]	93.07%	0.61%	1.01E8 (60.0%)	-	-	-
	HRank [31]	93.36%	0.87%	1.06E8 (58.2%)	-	-	-
	DAIS	<b>95.02%</b>	<b>-0.60%</b>	1.01E8 ( <b>60.0%</b> )	<b>74.69%</b>	<b>-0.65%</b>	1.14E8 ( <b>56.7%</b> )

TABLE II

RESULTS OF LIGHT-WEIGHTED NETWORKS ON CIFAR-10. “M.NET” AND “R-20” DENOTE THE RESNET-20 AND MOBILENET RESPECTIVELY.

		Pruning Acc	Acc Drop	FLOPs
M.Net	M.Net-0.75	91.65%	1.22%	1.95E8 (43.3%)
	CGNet [55]	87.56%	<b>0.29%</b>	1.19E8 (65.3%)
	DAIS	<b>91.87%</b>	1.00%	1.15E8 ( <b>66.6%</b> )
R-20	LCCL [35]	91.68%	1.06%	2.61E7 (36.0%)
	SFP [28]	90.83%	1.37%	2.43E7 (42.2%)
	FPGM [11]	91.09%	1.11%	2.43E7 (42.2%)
	CNN-FCF [29]	91.13%	1.07%	2.38E7 (41.6%)
	TAS [7]	92.88%	0.00%	2.24E7 (45.0%)
	DAIS	<b>92.89%</b>	<b>-0.36%</b>	1.91E7 ( <b>51.1%</b> )

pruned models, we used the feature distillation method [62] on the ImageNet experiments. We leveraged 4 NVIDIA P40 GPUs in the ImageNet experiments.

### B. Comparisons with State-of-the-Art Methods

**Results on CIFAR.** We first evaluated the performance of DAIS on ResNet-32/56/110 with CIFAR-10 and CIFAR-100.

As illustrated in TABLE I, DAIS consistently outperformed various state-of-the-art pruning approaches on CIFAR. For ResNet-32 on CIFAR-100, DAIS reduced 42.9% FLOPs and increased accuracy by 1.04% compared with the original network. Besides, the differentiable annealing indicator search was efficient: DAIS only spent 95 minutes on the search procedure, while TAS finished the search in 228 minutes. For ResNet-56 on CIFAR-10 and CIFAR-100, DAIS obtained fewer FLOPs than other pruning methods. Furthermore, since DAIS was a one-shot solution, it required less training effort than the iterative pruning approaches like GBN [16]. In the case of deeper architecture ResNet-110, on both CIFAR-10 and CIFAR-100, our DAIS obtained the best FLOPs reduction ratio, accuracy drop, and accuracy, indicating that the proposed symmetry regularizer perfectly improved the capability of gradient propagation in the deeper layers.

We also evaluated the performance of DAIS on light-weighted networks like MobileNet [63] and ResNet-20. For MobileNet, DAIS got better accuracy with fewer FLOPs compared with MobileNet-0.75 and CGNet [55], as shown in TABLE II. For ResNet-20 on CIFAR-10, DAIS reduced 51.1% FLOPs and increased accuracy by 0.36% compared with the

TABLE III  
COMPARISON WITH BASELINE METHODS ON IMAGENET WITH RESNET-18/34/50. “LATENCY” = LATENCY (SPEEDUP).

Depth	Method	Top-1		Top-5		FLOPs	Latency
		Prune	Acc Drop	Prune	Acc Drop		
18	Uniform	66.12%	3.64%	87.25%	1.83%	1.06E9 (41.8%)	0.21s (1.52×)
	LCCL [35]	66.33%	3.65%	86.94%	2.29%	1.19E9 (34.6%)	-
	SFP [28]	67.10%	3.18%	87.78%	1.85%	1.06E9 (41.8%)	-
	ABCPruner [32]	67.28%	2.38%	-	-	1.01E9 ( <b>44.9%</b> )	-
	DAIS	<b>67.56%</b>	<b>2.20%</b>	<b>87.90%</b>	<b>1.18%</b>	1.03E9 (43.3%)	0.19s ( <b>1.68×</b> )
34	Uniform	71.38%	1.93%	90.52%	0.90%	2.13E9 (41.9%)	0.33s (1.83×)
	PFEC [27]	72.17%	1.06%	-	-	2.78E9 (24.2%)	-
	SFP [28]	71.83%	2.09%	90.33%	1.29%	2.16E9 (41.1%)	-
	FPGM [11]	72.54%	1.38%	<b>91.13%</b>	0.49%	2.16E9 (41.1%)	-
	ABCPruner [32]	70.98%	2.30%	-	-	2.17E9 (41.0%)	-
	DAIS	<b>72.77%</b>	<b>0.54%</b>	90.99%	<b>0.43%</b>	2.13E9 ( <b>41.9%</b> )	0.31s ( <b>1.93×</b> )
50	SFP [28]	62.14%	14.0%	84.60%	8.27%	2.38E9 (41.8%)	-
	SSS [36]	71.82%	4.30%	90.79%	2.07%	2.33E9 (43.0%)	-
	FPGM [11]	74.13%	2.02%	91.94%	0.93%	1.90E9 (53.5%)	-
	ABCPruner [32]	73.86%	2.15%	91.69%	1.27%	1.89E9 (54.1%)	-
	SCP [33]	74.20%	<b>1.69%</b>	92.00%	0.98%	1.87E9 (54.3%)	-
	DAIS	<b>74.45%</b>	1.70%	<b>92.21%</b>	<b>0.66%</b>	1.83E9 ( <b>55.3%</b> )	0.31s ( <b>1.77×</b> )

