Evolutionary Shallowing Deep Neural Networks at Block Levels

Yao Zhou, *Member, IEEE*, Gary G. Yen, *Fellow, IEEE*, and Zhang Yi, *Fellow, IEEE*

***Abstract*— Neural networks have been demonstrated to be trainable even with hundreds of layers, which exhibit remarkable improvement on expressive power and provide significant performance gains in a variety of tasks. However, the prohibitive computational cost has become a severe challenge for deploying them on resource-constrained platforms. Meanwhile, widely adopted deep neural network architectures, for example, ResNets or DenseNets, are manually crafted on benchmark datasets, which hamper their generalization ability to other domains. To cope with these issues, we propose an evolutionary algorithm-based method for shallowing deep neural networks (DNNs) at block levels, which is termed as ESNB. Different from existing studies, ESNB utilizes the ensemble view of block-wise DNNs and employs the multiobjective optimization paradigm to reduce the number of blocks while avoiding performance degradation. It automatically discovers shallower network architectures by pruning less informative blocks, and employs knowledge distillation to recover the performance. Moreover, a novel prior knowledge incorporation strategy is proposed to improve the exploration ability of the evolutionary search process, and a correctness-aware knowledge distillation strategy is designed for better knowledge transferring. Experimental results show that the proposed method can effectively accelerate the inference of DNNs while achieving superior performance when compared with the state-of-the-art competing methods.**

***Index Terms*— Deep neural networks, evolutionary algorithm, multiobjective optimization, network pruning.**

I. Introduction

DEEP neural networks (DNNs) have proven extraordinarily effective in various fields, and achieved surpassing human-level performance from visual recognition [1] to video games playing [2]. It has been empirically recognized that building very deep networks is of crucial importance for these problems [3], which can be evidenced by the drastically increasing depth of models won the ImageNet competitions in recent years [4], [5]. However, the notorious gradient vanishing/exploding problems [6] have hampered the convergence when stacking more layers in neural networks, and the model size becomes cumbersome during the depth expansion [7]. To cope with these issues, block-wise design and skip connections have been introduced to enable the training of DNNs with up to thousands of layers [8], and the resulting network architectures have been extensively adopted as the basic component in a variety of applications, such as Go game playing [9], speech recognition [10], medical image analysis [11], just to name a few. Although block-wise design can generally obtain better results when compared with those of previous DNNs, the computational overhead has significantly increased which limits their utilization under many practical scenarios. For instance, ResNet-152 uses over 6×10^7 parameters that requires approximately 11.3×10^9 FLOPs for a single inference, which is a heavy burden for deploying them on resource constrained platforms and as a result severely limits their applicability. Therefore, reducing the computational cost of block-wise DNNs is in critical need to handle this problem. Meanwhile, the skip connections impose restrictions on the network architecture, therefore complicate the process of model pruning [12], since subtle change at connection or filter level could result in invalid model. Fortunately, the ensemble view [13] reveals that a residual network behaves such as an ensemble of relatively shallow networks, which provides the inspiration of shallowing DNNs at block levels by modeling the problem as ensemble pruning [14]. However, a systematic approach for shallowing DNNs is yet to be investigated.

Many studies have been focused on improving the efficiency of DNNs. For instance, *sparse convolution* [15] introduces sparsity to reduce the number of active neurons, while it generally requires extra mechanism or specified hardware to support their running. *Filter pruning* [12] uses already-trained neural network as input, and removes less important parameters at filter level. However, the conflicting nature between performance degradation and parameters reduction makes it difficult to determine the target pruned model, thus impedes their practical applicability. In addition, the skip connections impose restrictions for filter pruning. For example, the numbers of filters need to be the same for those layers involved in the identity mappings [16], thus a large proportion of layers cannot be pruned directly. On the other hand, it has been shown that removing blocks results in little damage to the test accuracy [13], which indicates that some blocks might be redundant. However, the performance will drop severely when

Manuscript received 30 December 2019; revised 14 May 2020, 16 October 2020, and 25 January 2021; accepted 11 February 2021. Date of publication 26 February 2021; date of current version 2 September 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0100201, in part by the National Natural Science Foundation of China under Grant 62006163, and in part by the National Postdoctoral Program for Innovative Talents under Grant BX20200226. *(Corresponding author: Gary G. Yen.)*

Yao Zhou and Zhang Yi are with the College of Computer Science, Sichuan University, Chengdu 610065, China (e-mail: zy3381@gmail.com; zhangyi@scu.edu.cn).

Gary G. Yen is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA (e-mail: gyen@okstate.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2021.3059529.

Digital Object Identifier 10.1109/TNNLS.2021.3059529

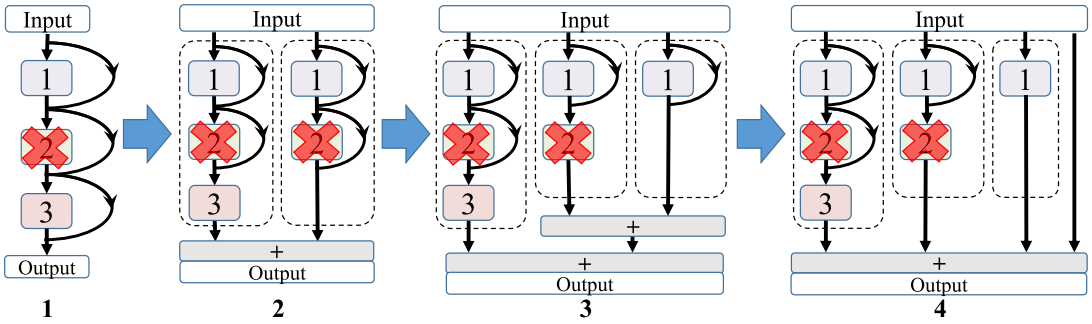

Fig. 1. Unfolding a 3-block residual network. The original network is shown in step 1, and the next two steps show the intermediate results of unfolding the skip connection in the last two blocks. The final step shows the ensemble view. Each dotted square represents a path, and blocks marked the same number are shared. The cross mark implies removing that block. As can be seen in step 4, removing the second block only affects 2 paths out of 4 in the ensemble.

a considerable number of blocks are removed [13], which is conflict to the objective of improving model efficiency. As a remedy, Pareto ensemble pruning [14] can be adopted for handling this issue, since it has shown effectiveness in simultaneously maximizing the generalization performance and minimizing the number of base learners. Intuitively, a block-wise DNN can be unfolded as an ensemble with multiple paths. Take the ResNet as an illustrative example, unfolding the skip connection in the last block results in a two-path view, which is shown as step 2 in Fig. 1. Similarly, the second block from the bottom can be unfolded in the same manner shown as step 3 in Fig. 1. Lastly, the sum operations are merged to obtain the final ensemble view shown as step 4 in Fig. 1. It can be seen that removing the second block is equivalent to pruning base classifiers in the ensemble.

Motivated by the above observations, we propose an evolutionary algorithm (EA)-based method for shallowing deep neural networks at block levels (ESNB). Different from aforementioned studies, our method formulates the problem of DNN acceleration using a dual-objective function which minimizes error and number of blocks, and further applies multiobjective EA [17] to solve it. In particular, the weight distribution of a well-trained model is exploited as a prior knowledge (PK) and incorporated into the EA, which aims to speed up the convergence. To compensate the performance degradation of the shallowed models, knowledge distillation (KD) [18] is further utilized to form a teacher–student scheme for transferring knowledge.

The overall framework of ESNB is depicted in Fig. 2. We first extract PK from the original model, and use it to guide the population initialization of EA. Then the evolutionary shallowing is performed in a multiobjective optimization fashion based on the ensemble explanation, in which the number of blocks and the performance drop are simultaneously optimized as two conflicting objectives. To recover the performance after shallowing, KD is adopted to encourage the output of the pruned model to match the softened output of the original model, as well as the hard label from the ground truth. This design provides the advantage for automatically transferring knowledge from not only the prediction of the original model, but also its architecture and weights. The main contributions of this article can be summarized as follows:

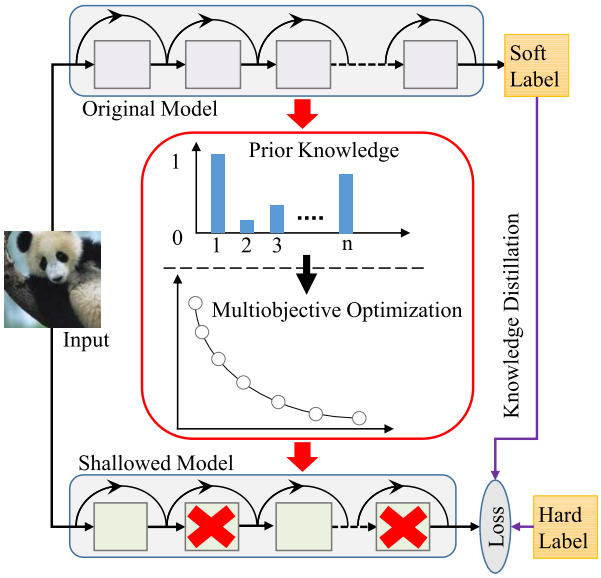

Fig. 2. Overall framework of the proposed ESNB. The left side shows the input image. For the original model, the distribution of the weights is extracted as a PK, and further incorporated into the EA for shallowing the original model in a multiobjective optimization fashion. Then KD is employed to compensate the performance degradation of the shallowed model by utilizing both soft and hard labels.

