# Surrogate-Assisted Particle Swarm Optimization for Evolving Variable-Length Transferable Blocks for Image Classification

Bin Wang, *Student Member, IEEE*, Bing Xue, *Member, IEEE*, and Mengjie Zhang, *Fellow, IEEE*

***Abstract*— Deep convolutional neural networks (CNNs) have demonstrated promising performance on image classification tasks, but the manual design process becomes more and more complex due to the fast depth growth and the increasingly complex topologies of CNNs. As a result, neural architecture search (NAS) has emerged to automatically design CNNs that outperform handcrafted counterparts. However, the computational cost is immense, e.g., 22 400 GPU-days and 2000 GPU-days for two outstanding NAS works named NAS and NASNet, respectively, which motivates this work. A new effective and efficient surrogate-assisted particle swarm optimization (PSO) algorithm is proposed to automatically evolve CNNs. This is achieved by proposing a novel surrogate model, a new method of creating a surrogate data set, and a new encoding strategy to encode variable-length blocks of CNNs, all of which are integrated into a PSO algorithm to form the proposed method. The proposed method shows its effectiveness by achieving the competitive error rates of 3.49% on the CIFAR-10 data set, 18.49% on the CIFAR-100 data set, and 1.82% on the SVHN data set. The CNN blocks are efficiently learned by the proposed method from CIFAR-10 within 3 GPU-days due to the acceleration achieved by the surrogate model and the surrogate data set to avoid the training of 80.1% of CNN blocks represented by the particles. Without any further search, the evolved blocks from CIFAR-10 can be successfully transferred to CIFAR-100, SVHN, and ImageNet, which exhibits the transferability of the block learned by the proposed method.**

***Index Terms*— Convolutional neural networks (CNNs), evolutionary deep learning, image classification, neural architecture search (NAS).**

## I. Introduction

CONVOLUTIONAL neural networks (CNNs) have shown promising performance in tackling image classification tasks [1], [2], and the state-of-the-art classification accuracy records have been constantly broken in recent years. One obvious trend has been growing the depth of CNNs to improve the classification accuracy, from only five convolutional layers of AlexNet [3] to tens of layers of VGGNet [4] and hundreds of layers of ResNet [5] and DenseNet [6]. Another observation from the recent CNNs is that the shortcut connections [5] have been introduced to connect the layers that are not next to each other in CNNs, e.g., ResNet [5], DenseNet [6], wide residual networks [7], and PyramidNets [8]. The shortcut connections have broken the traditional feed-forward topology of CNNs but enable more flexible topologies of CNNs. A serious side effect of the depth increase and topology flexibility is that handcrafting CNNs has become much more complex because the optimal depth and optimal topology are extremely difficult to be found due to the indefinite search space comprised of the depth and the topology of CNNs. Apart from the difficulty, the manual design process requires high-level expertise in both CNNs and the data sets, which is also time-consuming because every design trial has to be trained by the stochastic gradient descent (SGD) algorithm and the training process is slow, especially for deep CNNs.

Consequently, a research area of automatically searching for optimal CNNs has been surging in recent years. Two machine learning techniques have been widely used in this area—reinforcement learning (RL) and evolutionary computation (EC). For example, the research works [9]–[11] have demonstrated that the CNNs designed automatically by RL methods can outperform the state-of-the-art handcrafted CNNs. The counterpart of using EC methods to automatically evolve CNNs has also achieved promising performance similar to the RL methods (see [12]–[15]). However, most of the methods obtained good CNNs via evaluating a large number of CNNs, which requires very expensive computational cost for training CNNs. For example, Zoph and Le [9] and Zoph *et al.* [11] achieved the state-of-the-art classification accuracy with the computational cost of 22 400 GPU-days and 2000 GPU-days, respectively. They ran the experiments on hundreds of GPUs to acquire good CNNs within a reasonable time frame, but most of the researchers or practitioners do not have luxurious computing resources. In this article, a surrogate-assisted EC method will be proposed to significantly mitigate the expensive computational cost issue.