TABLE IV  
COMPARISON WITH BASELINE METHODS ON IMAGENET WITH VGG-16.  
THE PRUNING RESULTS OF 2×, 4× SPEEDUP RATIO ARE REPORTED.

Method	2×		4×	
	Top-1 Acc Drop	Top-5 Acc Drop	Top-1 Acc Drop	Top-5 Acc Drop
Jaderberg <i>et al.</i> [56]	-	-	-	9.70%
Zhang <i>et al.</i> [57]	-	-	-	3.84%
Li <i>et al.</i> [27]	-	-	-	8.60%
SSS [36]	-	-	3.93%	2.64%
CC-GAP [58]	2.78%	1.68%	-	-
DAIS	<b>1.64%</b>	<b>0.79%</b>	<b>2.91%</b>	<b>1.48%</b>

original network. It also outperformed state-of-the-art methods like SFP [28] and FPGM [11], which revealed that the simple intuitive pruning rates design (evenly pruning for each layer) could not surpass the automatic search-based pruning approach in DAIS.

**Results on ImageNet.** To evaluate the effectiveness of DAIS, extensive experiments were performed on ResNet-18/34/50 [54] and VGG-16 with the ImageNet dataset. TABLE III reports the accuracy, pruning rate and latency of the pruned models on ResNet-18/34/50. The latency is calculated by the implementation on Galaxy S9 with PyTorch Mobile<sup>1</sup>. For ResNet-18, the derived model got 1.68× speedup on Galaxy S9, and it exceeded the uniform pruning model by 1.44% accuracy with fewer FLOPs. For ResNet-34, the searched model by DAIS got minimized accuracy drop and maximized pruning ratio compared with baseline methods. On deep models like ResNet-50, DAIS still outperformed other methods. The pruning results of VGG-16 are shown in TABLE IV. DAIS got less Top-1/5 accuracy drop compared with other methods on 2×, 4× speedup ratio of FLOPs. These

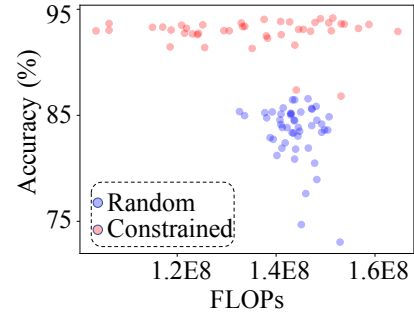


Fig. 3. Statistics of 100 network instances generated from the “Random” and “Constrained” design spaces.

experimental results verified the generality of DAIS on large-scale datasets.

### C. Importance of Identity Mapping on Deep Residual Network

To verify the importance of identity mapping in the shortcut connection, we built up two network design spaces [64], where multiple network instances were randomly generated with 110 layers, simulating channel pruning results of ResNet-110. All these network instances were trained 300 epochs on the CIFAR-10 dataset. Specifically, the “Random” design space had no constraint on the number of output channels, while in “Constrained” design space, every residual block was restricted to have the same number of input and output channels. The details of the two network design spaces are as follows:

- **“Random”.** For each pruned model instance, we randomly sampled the number of output channels  $c_l^{\text{Random}} = c_l * \mathcal{U}(0.5, 1)$ , where  $c_l$  denotes the number of output channels in the  $l$ -th layer of the original ResNet-110, and  $\mathcal{U}(0.5, 1)$  denotes a uniform sampling function ranging from 0.5 to 1.

<sup>1</sup><https://pytorch.org/mobile/home/>

TABLE V  
RESULTS OF ABLATION STUDY WITH RESNET-110 ON CIFAR-10.  
“FLOPs” = FLOPs (PRUNING RATIO).