1) An EA-based method for shallowing DNNs is proposed, in which a PK incorporation strategy is designed to extract known information from given neural network models and then guide the initialization of EA, thus accelerating the evolution process.
2) A correctness-aware knowledge distillation (CKD) strategy is designed for better knowledge transferring, in which the correctness of the teacher model's output distribution is explicitly considered in an improved cost function.
3) The proposed ESNB is empirically evaluated on datasets including CIFAR-10, CIFAR-100 ImageNet-16-120, and ImageNet datasets. Experimental results show that the proposed method can effectively improve the efficiency of block-wise DNNs, while achieving even better performance compared with the original models.

The rest of the article is organized as follows. Section II reviews related works about block-wise DNNs, pruning, and

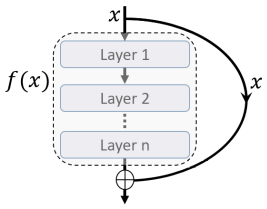

Fig. 3. Residual block. The dotted square represents the residual mapping to be learned, which contains parameterized layers and is denoted as $f(x)$.

EAs. Then the proposed ESNB is elaborated in Section III. Section IV shows experimental results, as well as corresponding analysis. Conclusions are drawn in Section V.

II. Related Work

A. Block-Wise DNNs

Blockwise DNNs have been widely employed in many tasks and have shown significant performance improvements, here we briefly review ResNet [8] and DenseNet [19] as representatives. The ResNet introduces a basic component named residual block for constructing very deep network architectures. An example of the residual block is shown in Fig. 3.

Formally, for a given input x, the computation of a residual block can be defined as $y = f(x) + x$, which provides a nice property for back-propagating gradients without vanishing since

$$\frac{\partial y}{\partial x} = \frac{\partial f(x)}{\partial x} + 1. \tag{1}$$

Meanwhile, as illustrated in Fig. 1, the recursive nature implies that ResNet behaves as ensembles of relatively shallow networks [13], which can be analytically understood by expanding the computation of a residual block into the following form:

$$\begin{aligned} y &= y_2 + f_3(y_2) \\ &= y_1 + f_2(y_1) + f_3(f_2(y_1) + y_1) \\ &= [x + f_1(x)] \\ &\quad + [f_2(f_1(x) + x)] \\ &\quad + [f_3(x + f_1(x) + f_2(x + f_1(x)))] \end{aligned}$$

where y_i and f_i represent the output and residual mapping of the i-th block ($i \in \{1, 2, 3\}$), respectively.

In fact, the residual connection scheme has been widely adopted in other block-wise DNNs, such as Inception-ResNet [20] and MobileNetV2 [21], which can be regarded as ensembles as well. Different from ResNet, DenseNet [19] introduces dense block, in which dense layers are progressively built to receive concatenated feature maps from all preceding layers. As illustrated in Fig. 4, this design encourages feature reuse and enhances gradient flow. It is worth noting that a dense layer in the DenseNet generally contains multiple convolutional layers, which is conceptually similar to residual block. In addition, the skip connections provide the property of ensembles since removing any dense layer will not result in a completely disconnected computation flow.

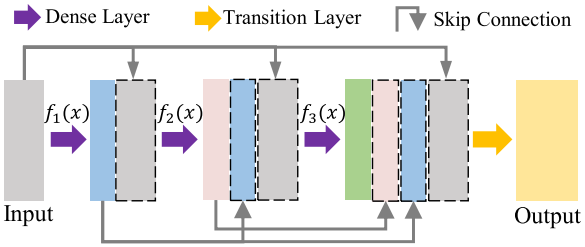

Fig. 4. Dense block with 3 dense layers. The dotted squares represent concatenated feature maps from the skip connections, each dense layer could contain multiple convolutional layers. The transition layer uses 1×1 kernels to squash feature maps between dense blocks.

B. Network Compression

Various network compression approaches have been investigated for accelerating DNNs [22]. In [23], low-rank approximation is employed to approximate the convolutional layers and fully connected layers with low-rank matrices. Also, using low-precision floating-point numbers to represent the parameters within DNNs can considerably reduce the computational overhead and storage for a complex model [24], [25]. Designing compact and efficient architectures [21] by introducing depth-wise convolution [26] can considerably decrease the number of operations and memory consumption. Moreover, KD [18] has become popular for speeding up DNNs, which constructs a small model to mimic the softened target from the original cumbersome model. In [27], KD is utilized to train student network that is deeper and thinner than the teachers. In [28], it shows that the flow between layers can be transferred as knowledge for compressing DNNs. Interestingly, the results in [29] suggest that label smoothing regularization can be adopted as a virtual teacher for KD. Also, another study [30] reveals that the knowledge of a neural network is represented by both parameters and architecture, and KD-guided reward is leveraged to search for the best student architectures.

Network pruning [12], [31] is another type of network compression approach, which aims to eliminate the redundant parameters and accelerate the inference of DNNs. The basic idea is to remove parameters according to the estimated importance in a structured or nonstructured fashion, while leading to imperceptible performance loss. Many pruning criteria have been proposed for measuring the redundancies of parameters in DNNs [32]–[39]. Particularly, HRank [35] suggests that the ranks of feature maps can be adopted as importance indicator of the corresponding weights, where low-rank feature maps contain less information, thus filters generating feature maps with lower ranks can be pruned first. In [36], the temporal Jacobian is introduced to determine the effect of pruning parameters from recurrent neural networks. Compression using residual connections and limited data (CURL) [37] prunes the filters both inside and outside the residual blocks, and uses KL-divergence to calculate the information loss of output probability distributions when removing filters. In particular, all residual blocks in the same stage are pruned simultaneously in CURL, since the shortcut connections and the sum operations require the channel number of each block in the same group to be consistent. The lottery ticket hypothesis [38]

suggests that the initial weight has a significant impact on the accuracy of pruned subnetworks. In addition, the results in [39] indicate that the essence of structured pruning lies in finding optimal pruned architecture.

Recently, neural architecture search (NAS) has shown the potential in automatically designing efficient network models [40]. Inspired by the design of stacked blocks in handcrafted DNNs [8], [20], earlier NAS approaches aim at searching building blocks [41]. In particular, reinforcement learning [42], differentiable search [43], and evolutionary search [44] have been applied to discovering well-performed networks, in which the searched models achieve better accuracies while using orders reduced parameters compared with previous neural network architectures [8], [19]. However, evaluating each of the searched network via individually training is computationally expensive, which impedes its use in practice. Moreover, the cost for training searched networks grows linearly as the number of different efficiency constraints increases in diversified deployment scenarios. To alleviate these issues, weight-sharing methods have been investigated to bridge the performance gap between the supernet and subnets during training [45]; thus, the evaluation of a candidate network could be much more efficient. To accelerate the deployment on different hardware platforms, once-for-all (OFA) [46] trains a supernet using the architecture space such as MobileNetV3 [47], from which a subnet with different architectural configurations can be directly selected without retraining. BigNAS [48] further eliminates the post-processing steps including fine-tuning for instant deployment with respect to the given constraints. While this study is related to the NAS methods, the main objective is to reduce the redundancies by shallowing a given model instead of searching the optimal architecture.

Different from existing studies, the proposed method aims at evolutionary shallowing the network structure at block levels in a multiobjective fashion. It transfers the knowledge of DNNs including architecture, weight, and output distribution. In addition, the proposed method is orthogonal to many other pruning strategies, which could also be explored as post-process steps in the EA framework.

