Two-Stage Evolutionary Neural Architecture Search for Transfer Learning

Yu-Wei Wen, *Graduate Student Member, IEEE*, Sheng-Hsuan Peng, and Chuan-Kang Ting, *Senior Member, IEEE*

***Abstract*—Convolutional neural networks (CNNs) have achieved state-of-the-art performance in many image classification tasks. However, training a deep CNN requires a massive amount of training data, which can be expensive or unobtainable in practical applications, such as defect inspection and medical diagnosis. Transfer learning has been developed to address this issue by transferring knowledge learned from source domains to target domains. A common approach is fine-tuning, which adapts the parameters of a trained neural network for the new target task. Nevertheless, the network architecture remains designed for the source task rather than the target task. To optimize the network architecture in transfer learning, we propose a two-stage evolutionary neural architecture search for transfer learning (EvoNAS-TL), which searches for an efficient subnetwork of the source model for the target task. EvoNAS-TL features two search stages: 1) structure search and 2) local enhancement. The former conducts a coarse-grained global search for suitable neural architectures, while the latter acts as a fine-grained local search to refine the models obtained. In this study, neural architecture search (NAS) is formulated as a multiobjective optimization problem that concurrently minimizes the prediction error and model size. The knee-guided multiobjective evolutionary algorithm, a modern multiobjective optimization approach, is employed to solve the NAS problem. In this study, several experiments are conducted to examine the effectiveness of EvoNAS-TL. The results show that applying EvoNAS-TL on VGG-16 can reduce the model size by 52%–85% and simultaneously improve the testing accuracy by 0.7%–6.9% in transferring from ImageNet to CIFAR-10 and NEU surface detection datasets. In addition, EvoNAS-TL performs comparably to or better than state-of-the-art methods on the CIFAR-10, NEU, and Office-31 datasets.**

***Index Terms*—Convolutional neural network (CNN), multiobjective evolutionary algorithm, neural architecture search (NAS), transfer learning.**

Manuscript received October 20, 2020; revised March 7, 2021 and May 18, 2021; accepted July 6, 2021. Date of publication July 16, 2021; date of current version October 1, 2021. This work was supported by the Ministry of Science and Technology of Taiwan, under Grant MOST 110-2634-F-007-026 and Grant MOST 110-2221-E-007-082-MY3. *(Corresponding author: Chuan-Kang Ting.)*

Yu-Wei Wen is with the Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 62102, Taiwan.

Sheng-Hsuan Peng and Chuan-Kang Ting are with the Department of Power Mechanical Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan (e-mail: ckting@pme.nthu.edu.tw).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TEVC.2021.3097937.

Digital Object Identifier 10.1109/TEVC.2021.3097937

I. Introduction

CONVOLUTIONAL neural networks (CNNs) have been widely applied to various image classification tasks. Deeper or more complex CNNs can usually achieve higher accuracy [1]. However, training a deep CNN ordinarily requires a substantial amount of training data, which could be unavailable in some applications due to the expensive cost or other potential difficulties in data collection. Transfer learning renders an effective way to address this issue by applying the knowledge learned from source tasks to target tasks [2], [3]. This machine learning paradigm has improved the prediction accuracy in low-resource image classification tasks [4]. Transfer learning techniques involve fine-tuning, a loss function, and feature mapping [2]–[4]. The network architecture plays a crucial role in the performance of neural networks; however, it receives less attention in transfer learning. Transfer learning methods typically use the same network architecture for both the source task and the target task, which may cause overparametrization when the target task has only a small amount of data. Moreover, the network architecture designed originally for the source task is not optimized and is likely unsuitable for the target task [5]–[14]. The transferred networks, consequently, can suffer from overparametrization and consume much more computational cost and storage than necessary.

Neural architecture search (NAS) aims to optimize the network architecture for machine learning tasks. By repeatedly generating and testing various architectures, NAS algorithms search for architectures with better performance than those handcrafted by humans. Nevertheless, NAS can be time consuming due to its repeated evaluation of generated network architectures. In addition, it is commonly assumed that the amount of data is sufficient to train the network, but this amount is hardly attainable in many real-world applications. Some studies have attempted to combine NAS with transfer learning. These methods transfer the architectures discovered in the source task to the target task [15]–[18] or transfer the NAS controllers or architecture performance predictors to facilitate NAS in new tasks [19]–[22]. Although reusing trained parameters has been shown to benefit learning a new task [2], [4], [5], [12], only a few studies transfer the parameters trained in the source model to the target task [23].

This study presents an evolutionary system, two-stage evolutionary NAS for transfer learning (EvoNAS-TL), to address the above issues of transfer learning. The proposed

EvoNAS-TL enables NAS in transfer learning to adapt the network architecture and parameters jointly for the target task. Restated, EvoNAS-TL transfers parameters from the source model and searches for the optimal neural architecture for the target task. The transfer of parameters, in addition to the architecture, exploits the parameter-related knowledge learned from the source model while searching architectures for the target task. Based on this novel integration, EvoNAS-TL can enhance the effectiveness of the transfer and generate size-efficient neural architectures for the target task. To reduce the high computational cost of the extensive network architecture search, EvoNAS-TL confines the NAS search space to subnetworks of the source model and focuses on seeking the subnetwork that would achieve the best performance on the target task. EvoNAS-TL includes two stages of the subnetwork search: 1) structure search (SS) and 2) local enhancement (LE). The former aims to manipulate the source model layerwise, whereas the latter proceeds to remove unnecessary operation units from the transferred network. Both optimization stages are formulated as a multiobjective optimization problem (MOP), of which the objectives are the minimization of the prediction error and model size. The knee-guided multiobjective evolutionary algorithm (KGEA) [24] is adopted to solve the MOP in both stages. Moreover, we propose two representations for the SS: 1) sequential representation and 2) selective representation. The two representations allow KGEA to reshape the source model in different levels of detail. The proposed EvoNAS-TL is tested on three transfer learning scenarios, where the first two use ImageNet [25] as the source task and adopt the CIFAR-10 dataset [26] and the NEU surface defect database [27] as the target tasks, whereas the third scenario transfers across domains on the Office-31 dataset [28].

The major contributions of this article are summarized as follows.

1) A two-stage multiobjective evolutionary NAS system is developed to optimize the model structure for the target task in transfer learning.
2) The first stage (SS) serves as a coarse-grained global search, seeking suitable neural architectures for the target tasks and transferring knowledge from the source model to the candidate models. Sequential representation and selective representation are proposed to encode different levels of structural information.
3) The second stage (LE) acts as a fine-grained local search, which refines the model obtained from the SS through evolutionary network pruning.
4) An empirical study is conducted to examine the performance of EvoNAS-TL. The experimental results show that EvoNAS-TL effectively improves the classification error and reduces the model size in transfer learning.

The remainder of this article is organized as follows. Section II reviews the related studies on transfer learning and NAS. Section III elaborates on the proposed EvoNAS-TL system. Section IV presents the experimental settings and results. Finally, concluding remarks are given in Section V.

II. Related Work

The architecture of CNNs has become increasingly complex to improve classification performance and deal with complex image classification tasks [29]. For example, AlexNet [30] uses an eight-layer architecture with over 60 million parameters, VGG-16 [31] has a 16-layer architecture with 138 million parameters, and ResNet [1] reaches more than 100 layers. Transfer learning has shown to be capable of addressing the previously mentioned issues with low training resources by leveraging the prelearned knowledge from one task to another task [2]–[5], [32], as evidenced by practical applications, such as defect inspection [12] and medical diagnosis [8]. Some existing transfer learning methods focus on adapting the source model to the target task through parameter tuning. Although post-transfer pruning of the neural architecture has been proposed [33], [34], the transfer and pruning processes in these studies are performed separately without considering their interaction. Thus, the challenge of finding a suitable neural architecture for transfer learning tasks remains.

NAS deals with the automatic design of network architectures that are suitable for a given machine learning task. Modern NAS systems commonly use evolutionary algorithms [35]–[37] and reinforcement learning [19], [38] to search network structures. However, the generate-and-test NAS systems usually require enormous computational resources for the search process, as it involves repeatedly training candidate models from scratch [17], [38], [39]. For example, Zoph and Le [38] used 800 GPUs over 21–28 days for a single task.

Some approaches have been developed to reduce the computational cost of NAS systems by restricting the search space [16], [17], [40], [41] and parameter sharing [16], [41], [42]. Zoph *et al.* [17] proposed searching only computational cells rather than the entire network because the intracell structure forms a smaller search space. After optimizing the cells, human experts need to arrange and stack the cells to obtain a near-optimal network structure. Similar to this idea, Xie and Yuille [43] attempted to apply the cell structure learned from CIFAR-10 to ImageNet. The cells are stacked in a more complicated way, but the resultant model achieves no improvement in classification accuracy over the networks designed by human experts. For parameter sharing, NAS is formulated to search the optimal subgraph of a massive model represented by a directed acyclic graph. Bayesian learning [44] and gradient-based learning [16], [17], [39], [41], [42] have been used in subgraph search. Sun *et al.* [45] proposed a fully automated evolutionary NAS system, where NAS is formulated as a combinatorial problem based on the existing convolutional blocks, for example, residual blocks and dense blocks. This approach is effective in improving search efficiency. Nonetheless, these types of methods still require massive amounts of data to train the large prototype model, which can be unattainable in real-world applications.