	Pruning Acc	Acc Drop	FLOPs
Slimming	84.94%	9.48%	1.15E8 (54.46%)
Random	86.56%	7.86%	1.46E8 (42.15%)
Constrained	94.18%	0.24%	1.51E8 (40.17%)
DAIS	<b>95.02%</b>	<b>-0.60%</b>	1.01E8 ( <b>60.00%</b> )
w/o $\mathcal{R}_{\text{sym}}$	88.97%	5.45%	1.20E8 (52.62%)
w/o $\mathcal{R}_{\text{FLOPs}}$	89.15%	5.27%	1.45E8 (42.92%)
DAIS	<b>95.02%</b>	<b>-0.60%</b>	1.01E8 ( <b>60.00%</b> )
w/o annealing	93.57%	0.85%	1.54E8 (39.17%)
DAIS	<b>95.02%</b>	<b>-0.60%</b>	1.01E8 ( <b>60.00%</b> )
w/o bi-level	94.73%	-0.31%	1.04E8 (59.07%)
DAIS	<b>95.02%</b>	<b>-0.60%</b>	1.01E8 ( <b>60.00%</b> )

- **“Constrained”**. For the model instances in the “Constrained” design space, we hope to reserve most of the identity functions in the shortcut path of the residual block, which requires the same number of input and output channels in each residual block. Therefore, we designed the sampling procedure as follows. For all residual blocks in the first stage (output feature map size:  $32 \times 32$ ), we set the output channels to be  $16 * \mathcal{U}(0.5, 1)$ . For all residual blocks in the second stage (output feature map size:  $16 \times 16$ ), we set the output channels to be  $32 * \mathcal{U}(0.5, 1)$ . For all residual blocks in the third stage (output feature map size:  $8 \times 8$ ), we set the output channels to be  $64 * \mathcal{U}(0.5, 1)$ . The rest convolutional layers were randomly sampled *in the same pattern* as “Random”:  $c_l^{\text{Constrained}} = c_l * \mathcal{U}(0.5, 1)$ .

As shown in Fig. 3, With similar FLOPs  $F \in [1.1\text{E}8, 1.5\text{E}8]$ , the model instances generated from the “Constrained” design space obtained a consistent performance improvement (in terms of accuracy and pruning ratio) compared with the “Random” design space, indicating the importance of consistency between the input and output channels within a residual block.

#### D. Ablation Study

**Comparison with other search methods.** To verify the effectiveness of the search procedure of DAIS, we compared the differentiable search with three other search methods. “Slimming” denoted the method [6] in which the channel indicator was represented by the weights of BN layers, and the channels with small BN weights would be filtered out. “Random” and “Constrained” are the methods mentioned in Sec. IV-C, where 50 model instances were randomly generated by each method and the instances with the best accuracy were collected in TABLE V. DAIS outperformed these three methods on both the accuracy and the pruning rate, verifying the effectiveness of the differentiable annealing indicator search.

**Effectiveness of  $\mathcal{R}_{\text{FLOPs}}$  and  $\mathcal{R}_{\text{sym}}$ .** We conducted additional experiments to verify the advantages of  $\mathcal{R}_{\text{FLOPs}}$  and  $\mathcal{R}_{\text{sym}}$ . In TABLE V, “w/o  $\mathcal{R}_{\text{FLOPs}}$ ” denotes a model variant that replaced  $\mathcal{R}_{\text{FLOPs}}$  by  $\mathcal{R}_{\text{lasso}}$ . The replacement led to a huge accuracy drop

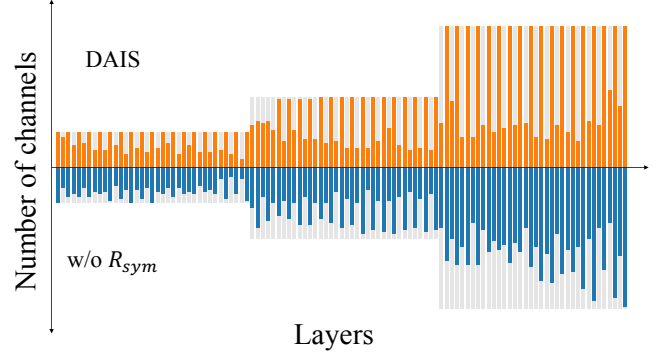


Fig. 4. The pruned model for ResNet-110 on CIFAR-10.

with less pruning rate, which indicated the effectiveness of  $\mathcal{R}_{\text{FLOPs}}$ . The second variant “w/o  $\mathcal{R}_{\text{sym}}$ ” removed the symmetry regularizer from DAIS and suffered a 6.05% accuracy reduction. The comparison revealed the importance of the symmetry regularizer on the ultra-deep residual networks.