III. Multiobjective Evolutionary Algorithms Based Blockwise DNN Shallowing

This section details the proposed EA for shallowing DNNs. First, we present the overall framework of ESNB, in which we use the NSGA-II algorithm [17], with a novel modification to incorporate PK into the initial population, plus the modeling of the multiobjective optimization problem (MOP). Then, the computation of the PK and its incorporation in the proposed method are elaborated. Lastly, details about knowledge transferring to build a teacher–student paradigm are given.

A. Overall Framework

The overall framework of ESNB is shown in Algorithm 1. A well-trained DNN model $\mathcal{M}$ is provided as input, and the PK extraction is processed as the first step. The basic idea is to estimate the importance of each block according to the magnitude of its weight. More details about the PK extraction will be elaborated in the following subsection. We use binary representation in the EA to encode the blocks, where the values of 0 and 1 mean a removed block and intact block, respectively. Thus, the size of the search space is 2^L, where L is the length of the representation. A population $\mathcal{P}_0$ with N individual solutions is then initialized according to the extracted PK $\mathcal{K}$, which is expected to provide a better initial point in the search space than that of the randomly initialized one. The following evaluation step will assign the objective values to each individual solution in the population $\mathcal{P}_0$ by evaluating them on the given dataset. Next, pairs of solutions are randomly selected from the population, and used as parents (I_1, I_2) to reproduce offspring $(\hat{I}_1, \hat{I}_2)$. All the evaluated offspring will be combined into the population from the previous generation, and a number of N solutions could survive to the next generation through the selection. Lastly, the optimal solution in the population can be selected for shallowing the DNN model according to desired preference information.

Algorithm 1 Overall Framework of ESNB

Input : A well-trained model $\mathcal{M}$, the number of generations G, the population size N, the mutation probability p, and the dataset $\mathcal{D}$
Output: The pruned model $\hat{\mathcal{M}}$

$\mathcal{K} \leftarrow$Prior-Knowledge-Extraction($\mathcal{M}$) ;
$\mathcal{P}_0 \leftarrow$Population-Initialization(N, $\mathcal{K}$) ;
Evaluation($\mathcal{P}_0, \mathcal{M}, \mathcal{D}$);
for $g = 0, 1, \ldots, G$ **do**
  $\mathcal{Q}_g \leftarrow \emptyset$;
  for $n = 0, 1, \ldots, N/2$ **do**
    $I_1, I_2 \leftarrow$ Sample $(\mathcal{P}_g)$;
    $\hat{I}_1, \hat{I}_2 \leftarrow$ Variation (I_1, I_2) ;
    Add $\{\hat{I}_1, \hat{I}_2\}$ to $\mathcal{Q}_g$;
  end
  Evaluation($\mathcal{P}_g, \mathcal{M}, \mathcal{D}$);
  $\mathcal{P}_{g+1} \leftarrow$ Selection $(\mathcal{P}_g \cup \mathcal{Q}_g, N)$;
  Terminate when stop condition satisfied;
end
Shallowing $\mathcal{M}$ according to the selected solution from $\mathcal{P}_{g+1}$, and retrain with knowledge distillation;

1) Variation: Variation utilizes crossover and mutation genetic operators to explore the search space. Based on the binary representation, the conventional one-point crossover [49] is introduced to recombine information from selected parents. Specifically, offspring are generated by recombining the genetic materials of parents. Similarly, bit-flip mutation is applied to the generated offspring to promote the diversity of the population and encourage exploration in the search space.

2) Selection: To handle the conflicting nature between performance degradation and parameters reduction when shallowing block-wise DNNs, the selection process utilizes Pareto-optimality to optimize the above two objectives simultaneously. The basic idea is to explicitly preserve the

nondominated solutions, which are not completely worse than any other solution in terms of objective values. For instance, assuming there are two candidate solutions that prune the same amount of blocks and result in two distinct shallowed models, a solution is said to dominate the other if it finds a model with better accuracy than the other one, since the goal is to prune parameters while maintaining performance. In this case, the solution dominated by the other is considered less important one in the population. Since the size of the population is set to a fixed value N, the nondominated solutions less beneficial to the diversity of the population are discarded during the evolution. Specifically, crowding distance [17] is adopted as a measurement for estimating the diversity, which encourages finding a set of widespread solutions in the objective space.

3) Objective Definition: Formally, the mappings of blocks in a DNN are denoted as $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$, where n is the number of blocks. Meanwhile, a mask vector $\mathcal{B}$ can be employed for selecting a subset of blocks, and its length is the same as that of $\mathcal{F}$. By defining the cost function of the DNN on a given dataset $\mathcal{D}$ as $\mathcal{C}(\mathcal{D}; \mathcal{F})$, shallowing DNN can be formulated as the following MOP:

$$\begin{cases} \min_{\mathcal{B}} \ \mathcal{C}(\mathcal{D}; \mathcal{B} \odot \mathcal{F}) \\ \min_{\mathcal{B}} \|\mathcal{B}\|_1 \end{cases} \tag{2}$$

where $\odot$ and $\|\cdot\|_1$ represent element-wise multiplication and ℓ_1 norm, respectively.

Since our focus is not on proposing a brand new EA, it is reasonable to follow the main components of EA. Nevertheless, the technical contribution from the side of evolution is twofold. First, the evaluation step is designed by considering the properties of the DNN shallowing problem, especially the perspective in dual-objective function is new to the basic EA, in which minimizing the number of blocks is considered as the driving force for shallowing DNNs, while the conflicting nature of objectives is utilized to enable Pareto optimization. Second, a prior knowledge incorporation strategy is proposed to initialize the population in the EA, which integrates known information of neural network models into EA. In summary, the overall framework has contributions on the objective function and population initialization from the side of evolution.

B. Prior Knowledge Incorporation

The weight distribution of a well-trained model is utilized as the PK for accelerating the EA, since it is known before the evolutionary optimization. Specifically, assuming there are m parameterized layers in the i-th block, which are denoted as $\{W_1^i, W_2^i, \ldots, W_m^i\}$, and the size of the parameters of a layer in the i-th block is denoted as $(k_j^i, c_j^i, w_j^i, h_j^i)$, where j ranges from 1 to m, and k represents the number of kernels, c means the number of channels in each kernel, w and h denote the width and height of kernels, respectively. Based on the above notations, the PK of the i-th block is calculated as:

$$\mathcal{K}_i = \frac{1}{m} \cdot \sum_{j=1}^{m} \frac{\left\| W_j^i \right\|_1}{d_j^i \cdot c_j^i \cdot w_j^i \cdot h_j^i}. \tag{3}$$

To transform PK into the mask vector representation $\mathcal{B}$ in (2), all PK values are normalized by dividing the maximum value, which transforms them into the range of 0 to 1. Furthermore, a set of Bernoulli random variables parameterized by the normalized PK values is created, from which each decision variable of the candidate solution in the population is independently sampled. This process is depicted as:

$$\mathcal{B}_i \sim \text{Bernoulli}\left(\frac{\mathcal{K}_i}{\max_j \mathcal{K}_j}\right). \tag{4}$$

The PK is a coarse estimation of the importance about each block, since it ignores the correlation between blocks and make assumptions on the weight magnitude without theoretical analysis. However, combined with EAs could considerably mitigate this issue, since the solution will be further optimized. Meanwhile, PK provides an informative way to initialize the population, and is expected to improve the exploration of EA at the early stage of the evolution. Therefore, PK and EA could be beneficial to each other. It is worth noting that this is different from the encoding strategies in EA-based NAS methods [44], [50], since here we focus on determining the initial value based on the weights of a given model, instead of designing a general representation to cover a wide range of network architectures. Also, this strategy has not been investigated in EA-based pruning methods [51], [52], though the pretrained weight has been widely recognized as an informative indicator for measuring the importance of parameters [12], [53].

C. Knowledge Transfer

To reduce the performance gap between the shallowed model and the original model caused by block reduction, KD [18] is adopted for transferring knowledge from the original model to the pruned one. Specifically, a distillation term is introduced into the objective function for retraining, which encourages the pruned model to mimic predictions of the original model and ground truth labels simultaneously. Formally, the objective function with KD is denoted as follows:

$$L(y, \hat{y}) = -\alpha \cdot \sum_i p_i \log(q_i) - (1 - \alpha) \cdot \sum_i y_i \log(\hat{y}_i). \tag{5}$$

Here y_i and $\hat{y}_i$ represent the ground truth label and the predicted probability of the i-th class, respectively. α is introduced as the weighting term. In particular, given the temperature T, p_i and q_i can be computed as:

$$p_i = \frac{\exp(v_i/T)}{\sum_j \exp(v_j/T)} \text{ and } q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)} \tag{6}$$

where v and z are unscaled predictions of the original model and pruned model, respectively.