Because of the respective issues of transfer learning and NAS, certain methods have been proposed to integrate them, seeking solutions to enable both the exploitation of architectural information in architectural optimization for the target task and the reuse of learned knowledge from the source task to the target task. Baker *et al.* [15] proposed transferring the architecture discovered on a source task along with pretrained weights to the target task with parameter fine-tuning. This is one of the methods that perform NAS and transfer learning separately. Wong *et al.* [19] developed a reinforcement learning-based NAS system in which the searching controller is pretrained on a set of tasks and reused for new tasks. Wistuba and Pedapati [22] presented inductive transfer, which maintains an architecture pool and selects a promising architecture from the pool for a new task. Other studies [20], [21] suggested transferring architecture search experience by reusing the surrogate model trained as a performance predictor for the candidate architecture. However, these techniques center on the transfer of search experience with no prelearned knowledge involved.

Network pruning aims to remove redundant parameters from a trained model and further prevents overfitting caused by overparametrization [46]. Pruning mechanisms can be roughly categorized into nonstructural pruning and structural pruning [47]. Nonstructural pruning considers the removal of parameters individually, whereas structural pruning also takes the structural relationship of parameters into account. The studies on nonstructural pruning face a significant challenge in that the pruned networks are relatively slow on GPUs. Although specialized hardware can alleviate the problem [48], implementing the hardware can be expensive and challenging to general users. In contrast, structural pruning aims to remove computational units, such as filters and neurons. The models obtained from structural pruning preserve the dimensional coherence of tensor operations and, thus, are supported by off-the-shelf deep learning frameworks. Identifying the filters and neurons to be removed remains a major challenge in pruning CNNs. Greedy algorithms [49]–[51], reinforcement learning [52], and group sparsity measurement [53] have been employed to solve network pruning tasks. These approaches prune the network step-by-step and tend not to recover pruned units in later phases. Some studies use evolutionary algorithms as a global optimizer for pruning [24], [54]. These structural pruning methods downsize the width of the networks, but the network depth remains unchanged. Considering that different tasks may need different model depths, the present study seeks to simultaneously optimize the network structure both widthwise and depthwise for transfer learning tasks by combining NAS and structural network pruning.

III. Evolutionary Neural Architecture Search for Transfer Learning

The proposed EvoNAS-TL is a two-stage multiobjective optimization system searching for the subnetworks of a given source model that are also effective after being transferred to the target domain. The first stage of EvoNAS-TL performs a SS on the source model, and the second stage conducts LE to refine the models through evolutionary network pruning. The MOP for both stages is formulated as follows:

$$\begin{aligned} &\underset{M\in\Omega}{\operatorname{argmin}} \quad \boldsymbol{f}(M) \\ &\boldsymbol{f}(M) = \{f_1(M), f_2(M)\}. \end{aligned} \tag{1}$$

The first objective is to minimize the classification error on the target domain

$$f_1(M) = \min\ \mathrm{E}\big(M,\ \mathcal{D}_{\text{target}}\big) \tag{2}$$

where $\mathrm{E}(\cdot)$ measures the validation error of model M on the dataset of target domain $\mathcal{D}_{\text{target}}$.

The second objective is to minimize the model size. In this study, the model size is determined by the number of parameters in the model. The second objective is formulated by

$$f_2(M) = \min\ |M| \tag{3}$$

where the cardinality $|\cdot|$ stands for the total number of parameters in model M.

EvoNAS-TL undertakes the above biobjective optimization in both stages. Fig. 1 illustrates the workflow of EvoNAS-TL. First, the SS stage explores the source model subnetworks for the structures with the optimal accuracy and model size. Second, the LE stage applies evolutionary network pruning to refine the subnetworks obtained from SS. More details about the two stages are described in the following sections.

A. Structure Search

SS is designed to seek the subnetworks of the source model while considering minimal prediction error and minimal model size. More specifically, SS repeatedly generates subnetworks originated from the source model and then reshapes the subnetworks according to their performance on the target domain. This process transfers and adapts the source model to the target domain, but its computational cost is generally high because of the considerable number of evaluations. To address this issue, we propose two types of representation for candidate solutions of the biobjective optimization problem: 1) sequential representation and 2) selective representation. Each representation encodes the layer information that confines the search space to promote efficiency.

Given a source model based on a CNN

$$M_{\text{source}} = \left\{\boldsymbol{L}_1^{\text{conv}}, \ldots, \boldsymbol{L}_p^{\text{conv}}, \boldsymbol{L}_1^{\text{fc}}, \ldots, \boldsymbol{L}_q^{\text{fc}}\right\}$$

$\boldsymbol{L}_i^{\text{conv}}$ is the ith convolutional layer, $\boldsymbol{L}_j^{\text{fc}}$ is the jth fully connected layer, and p and q indicate the numbers of convolutional layers and fully connected layers, respectively.

1) Sequential Representation: Sequential representation encodes the depth of the convolutional layers and the number of neurons of each fully connected layer. This representation considers the connection between adjacent layers of a trained model and allows for controlling the complexity of the model without breaking the inferencing coherence in convolutional layers. According to the sequential representation, a candidate solution is encoded as

$$\boldsymbol{x} = \left(x^{\text{conv}}, x_1^{\text{fc}}, \ldots, x_q^{\text{fc}}\right) \tag{4}$$

where $x^{\text{conv}} \in \{1, \ldots, p\}$ indicates the number of convolutional layers and $x_i^{\text{fc}} \in [0, 1]$ represents the proportion of neurons to be used in the ith fully connected layer. The candidate model corresponding to a candidate solution $\boldsymbol{x}$ can be written by

$$M_{\text{cand}} = \left\{\boldsymbol{L}_1^{\text{conv}}, \ldots, \boldsymbol{L}_{x^{\text{conv}}}^{\text{conv}}, \hat{\boldsymbol{L}}_1^{\text{fc}}, \ldots, \hat{\boldsymbol{L}}_q^{\text{fc}}\right\}. \tag{5}$$

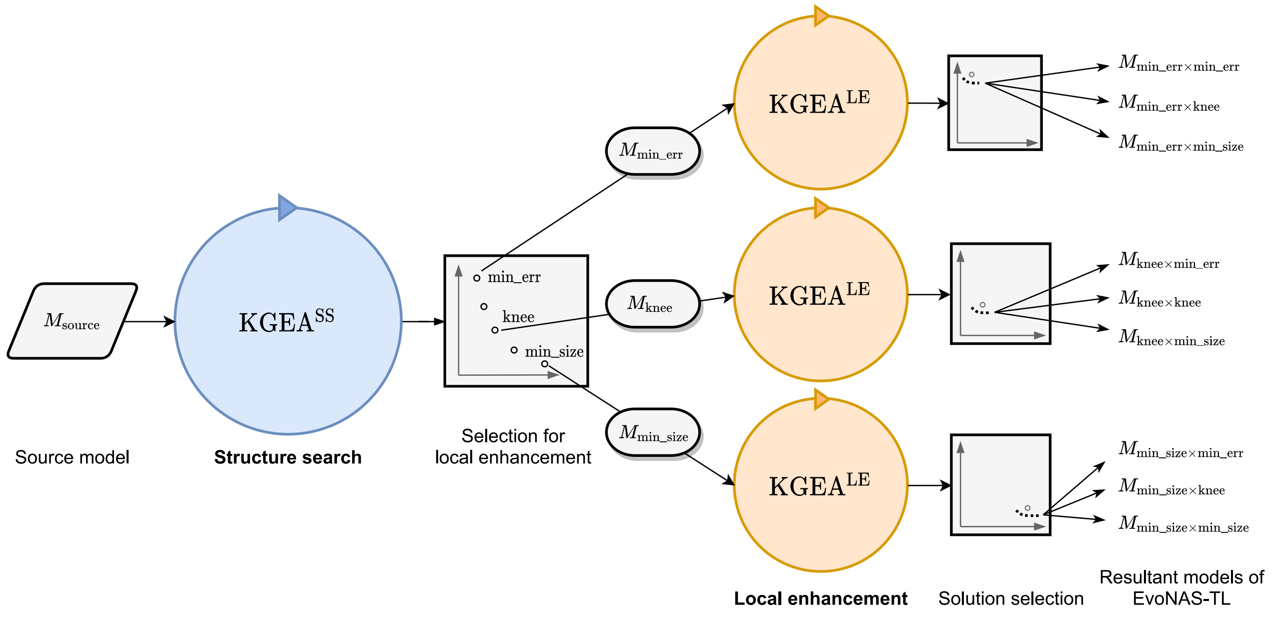
M_{source}
KGEA$^{\text{SS}}$
min_err
knee
min_size
$M_{\text{min\_err}}$
M_{knee}
$M_{\text{min\_size}}$
KGEA$^{\text{LE}}$
KGEA$^{\text{LE}}$
KGEA$^{\text{LE}}$
$M_{\text{min\_err}\times\text{min\_err}}$
$M_{\text{min\_err}\times\text{knee}}$
$M_{\text{min\_err}\times\text{min\_size}}$
$M_{\text{knee}\times\text{min\_err}}$
$M_{\text{knee}\times\text{knee}}$
$M_{\text{knee}\times\text{min\_size}}$
$M_{\text{min\_size}\times\text{min\_err}}$
$M_{\text{min\_size}\times\text{knee}}$
$M_{\text{min\_size}\times\text{min\_size}}$
Source model
Structure search
Selection for local enhancement
Local enhancement
Solution selection
Resultant models of EvoNAS-TL

Fig. 1. Flowchart of EvoNAS-TL.