Furthermore, we visualized the pruning result of DAIS and “w/o  $\mathcal{R}_{\text{sym}}$ ” in Fig. 4. Compared to the pruned model derived from “w/o  $\mathcal{R}_{\text{sym}}$ ”, in each residual block, the pruned model derived from DAIS was intended to prune more channels in the first  $3 \times 3$  convolution, and reserved most channels in the second  $3 \times 3$  convolution. Therefore, the pruned model could reserve most identity functions in the shortcut path, which improved the capability of gradient propagation across multiple layers.

**Effectiveness of the annealing-relaxed function.** With the help of the annealing-relaxed function, the channel indicators converge to the binarization automatically with no hand-crafted threshold. In this experiment, we explored the influences of removing the annealing-relaxed function. We designed a variant “w/o annealing” which kept the channel indicator in Eq. (6) fixed without any temperature annealing. Without temperature annealing, the  $\tilde{I}_l^i$  could not converge to the binarization automatically, and therefore we manually filtered out the channels with  $\tilde{I}_l^i < 0.55$ . Results showed that DAIS performed better in terms of both accuracy and pruning ratios, verifying the necessity of the annealing-relaxed function.

**Effectiveness of the bi-level optimization.** We implemented the bi-level optimization on the differentiable annealing indicator search, which updated the model parameters on the training set and updated the channel indicator on the validation set. To verify the necessity of the bi-level optimization, we designed the variant “w/o bi-level” in TABLE V, which jointly optimized the model parameters and channel indicators on the training set. The pruned model searched by “w/o bi-level” might overfit the training data, and results showed that it performed worse on both accuracy and pruning ratio than DAIS, verifying the effectiveness of the bi-level optimization.

#### E. Case Study

**Recoverability.** According to Guo *et al.* [20], over-pruning or incorrect pruning might happen in the pruning process and lead to degraded performance, and therefore the recoverability

TABLE VI

THE INDICATOR SEARCH PROCEDURE ON RESNET-20 ON CIFAR-10 DATASET. THE NUMBER OF OUTPUT CHANNELS WITH  $\tilde{I}_l^i > 0.5$  ARE COLLECTED. SOME CHANNELS WILL BE FIRSTLY PRUNED AND THEN RECOVERED IN THE SEARCH PROCEDURE.

Epochs	Number of Output Channels with $\tilde{I}_l^i > 0.5$ (From Layer 1 to Layer 19).																	
0	16	16	16	16	16	16	16	32	32	32	32	32	32	64	64	64	64	64
20	16	3	15	2	16	0	16	26	32	13	32	12	32	61	64	61	62	46
40	16	3	15	4	16	2	15	26	30	17	30	14	32	53	64	54	60	42
60	16	3	14	4	16	5	16	25	30	17	29	14	31	52	64	51	57	39
80	16	4	14	5	16	6	15	24	31	17	29	16	31	52	63	48	56	37
100	16	7	14	4	16	7	15	24	30	17	29	13	31	52	63	47	56	34

TABLE VII

COMPARISONS ON TRAINING SCHEMES AND TEMPERATURE DECAY SCHEMES WITH RESNET-110 ON CIFAR-10. “FLOPs” = FLOPs (PRUNING RATIO).

	Pruning Acc	Acc Drop	FLOPs
DAIS.e50	94.40%	0.02%	1.02E8 (59.69%)
DAIS.cosine	95.00%	-0.58%	1.04E8 (59.06%)
DAIS.smallT	94.61%	-0.19%	1.05E8 (58.70%)
DAIS	<b>95.02%</b>	<b>-0.60%</b>	1.01E8 ( <b>60.00%</b> )

is necessary for a good pruning algorithm. Therefore, we visualized the differentiable annealing indicator search procedure on ResNet-20 in TABLE VI. For every 20 epochs, we collected the number of output channels with  $\tilde{I}_l^i > 0.5$ . We observed that the number of output channels did not decrease monotonically, and some channels were recovered as the training proceeded. For example, in the second convolutional layer, the number of output channels was decreased to 3, and then recovered to 7. The recoverability revealed that DAIS could rectify the mistakes of over-pruning or incorrect pruning, which might also explain the effectiveness of DAIS.

**Robustness of DAIS.** We first explored the impact of a shorter training scheme. The “DAIS.e50” trained 50 epochs for the differentiable annealing indicator search procedure and got similar performance with the original DAIS, indicating the efficiency of DAIS. The second experiment explored the impact of different temperature decay schemes. “DAIS.cosine” leveraged a cosine decay scheme  $\psi(n) = 49 \times (1 - \cos(\frac{\pi}{2}n/N_{\max})) + 1$ , while “DAIS.smallT” adopted a smaller termination temperature by  $\psi(n) = 99 \times n/N_{\max} + 1$ . All the corresponding results were shown in TABLE VII. We also explored the effect of the  $\alpha$  initialization in TABLE VIII. All variants got similar performance, which verified the robustness of DAIS to the normal temperature decay schemes and  $\alpha$  initialization.