The proposed method is mainly inspired by NasNet [11] and DenseNet [6]. NasNet established a new approach toward the

Manuscript received 3 July 2020; revised 23 October 2020; accepted 21 January 2021. Date of publication 8 February 2021; date of current version 4 August 2022. This work was supported in part by the Marsden Fund of New Zealand Government under Contract VUW1509, Contract VUW1615, Contract VUW1913, and Contract VUW1914; in part by the Science for Technological Innovation Challenge (SfTI) Fund under Grant E3603/2903; in part by the University Research Fund at Victoria University of Wellington under Grant 223805/3986; in part by the MBIE Data Science SSIF Fund under Contract RTVU1914; and in part by the National Natural Science Foundation of China (NSFC) under Grant 61876169. *(Corresponding author: Bin Wang.)*

The authors are with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: bin.wang@ecs.vuw.ac.nz; bing.xue@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3054400.

Digital Object Identifier 10.1109/TNNLS.2021.3054400

efficient search of CNNs by seeking optimal blocks instead of whole CNN architectures. Since the block is much smaller than the whole CNN architecture, the computational cost of training the block is much lower, which, therefore, can accelerate the automatic search process. NasNet also found that the optimal blocks learned from one data set could be transferred to another data set. This article will adopt the strategy of evolving single blocks and explore the transferability of the evolved blocks. Another motivation is from DenseNet. DenseNet has demonstrated the performance improvement by building densely connected CNNs, but it uses a fixed hyperparameter called growth rate for each layer in a dense block, which might not be an optimal solution. Hence, this article will propose an EC approach to exploring various growth rates for each layer. Particle swarm optimization (PSO) will be used as the EC algorithm because PSO is a relatively simple EC algorithm, which is computationally inexpensive and effective for optimizing a wide range of functions [16]–[18].

*Goals:* The overall goal of this article is to propose a new surrogate-assisted PSO method to efficiently and effectively evolve transferable blocks with the highlights of a novel surrogate method and the transferability of the evolved block. The goal will be achieved by accomplishing the following tasks.

1) A novel surrogate model will be proposed to predict the result of performance comparison between two CNNs to avoid the expensive computational cost of training CNNs. The proposed method will transform the performance estimation of CNNs into a simple binary classification task. Support vector machine (SVM) [19], [20] is chosen to solve the classification task.
2) In-depth analysis and visualization will be performed to verify the reliability of the proposed surrogate model. First, the pattern of the data to train the surrogate model will be discovered and visualized. Second, the surrogate model will be evaluated on different feature combinations of the data to figure out the best feature combinations for the surrogate model. Finally, the performance of the surrogate model during the evolutionary process will be analyzed.
3) A new method of creating a surrogate data set will be proposed. The surrogate data set is sampled from the original data set by reducing the sample size and the image resolution to reduce the computational cost.
4) A new surrogate-assisted PSO method will be proposed by integrating the surrogate model and surrogate data set into PSO to automatically evolve CNNs. The surrogate model will filter out the unnecessary evaluations of underperformed particles to accelerate the evolution. Moreover, the analysis and visualization of the evolutionary process will be done to gain deeper insights into the convergence of the proposed method.
5) An encoding strategy being able to explore various growth rates for each layer in variable-length blocks will be proposed. Further analysis will also be done to figure out the growth rates that are preferred by different layers in the block.

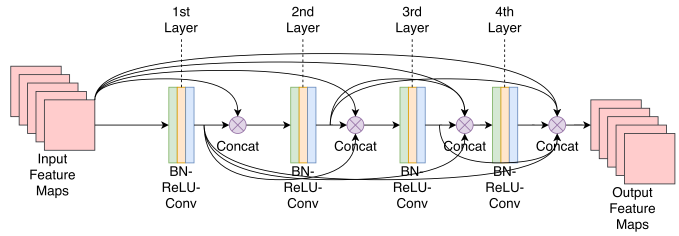



Fig. 1. Example of a dense block comprised of four layers.