In this setting, the original model and the pruned model can be regarded as the teacher and the student, respectively. The main benefit of employing this training scheme is that the prediction of the teacher model provides more intraclass information than that of the hard label. However, the learning of the student model could potentially be misled since the teacher model is not guaranteed to be completely correct. Simply ignore the incorrect predictions is not a promising remedy, since they could also contain some noisy hints about the relevant classes. Motivated by this observation,

we propose the CKD. Specifically, a new cost function shown in (7) is designed for the KD learning, in which incorrect predictions are considered but less encouraged to be simulated by the student model, thus the learning can be processed with different sensitivities based on the correctness of the teacher model's output distribution.

$$L(y,\hat{y}) = -\alpha\left(\delta_y^{\hat{y}}\sum_i p_i\log(q_i) + \left(1-\delta_y^{\hat{y}}\right)\lambda_y^{\hat{y}}\sum_i p_i\log(q_i)\right) -(1-\alpha)\sum_i y_i\log(\hat{y}_i). \quad (7)$$

In this equation, $\delta_y^{\hat{y}} = 1$ when the output of the teacher model equals to the ground truth label, otherwise $\delta_y^{\hat{y}} = 0$, and $\lambda_y^{\hat{y}}$ measures the correctness of the teacher model's output distribution.

In summary, the proposed method models the DNN shallowing as a MOP, and an EA-based framework is constructed to solve it. In particular, a PK extraction strategy is proposed, which utilizes the information from the parameters of the blocks to guide the population initialization. In addition, by considering the impacts of incorrect predictions from the teacher model, an improved KD loss is proposed to compensate the performance gap between the original model and the pruned model.

IV. Experimental Studies

In this section, we empirically evaluate the performance of ESNB on shallowing DNNs on four datasets including CIFAR-10, CIFAR-100, ImageNet-16-120 [54], and ImageNet. Several state-of-the-art methods are adopted as competitors to show the effectiveness of the proposed method.

A. Implementation Details

We first evaluate the proposed method on ResNet-110, which is adopted as the baseline model, since it achieves the highest test accuracy as reported in [8]. According to the original design, each residual block contains two convolutional layers. There are 54 blocks, plus a convolutional layer and a fully connected layer in the first and last layer, respectively. Moreover, the 54 blocks are divided into three stages, where the first block in the 2nd and 3rd stages uses strided convolution for down-sampling on the intermediate feature maps. Besides, the ResNet-152 is adopted as another baseline for investigating the performance on processing high-resolution images. It is worth noting that ResNet-152 use bottleneck blocks to mitigate the rapidly increased number of intermediate feature maps, which is different from that in the ResNet-110. Accordingly, there are totally 50 blocks distributed in four stages.

To measure the reduction of computational cost, the number of parameters and the FLOPs reduced from the original model are adopted as evaluation metrics. Intuitively, a model with less parameters means it consumes less storage and computational resource. Likewise, a model with less FLOPs is considered more efficient. Due to the heuristic nature and computational consideration, the experiments are independently repeated for 10 times, the median results are selected for detailed analysis, and the mean and standard deviation of the performance are reported. For the EA, the probability value of flip mutation is set to $p = 0.1$ through grid search on a validation set. Based on the computational cost consideration, the population size N and the maximum number of generation G are empirically set to 30 and 200, respectively. In particular, we adopt a simple implementation to measure the correctness in (7), which is represented as $\lambda_y^{\hat{y}} = \sum_i y_i\hat{y}_i$. The temperature T and α in the KD term in (6) are empirically set to 10 and 0.5, respectively. The overall computational complexity of Algorithm 1 is governed by the nondominated sorting part, which is $\mathcal{O}(GN^2)$ [17], where N refers to the population size. To discover a compact model that is efficient in inference, all the residual blocks are left intact in the initialization step, then less important blocks are gradually identified and removed during the evolution. Particularly, the PK incorporation is applied to half of the solutions in the population, which aims to evenly distribute the searching power to exploitation and exploration at the beginning.

The conflicting nature of the two objectives in ESNB results in a set of Pareto-optimal solutions, in which any solution is not completely worse than any other one. Therefore, decision making is required to come up with an optimal solution. Here we consider the knee point on the Pareto front as a representative solution, where any further improvement in one objective could result in severe degradation in another one [55]. In this case, it can be regarded as a quality trade-off between the residual block reduction and the performance degradation. To automatically select the knee solution, the minimum Manhattan distance (MMD) approach [56] is adopted for locating the knee point, since it exhibits rich geometric interpretations and can globally characterize the knee. Moreover, selecting such a solution is similar to finding a dedicated trade-off model in importance estimation based approaches [12], [57]. In real life, the solution could be chosen by some practical criteria according to the available resource on the target platform. Therefore, ESNB provides a cheap and systematic way to make DNNs shallower. The code is available at https://github.com/ecdnn/ESNB.

B. Experiments on the CIFAR-10

1) Experimental Settings: The CIFAR-10 dataset contains 50 and 10K 32×32 color images in 10 classes for training and testing, respectively. Particularly, we split the original training set into training and validation sets using the ratio of 0.9 to 0.1, and report the accuracy on the original testing set. The data augmentation technique as described in [8] is adopted in the training stage: four pixels are padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. The validation set is used to tune hyper-parameters and evaluating the MOP formulated in (2). The learning rate for stochastic gradient descent (SGD) is initially set to 0.1 and is divided by 10 at 50% and 75% again of the total number of training epochs. To recover the performance of pruned models, retraining is performed with $1/10$ of the initial learning rate under the same learning schedule.

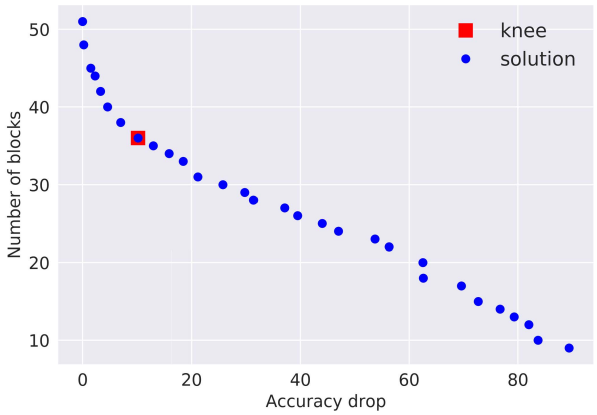

Fig. 5. Pareto-optimal solutions of ESNB on ResNet-110. (The knee solution is highlighted in red square).

2) Performance Evaluation: The Pareto-optimal solutions found on the CIFAR-10 dataset are shown in Fig. 5. As can be seen, the boundary solution on the left top position has no accuracy drop, while the number of blocks is nearly intact. A possible reason is that removing blocks in ResNet-110 is relatively sensitive to the performance, and a small proportion of the solutions in the population are in exploitation throughout the evolution.

On the other hand, the accuracy drops appreciably when there are more blocks get pruned, which provides an intuitive view about the trade-off between the two objectives. Furthermore, the knee solution on the Pareto front, as indicated by the red square in Fig. 5, can be adopted as a promising solution, since it achieves quality trade-off for balancing the two conflicting objectives. Therefore, the knee solution is selected as a representative one on the Pareto front using MMD, and further retraining is employed to recover its performance degradation.

The representative solution is detailed and depicted in Fig. 6. As can be seen, most of the parameters are in the third stage, while the FLOPs are nearly the same in each of the blocks. Interestingly, the distribution of the reduced blocks is relatively even in the three stages of the ResNet-110, which indicates that the intermediate features from low blocks to high blocks are equally important. Also, it can be observed that the redundancy in the 2nd stage is slightly higher than that of other stages.