**One-shot capability of DAIS.** We conducted an iterative pruning scheme on MobileNet, denoted by “Iterative- $i$ ” in Table IX. Similar to most iterative pruning schemes [24], [65], the pruning rate was set to be large (45%) in the first round and gradually increased 10% in latter rounds. The experimental result showed that the original one-shot DAIS obtained slightly better performance than “Iterative-3”, indicating the differentiable search procedure could effectively search the pruned models in a one-shot manner.

TABLE VIII

COMPARISONS ON DIFFERENT  $\alpha$  INITIALIZATION WITH RESNET-56 ON CIFAR-10. “FLOPs” = FLOPs (PRUNING RATIO).

	Pruning Acc	Acc Drop	FLOPs
DAIS. $\mathcal{N}(0, 0.1^2)$	93.11%	1.42%	5.41E7 (57.01%)
DAIS. $\mathcal{N}(0, 0.05^2)$	93.36%	1.17%	5.34E7 ( <b>57.54%</b> )
DAIS. $\mathcal{N}(1, 0.05^2)$	92.97%	1.56%	5.55E7 (55.89%)
DAIS	<b>93.71%</b>	<b>0.82%</b>	5.61E7 (55.40%)

TABLE IX

RESULTS OF ITERATIVE PRUNING OF DAIS ON CIFAR-10. “Iterative- $i$ ” DENOTES THE  $i$ -TH ROUND PRUNING RESULT BY DAIS.

	Pruning Acc	Acc Drop	FLOPs
MobileNet	92.87%	—	3.44E8
Iterative-1	91.66%	1.21%	1.88E8 (45.34%)
Iterative-2	91.76%	1.11%	1.50E8 (56.40%)
Iterative-3	91.77%	1.10%	1.18E8 (65.70%)
DAIS	<b>91.87%</b>	<b>1.00%</b>	1.15E8 ( <b>66.60%</b> )

**Trade-off between accuracy and FLOPs.** DAIS can perform fine-grained search on the pruned model with a target FLOPs  $F$  by  $\mathcal{R}_{\text{FLOPs}}$ , which controls the trade-off between FLOPs and accuracy. We derived models with different pruning rates on ResNet-56 formatted by (pruning rate, accuracy): (27.43%, 94.91%), (48.85%, 94.24%), (55.36%, 93.71%), (70.92%, 93.53%).

#### F. Transfer Learning

We have demonstrated the effectiveness of DAIS in classification tasks, and we explored its performance on some other computer vision tasks like semantic segmentation and scene text recognition.

**Semantic segmentation.** We selected DeepLabV3+ [66] with ResNet-34 as the baseline model and performed experiments on PASCAL VOC 2012 [67]. As suggested by Liu *et al.* [12], we first pruned the ResNet-34 backbone with DAIS on ImageNet, and then finetuned a DeepLabV3+ with the pruned ResNet-34 as its backbone on PASCAL VOC 2012. In the finetuning process on PASCAL VOC 2012 dataset, we followed the same training strategies as the original DeepLabV3+ paper. All models were trained for 50 epochs with the training set of PASCAL VOC 2012 and an

TABLE X  
SEMANTIC SEGMENTATION RESULTS BASED ON DEEPLABV3+ ON  
PASCAL VOC 2012.

	PA	MPA	MIoU	FWIoU	FLOPs
DeepLabV3+	92.87%	81.71%	72.52%	87.16%	5.67E10
Uniform	91.39%	73.63%	68.42%	84.22%	4.94E10 (12.78%)
DAIS	<b>92.51%</b>	<b>79.23%</b>	<b>71.29%</b>	<b>86.48%</b>	4.80E10 ( <b>15.39%</b> )

TABLE XI  
SCENE TEXT RECOGNITION ACCURACIES BASED ON STN ACROSS 7 REAL  
WORLD DATASETS.

	IIIT5k	SVT	IC03	IC13	IC15	SVTP	CUTE
STN	92.9%	88.4%	91.9%	89.5%	78.3%	78.9%	78.5%
Uniform	<b>93.4%</b>	88.9%	91.7%	<b>89.3%</b>	<b>79.0%</b>	78.6%	76.4%
DAIS	93.2%	<b>89.8%</b>	<b>93.2%</b>	<b>89.3%</b>	78.8%	<b>78.8%</b>	<b>79.2%</b>

augmentation dataset [68]. The performance was measured in terms of Pixel Accuracy (PA), Mean Pixel Accuracy (MPA), Mean Interaction over Union (MIoU) and Frequency Weighted Interaction over Union (FWIoU). As illustrated in TABLE X, DAIS reduced 15.39% FLOPs on DeepLabV3+ with small pixel accuracy drop (92.87%  $\rightarrow$  92.51%), indicating that the pruned model derived by DAIS could be transferred into another task without major performance changes.