The transfer process directly copies $\boldsymbol{L}_1^{\text{conv}}, \ldots, \boldsymbol{L}_{x^{\text{conv}}}^{\text{conv}}$ from the first x^{conv} convolutional layers of source model M_{source} and uses $\text{N}(\hat{\boldsymbol{L}}_1^{\text{fc}}), \ldots, \text{N}(\hat{\boldsymbol{L}}_q^{\text{fc}})$ neurons for the q fully connected layers, where $\text{N}(\cdot)$ gives the number of neurons in the given fully connected layer. The number $\text{N}(\hat{\boldsymbol{L}}_i^{\text{fc}})$ is computed by

$$\text{N}\left(\hat{\boldsymbol{L}}_i^{\text{fc}}\right) = \left\lfloor x_i^{\text{fc}} \times \text{N}\left(\boldsymbol{L}_i^{\text{fc}}\right)\right\rceil \tag{6}$$

where $\lfloor\cdot\rceil$ rounds its argument to the nearest integer. The fully connected layers in M_{cand} are reconstructed in accordance with $\text{N}(\hat{\boldsymbol{L}}_1^{\text{fc}}), \ldots, \text{N}(\hat{\boldsymbol{L}}_q^{\text{fc}})$, in which the parameters are initialized randomly. Then, M_{cand} is fine-tuned with the training data of the target domain $\mathcal{D}_{\text{target\_train}}$; that is, the convolutional and fully connected layers are tuned. The validation error and model size of M_{cand} then serve as the objective values of $\boldsymbol{x}$. Fig. 2 illustrates the association of the sequential representation with a source model VGG-16 [31].

2) Selective Representation: Selective representation enables subtler manipulation of convolutional layers than sequential representation. Using selective representation, a candidate solution is encoded as

$$\boldsymbol{x} = \left(x_1^{\text{conv}}, \ldots, x_p^{\text{conv}}, x_1^{\text{fc}}, \ldots, x_q^{\text{fc}}\right). \tag{7}$$

The selective representation includes two parts: the first part $x_i^{\text{conv}} \in \{0, 1, 2\}$ indicates the operation for the ith convolutional layer, and the second part $x_j^{\text{fc}} \in [0, 1]$ resembles the corresponding term in sequential representation, which accounts for the proportion of neurons used in the jth fully connected layer. The first part considers three operations: discarding, fixing, or fine-tuning a convolutional layer

$$\begin{cases} \text{discard}, & x_i^{\text{conv}} = 0 \\ \text{fix}, & x_i^{\text{conv}} = 1 \\ \text{fine-tune}, & x_i^{\text{conv}} = 2. \end{cases}$$

Fig. 3 illustrates the selective representation associated with a VGG-16 source model.

Algorithm 1 Structure Search

```
Input: pretrained source model $M_{\text{source}}$,
       training data of target domain $\mathcal{D}_{\text{target\_train}}$,
       and validation data of target domain $\mathcal{D}_{\text{target\_valid}}$
Output: $\mathcal{M}_{\text{nondominated}}$
 1: $\mathcal{Q}_0 \leftarrow$ Population-initialization
 2: $\mathcal{M}_0 \leftarrow$ Model-construction($\mathcal{Q}_0$, $M_{\text{source}}$)
 3: $\mathcal{M}_0 \leftarrow$ Fine-tuning($\mathcal{M}_0$, $\mathcal{D}_{\text{target\_train}}$)
 4: Evaluation($\mathcal{M}_0$, $\mathcal{D}_{\text{target\_valid}}$)
 5: $t \leftarrow 0$
 6: while termination criterion is not satisfied do
 7:     $\mathcal{P} \leftarrow$ Parent-selection($\mathcal{Q}_t$)
 8:     $\mathcal{Q}_{t+1} \leftarrow$ Crossover-and-mutation($\mathcal{P}$)
 9:     $\mathcal{M}_{t+1} \leftarrow$ Model-construction($\mathcal{Q}_{t+1}$, $M_{\text{source}}$)
10:     $\mathcal{M}_{t+1} \leftarrow$ Fine-tuning($\mathcal{M}_{t+1}$, $\mathcal{D}_{\text{target\_train}}$)
11:     Evaluation($\mathcal{M}_{t+1}$, $\mathcal{D}_{\text{target\_valid}}$)
12:     $\mathcal{Q}_{t+1} \leftarrow$ Survivor-selection($\mathcal{Q}_t \cup \mathcal{Q}_{t+1}$)
13:     $t \leftarrow t + 1$
14: end while
15: $\mathcal{M}_{\text{nondominated}} \leftarrow$ Approximate-Pareto-front($\mathcal{M}_t$)
```

Algorithm 1 outlines the SS procedure. EvoNAS-TL uses either of the two representations and applies KGEA [24] to solve the biobjective optimization problem in the SS. Regarding the evolutionary process, the population is initialized at uniform random. Candidate solutions, namely, offspring, are generated by crossover and mutation. To evaluate a candidate solution $\boldsymbol{x}$, a model M_{cand} is constructed with reference to source model M_{source} and the representation of $\boldsymbol{x}$; then, M_{cand} is fine-tuned using training data of target domain $\mathcal{D}_{\text{target\_train}}$. The performance of $\boldsymbol{x}$ is assessed by the two objective functions (2) and (3) with the validation data of target domain $\mathcal{D}_{\text{target\_valid}}$. At the end of the SS, KGEA yields an

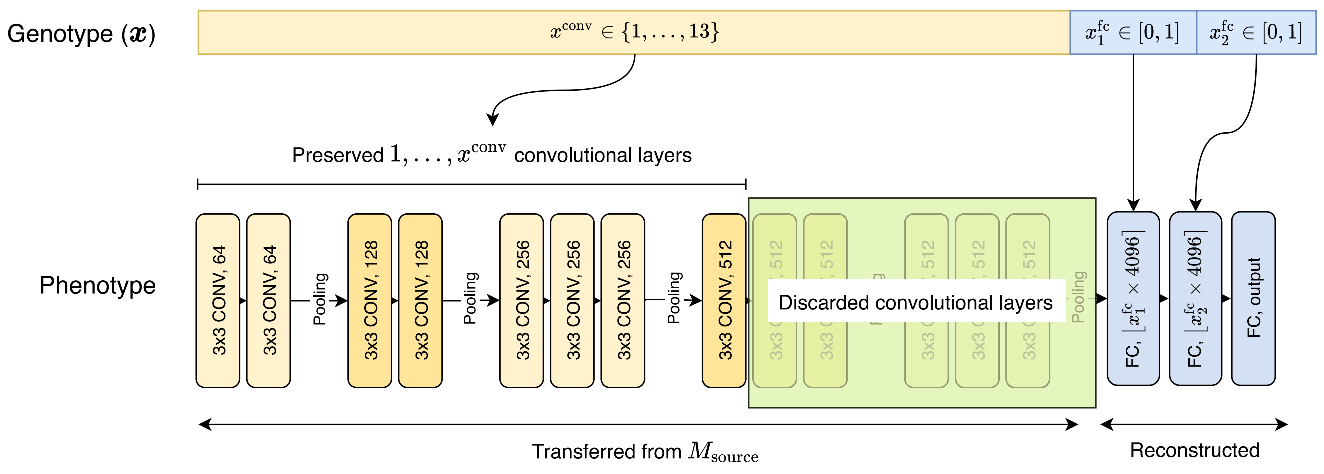
Genotype ($\boldsymbol{x}$)
$x^{\text{conv}} \in \{1, \ldots, 13\}$
$x_1^{\text{fc}} \in [0,1]$
$x_2^{\text{fc}} \in [0,1]$
Preserved $1, \ldots, x^{\text{conv}}$ convolutional layers
Phenotype
3x3 CONV, 64
3x3 CONV, 128
3x3 CONV, 256
3x3 CONV, 512
Pooling
Discarded convolutional layers
FC, $\lfloor x_1^{\text{fc}} \times 4096 \rceil$
FC, $\lfloor x_2^{\text{fc}} \times 4096 \rceil$
FC, output
Transferred from M_{source}
Reconstructed

Fig. 2. Mapping between the sequential representation $\boldsymbol{x}$ and its corresponding model with M_{source} VGG-16.