3) Analysis of Prior Knowledge Incorporation: To validate the effectiveness of PK incorporation in ESNB, we further conduct experiments with different initialization strategies by either fixing all the blocks to be intact or incorporating random noise sampled from uniform distribution. The populations obtained with the same number of generations under different strategies are shown in Fig. 7. As can be seen, keeping the decision variables fixed in the initialization leads to slower speed in convergence than those of the random noise and PK-incorporated strategies. A possible reason is that fixing all solutions in the population to be intact at the beginning could severely hamper the exploration capability of ESNB. Moreover, incorporating PK can achieve better converged population than that of random noise, which indicates that the PK incorporation is an effective way to accelerate the convergence of ESNB. Since the true Pareto front is unknown

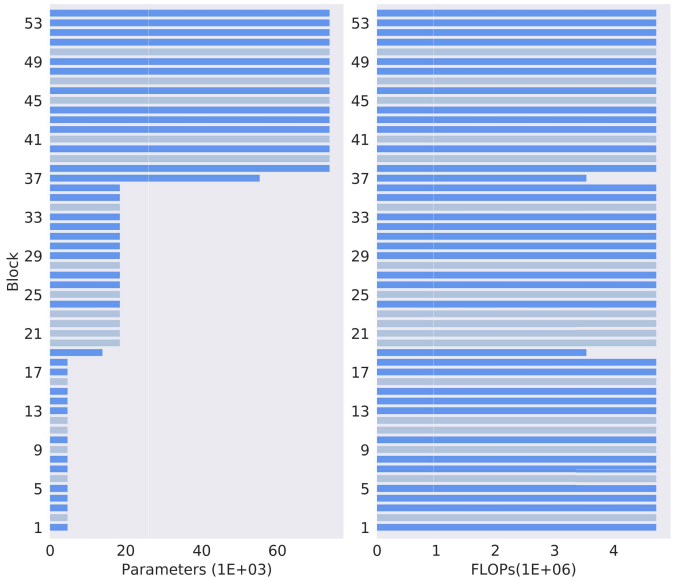

Fig. 6. Searched solution for shallowing ResNet-110. The bars with gray color represent the blocks considered less important.

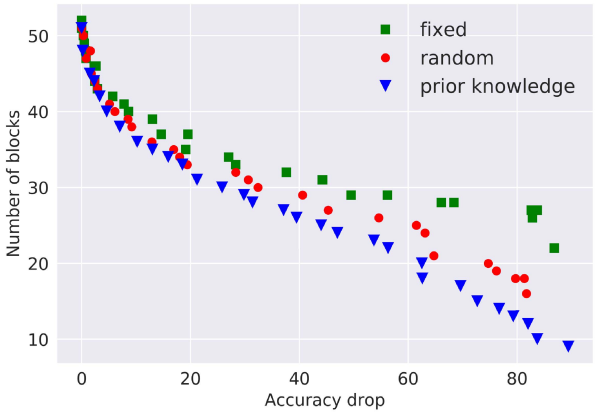

Fig. 7. Evolved populations under different initialization strategies (i.e., green square, red circle, and blue triangle represent fixed, random (uniform), and PK initialization strategies, respectively) obtained from shallowing ResNet-110.

for this problem, we use hypervolume [58] as a metric to measure the performance of these initialization strategies. Since both the number of blocks and accuracy drop have a definite upper bound, objective values of the solutions can be conveniently normalized into the range from 0 to 1. Therefore, (1,1) is adopted as the reference point, the mean and standard deviation of the hypervolume are reported in Table I, which is calculated based on the results from 10 independent runs. It shows that incorporating PK can generally achieve a hypervolume higher than those of fixed and random ones, which indicates it brings benefits in finding a better approximated Pareto front.

To better understand the PK incorporation, we visualize its values computed by (3). The PK values are normalized by dividing the maximum, which makes them interpretable as probabilities and could further be adopted for the population initialization. The normalized PK values of residual blocks are shown in Fig. 8. It can be clearly observed that the 19th and the 37th blocks have much larger PK values than their

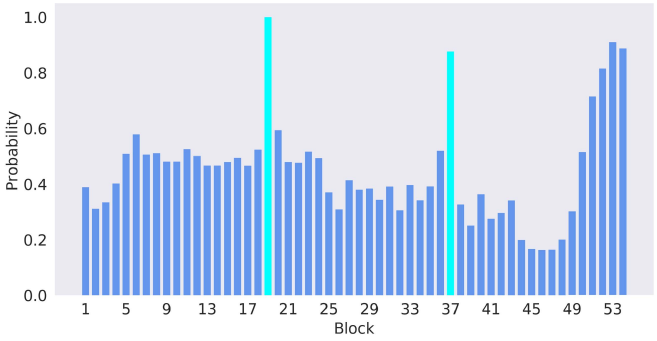

Fig. 8. PK extracted from blocks in the ResNet-110. Blocks with cyan color applies down-sampling on feature maps.

TABLE I
COMPARISON OF DIFFERENT INITIALIZATION STRATEGIES IN ESNB

	fixed	random	prior knowledge
Hypervolume	0.4204±0.0009	0.4971±0.0021	0.5607±0.0013

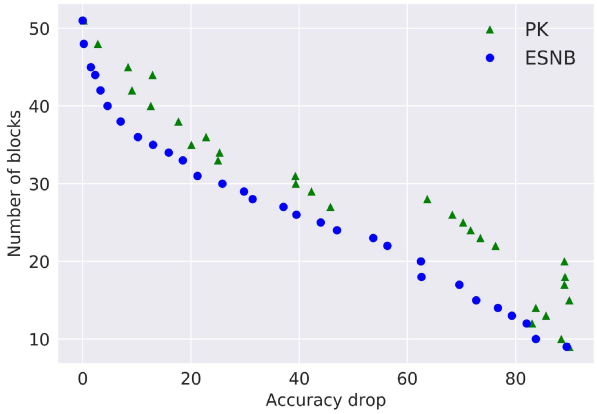

Fig. 9. Comparison between PK (triangles) and ESNB (circles) on shallowing the ResNet-110 before KD training.

adjacent blocks. The main reason is because they contain down-sampling layers, which are of critical importance to the network [13]. Besides, the last four blocks have much higher PK values as well, which is consistent to the result in the representative solution. This observation provides a possible explanation for the accelerated speed of convergence, since the extracted PK is informative.

In fact, PK can be utilized alone as a pruning criterion, where blocks with lower PK are considered less important. To investigate its effectiveness, we adopt the Pareto optimal solutions from ESNB to construct the target network architectures, and prune those blocks that have lower PK values. The obtained results are shown in Fig. 9. Without applying distillation training, it can be seen that simply using PK results in a worse performance than that of ESNB, which verifies that evolution is beneficial to identifying less informative blocks.

Since the solutions shown in Fig. 9 need to be refined by KD, the performance degradation could be affected. Therefore, we further investigate the performance of solutions obtained from PK and ESNB after knowledge distillation training, the results are shown in Fig. 10. As can be seen, the curve shape

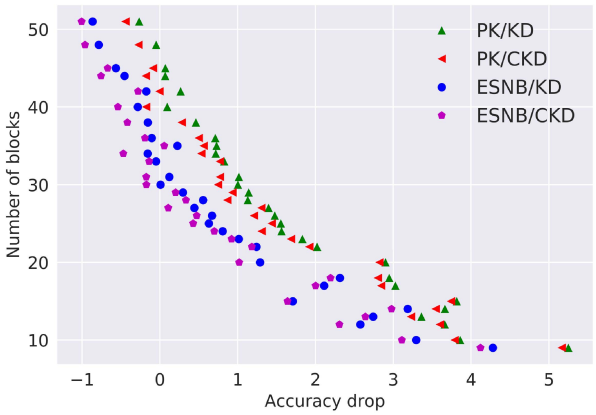

Fig. 10. Comparison between PK (up-pointing triangles and left-pointing triangles) and ESNB (circles, pentagons) on shallowing the ResNet-110 after training with KD and CKD.

generally remains steadily for both PK and ESNB, as well as the performance gap between them. It indicates that KD has no significant impact on the performance gap between these two different pruning strategies. On the other hand, it can be seen that CKD provides better performance than KD for both PK and ESNB, since the accuracy drop with CKD training is lower than that of KD over all the found solutions.

4) Comparison to the State-of-the-Art Pruning Techniques: We further make comparison to the state-of-the-art approaches to validate the effectiveness of the proposed method. First, we investigate the performance of filter-level pruning, which focuses on reducing the number of filters in residual network. We adopt the PFEC [12] as a competitor for block-level pruning, which prunes the first convolutional layer of blocks in the residual network according to the magnitude of the weight parameters. Different from block shallowing, it does not change the number of layers during the pruning, which might be a barrier of pruning DNNs. Then, we further compare the proposed method with competitive DNN shallowing approaches. The random pruning (RP) [13] is also adopted as a baseline for comparison. Another study [57] to be compared utilizes the feature representations of intermediate blocks to measure the importance of each block. Specifically, linear classifiers are required to be trained on features extracted at different layers within the network, and a predefined threshold needs to be manually tuned to find an appropriate trade-off between model performance and computational cost. Thus, we represent it as a feature representation (FR)-based approach.