**Scene text recognition.** We also explored the performance of DAIS on scene text recognition tasks. We adopted STN [69], [70] with ResNet-34 as the baseline model. Similar to the segmentation experiments in Sec. 4.5, we replaced the original ResNet-34 with the pruned model searched by DAIS on ImageNet [18]. All models were trained on SynthText [71] and Synth90K [72] and evaluated on 7 real-world datasets: IIIT5k [73], SVT [74], IC03 [75], IC13 [76], IC15 [77], SVTP [78] and CUTE [79]. We adopted the same training hyperparameters as Aster [80]. DAIS had the pruning rate of 17.06% and the uniform pruning method prunes 16.98% FLOPs. As illustrated in TABLE XI, DAIS outperformed the uniform pruning on 4 datasets with less FLOPs. These experiments indicated the generality of our method on several vision tasks including image classification, semantic segmentation and scene text recognition.

## V. DISCUSSIONS

DAIS focuses on the automatic channel pruning, which also implicitly decreases the latency, memory footprint, and energy consumption of the original models. Specifically, DAIS reduces the FLOPs, model parameters, and the memory footprint. As shown in TABLE III, the pruned models also have lower latency on mobile devices. In terms of energy consumption, DAIS has a shorter search time compared to other methods [13], [7], which implies lower CO2 marginal emission costs and cloud computing costs according to APQ [81]. Besides, DAIS has lower calculation cost and calculation delay as shown in TABLE III, which also decrease the CO2 emissions following [82].

## VI. CONCLUSION

In this paper, we propose Differentiable Annealing Indicator Search (DAIS), which leverages the channel indicator to represent the sparsity and searches for an appropriate pruned model with the computation cost constraints. Specifically, DAIS approximates the binarized channel indicator with the annealing-relaxed indicator and then jointly optimizes the indicator and model parameters with gradient-based bi-level optimization. The annealing-relaxed indicator will automatically converge to the binarized state as the optimization proceeds and temperature anneals. Furthermore, DAIS proposes a continuous FLOPs estimator regularizer to precisely constrain model sizes and a symmetry regularizer to optimize the gradient propagation on very deep residual networks. Experimental results show that DAIS outperforms state-of-the-art methods on CIFAR-10, CIFAR-100, and ImageNet on different architectures, verifying the effectiveness of the differentiable search.

## ACKNOWLEDGMENT

This work is partially supported by National Key R&D Program of China (2020YFB2103801), Beijing Academy of Artificial Intelligence (BAAI), NSFC (National Natural Science Foundation of China) 62032003, and BJNSF (Beijing Municipal Natural Science Foundation) L192004.

## REFERENCES

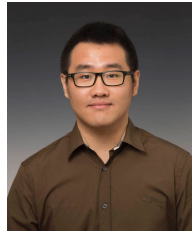
- [1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.
- [2] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.
- [3] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *ECCV*. Springer, 2016, pp. 483–499.
- [4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.
- [5] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021. [Online]. Available: <https://doi.org/10.1016/j.neucom.2021.07.045>
- [6] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.
- [7] X. Dong and Y. Yang, "Network pruning via transformable architecture search," in *Advances in Neural Information Processing Systems*, 2019, pp. 760–771.
- [8] T. Chen, T. Moreau, Z. Jiang, L. Zheng, E. Yan, H. Shen, M. Cowan, L. Wang, Y. Hu, L. Ceze *et al.*, "TVM: An automated end-to-end optimizing compiler for deep learning," in *OSDI*, 2018.
- [9] "Tensorflow lite," <https://www.tensorflow.org/lite>, 2017.
- [10] X. Jiang, H. Wang, Y. Chen, Z. Wu, L. Wang, B. Zou, Y. Yang, Z. Cui, Y. Cai, T. Yu, C. Lv, and Z. Wu, "Mnn: A universal and efficient inference engine," <https://github.com/alibaba/MNN>, 2020.
- [11] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4340–4349.
- [12] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *ICLR*, 2018.
- [13] 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 *ECCV*. Springer, 2018, pp. 815–832.