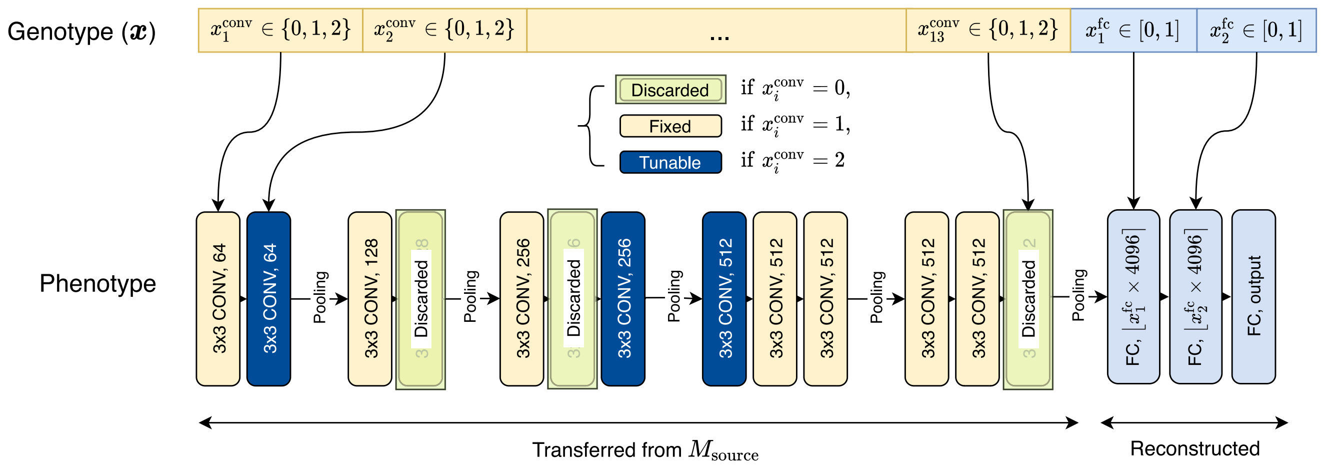
Genotype ($\boldsymbol{x}$)
$x_1^{\text{conv}} \in \{0,1,2\}$
$x_2^{\text{conv}} \in \{0,1,2\}$
...
$x_{13}^{\text{conv}} \in \{0,1,2\}$
$x_1^{\text{fc}} \in [0,1]$
$x_2^{\text{fc}} \in [0,1]$
Discarded if $x_i^{\text{conv}} = 0$,
Fixed if $x_i^{\text{conv}} = 1$,
Tunable if $x_i^{\text{conv}} = 2$
Phenotype
3x3 CONV, 64
3x3 CONV, 128
3x3 CONV, 256
3x3 CONV, 512
Discarded
Pooling
FC, $\lfloor x_1^{\text{fc}} \times 4096 \rceil$
FC, $\lfloor x_2^{\text{fc}} \times 4096 \rceil$
FC, output
Transferred from M_{source}
Reconstructed

Fig. 3. Mapping between the selective representation $\boldsymbol{x}$ and its corresponding model with M_{source} VGG-16.

approximate Pareto front (APF) formed by the resultant models. EvoNAS-TL selects three models from the results: 1) the knee solution; 2) the solution with minimal error; and 3) the solution with minimal model size, and it proceeds with LE on each model in the second stage.

B. Local Enhancement

LE uses evolutionary network pruning [24] on the models obtained from SS to reduce the model size and improve the classification performance. For the latter, several studies (e.g., [55]–[57]) have shown that network pruning can address the overparametrization issue of deep neural networks. By removing less relevant units, network pruning can enhance the network performance and generalization; that is, network pruning aims to refine models by retaining useful units and removing trivial ones.

Given a resultant model from the SS

$$M_{\text{SS}} = \left\{\boldsymbol{L}_1^{\text{conv}}, \ldots, \boldsymbol{L}_k^{\text{conv}}, \boldsymbol{L}_1^{\text{fc}}, \ldots, \boldsymbol{L}_m^{\text{fc}}\right\}$$

with

$$\boldsymbol{L}_i^{\text{conv}} \in \mathbb{R}^{H_i \times W_i \times C_i \times F_i} \quad \text{and} \quad \boldsymbol{L}_i^{\text{fc}} \in \mathbb{R}^{1 \times 1 \times I_i \times N_i}$$

where H_i and W_i are the height and width of the filters, and C_i and F_i are the numbers of channels and filters in $\boldsymbol{L}_i^{\text{conv}}$, respectively. I_i is the number of inputs in $\boldsymbol{L}_i^{\text{fc}}$, and N_i denotes the number of neurons in $\boldsymbol{L}_i^{\text{fc}}$. A candidate solution $\boldsymbol{z}$ in LE is represented by

$$\boldsymbol{z} = \left(\boldsymbol{z}^{\text{conv}}, \boldsymbol{z}^{\text{fc}}\right). \tag{8}$$

The first part $\boldsymbol{z}^{\text{conv}}$ is a binary string

$$\boldsymbol{z}^{\text{conv}} = \left(z_{1,1}^{\text{conv}}, \ldots, z_{k,F_k}^{\text{conv}}\right)$$

where the binary variable $z_{i,f}^{\text{conv}} \in \{0, 1\}$ indicates whether the fth filter in the ith convolutional layer is activated (value 1) or not (value 0). The second part $\boldsymbol{z}^{\text{fc}}$ is a binary string

$$\boldsymbol{z}^{\text{fc}} = \left(z_{1,1}^{\text{fc}}, \ldots, z_{m,N_m}^{\text{fc}}\right)$$

where the binary variable $z_{j,n}^{\text{fc}} \in \{0, 1\}$ indicates whether the nth neuron in the jth fully connected layer is activated or not.

The pruning mask associated with $z_{i,f}^{\text{conv}}$ can be written by

$$\boldsymbol{B}_i^{\text{conv}}(:, :, :, f) = \begin{cases} 0, & \text{if } z_{i,f}^{\text{conv}} = 0 \\ 1, & \text{otherwise} \end{cases} \tag{9}$$

Algorithm 2 Local Enhancement

```
Input: a model from structure search $M_{\text{SS}}$ and
        validation data of target domain $\mathcal{D}_{\text{target\_valid}}$
Output: $\mathcal{M}_{\text{nondominated}}$
1:  $\mathcal{Z}_0 \leftarrow$ Population-initialization
2:  $\mathcal{M}_0 \leftarrow$ Model-masking($\mathcal{Z}_0$, $M_{\text{SS}}$)
3:  Evaluation($\mathcal{M}_0$, $\mathcal{D}_{\text{target\_valid}}$)
4:  $t \leftarrow 0$
5:  while termination criterion is not satisfied do
6:      $\mathcal{P} \leftarrow$ Parent-selection($\mathcal{Z}_t$)
7:      $\mathcal{Z}_{t+1} \leftarrow$ Crossover-and-mutation($\mathcal{P}$)
8:      $\mathcal{M}_{t+1} \leftarrow$ Model-masking($\mathcal{Z}_{t+1}$, $M_{\text{SS}}$)
9:      Evaluation($\mathcal{M}_{t+1}$, $\mathcal{D}_{\text{target\_valid}}$)
10:     $\mathcal{Z}_{t+1} \leftarrow$ Survivor-selection($\mathcal{Z}_t \cup \mathcal{Z}_{t+1}$)
11:     $t \leftarrow t + 1$
12: end while
13: $\mathcal{M}_{\text{nondominated}} \leftarrow$ Approximate-Pareto-front($\mathcal{M}_t$)
```

and the pruning mask associated with $z_{i,n}^{\text{fc}}$ is denoted as

$$\boldsymbol{B}_i^{\text{fc}}(1, 1, :, n) = \begin{cases} 0, & \text{if } z_{i,n}^{\text{fc}} = 0 \\ 1, & \text{otherwise.} \end{cases} \tag{10}$$

Let $\boldsymbol{B} = (\boldsymbol{B}_1^{\text{conv}}, \ldots, \boldsymbol{B}_k^{\text{conv}}, \boldsymbol{B}_1^{\text{fc}}, \ldots, \boldsymbol{B}_m^{\text{fc}})$ denote the weight mask based on z. The resultant model of LE by applying model masking on M_{SS} with weight mask $\boldsymbol{B}$ is

$$M_{\text{LE}} = M_{\text{SS}} \circ \boldsymbol{B}$$

where operator $\circ$ refers to the elementwise product.

Like the SS stage, the LE stage also uses KGEA to search for the optimal models in terms of prediction error and model size. The evolutionary optimization process of LE is similar to that of SS, except LE does not apply the fine-tuning process. To evaluate a candidate solution z, the model masking procedure is used to determine from z whether a filter or neuron is activated or deactivated in model M_{SS}. More specifically, model masking performs elementwise multiplication of the weights of M_{SS} with the weight mask $\boldsymbol{B}$ of candidate solution z. The masked model is then tested on $\mathcal{D}_{\text{target\_valid}}$ for the classification error. The size of a pruned model is simply the number of activated weights in the masked model. Algorithm 2 outlines the pseudocode of LE.

IV. Experimental Studies

In this study, several experiments were conducted to examine the effectiveness and efficiency of EvoNAS-TL compared to the baseline and state-of-the-art methods. In this section, we first introduce the experimental settings, including the source task, source model, target tasks, data processing, and settings for optimization and fine-tuning. Next, we investigate the experimental results on each transfer task and discuss the advantages and limitations of EvoNAS-TL.