The remainder of this article is organized as follows. Section II introduces the essential background to understand the proposed method, and Section III describes the details of the proposed method. The experiments are designed and illustrated in Section IV, and the experimental results are presented and analyzed in Section V. In the end, the conclusions are made and the future works are envisaged in Section VI.

## II. Background and Related Works

### A. DenseNet

Since the dense block is the most fundamental unit in DenseNet [6] and will be used in this article, the details of the dense block are explained. Fig. 1 shows a dense block of four layers. From the left of the figure, the input is a set of feature maps that are extracted by a convolutional layer [3]. The first layer takes the input and produces a set of output feature maps. Instead of passing the output feature maps to the next layer as the input feature maps in other CNNs such as VGGNet [4] and Fitnets [21], the output feature maps from the first layer are concatenated with the input feature maps of the first layer to form the input of the second layer. The same strategy applies to the following layers as well. In general, suppose that the input feature maps of the dense block are considered as the output feature maps of the layer 0, and the input of the $l_{\text{th}}$ layer comes from the concatenation of all the output feature maps from the layers of $0, 1, \ldots, l-1$. In the end, the output feature maps of the dense block are obtained by concatenating the output feature maps of all the layers in the block and the input feature maps of the dense block.

A layer in the dense block is actually a composite layer, which consists of three consecutive layers—a batch normalization (BN) layer [22], a rectified linear unit (ReLU) layer, and a convolutional layer with $3 \times 3$ filters. The composite layers are written as BN-ReLU-Conv in Fig. 1

$$n_{l+1} = \sum_{0}^{l} k_i. \tag{1}$$

The number of input feature maps of the $(l+1)_{\text{th}}$ layer, i.e., $n_{l+1}$, can be derived from (1), where $k_0$ is the number of input feature maps of the dense block and $k_l$ is the number of output feature maps of the $l_{\text{th}}$ layer. $k_i$ is a key hyperparameter for the composite layer in the dense block, which is called the growth rate [6]. The definition of the growth rate is the number of feature maps that the composite layer produces. In the experiments of DenseNet paper [6], it takes the approach of having the same growth rate $r$ for all of the layers in the same dense block. Therefore, (1) can be transformed as $n_{l+1} = k_0 + r \times l$.

### B. Particle Swarm Optimization

PSO [16], [23] is essentially inspired by behaviors, such as bird flocking, fish schooling, and swarm theory. Similar to the genetic algorithm (GA), and it initializes the population with random solutions, which are called particles. However, in additional to GA, PSO also assigns a random velocity to every particle

$$v_{id}(t+1) = wv_{id}(t)+c_1\epsilon_1(p_{id}-x_{id}(t))+c_2\epsilon_2(p_{\text{gd}} - x_{id}(t)) \tag{2}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1). \tag{3}$$

The particles fly through the hyperspace only based on primitive mathematical operators shown in (2) and (3), called PSO update equations. Both the particle's position and its velocity are represented by vectors. In the update equations, $x_{id}$ means the $d_{\text{th}}$ dimension of the $i_{\text{th}}$ particle's position, $v_{\text{id}}$ represents the $d_{\text{th}}$ dimension of the $i_{\text{th}}$ particle's velocity, $p_{\text{id}}$ is the particle's best-so-far solution at the $d_{\text{th}}$ dimension, and $p_{\text{gd}}$ keeps the best-so-far solution at the $d_{\text{th}}$ dimension across all neighbors of the particle. $(t)$ indicates the current state of the position or velocity, whereas $(t+1)$ denotes the next state. Apart from the particle's values, there are also several parameters in the update equations. Both $\epsilon_1$ and $\epsilon_2$ are random values between 0 and 1, $c_1$ and $c_2$ are two acceleration coefficients, and $w$ is called inertia weight. $w$ is a value between 0 and 1, which is used to avoid the explosion of the velocity.