- [14] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *ICLR*, 2019.
- [15] J.-H. Luo, J. Wu, and W. Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [16] Z. You, K. Yan, J. Ye, M. Ma, and P. Wang, "Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 2130–2141.
- [17] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Master's thesis, Department of Computer Science, University of Toronto*, 2009.
- [18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 248–255.
- [19] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *NIPS*, 2015, pp. 1135–1143.
- [20] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Advances in Neural Information Processing Systems*, 2016.
- [21] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1398–1406.
- [22] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9194–9203.
- [23] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 875–886.
- [24] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, pp. 4876–4883.
- [25] L. S. Blackford, A. Petit, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry *et al.*, "An updated set of basic linear algebra subprograms (blas)," *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 135–151, 2002.
- [26] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems*, 2016.
- [27] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *ICLR*, 2017.
- [28] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, J. Lang, Ed. ijcai.org, 2018, pp. 2234–2240.
- [29] T. Li, B. Wu, Y. Yang, Y. Fan, Y. Zhang, and W. Liu, "Compressing convolutional neural networks via factorized convolutional filters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3977–3986.
- [30] Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, and Y. Yang, "Learning filter pruning criteria for deep convolutional neural networks acceleration," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [31] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, "Hrank: Filter pruning using high-rank feature map," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [32] M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, and Y. Tian, "Channel pruning via automatic structure search," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, C. Bessiere, Ed., 2020*, pp. 673–679.
- [33] M. Kang and B. Han, "Operation-aware soft channel pruning using differentiable masks," in *Proceedings of Machine Learning and Systems 2020*, 2020, pp. 2505–2514.
- [34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 448–456.
- [35] X. Dong, J. Huang, Y. Yang, and S. Yan, "More is less: A more complicated network with less inference complexity," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5840–5848.
- [36] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [37] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured cnn pruning via generative adversarial learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2790–2799.
- [38] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *ICLR*, 2017.
- [39] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *ICLR*, 2019.
- [40] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.
- [41] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *ICLR*, 2017.
- [42] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.
- [43] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*. JMLR. org, 2017, pp. 2902–2911.
- [44] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4780–4789.
- [45] Y. Zhou, X. Sun, C. Luo, Z. Zha, and W. Zeng, "Posterior-guided neural architecture search," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence*. AAAI Press, 2020, pp. 6973–6980.
- [46] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *International Conference on Machine Learning*, 2018, pp. 550–559.
- [47] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International Conference on Machine Learning*, 2018, pp. 4095–4104.
- [48] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "Detnas: Backbone search for object detection," in *Advances in Neural Information Processing Systems*, 2019, pp. 6638–6648.
- [49] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI*, 2019.
- [50] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10734–10742.
- [51] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," in *ICLR*, 2019.
- [52] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair DARTS: eliminating unfair advantages in differentiable architecture search," in *ECCV*, ser. Lecture Notes in Computer Science, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., vol. 12360. Springer, 2020, pp. 465–480.
- [53] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *ICLR*, 2020.
- [54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [55] W. Hua, Y. Zhou, C. M. De Sa, Z. Zhang, and G. E. Suh, "Channel gating neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 1884–1894.
- [56] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *British Machine Vision Conference, BMVC 2014, Nottingham, UK, September 1-5, 2014*, M. F. Valstar, A. P. French, and T. P. Pridmore, Eds. BMVA Press, 2014.
- [57] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, 2016.
- [58] Y. Li, S. Lin, J. Liu, Q. Ye, M. Wang, F. Chao, F. Yang, J. Ma, Q. Tian, and R. Ji, "Towards compact cnns via collaborative compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 6438–6447.

- [59] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2015.
- [60] I. Loshchilov and F. Hutter, "SGDR: stochastic gradient descent with warm restarts," in *ICLR*, 2017.
- [61] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *AAAI*. AAAI Press, 2020, pp. 13 001–13 008.
- [62] B. Heo, J. Kim, S. Yun, H. Park, N. Kwak, and J. Y. Choi, "A comprehensive overhaul of feature distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
- [63] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
- [64] I. Radosavovic, R. P. Kosaraju, R. B. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Computer Vision Foundation / IEEE, 2020, pp. 10 425–10 433.
- [65] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," in *ICLR Workshop*, 2018.
- [66] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.
- [67] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," <http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html>.
- [68] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 991–998.
- [69] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in neural information processing systems*, 2015, pp. 2017–2025.
- [70] S. Long, Y. Guan, B. Wang, K. Bian, and C. Yao, "Rethinking irregular scene text recognition," *arXiv*, pp. arXiv–1908, 2019.
- [71] A. Gupta, A. Vedaldi, and A. Zisserman, "Synthetic data for text localisation in natural images," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [72] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," in *Workshop on Deep Learning, NIPS*, 2014.
- [73] A. Mishra, K. Alahari, and C. Jawahar, "Scene text recognition using higher order language priors," in *BMVC-British Machine Vision Conference*. BMVA, 2012.
- [74] K. Wang, B. Babenko, and S. Belongie, "End-to-end scene text recognition," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1457–1464.
- [75] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young, "Icdar 2003 robust reading competitions," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings*. Citeseer, 2003, pp. 682–687.
- [76] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. i Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. De Las Heras, "Icdar 2013 robust reading competition," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pp. 1484–1493.
- [77] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu *et al.*, "Icdar 2015 competition on robust reading," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2015, pp. 1156–1160.
- [78] T. Quy Phan, P. Shivakumara, S. Tian, and C. Lim Tan, "Recognizing text with perspective distortion in natural scenes," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 569–576.
- [79] A. Risnumawan, P. Shivakumara, C. S. Chan, and C. L. Tan, "A robust arbitrary text detection system for natural scene images," *Expert Systems with Applications*, vol. 41, no. 18, pp. 8027–8048, 2014.
- [80] B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai, "Aster: An attentional scene text recognizer with flexible rectification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2035–2048, 2018.
- [81] T. Wang, K. Wang, H. Cai, J. Lin, Z. Liu, H. Wang, Y. Lin, and S. Han, "APQ: joint search for network architecture, pruning and quantization policy," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. IEEE, 2020, pp. 2075–2084.
- [82] A. Fu, M. S. Hosseini, and K. N. Plataniotis, "Reconsidering CO2 emissions from computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 2311–2317. [Online]. Available: [https://openaccess.thecvf.com/content/CVPR2021W/RCV/html/Fu\\_Reconsidering\\_CO2\\_Emissions\\_From\\_Computer\\_Vision\\_CVPRW\\_2021\\_paper.html](https://openaccess.thecvf.com/content/CVPR2021W/RCV/html/Fu_Reconsidering_CO2_Emissions_From_Computer_Vision_CVPRW_2021_paper.html)