A. Experimental Settings

The experiments consisted of three transfer learning scenarios that pose different challenges to transfer learning. The first two scenarios shared the same source task, namely, ImageNet [25], whereas their target tasks were the CIFAR-10 dataset [26] and the NEU surface defect database [27], respectively. The third scenario used the Office-31 dataset [28] as a benchmark. Fig. 4 shows image samples of the three datasets. ImageNet is widely used as a source task in transfer learning because of its comprehensive image contents and labels. A model well trained on ImageNet is deemed to implicitly possess a variety of feature extractors that can be reused in other image recognition tasks through transfer learning. The major challenge of transferring from ImageNet to CIFAR-10 lies in the significant difference in image size, whereas the challenge of transferring from ImageNet to the NEU dataset arises from the nontrivial intertask correlation and low data volume. The Office-31 dataset comprises three domains: Amazon (A), DSLR (D), and Webcam (W). The three domains share the same 31-class image classification task. In light of the considerable successes of VGG-16 [31] in transfer learning [9], [11], [58], VGG-16 pretrained on ImageNet was adopted as the source model in our experiments. Each target dataset was partitioned into three disjoint parts: 1) training; 2) validation; and 3) testing sets. For Office-31, we conducted experiments on all domain combinations, which resulted in six transfer tasks in total. The experimental settings for the baseline methods and EvoNAS-TL followed the settings in [6], [14], and [59]. Common data augmentation techniques, including rotating, shifting, and flipping, were applied to extend the target datasets. For all training and fine-tuning processes in the experiments, we employed stochastic gradient descent as the optimizer with an initial learning rate 0.005 decaying 10^{-6} per batch, momentum 0.9, and batch size 256. Training and fine-tuning were terminated if the validation loss ceased improving for three consecutive epochs.

1) Baseline Models: Three baseline methods were used for performance comparison: 1) training from scratch; 2) transfer learning [5]; and 3) network pruning [24] after transfer learning. Our preliminary tests showed that directly using standard VGG-16 resulted in low classification accuracy due to the overly complex architecture. To improve the baseline results, we modified VGG-16 as mVGG-16 with a global average pooling (GAP) layer [60] between the last convolutional layer and the first fully connected layer. Accordingly, the source model size was reduced from 138 million parameters (standard VGG-16) to 33.6 million parameters (mVGG-16). It is worth noting that in our experiments, this modification preserved pretrained filters in the convolutional layers while the fully connected layers were reconstructed.

The train-from-scratch model was obtained by training mVGG-16 from scratch with the data of the target task. Performance comparison included the common transfer learning approach based on the fine-tuning method [5]. In this approach, a model based on mVGG-16 for the target domain was built by copying the convolutional weights of the source model VGG-16 pretrained on ImageNet, randomly initializing the weights of its fully connected layers, and then fine-tuning the whole network with the training data of the target task. To demonstrate the difference between simultaneous transfer learning and subnetwork search featured in EvoNAS-TL

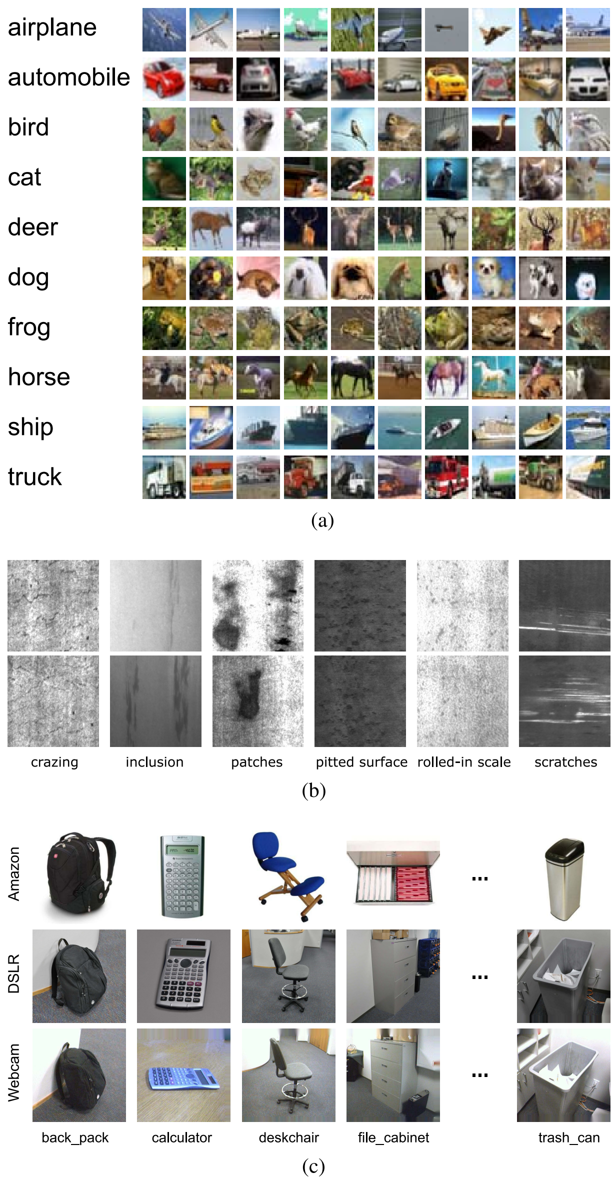

Fig. 4. Demonstration of the (a) CIFAR-10, (b) NEU surface defect, and (c) Office-31 datasets.

and the sequential procedure of transfer learning followed by network pruning, we included another baseline model that applied the state-of-the-art network pruning method [24] to the fine-tuned model. Considering that network training, fine-tuning, evolutionary network pruning, and EvoNAS-TL all involve stochasticity, our empirical study considered the average performance over ten independent runs for the CIFAR-10 and NEU datasets and five independent runs for each task in the Office-31 dataset.

2) EvoNAS-TL: The proposed EvoNAS-TL performed two stages of biobjective optimization for transfer learning. In the first stage, SS of EvoNAS-TL used the modified VGG-16 as the source model, searched for suitable subnetworks for the target domain, and yielded an APF of candidate models. From the APF, the model with the minimal validation error, the model with the minimal number of parameters, and the knee model were used individually in the next stage. In the second stage, EvoNAS-TL local enhancement received a modulated transferred model from SS and applied evolutionary network pruning to further improve the model regarding both objectives. Fig. 1 illustrates that SS produces three models corresponding to the minimal validation error, minimal number of parameters, and knee model. The LE was applied based on these three models and yielded an APF for each. Accordingly, EvoNAS-TL outputs nine models for each transfer learning task. Table I summarizes the parameter settings for the KGEA used in EvoNAS-TL.

B. Results and Discussions

First, we investigated the performance of EvoNAS-TL in transferring from ImageNet to CIFAR-10. The visual components of the two datasets have observable interconnections. The common features between the source and target tasks were expected to facilitate the transfer of knowledge. Note that the image size of the source task (224×224 pixels) significantly differs from that of the target task (32×32 pixels) in this test case, posing a challenge for the transfer. Second, we assessed EvoNAS-TL in transferring from ImageNet to the NEU surface defect database, which consists of 1800 grayscale images containing six types of surface defect patterns. The main challenge of this test case was its low visual correlation between the source task and the target task. Finally, we evaluated EvoNAS-TL in comparison with state-of-the-art methods on Office-31 [28], a popular benchmark for transfer learning and domain adaptation [3].

Tables II–IV compare the performance of EvoNAS-TL with the baseline and state-of-the-art methods on the learning tasks of transferring from ImageNet to CIFAR-10, from ImageNet to the NEU dataset, and from domain to domain in Office-31, respectively. The tables specifically present the performance of EvoNAS-TL with minimal-error selection for the final candidate solutions. For transferring from ImageNet to CIFAR-10, performance comparison includes the state-of-the-art automated machine learning methods that use NAS and hyperparameter tuning, i.e., Hyperband [61], Warmstart [62], and TrAutoML [23]. For transferring from ImageNet to the NEU dataset, we compared EvoNAS-TL with the baseline methods, i.e., training from scratch, transfer through fine-tuning [5], and applying state-of-the-art network pruning [24] after transfer through fine-tuning. For Office-31, the performance of EvoNAS-TL was evaluated by comparing with baseline methods and state-of-the-art supervised domain adaptation methods using VGG-16, including FADA [63], CCSA [13], *d*-SNE [14], and DAGE-LDA [59].