### C. Related Work

In recent years, RL methods have been widely investigated to automatically design CNNs following the successful research [9] named neural architecture search (NAS). NAS uses a recurrent network as a controller to generate variable-length strings, which represent CNN architectures, and RL optimizes the controller to find the optimal CNNs. NAS achieved the state-of-the-art CNNs by consuming the computing resource of 22 400 GPU-days. As the computational cost is too expensive, several excellent RL methods have been proposed to reduce the computational cost. For example, NASNet [11] adopts a similar search strategy as NAS but only searching for a single CNN block, which successfully reduced the computational cost to 2000 GPU-days. PNASNet [24] and BockQNN [25] have further reduced the computational cost, but their classification accuracy was sacrificed comparing to NASNet.

Another emerging approach to automatically searching for optimal CNNs is EC-based methods. Early research is LS-Evolution [26], which employs evolutionary algorithms to evolve CNNs. It achieved promising performance by evolving CNNs for more than 2730 GPU-days. After that, there are a few successful EC-based methods proposed, such as GeNet [12], CGP-CNN [27], and EIGEN [28], which reduces the computational cost by compromising the classification accuracy comparing to LS-Evolution. Recently, AmoebaNet [14] and AECNN [29] were proposed. AmoebaNet proposed a regularized evolutionary algorithm to evolve CNNs, which achieved the state-of-the-art CNNs by using EC methods for the first time, but it took 3150 GPU-days. Conversely, AECNN accelerated the search process by evaluating fewer CNNs, which only consumed 27 GPU-days, but the classification accuracy was compromised comparing to AmoebaNet.

From the existing work, it can be observed that the tradeoff between the classification accuracy and the computational cost is a challenging problem in automatically designing CNNs both for RL methods and EC methods. In this article, a novel reliable surrogate model will be proposed to assist PSO to reduce the computational cost by maintaining or even increasing the classification accuracy. The main reason for adopting the EC method is that EC methods tend to converge faster in searching for CNN architectures [14].

## III. Proposed Method—EffPNet

In this section, the details of the proposed method, which is referred to as EffPNet (Efficient PSO Network), will be illustrated and discussed. Since the evolved blocks learned from one data set are expected to be transferable to other data sets, the evolved blocks are called transferable blocks. The overall framework will be outlined in Section III-A, and the encoding strategy will be described in Section III-B. The fitness evaluation, the feature construction of the surrogate model, the surrogate model, the surrogate data set, and the evolutionary process with the surrogate assistance will be detailed in Sections III-C–III-G, respectively. Finally, the method of stacking the evolved block will be explained in Section III-H.

### A. Overall Framework

The overall framework of the proposed method is shown in Fig. 2. First of all, a surrogate data set is sampled from the training set of the given data set according to the method detailed in Section III-F. The surrogate data set is a small subset of the training set, which is passed to the evolutionary process. Second, the surrogate-assisted PSO method searches for the optimal dense block by only using the surrogate data set. A surrogate model is trained on the data extracted during the evolutionary process (to be described in Sections III-D and III-E), comprised of the encoded vectors and the corresponding fitness values, which is used to predict the CNN performance instead of training the CNNs by SGD. The details of the surrogate model will be described in Sections III-D and III-E. By combining the surrogate data set and the surrogate model, the computational cost can be significantly reduced. Third, the evolved dense block is stacked by various numbers of times, e.g., once, twice, and three times in Fig. 2, to produce various CNN architectures as the final candidates. Only the training set is taken to evaluate the stacked CNNs, and the best candidate is selected as the final CNN. In the end, the final CNN is retrained on the training set and evaluated on the test set. The classification accuracy on the test set will be reported.