As shown in Table II, the ResNet-110 is selected as baseline model (RN110), of which the FLOPs and number of parameters are 253 M and 1.72 M, respectively. From Table II, the adopted approaches pruned a moderate number of parameters from the model without resulting in significant performance degradation. In particular, we use PFEC and adopt the same setting as described in [12], which reduces 15.90% of the FLOPs and improves the error rate by 0.07% after retraining with pretrained weights. Meanwhile, retraining the network pruned by PFEC from scratch (PFEC/S) results in a worse performance, which implies the pretrained weight is important

TABLE II
EXPERIMENTAL RESULTS ON THE CIFAR-10 DATASET USING RESNET-110. (RN REPRESENTS RESNET, /S, /KD AND /CKD MEAN SCRATCH TRAINING, KD AND CKD, RESPECTIVELY)

Model	Error	FLOPs	PR[1]	Param.	PR
RN110	6.75	253M	0	1.72M	0
PFEC [12]	6.68	213M	15.90	1.68M	2.28
PFEC/S [12]	6.95				
PFEC/KD	6.65				
PFEC/CKD	6.53				
RP [13]	7.50±0.16	**177 ±5M**	**29.89 ±1.86**	**1.28 ±0.02M**	**25.19 ±1.07**
RP/KD	7.43±0.13				
RP/CKD	7.38±0.14				
FR [57]	6.72±0.02				
FR/KD [57]	6.65±0.03				
FR/CKD	6.63±0.02				
ESNB	6.63±0.02				
ESNB/S	6.68±0.04				
ESNB/KD	6.07±0.03				
ESNB/CKD	**5.60±0.02**				

for preserving the performance. Next, we employ the proposed ESNB to prune the blocks, around 25.19% of the parameters are pruned at block level, which is 11 times more than that of the filter-level pruning (PFEC). It demonstrates that block pruning could be more effective than filter pruning. Also, around 29.89% of the FLOPs are reduced in the block-level case, which means the acceleration of ESNB is nearly 2 times more than that of PFEC. The restriction of the skip connection could be one of the possible reasons, since it requires the second convolutional layer in the residual block to be excluded from the pruning, thus severely limits the number of filters that can be pruned. Nevertheless, block pruning and filter pruning are orthogonal to each other, and can be applied for accelerating deep residual networks together in a cooperative manner.

5) Effect of Knowledge Distillation in the Pruned Networks: Regarding block pruning and KD with different strategies, we adopt the pruned model found by ESNB as the target network architecture, then use RP and FR to prune blocks from the well-trained ResNet-110 model. The number of experiments required for each of the compared block pruning approaches is set to the same as that of ESNB (i.e., 10), and the mean and standard deviation of the results are reported. As shown in Table II, the error rates achieved by RP and FR are worse than those of ESNB when KD or CKD is not applied. Since the subset selection is a combinatorial optimization problem, RP may suffer from the search space overhead. Besides, FR assumes that the discriminative power of a specified block relies on all its previous blocks, and sequentially estimates the importance of each block from the input side to the output side. However, the ensemble view indicates that a subset of the blocks could have the potential of being effective without incurring perceptible performance degradation. In other words, the discriminative power of a block is not necessarily dependent on all its preceding blocks since they could be skipped by the identity connections. Similar to that of filter-level pruning, training from scratch (ESNB/S) without any block importance estimation technique results in inferior performance. To make a fair comparison, we apply KD and CKD to each of the block pruning algorithms. The results show that the error rates of PFEC, RP, FR, and ESNB get improved consistently, and the performance boost of CKD surpasses that of KD for all those methods. Also, ESNB achieves a lower error rate than those of PFEC, RP, and FR, which verifies the effectiveness of the CKD and the superiority of ESNB on shallowing DNNs at block levels.

6) Time Complexity & Computational Resources: We conduct the experiments on a single NVIDIA Tesla P100 GPU. The ESNB with KD training (ESNB/KD) costs around 4 hours, and the CKD case costs nearly the same time since the computational burden for calculating the correctness score is ignorable. On the other hand, FR/KD [57] sequentially trains an auxiliary classifier to determine the importance of each block, which consumes around 6 hours to finish the importance estimation and KD training. RP/KD [13] takes about 2 hours to find the pruned model with transferred knowledge from the original model, while PFEC/KD [12] costs around 2.5 hours since it has more parameters to be trained. As can be seen, ESNB is more efficient than FR but slower than PFEC and RP. This is intuitive since FR requires extra training for the auxiliary classifiers which is computational more expensive. PFEC and RP are more efficient, but they use ad hoc strategies to select blocks, thereby resulting in inferior error rate.

C. Comparisons of Different Pruning algorithms

We further compare ESNB with other pruning algorithms on CIFAR-10, CIFAR-100, ImageNet-16-120, and ImageNet datasets. CIFAR-100 has 50K training images and 10K test images with 100 classes, the training setting is the same as that of CIFAR-10. ImageNet-16-120 is down-sampled (16×16) from the original ImageNet dataset, it contains 151.7K training images, 3K validation images, and 3K test images with 120 classes [59]. We use a training setting similar to that of CIFAR-10, while changing the batch size to 256. ImageNet contains 1.28 million training images and 50K test images with 1000 classes; we randomly extract 10 images for each class from the training set for validation, the batch size is set to 128 and 90 epochs are given for training. The learning rate is set as 0.01, and divided by 10 every 30 epochs. For ImageNet, data augmentation techniques including random resized crop and randomly horizontal flipping are applied. Moreover, we use CKD in ESNB as default setting, and make comparisons to investigate its overall performance.

The results on pruning ResNet-56 on CIFAR-10 dataset is shown in Table III. As can be seen, ESNB achieves an error rate of 6.25 with 59.4M FLOPs, while over 52.6% of the FLOPs are reduced from the original model. Meanwhile, the pruned models obtained from other state-of-the-art approaches including generative adversarial learning (GAL) [60], low-cost collaborative layer (LCCL) [61], and neuron importance score propagation (NISP) [62] have not only higher error rates, but also more FLOPs consumptions. The results of transformable

TABLE III

COMPARISON OF DIFFERENT PRUNING ALGORITHMS FOR RESNET-56 ON CIFAR-10 (- INDICATES NOT AVAILABLE)

Method	Error	Err↑	FLOPs	PR	Param.	PR
GAL [60]	7.02	-0.12	78.3M	37.6	0.75M	11.80
LCCL [61]	7.19	1.54	78.1M	37.9	-	-
NISP [62]	6.99	0.03	81.0M	35.5	0.49M	42.40
TAS [40]	6.31	0.77	59.5M	52.7	-	-
DMC [63]	6.31	-0.07	-	50.0	-	-
HRank [35]	6.83	0.09	62.7M	50.0	0.49M	42.4
SCP [64]	6.77	0.46	-	51.5	-	48.47
ESNB	6.25	0.62	59.4M	52.6	0.37M	56.43

TABLE IV

COMPARISON OF DIFFERENT PRUNING ALGORITHM FOR DENSENET-40 ON CIFAR-100 (- INDICATES NOT AVAILABLE)

Method	Error	Err↑	FLOPs	PR	Param.	PR
VCNN [65]	27.81	2.45	218M	22.67	0.65M	38.68
SCP [64]	26.16	0.40	-	46.25	-	55.22
CondenseNet [66]	25.28	0.08	190M	32.62	0.66M	37.73
ESNB	25.10	-0.17	186M	34.04	0.58M	45.28

architecture search (TAS) [40], discrete model compression (DMC) [63], high rank of feature maps (HRank) [35], and soft channel pruning (SCP) [64] show that they can reduce similar number of FLOPs; however, their error rates are slightly higher at the same time, which verifies that the overall performance of ESNB is very competitive.

To demonstrate the generalization ability of ESNB on other block-wise DNNs, we test its performance via pruning DenseNet-40 [19] on CIFAR-100 dataset. The results are provided in Table IV. It shows that ESNB reduces 34.04% FLOPs from the original model, and achieves an error rate of 25.10%. Specifically, 2, 3, and 5 dense layers are pruned from the first, second, and the third dense blocks, respectively. To maintain the validity of the pruned model, the corresponding channels in the involved dense layers and transition layers are pruned simultaneously. On the other hand, variational convolutional neural network (VCNN) [65] and CondenseNet [66] obtain slightly higher error rates, and the reduced FLOPs are less than that of ESNB. Though SCP [64] adopts a higher pruning ratio, the error rate is increased and higher than that of ESNB. Also, it shows that DenseNet-40 is more compact than that of ResNet56, while ESNB still gains considerable performance improvement when compared with other competitive approaches.