1) Results of Baseline Models: Table II shows that training from scratch resulted in low prediction accuracy when using the modified VGG-16 to deal with CIFAR-10. This poor performance was caused by the attempt to train a deep model for small-sized images. By transferring the filters pretrained on ImageNet, the model obtained showed a substantial improvement in classification accuracy, confirming the effectiveness of the transfer. The results also indicate that evolutionary network pruning downsized the transferred model by 13% in the number of parameters, which slightly compromised the

TABLE I
PARAMETER SETTINGS FOR EVONAS-TL

	Structure search		Local enhancement
	Sequential representation	Selective representation	
Representation	$\boldsymbol{x} \in \{1, ..., 13\} \times [0, 1]^2$	$\boldsymbol{x} \in \{0, 1, 2\}^{13} \times [0, 1]^2$	$\boldsymbol{z} \in \{0, 1\}^l$, $l \leq 12416$
Crossover	Uniform ($\boldsymbol{x}^{\text{conv}}$); whole-arithmetic ($\boldsymbol{x}^{\text{fc}}$)		Uniform
Mutation	Random resetting ($\boldsymbol{x}^{\text{conv}}$); uniform ($\boldsymbol{x}^{\text{fc}}$)		Bit-flip
Population size	30		200
Parent selection	2-tournament		2-tournament
Crossover rate	1.0		1.0
Mutation rate	0.333	0.066	$1 / l$
Survivor selection	$(\mu + \lambda)$		$(\mu + \lambda)$
#generations	30		500

TABLE II
EXPERIMENTAL RESULTS OF TRANSFERRING FROM IMAGENET TO CIFAR-10. BOLDFACE DENOTES THE BETTER OF SS AND SS+LE RESULTS

Method		Accuracy Val (%)	Accuracy Test (%)	#Parameters ($\times 10^6$)	Reduced size (%)	FLOPs ($\times 10^7$)	Time of NAS (GPU days)
Train-from-scratch		52.9	53.2	33.64	-	33.2	-
Fine-tune [5]		80.6	80.3	33.64	-	33.2	-
Fine-tune + prune [24]		81.0	79.9	29.21	13%	30.0	0.67
Hyperband [61]		-	78.5	-	-	-	0.67
Warmstart + hyperband [62]		-	73.5	-	-	-	0.67
TrAutoML (MH) + hyperband [23]		-	77.5	-	-	-	0.67
EvoNAS-TL (sequential)	SS	87.8	**87.2**	16.20	52%	30.0	2.62
	SS + LE	**88.2**	**87.2**	**14.69**	**56%**	**28.1**	3.10
EvoNAS-TL (selective)	SS	87.6	**87.1**	12.60	63%	26.3	2.32
	SS + LE	**88.1**	86.8	**11.21**	**67%**	**24.7**	2.78

testing accuracy. The results of the experiments on the NEU dataset show similar tendencies: training from scratch performed poorly in training the huge and deep model. Reusing the filters trained on ImageNet nevertheless showed the potential to solve low-resource problems. In addition, the size of the transferred model was reduced by 35% through evolutionary network pruning, while the testing accuracy was slightly affected. The results of pruning after fine-tuning suggest the utility of pruning the complex architecture of the source model in CIFAR-10 and NEU, but they tend to overfit in Office-31. The evolutionary network pruning can increase the validation accuracy and reduce the model size, but this advantage is not generalized to test data. This overfitting may have been caused by the scarcity of data in Office-31, which contains only 7–100 images for each category in a domain.

2) Results of EvoNAS-TL: The optimization process in EvoNAS-TL comprises two stages: SS and LE. As previously stated, this work proposes sequential representation and selective representation for SS, thereby considering the high-level structural information of the source model in different ways. The experiments investigated the effects of these two representations with and without LE on the performance of EvoNAS-TL.

As indicated in Tables II and III, EvoNAS-TL achieved significantly better testing accuracy than the three baseline methods in the CIFAR-10 test case and comparable accuracy in the NEU test case. In addition, EvoNAS-TL effectively reduced the number of parameters in the resultant models in a reasonable search time. Both test cases showed that EvoNAS-TL was highly capable of tailoring neural architecture for the target tasks in transfer learning. Regarding the effect of representation, both representations yielded a similar improvement in the classification error. Selective representation, in particular, was more effective in reducing the model size because it enabled more precise manipulation of the neural architecture than sequential representation. As for the improvement exerted by the two stages, the experimental results indicate that SS significantly contributed to both validation accuracy and model size; to be precise, it reduced the number of parameters by 52%–78% and improved the test accuracy by up to 6.9%. LE could further downsize the models by 4%–15%.

On Office-31 in Table IV, EvoNAS-TL using SS with sequential representation achieved comparable or better average classification accuracy and model size than the state-of-the-art methods. Note that the Office-31 dataset was originally developed for domain adaptation, instead of general transfer learning. In addition, the domain adaptation methods used in the experiments assume that the target task is the same as the source task, whereas EvoNAS-TL does not rest on such an advantageous assumption. Even so, EvoNAS-TL performed comparably to or better than the domain adaptation methods. Table IV further reveals that network pruning was vulnerable to overfitting, which may have been caused by overly pursuing classification accuracy despite having only limited data. This negative effect also occurred in LE and caused the deterioration in the average accuracy of EvoNAS-TL.

TABLE III
EXPERIMENTAL RESULTS OF TRANSFERRING FROM IMAGENET TO THE NEU DATASET. BOLDFACE DENOTES THE BETTER OF SS AND SS+LE RESULTS

Method		Accuracy Val (%)	Accuracy Test (%)	#Parameters ($\times 10^6$)	Reduced size (%)	FLOPs ($\times 10^9$)	Time of NAS (GPU days)
Train-from-scratch		31.6	22.4	33.62	-	15.37	-
Fine-tune [5]		98.4	98.9	33.62	-	15.37	-
Fine-tune + prune [24]		99.4	97.7	22.02	35%	10.58	1.01
EvoNAS-TL (sequential)	SS	98.6	**99.6**	12.40	63%	14.38	1.15
	SS + LE	**99.8**	98.9	**7.36**	**78%**	**8.90**	1.84
EvoNAS-TL (selective)	SS	99.2	98.6	7.55	78%	11.33	1.09
	SS + LE	**99.9**	**98.7**	**5.13**	**85%**	**8.01**	1.66

TABLE IV
EXPERIMENTAL RESULTS OF TRANSFERRING BETWEEN DOMAINS IN THE OFFICE-31 DATASET, INCLUDING THE AVERAGE TEST ACCURACY AND AVERAGE MODEL SIZE (MILLIONS OF PARAMETERS). BOLDFACE DENOTES THE BETTER OF SS AND SS+LE RESULTS

Method		Test Accuracy (%) A→D	A→W	D→A	D→W	W→A	W→D	Avg. acc. (%)	Avg. size ($\times 10^6$)
Fine-tune [5]		59.1	92.8	78.3	85.8	75.1	68.1	76.5	33.7
Fine-tune + prune [24]		67.5	82.7	72.3	83.8	65.4	58.5	71.7	23.5
FADA (VGG-16) [63]		88.2	88.1	68.1	96.4	71.1	97.5	84.9	15.4
CCSA (VGG-16) [13]		89.0	88.2	71.8	96.4	72.1	97.6	85.8	15.4
d-SNE (VGG-16) [14]		91.4	90.1	71.1	97.1	71.7	97.5	86.5	15.4
DAGE-LDA (VGG-16) [59]		85.9	88.7	66.2	97.9	64.2	99.5	83.6	15.4
EvoNAS-TL (sequential)	**SS**	**89.6**	**93.8**	**79.1**	**94.1**	**77.7**	**83.8**	**86.4**	15.2
	SS + LE	66.3	82.8	74.9	81.6	74.5	61.9	73.7	**12.8**
EvoNAS-TL (selective)	**SS**	**77.6**	**95.3**	**77.3**	**90.9**	**77.0**	**83.8**	**83.7**	18.4
	SS + LE	64.1	80.3	75.2	81.2	71.6	68.8	73.5	**14.5**

TABLE V
PERFORMANCE OF EVONAS-TL USING THREE DIFFERENT STRATEGIES IN SS AND LE IN TRANSFERRING FROM IMAGENET TO CIFAR-10. THE PAIRWISE VALUES OF EACH CELL INDICATE TESTING ACCURACY IN PERCENTAGE AND REDUCTION PERCENTAGE OF THE NUMBER OF PARAMETERS, RESPECTIVELY

		SS (sequential representation)			SS (selective representation)		
		min_err	knee	min_par	min_err	knee	min_par
LE	–	87.2 / 51.8	82.3 / 97.3	22.2 / 99.9	87.1 / 62.6	84.3 / 93.4	62.1 / 97.9
	min_err	87.2 / 56.3	82.4 / 97.5	23.2 / 99.9	86.8 / 66.7	84.2 / 94.0	62.7 / 98.3
	knee	75.4 / 81.5	65.1 / 99.2	21.5 / 99.9	75.5 / 84.3	66.1 / 97.9	51.3 / 99.3
	min_par	12.2 / 90.6	15.0 / 99.7	11.7 / 99.9	11.3 / 92.7	14.9 / 99.0	19.0 / 99.8

Fig. 5 shows the APFs produced by SS and LE. The solid triangles and diamond in Fig. 5(a) and (b), respectively, plot the minimal-error model, the minimal-size model, and the knee model in the APF obtained from SS. The hollow symbols in Fig. 5(b) depict the APFs gained by applying LE to the three models. The distribution of each APF shows a clear conflicting relationship between the objectives of minimizing prediction error and minimizing model size.

To further inspect the performance of models located in different regions of objective space, Tables V and VI compare the performance of three decision-making policies for selecting models from an APF. As the flowchart in Fig. 1 indicates, the knee solution and the best solutions of the two objectives are selected from the resultant APF of KGEA. The models that aim for minimal parameters are undesirable in practice due to their poor classification accuracy. The knee selection strategy presents a compromise between the two conflicting objectives [24], [64]. Considering the synthetic effects, EvoNAS-TL using the knee strategy for SS and the minimal-error solution for LE can strike a balance between the two objectives and generate models that have acceptable classification error with a substantial reduction in model size. In contrast, applying knee selection to both stages causes KGEA to trade too much accuracy to reduce the model size. The preferable outcomes above indicated that EvoNAS-TL showed strong capabilities in advancing classification accuracy through minimal-error selection in both stages and in finding satisfactory results through the knee selection in SS and minimal-error selection in LE.