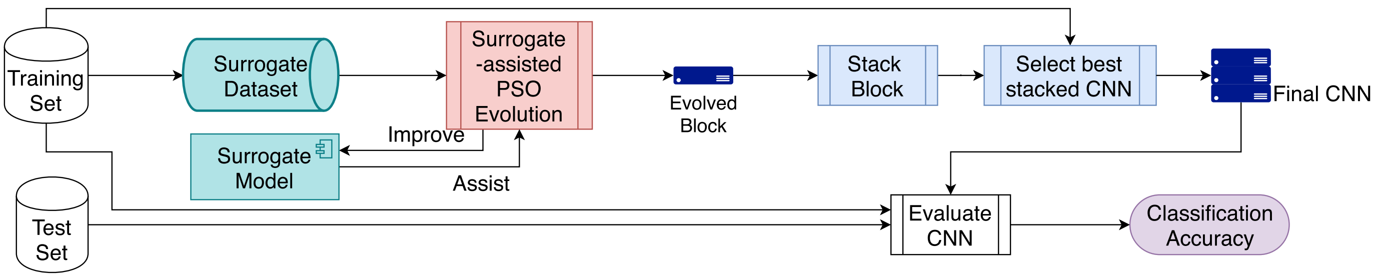



Fig. 2. Framework of the proposed surrogate-assisted PSO method.

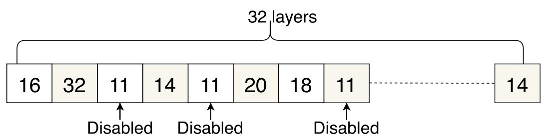



Fig. 3. Example of an encoded vector with the maximum number of layers set to 32.

### B. Encoding Strategy

The encoding strategy is designed to encode the hyperparameters of dense blocks [6] with variable lengths. In DenseNet, there are two hyperparameters—the number of layers and the growth rate. It uses the same growth rate for all of the layers in a dense block. However, the fixed growth rate of each layer in a dense block is not necessarily optimal, so the proposed method explores dense blocks with various growth rates for each layer besides the number of layers in a dense block in DenseNet [6].

In order to encode the above dense blocks in each particle, a fixed-length vector shown in Fig. 3 with the maximum number of layers in a dense block as the dimensionality is proposed to accommodate the hyperparameters. Each dimension in the vector represents the growth rate of the corresponding layer. A special value is introduced to indicate that the corresponding layer is disabled to achieve variable-length dense blocks. There are three hyperparameters that need to be defined to accomplish the encoding vector. The first one is the maximum number of layers ($ml$) in a dense block. The second one is the range of the growth rates—the lower bound ($g_l$) and the upper bound ($g_u$). The last is the special value ($sv$) to disable a layer in the encoded vector. In the example of Fig. 3, $ml$ is set to 32, which is based on the capacity of hardware resource and the complexity of the image data set; 12 is used for $g_l$ based on the experimental experience in DenseNet [6] because if the growth rate is too small, it will not be able to capture the features in the output feature maps, and 32 is used for $g_u$ due to the hardware capacity. After defining the range of the growth rates, $sv$ is defined as the value of $g_l - 1$, so the value of each dimension in the vector is a continuous value between 12 and 32 inclusive; 11 is used to represent the disabled layer.

### C. Fitness Evaluation

The details of the fitness evaluation method are illustrated in Algorithm 1. The dense block represented by a particle in the PSO swarm is passed to the fitness evaluation function along with the *fitness evaluation data set*. The *fitness evaluation data set* can be the whole training data set or a subset of the training data set, which is split into the *training part* and the *test part*. The training part is used to train the dense block by SGD. The test part is used to evaluate the trained dense block to produce the classification accuracy, which is used as the fitness value.