**Yushuo Guan** received the Master degree at the School of Electronics Engineering and Computer Science in Peking University, China, in 2021. He received his Bachelor degree from Peking University in 2018. His research focuses on model compression, scene text recognition and other machine learning applications.



**Ning Liu** received the Ph.D. degree in computer engineering from the Northeastern University, Boston, MA, USA, in 2019. He is a researcher at Midea Group. His current research interests lie in deep learning, deep model compression and acceleration, deep reinforcement learning, and edge computing.



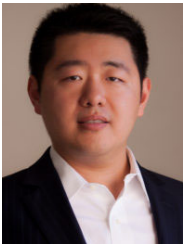
**Pengyu Zhao** received his Bachelor and Master degrees, both at the School of Electronics Engineering and Computer Science from Peking University, China, in 2017 and 2020 respectively. His research focuses on model compression, neural architecture search and other machine learning applications.



**Zhengping Che** received the Ph.D. degree in Computer Science from the University of Southern California, Los Angeles, CA, USA, in 2018, and the B.Eng. degree in Computer Science from Tsinghua University, Beijing, China, in 2013. He is now with AI Innovation Center, Midea Group. His current research interests lie in the areas of machine learning, deep learning, computer vision, and time series analysis with applications to robot learning.



**Kaigui Bian** received the PhD degree in computer engineering from Virginia Tech, Blacksburg, USA in 2011. He is currently an associate professor in the Institute of Network Computing and Information Systems, School of EECS, Peking University. His research interests include mobile computing, cognitive radio networks, network security, and privacy.



**Yanzhi Wang** received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2009, and the Ph.D. degree in computer engineering from the University of Southern California, Los Angeles, CA, USA, in 2014. His research interests include energy-efficient and high-performance implementations of deep learning and artificial intelligence systems, emerging deep learning algorithms/systems, generative adversarial networks, and deep reinforcement learning.



**Jian Tang** received his Ph.D degree in Computer Science from Arizona State University in 2006. He is an IEEE Fellow and an ACM Distinguished Member. He is with Midea Group. His research interests lie in the areas of AI, IoT, Wireless Networking, Mobile Computing and Big Data Systems. Dr. Tang has published over 160 papers in premier journals and conferences. He received an NSF CAREER award in 2009. He also received several best paper awards, including the 2019 William R. Bennett Prize and the 2019 TCBD (Technical Committee on

Big Data) Best Journal Paper Award from IEEE Communications Society (ComSoc), the 2016 Best Vehicular Electronics Paper Award from IEEE Vehicular Technology Society (VTS), and Best Paper Awards from the 2014 IEEE International Conference on Communications (ICC) and the 2015 IEEE Global Communications Conference (Globecom) respectively. He has served as an editor for several IEEE journals, including IEEE Transactions on Big Data, IEEE Transactions on Mobile Computing, etc. In addition, he served as a TPC co-chair for a few international conferences, including the IEEE/ACM IWQoS'2019, MobiQuitous'2018, IEEE iThings'2015. etc.; as the TPC vice chair for the INFOCOM'2019; and as an area TPC chair for INFOCOM 2017-2018. He is also an IEEE VTS Distinguished Lecturer, and the Chair of the Communications Switching and Routing Committee of IEEE ComSoc.