We further investigate the performance of ESNB on ImageNet-16-120, which is a middle-level image classification dataset. As can be seen from Table V, the results indicate that ESNB is capable of reducing over half of the FLOPs while obtaining relative high validation and test accuracies. Comparing to other pruning approaches including PEFC [12], RP [13] and FR [57], ESNB achieves better performance in terms of accuracy and FLOPs reduction, which is similar to the results on CIFAR-10. The results show the potential of ESNB on scaling to datasets with a relatively large number of classes.

To test the effectiveness of ESNB for pruning DNNs on large-scale dataset, we experiment with ResNet-101 on ImageNet dataset. As the results shown in Table VI, the top-1 and top-5 accuracies of ESNB are 77.50% and 93.78%,

TABLE V

COMPARISON OF DIFFERENT PRUNING ALGORITHMS FOR RESNET-110 ON IMAGENET-16-120

Method	validation		test		FLOPs	PR
	acc	acc ↓	acc	acc ↓		
PFEC [12]	44.83	0.49	44.62	0.42	52.61M	0.35
RP [13]	44.78	0.53	44.56	0.28	47.89M	0.33
FR [57]	45.21	0.15	45.27	0.09	40.81M	0.45
ESNB	45.79	0.13	45.83	0.11	30.20M	0.52

TABLE VI

COMPARISON OF DIFFERENT PRUNING ALGORITHMS FOR RESNET-101 ON IMAGENET (- INDICATES NOT AVAILABLE)

Method	Top-1		Top-5		FLOPs	PR
	acc	acc↓	acc	acc↓		
AOFP [67]	76.40	0.23	93.07	0.22	5.29G	30.11
RSNLIA [68]	75.27	1.36	-	-	4.47G	40.95
FPGM [69]	77.32	0.05	93.56	0.00	4.38G	42.20
ESNB	77.50	-0.02	93.78	-0.03	4.30G	43.20

TABLE VII

COMPARISON OF DIFFERENT PRUNING ALGORITHMS FOR INCEPTION-RESNET-V2 ON IMAGENET

Method	Top-1		Top-5		FLOPs	PR
	acc	acc↓	acc	acc↓		
CURL [37]	78.93	1.21	94.11	1.12	7.62G	41.38
HRank [35]	78.24	1.90	93.90	1.33	7.37G	43.31
FR [57]	78.54	1.32	93.67	1.54	7.65G	41.15
ESNB	80.02	0.14	95.05	0.18	6.83G	47.46

respectively. Moreover, 43.20% FLOPs are reduced from the original model. Compared with the results obtained by approximated oracle filter pruning (AOFP) [67], rethinking the smaller-norm-less-informative assumption (RSNLIA) [68] and filter pruning via geometric median (FPGM) [69], the proposed method achieves a better performance in terms of accuracies and FLOPs, which further validates its effectiveness in pruning DNN on large-scale dataset.

Due to the block-wise nature, inception networks [20] can be well handled by ESNB as well. Here we select Inception-Resnet-V2 as an example, and further investigate the performance of different pruning algorithms on ImageNet dataset. The experimental results are shown in Table VII. As can be seen, ESNB considerably reduces the FLOPs by 47.46% while maintaining a relatively higher top-1 accuracy at 80.2%. Meanwhile, CURL [37], HRank [35], and FR [57] show inferior performance in terms of the pruned accuracy and the reduced FLOPs, which empirically verifies that ESNB can generalize well on the Inception-Resnet-V2 network.

The results for ResNet-50 are shown in Table VIII. Compared with soft filter pruning (SFP) [70], Taylor expansion (Taylor) [31], and HRank [35], ESBN provides better parameter and FLOPs reductions, which demonstrates the superiority of evolutionary search with incorporated PK. In particular, ESNB automates the pruning process, and significantly relieves the burden of manual interventions. We further investigate the performance on pruning compact DNNs. In particular, we compress MobileNetV2 [21] on ImageNet dataset. The results are shown in Table IX, in which ESNB obtains an accuracy of 68.40%, which is higher than those of Thin net (ThiNet) [32], DCP [71], and DMC [63]. As clearly evidenced,

TABLE VIII

COMPARISON OF DIFFERENT PRUNING ALGORITHMS FOR RESNET-50 ON IMAGENET (- INDICATES NOT AVAILABLE)

Method	Top-1		Top-5		FLOPs	PR
	acc	acc↓	acc	acc↓		
SFP [70]	74.61	1.54	92.06	0.81	2.38G	41.80
Taylor [31]	74.50	1.68	-	-	2.25G	44.90
HRank [35]	74.98	1.17	92.33	0.54	2.30G	43.76
ESNB	76.13	1.14	93.02	0.59	2.12G	48.06

TABLE IX

COMPARISON OF DIFFERENT PRUNING ALGORITHMS FOR MOBILENETV2 ON IMAGENET (- INDICATES NOT AVAILABLE)

Method	Top-1		Top-5		PR
	acc	acc↓	acc	acc↓	
ThiNet [32]	63.71	6.40	-	4.60	44.70
DCP [71]	64.22	5.89	-	3.77	44.70
DMC [63]	68.37	3.51	90.29	1.83	46.00
ESNB	68.40	3.41	90.31	1.80	47.74

TABLE X

COMPARISON RESULTS OF PRUNING SEARCHED MODELS. ESNB-DARTS MEANS ESNB APPLIED TO DARTS, AND THE SAME MEANING FOR ESNB-OFA

Dataset	Method	Params	FLOPs	Acc (Top-1)
CIFAR-10	DARTS [43]	3.31M	452M	97.24
	NSGA-Net [44]	3.30M	535M	97.25
	ESNB/DARTS	2.85M	383M	97.45
ImageNet	OFA [46]	5.8M	230M	76.9
	BigNAS [48]	4.5M	242M	76.5
	ESNB/OFA	5.7M	208M	77.2

ESNB demonstrates that ENSB also works well on compact networks.

Since NAS approaches have shown promising results in searching for efficient network architectures, it would be interesting to explore the redundancy of those searched models. We first prune the model searched by differentiable architecture search (DARTS) [43]. As the results shown in Table X, the pruned DARTS model (ESNB/DARTS) achieves an error rate of 2.55, and the FLOPs has been reduced to $383M$, which is better than the original DARTS model and surpasses the NSGA-Net [44]. Furthermore, the supernet in OFA [46] is also adopted for pruning. The results in Table X indicate that the proposed method (ESNB/OFA) achieves slightly better performance than that of OFA [46] when reducing a similar number of parameters from the supernet model. Though the BigNAS [48] efficiently finds a model with less parameters, the consumed FLOPs is higher than the proposed method, and the accuracy is indeed lower than those of both OFA and ESNB/OFA. As can be seen, ESNB is effective in pruning searched models, and thus can be adopted as a complement to NAS approaches.

V. CONCLUSION

DNNs have been widely applied in artificial intelligence based applications. However, the intensive computational cost severely hampers their deployment on resource constrained platforms. Inspired by the skip connections and the ensemble view of block-wise DNNs, an EA-based method named ESNB is proposed for shallowing DNNs at block levels. ESNB employs multiobjective optimization paradigm to reduce the number of blocks while maintaining the performance. Particularly, a set of solutions with different trade-off can be discovered in a single run, which provides the flexibility for selecting satisfied solution according to practical configurations. Moreover, a novel PK incorporation strategy is proposed to initialize the population, which improves the exploration ability of the population and accelerates the convergence of the evolution process. In addition, a CKD is designed for transferring knowledge from the original model to the pruned one. A series of experiments are conducted on shallowing block-wise DNNs to demonstrate the effectiveness of ESNB. As indicated by the experimental results, ESNB can considerably reduce the computational cost of DNNs and improve the performance simultaneously. The superiority is further validated by comparing ESNB with other state-of-the-art approaches. Since the search space is bounded by the original DNN model, the ability to discover novel architectures could be very limited. Our future study will focus on designing a unified framework that prunes DNN models at different levels, as well as searching KD schemes. Besides, utilizing dynamic optimization techniques [72] to discover efficient network architectures [73]–[75] under the condition of streaming data is another direction to be investigated.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[2] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[3] Y. Zhou, G. G. Yen, and Z. Yi, "A knee-guided evolutionary algorithm for compressing deep neural networks," *IEEE Trans. Cybern.*, early access, Jul. 30, 2019, doi: 10.1109/TCYB.2019.2928174.
[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.
[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–14.
[6] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[7] Y. Zhou, G. G. Yen, and Z. Yi, "Evolutionary compression of deep neural networks for biomedical image segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2916–2929, Aug. 2020.
[8] 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.
[9] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.
[10] T. Tan, Y. Qian, H. Hu, Y. Zhou, W. Ding, and K. Yu, "Adaptive very deep convolutional residual network for noise robust speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 8, pp. 1393–1405, Aug. 2018.
[11] L. Yu, H. Chen, Q. Dou, J. Qin, and P.-A. Heng, "Automated melanoma recognition in dermoscopy images via very deep residual networks," *IEEE Trans. Med. Imag.*, vol. 36, no. 4, pp. 994–1004, Apr. 2017.
[12] 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.
[13] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 550–558.
[14] C. Qian, Y. Yu, and Z.-H. Zhou, "Pareto ensemble pruning," in *Proc. 29th Conf. Artif. Intell. (AAAI)*, 2015, pp. 2935–2941.