Fig. 6 further depicts examples of resultant models from EvoNAS-TL. The generated models using selective representation reflect that EvoNAS-TL tends to discard the last few convolutional layers. Additionally, the models using sequential representation tend to remove only a few layers. These tendencies are in line with the idea that the layers close to the input can catch low-level features and the layers close to the output act as high-level feature constructors as well as classifiers. On the other hand, extended from the sequential representation, selective representation can remove arbitrary convolutional layers and fix the parameters of arbitrary convolutional

TABLE VI

PERFORMANCE OF EVONAS-TL USING THREE DIFFERENT STRATEGIES IN SS AND LE IN TRANSFERRING FROM IMAGENET TO THE NEU DATASET. THE PAIRWISE VALUES OF EACH CELL INDICATE TESTING ACCURACY IN PERCENTAGE AND REDUCTION PERCENTAGE OF THE NUMBER OF PARAMETERS, RESPECTIVELY

		SS (sequential representation)			SS (selective representation)		
		min_err	knee	min_par	min_err	knee	min_par
LE	–	99.6 / 63.1	80.0 / 97.5	22.5 / 99.9	98.6 / 77.6	85.4 / 94.5	54.6 / 97.8
	min_err	98.9 / 78.1	85.8 / 98.6	34.9 / 99.9	98.7 / 84.8	90.4 / 97.1	67.2 / 98.9
	knee	87.4 / 90.3	61.6 / 99.7	32.3 / 99.9	87.9 / 93.7	63.4 / 99.3	50.5 / 99.9
	min_par	26.6 / 94.5	20.7 / 99.9	18.2 / 99.9	26.2 / 97.3	21.3 / 99.7	26.0 / 99.9

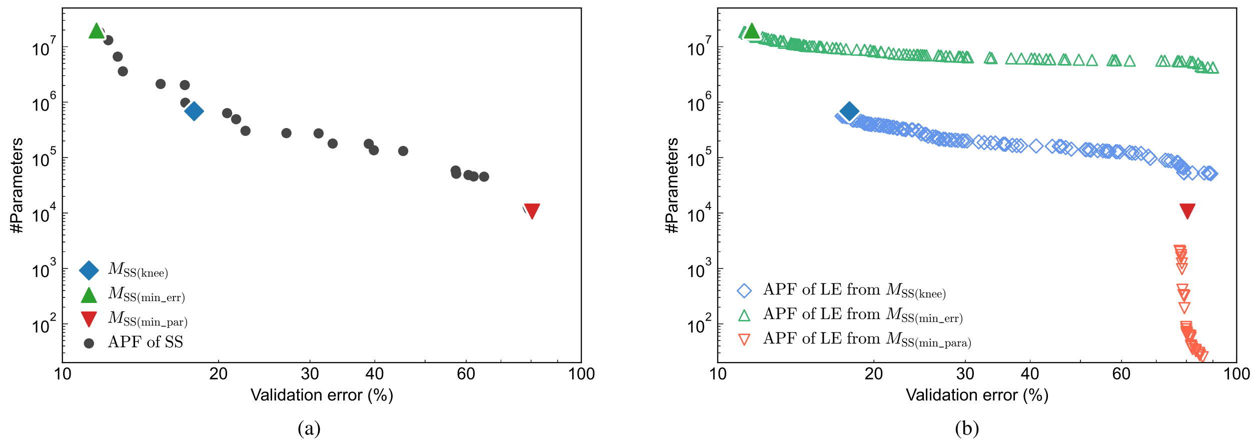

Fig. 5. Example of APFs generated by EvoNAS-TL using sequential representation in the case of transferring from ImageNet to the CIFAR-10 dataset. (a) Resultant APF of SS. (b) Resultant APF of LE.

layers. This extension allows EvoNAS-TL to flexibly manipulate the architecture of the source model and generate the models in an efficient model size. Although selective representation can fix the parameters of arbitrary convolutional layers, the experimental results show that EvoNAS-TL rarely opted to do so.

In short, EvoNAS-TL can automatically search for suitable subnetworks of the source model in transfer learning tasks. The two representations showed no significant difference in improving the prediction accuracy. However, selective representation was more effective in reducing the model size for the CIFAR-10 and NEU datasets, whereas the sequential representation is easier to implement. For applications seeking high classification performance, applying minimal-error selection to both stages gives satisfactory results. For applications pursuing a compact model with acceptable classification performance, applying the knee selection to SS and minimal-error selection to LE yields remarkable results in terms of both classification performance and model size.

V. CONCLUSION

Transfer learning has been used extensively to enhance deep learning models for learning from low-resource tasks. Despite many successful applications, one notable concern about transfer learning is the potential inefficiency of the neural architecture, as the transfer techniques focus on tuning parameters for the target task but seldom modify the neural architecture of the source model. This issue impedes those real-world applications with a limitation on the model size or inference latency, such as automatic optical inspection in manufacturing. This study proposed EvoNAS-TL, a two-stage biobjective evolutionary system that performs SS followed by LE for transfer learning tasks. The SS manipulates the source model over the structure by layer as a coarse-grained architectural search, while the LE attempts to remove unfavored filters and neurons as fine-grained architectural optimization. For SS, we proposed two representations that encode different levels of structural information.

The performance of EvoNAS-TL was examined and compared with three baseline methods and state-of-the-art methods over three transfer learning scenarios. The experimental results showed that, by automatically tailoring neural architecture for the target tasks, the proposed EvoNAS-TL improved classification performance and significantly reduced the model size. Using the minimal-error selection in both stages, EvoNAS-TL improved the testing accuracy by 6.9% in transferring to the CIFAR-10 dataset and 0.7% in transferring to the NEU dataset; meanwhile, it removed 52%–85% of the parameters from the source model. This strategy benefits applications that seek high classification performance. Moreover, by selecting the knee model in SS and the minimal-error model in LE, EvoNAS-TL can strike a balance between test accuracy and model size. On the Office-31 dataset, EvoNAS-TL achieved average accuracy and model size comparable to or better than state-of-the-art domain adaptation methods. These satisfactory outcomes validate that the proposed EvoNAS-TL, which aims to optimize the neural architecture in transfer learning tasks, shows promise in advancing the effectiveness and efficiency of the transferred models.