---

**Algorithm 1** Fitness Evaluation

---

**Input:** block $b$, *fitness evaluation data set* $d$;
$acc_{best}, epoch_{best}, epoch, acc \leftarrow 0, 0, 0, 0$;
$losses, acc\_history \leftarrow$ empty, empty;
$d_{train}, d_{test} \leftarrow$ Randomly split $d$ into 80% as the training part $d_{train}$ and 20% as the test part $d_{test}$;
**while** $acc >= acc_{best}$ **or** $epoch - epoch_{best} < 5$ **do**
  Apply Adam optimization [30] to train $b$ on $d_{train}$ for one epoch;
  $losses \leftarrow$ Append training loss to $losses$;
  $acc \leftarrow$ Evaluate $b$ on $d_{test}$;
  $acc\_history \leftarrow$ Append $acc \leftarrow$ to $acc\_history$;
  **if** $acc > acc_{best}$ **then**
    $acc_{best}, epoch_{best} \leftarrow acc, epoch$;
  **end if**
  $epoch \leftarrow epoch + 1$;
**end while**
Save $block, losses, acc\_history, acc\_best$ as one row to a file or database as *block training history*;
**return** $acc_{best}$;

---

During the above fitness evaluation process, a set of data are collected from the process of training the dense block, which will be used by the surrogate model later in Sections III-D–III-F. During the training process, the training loss and the classification accuracy on the test part of every epoch are recorded along with the best classification accuracy. These recorded data during the training and the parameters of the dense block are combined and saved as one row of records in a file or a database called *block training history*, which will later be used to construct the data to train the surrogate model in Sections III-D and III-E.

One important decision made for the fitness evaluation is that Adam optimization [30] is chosen as the SGD algorithm to train the dense blocks. The main advantage of Adam optimization is that the learning rate is adjusted based on the training status to optimize the learning rate at the specific epoch for the specific CNN. Due to the adjustment of the learning rate, Adam optimization demonstrates faster convergence compared with the SGD algorithm [31] with a fixed learning rate, which can accelerate the fitness evaluation. Besides, the adjustable learning rate can automatically optimize the training process for a CNN, so the classification accuracies obtained for different CNNs rely less on the prespecified settings of the SGD method, which provides a fairer comparison between CNNs [32].

### D. Feature Construction for Surrogate Model

Since PSO only updates the particle's best-so-far solution when the particle's new solution outperforms the best-so-far solution, it is not necessary to acquire the fitness value when the particle's new solution does not perform better. The objective of the proposed surrogate model is to predict whether the CNN represented by the particle's new solution would outperform that represented by the particle's best-so-far solution. First of all, the data used to train the surrogate model need to be constructed from the *block training history* obtained during the fitness evaluation, which transforms the direct prediction of a CNN's performance into a binary classification task of comparing the performance of a pair of CNNs. The class label will be 1 if the first CNN in the pair outperforms the other. Otherwise, the class label will be 0. First, the features in *block training history* need to be selected to represent the evolved block. Since the parameters of the transferable block are decisive to its performance, the parameters are selected. Both the training losses and the classification accuracies are chosen because the training losses on the training part of the fitness evaluation data set reflect how well the transferable block is trained, and the classification accuracies on the test part of the fitness evaluation data set represent a kind of generalization quality. Furthermore, a hyperparameter named the *feature-cutting epoch* is defined, which is the number of epochs used to extract the training losses and classification accuracies from the *block training history*. The *feature-cutting epoch* should be set based on the training process of the data set by finding the smallest cutting point of the training process where it would be sufficient to learn a trend of the whole training process. Finally, the data for training the surrogate model are formed by making pairs of the transferable blocks in the *block training history*. Fig. 4 shows an example of how an instance of the surrogate model's training data is constructed from a pair of records to form the binary classification task, each of which represents a CNN and its corresponding training process, in the *block training history*. The *feature-cutting epoch* is set to 10. A pair of transferable blocks are drawn in the first row, each of which has 16 growth rates in the block parameters, the first ten training losses, the first ten classification accuracies, and the best classification accuracy extracted from the *block training history*. The second row shows the constructed instance from the above pair. The total number of features of the constructed instance is 72, which includes the 16 growth rates, the training losses of the first ten epochs, and the classification accuracy of the ten epochs from each of the two blocks in the pair, i.e., the formula to calculate the total number of features is $(16+10+10)\times 2 = 72$. In this specific example, the class label of the binary classification task is set to 0 because, in the example, the best classification accuracy of the first block is less than that of the second block. Otherwise, the class label would be set to 1. A couple of benefits of doing so are that a much larger number of data instances can be obtained by making pairs of a small number of data to achieve an effect of the data augmentation, and it transforms the performance prediction into a binary classification task to simplify the performance prediction task.