[15] M. Ren, A. Pokrovsky, B. Yang, and R. Urtasun, "SBNet: Sparse blocks network for fast inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8711–8720.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Berlin, Germany: Springer-Verlag, 2016, pp. 630–645.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[18] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*. [Online]. Available: https://arxiv.org/abs/1503.02531

[19] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, vol. 1, no. 2, pp. 2261–2269.

[20] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. 31st Conf. Artif. Intell. (AAAI)*, 2017, pp. 1–7.

[21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[22] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 126–136, Jan. 2018.

[23] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 1269–1277.

[24] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jun. 2015, pp. 1737–1746.

[25] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 4107–4115.

[26] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.

[27] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–13.

[28] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4133–4141.

[29] L. Yuan, F. E. Tay, G. Li, T. Wang, and J. Feng, "Revisiting knowledge distillation via label smoothing regularization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3902–3910.

[30] Y. Liu *et al.*, "Search to distill: Pearls are everywhere but not the eyes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 7536–7545.

[31] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11264–11272.

[32] J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, and W. Lin, "ThiNet: Pruning CNN filters for a thinner net," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2525–2538, Oct. 2019.

[33] L. Liebenwein, C. Baykal, H. Lang, D. Feldman, and D. Rus, "Provable filter pruning for efficient neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–29.

[34] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, "Towards efficient model compression via learned global ranking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1515–1525.

[35] 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.

[36] S. Zhang and B. C. Stadie, "One-shot pruning of recurrent neural networks by jacobian spectrum evaluation," in *Proc. Int. Conf. Learn. Represent., (ICLR)*, 2020, pp. 1–12.

[37] J.-H. Luo and J. Wu, "Neural network pruning with residual-connections and limited-data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1455–1464.

[38] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–42.

[39] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–21.

[40] X. Dong and Y. Yang, "Network pruning via transformable architecture search," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2019, pp. 759–770.

[41] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[42] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–16.

[43] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–13.

[44] Z. Lu *et al.*, "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genet. Evol. Comput. Conf.*, Jul. 2019, pp. 419–427.

[45] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. 35th Int. Conf. Mach. Learn., (ICML)*, 2018, vol. 80, pp. 4092–4101.

[46] H. Cai, C. Gan, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–15.

[47] A. Howard *et al.*, "Searching for mobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.

[48] J. Yu *et al.*, "BigNAS: Scaling up neural architecture search with big single-stage models," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 702–717.

[49] R. Poli and W. B. Langdon, "Schema theory for genetic programming with one-point crossover and point mutation," *Evol. Comput.*, vol. 6, no. 3, pp. 231–252, Sep. 1998.

[50] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. 33rd Conf. Artif. Intell. (AAAI)*, 2018, pp. 4780–4789.

[51] Y. Wang, C. Xu, J. Qiu, C. Xu, and D. Tao, "Towards evolutionary compression," in *Proc. Int. Conf. Knowl. Discovery Data Mining, (KDD)*, Jul. 2018, pp. 2476–2485.

[52] H. Shu *et al.*, "Co-evolutionary compression for unpaired image translation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3234–3243.

[53] 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.

[54] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of ImageNet as an alternative to the CIFAR datasets," 2017, *arXiv:1707.08819*. [Online]. Available: https://arxiv.org/abs/1707.08819

[55] K. Zhang, G. G. Yen, and Z. He, "Evolutionary algorithm for knee-based multiple criteria decision making," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 722–735, Feb. 2021, doi: 10.1109/TCYB.2019.2955573.

[56] W.-Y. Chiu, G. G. Yen, and T.-K. Juan, "Minimum manhattan distance approach to multiple criteria decision making in multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 972–985, Dec. 2016.

[57] S. Chen and Q. Zhao, "Shallowing deep networks: Layer-wise pruning based on feature representations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 12, pp. 3048–3056, Dec. 2019.

[58] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 29–38, Feb. 2006.

[59] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-Bench: Benchmarking NAS algorithms for architecture topology and size," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Jan. 26, 2021, doi: 10.1109/TPAMI.2021.3054824.

[60] S. Lin *et al.*, "Towards optimal structured CNN pruning via generative adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2790–2799.

[61] X. Dong, J. Huang, Y. Yang, and S. Yan, "More is less: A more complicated network with less inference complexity," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5840–5848.

[62] R. Yu *et al.*, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9194–9203.

[63] S. Gao, F. Huang, J. Pei, and H. Huang, "Discrete model compression with resource constraint for deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1896–1905.
[64] M. Kang and B. Han, "Operation-aware soft channel pruning using differentiable masks," in *Proc. 37th Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 5122–5131.
[65] 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.
[66] G. Huang, S. Liu, L. V. D. Maaten, and K. Q. Weinberger, "CondenseNet: An efficient DenseNet using learned group convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2752–2761.
[67] X. Ding, G. Ding, Y. Guo, J. Han, and C. Yan, "Approximated oracle filter pruning for destructive CNN width optimization," in *Proc. 36th Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 1607–1616.
[68] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–11.
[69] 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. 4340–4349.
[70] 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.*, Jul. 2018, pp. 2234–2240.
[71] Z. Zhuang *et al.*, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 883–894.
[72] M. Jiang, Z. Huang, L. Qiu, W. Huang, and G. G. Yen, "Transfer learning-based dynamic multiobjective optimization algorithms," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 501–514, Aug. 2018.
[73] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 350–364, Apr. 2020.
[74] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 394–407, Apr. 2020.
[75] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1242–1254, Apr. 2020.

Yao Zhou (Member, IEEE) received the Ph.D. degree in computer science from Sichuan University, Chengdu, China, in 2019.

From 2017 to 2019, he was a Joint Ph.D. Student financed by the China Scholarship Council in the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA. He is currently a Post-Doctoral Research Fellow with the Machine Intelligence Laboratory, College of Computer Science, Sichuan University. His current research interests include deep neural networks, evolutionary computation, and their applications.

Gary G. Yen (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN, USA, in 1992.

He was with the Structure Control Division, U.S. Air Force Research Laboratory, Albuquerque, NM, USA. He joined Oklahoma State University (OSU), Stillwater, OK, USA, in 1997, where he is currently a Regents Professor with the School of Electrical and Computer Engineering. His current research interests include intelligent control, computational intelligence, conditional health monitoring, signal processing, and their industrial/defense applications.

Dr. Yen received the Andrew P. Sage Best Transactions Paper Award from the IEEE Systems, Man and Cybernetics Society in 2011 and the Meritorious Service Award from the IEEE Computational Intelligence Society in 2014. He has served as the General Chair for the 2003 IEEE International Symposium on Intelligent Control held in Houston, TX, USA, and the 2006 IEEE World Congress on Computational Intelligence held in Vancouver, BC, Canada. He has also served as the Vice President for the Technical Activities from 2005 to 2006 and then the President from 2010 to 2011 of the IEEE Computational intelligence Society. He was an Associate Editor of the *IEEE Control Systems Magazine*, the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY, *Automatica*, *Mechantronics*, the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS—PART B: CYBERNETICS, and the IEEE TRANSACTIONS ON NEURAL NETWORKS. He was the Founding Editor-in-Chief of the *IEEE Computational Intelligence Magazine* from 2006 to 2009. He is also serving as an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS.

Zhang Yi (Fellow, IEEE) received the Ph.D. degree in mathematics from the Institute of Mathematics, Chinese Academy of Science, Beijing, China, in 1994.

He is currently a Professor with the Machine Intelligence Laboratory, College of Computer Science, Sichuan University, Chengdu, China. He has coauthored three books entitled *Convergence Analysis of Recurrent Neural Networks* (Kluwer Academic Publishers, 2004), *Neural Networks: Computational Models and Applications* (Springer, 2007), and *Subspace Learning of Neural Networks* (CRC Press, 2010). His current research interests include neural networks and big data.

Dr. Yi was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS from 2009 to 2012. He is also an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS in 2014.