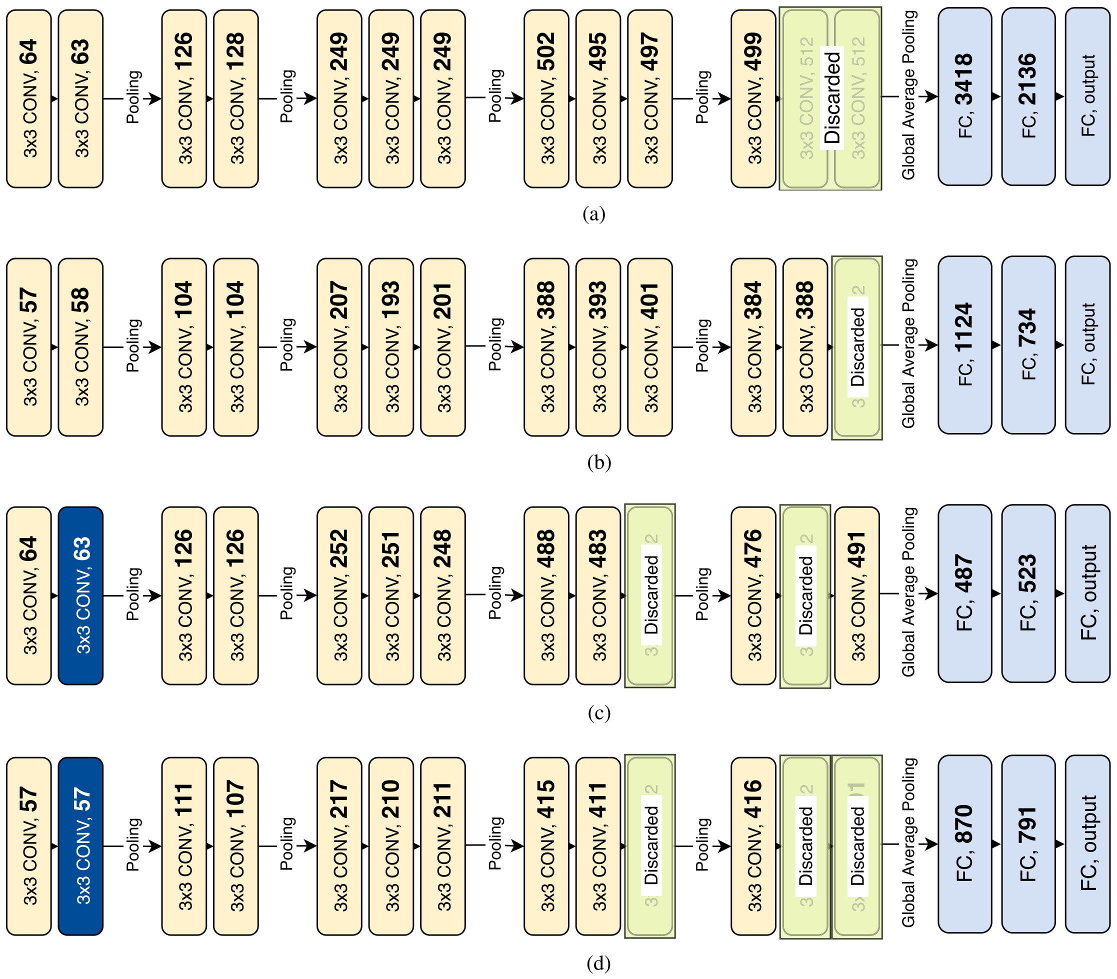

Fig. 6. Example models produced by EvoNAS-TL with minimal-error selection applied to both stages. (a) Resultant model of transferring from ImageNet to the CIFAR-10 dataset using sequential representation in SS. (b) Resultant model of transferring from ImageNet to the NEU dataset using sequential representation in SS. (c) Resultant model of transferring from ImageNet to the CIFAR-10 dataset using selective representation in SS. (d) Resultant model of transferring from ImageNet to the NEU dataset using selective representation in SS.

Acknowledgment

The authors would like to thank National Center for High-performance Computing (NCHC) in Taiwan for providing computational and storage resources.

References

[1] K.-M. He, X.-Y. Zhang, S.-Q. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

[2] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[3] F. Zhuang *et al.*, "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.

[4] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Proc. Int. Conf. Artif. Neural Netw. (ICANN)*, 2018, pp. 270–279.

[5] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 3320–3328.

[6] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2015, pp. 4068–4076.

[7] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 97–105.

[8] H.-C. Shin *et al.*, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1285–1298, May 2016.

[9] K. Gopalakrishnan, S. K. Khaitan, A. Choudhary, and A. Agrawal, "Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection," *Construct. Build. Mater.*, vol. 157, pp. 322–330, Dec. 2017.

[10] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 2208–2217.

[11] M. Hon and N. M. Khan, "Towards Alzheimer's disease classification through transfer learning," in *Proc. IEEE Int. Conf. Bioinformat. Biomed. (BIBM)*, 2017, pp. 1166–1169.

[12] S.-H. Kim, W.-Y. Kim, Y.-K. Noh, and F. C. Park, "Transfer learning for automated optical inspection," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 2517–2524.

[13] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, "Unified deep supervised domain adaptation and generalization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 5715–5725.

[14] X. Xu, X. Zhou, R. Venkatesan, G. Swaminathan, and O. Majumder, "d-SNE: Domain adaptation using stochastic neighborhood embedding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 2497–2506.

[15] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICRL)*, 2017, pp. 1–6.

[16] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 4095–4104.
[17] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 8697–8710.
[18] Y. Li, Z. Yang, Y. Wang, and C. Xu, "Adapting neural architectures between domains," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020, pp. 1–9.
[19] C. Wong, N. Houlsby, Y. Lu, and A. Gesmundo, "Transfer learning with neural AutoML," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 8366–8375.
[20] R. Istrate, F. Scheidegger, G. Mariani, D. Nikolopoulos, C. Bekas, and A. C. I. Malossi, "TAPAS: Train-less accuracy predictor for architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 3927–3934.
[21] E. Kokiopoulou, A. Hauth, L. Sbaiz, A. Gesmundo, G. Bartok, and J. Berent, "Fast task-aware architecture inference," 2019. [Online]. Available: arxiv.abs/1902.05781.
[22] M. Wistuba and T. Pedapati, "Inductive transfer for neural architecture optimization," 2019. [Online]. Available: arxiv.abs/1903.03536.
[23] C. Xue, J. Yan, R. Yan, S. M. Chu, Y. Hu, and Y. Lin, "Transferable automl by model sharing over grouped datasets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 9002–9011.
[24] Y. Zhou, G. G. Yen, and Z. Yi, "A knee-guided evolutionary algorithm for compressing deep neural networks," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1626–1638, Mar. 2021.
[25] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[26] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Rep., 2009.
[27] K.-C. Song and Y.-H. Yan, "A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects," *Appl. Surface Sci.*, vol. 285, pp. 858–864, Nov. 2013.
[28] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. 11th Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 213–226.
[29] W. Rawat and Z.-H. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Comput.*, vol. 29, no. 9, pp. 2352–2449, 2017.
[30] 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.
[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: arXiv:1409.1556.
[32] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 818–833.
[33] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–6.
[34] C. Reinhold and M. Roisenberg, "Filter pruning for efficient transfer learning in deep convolutional neural networks," in *Proc. Int. Conf. Artif. Intell. Soft Comput.*, 2019, pp. 191–202.
[35] E. Real *et al.*, "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 2902–2911.
[36] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020.
[37] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proc. ACM Genet. Evol. Comput. Conf. (GECCO)*, 2017, pp. 497–504.
[38] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016. [Online]. Available: arXiv:1611.01578.
[39] E. Real, A. Aggarwal, Y.-P. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4780–4789.
[40] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 2423–2432.
[41] H.-X. Liu, K. Simonyan, and Y.-M. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–6.
[42] B.-C. Wu *et al.*, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 10734–10742.
[43] L.-X. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 1379–1388.
[44] H.-P. Zhou, M.-H. Yang, J. Wang, and W. Pan, "BayesNAS: A Bayesian approach for neural architecture search," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 1–11.
[45] Y.-N. Sun, B. Xue, M.-J. 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.
[46] Y.-C. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," 2014. [Online]. Available: arXiv:1412.6115.
[47] W. Wen, C.-P. Wu, Y.-D. Wang, Y.-R. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 2074–2082.
[48] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–6.
[49] J.-H. Luo, J.-X. Wu, and W.-Y. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 5068–5076.
[50] 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. 1–6.
[51] H.-Y. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016. [Online]. Available: arXiv:1607.03250.
[52] Y.-H. He, J. Lin, Z.-J. Liu, H.-R. Wang, L.-J. Li, and S. Han, "AMC: AutoML for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 784–800.
[53] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 1398–1406.
[54] Y.-H. Wang, C. Xu, J.-Y. Qiu, C. Xu, and D.-C. Tao, "Towards evolutionary compression," in *Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2018, pp. 2476–2485.
[55] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 1989, pp. 598–605.
[56] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 1992, pp. 164–171.
[57] B. Bartoldson, A. Morcos, A. Barbu, and G. Erlebacher, "The generalization-stability tradeoff in neural network pruning," in *Proc. Conf. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1–13.
[58] M. Shaha and M. Pawar, "Transfer learning for image classification," in *Proc. Int. Conf. Electron. Commun. Aerosp. Technol. (ICECA)*, 2018, pp. 656–660.
[59] L. H. Morsing, O. A. Sheikh-Omar, and A. Iosifidis, "Supervised domain adaptation using graph embedding," 2020. [Online]. Available: arXiv:2003.04063.
[60] M. Lin, Q. Chen, and S.-C. Yan, "Network in network," 2013. [Online]. Available: arXiv:1312.4400.
[61] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "HyperBand: Bandit-based configuration evaluation for hyperparameter optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–6.
[62] M. Lindauer and F. Hutter, "Warmstarting of model-based algorithm configuration," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 1355–1362.
[63] S. Motiian, Q. Jones, S. M. Iranmanesh, and G. Doretto, "Few-shot adversarial domain adaptation," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 6673–6683.
[64] 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.

Yu-Wei Wen (Graduate Student Member, IEEE) received the B.S. degree in computer science and information engineering from National Chung Cheng University, Chiayi, Taiwan, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Information Engineering.

His research interests include evolutionary computation, memetic algorithm, machine learning, and computer music composition.

Sheng-Hsuan Peng received the B.S. degree in power mechanical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2018, where he is currently pursuing the M.S. degree with the Department of Power Mechanical Engineering.

His research interests include evolutionary computation, deep learning, and machine learning.

Chuan-Kang Ting (Senior Member, IEEE) received the B.S. degree from National Chiao Tung University, Hsinchu, Taiwan, in 1994, the M.S. degree from National Tsing Hua University, Hsinchu, in 1996, and the Dr. rer. nat. degree in computer science from Paderborn University, Paderborn, Germany, in 2005.

He is currently a Professor and the Chair of Department of Power Mechanical Engineering, National Tsing Hua University. His research interests include evolutionary computation, computational intelligence, machine learning, and their applications in machinery, manufacturing, ethics, music and arts.

Dr. Ting is the Editor-in-Chief of *IEEE Computational Intelligence Magazine* and *Memetic Computing*, an Associate Editor of IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, and an Editorial Board Member of *Soft Computing*. He served as the IEEE Computational Intelligence Society (CIS) Newsletter Editor, the IEEE CIS Webmaster, the Chair of IEEE CIS Chapters Committee, and the Chair of IEEE CIS Creative Intelligence Task Force. He is an Executive Board Member of Taiwanese Association for Artificial Intelligence.