### E. Surrogate Model

SVM is chosen as the surrogate model to predict whether one CNN could achieve better performance than the other mainly because of a couple of reasons. First, the data in the *block training history* collected from the fitness evaluation process are limited. SVM does not require a large number of examples as deep neural networks to achieve good performance. Second, the computational cost of fitting the SVM model is much less than training a deep learning model, so it can efficiently provide good predictions. The surrogate model is iteratively trained after each generation. The scores of tenfold cross validation of SVM are obtained to assess the quality of how SVM performs on the transformed binary classification task, which is used as the *scores of the surrogate model*. After that, the SVM model is fit by the training data, which is used as the surrogate model to perform predictions.

**Algorithm 2** Using the Surrogate Model to Perform Predictions

**Input:** the first transferable block $b_1$, the second transferable block $b_2$, *feature-cutting epoch* $c$;

1: **if** $b_1$ **in** *block training history* **then**
2: $f_1 \leftarrow$ Extract features for $b_1$;
3: **else**
4: $f_1 \leftarrow$ Extract the parameters of $b_1$, train $b_1$ for $c$ epochs, and then extract features for $b_1$;
5: **end if**
6: $f_2 \leftarrow$ Follow the above process (lines 1 to 5) to extract the features of $b_2$;
7: $f \leftarrow$ Concatenate the features $f_1$ and $f_2$;
8: $t \leftarrow$ Use surrogate model $m$ to predict the target class based on the combined features $f$;
9: **return** $t$;

The trained surrogate model is used to predict the comparison result between two transferable blocks. The prediction process is described in Algorithm 2. Three inputs are required in the prediction function—the two transferable blocks and the *feature-cutting epoch*. The *feature-cutting epoch* is the same as that of the training process. When building the features of an example for the prediction, if the transferable block exists in the *block training history*, the training losses and the classification accuracies will be extracted from the existing data; otherwise, the training losses and the classification accuracies are obtained from training the transferable block for the *feature-cutting epochs*. Following the data set construction process shown in Fig. 4, the features are built from the two transferable blocks. The surrogate model takes the built features and predicts the comparison result.

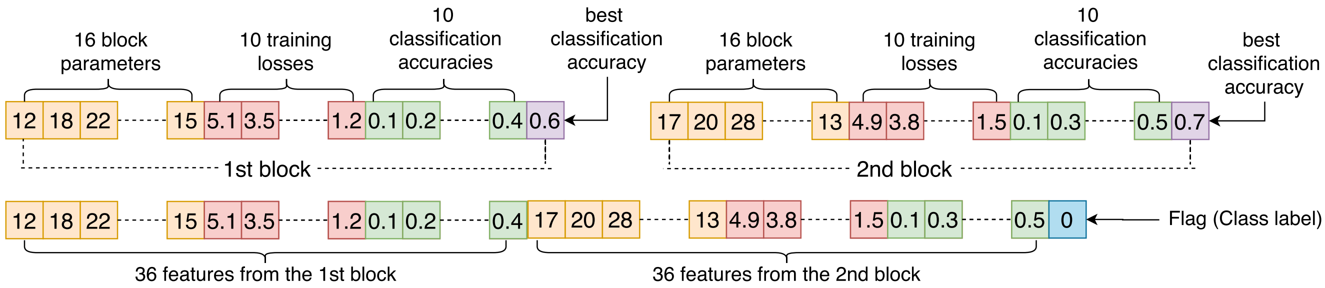



Fig. 4. Example of constructing the data set from the *block training history* to fit the SVM classifier.

At the beginning of each generation during the evolutionary process, the feature construction step for the surrogate model is performed, and the surrogate model is retrained based on the newly constructed data. This could help improve the surrogate model's coverage of the new CNNs found during the evolutionary process. Even though retraining the surrogate model may take a few minutes, it can still be neglectable because, without the surrogate model, the fitness evaluation of training a single particle, i.e., a single CNN, by using SGD usually takes about 1 h or so, and there are 30 particles that need to be evaluated.

### F. Surrogate Data Set

To reduce the computational cost of the fitness evaluation, the surrogate data set, which is a reduced training data set, is used to evaluate the transferable block instead of the whole training data set. Two techniques are utilized to form the surrogate data set. The first one is to downsize the *fitness evaluation data set* by sampling a small subset from the whole training data set according to a uniform distribution. By following the uniform distribution, the sample subset has a good chance to keep the characteristics of the whole training data set, so this could mitigate the sacrifice of the model performance. Another technique is to downsample the original images to smaller images. As the image resolution of the state-of-the-art benchmark data set could be very high, such as the ImageNet data set, the downsampled images are recognizable by humans, which means that the downsampled images can represent the original images very well. Therefore, the downsampled data set could be used to evaluate the transferable blocks.

Two hyperparameters are needed for the two techniques—the *data-reduction ratio* $p\%$ and the *downsampling factor* $n$. The *data-reduction ratio* is the percentage of the training data set that is used as the sample size of the surrogate data set. The *downsampling factor* is the downscaling ratio of the image size. For example, the downscaling scales down the image from the size of $w \times h$ to $w/n \times h/n$. Through downsizing the *fitness evaluation data set*, the computational cost of training the transferable block for one epoch is $p\%$ of the original cost. By downsampling the images, the feature map size is reduced to $1/n \times n$ of the original size, so both the memory and computational cost required to train the transferable block can be reduced to $1/n \times n$ of the original ones. In the combination of these two, the total computational cost of using the surrogate data set can be reduced to $1/n \times n \times p\%$ of the original cost.

### G. Evolving Transferable Blocks by Surrogate-Assisted PSO

**Algorithm 3** Evolving Dense Block by Surrogate-Assisted PSO

**Input:** generations $g$, surrogate model threshold $t$, *feature-cutting epoch* $c$;

```
p ← Random initialise the particles until the population is filled up;
g_best, j ← Empty, 0;
d_surrogate ← Generate the surrogate dataset from the training part of the fitness evaluation dataset according to Section III-F;
while j < g do
  Train the surrogate model according to Section III-E;
  for particle i in pop do
    i ← Apply standard PSO operations to update the position of i;
    if accuracy of the surrogate model ≥ t then
      flag ← Use surrogate model m to predict the performance comparison between the transferable blocks represented by the particle's best-so-far solution and the current solution;
      fitness ← Use Algorithm 1 to calculate the fitness for i if flag == 1. Otherwise, 0;
    else
      fitness ← Use Algorithm 1 to calculate the fitness for i;
    end if
    Update the fitness of i by fitness;
    Update the particle's best with the current solution when fitness > the fitness of the particle's best;
  end for
  g_best ← Update with the best particle among the current g_best and pop;
  j ← j + 1;
end while
return g_best;
```

The surrogate model and the surrogate data set are used to assist the PSO method in the evolutionary process to accelerate the proposed method. The details of the evolutionary process are illustrated in Algorithm 3. Since the proposed encoding strategy has transformed the variable-length parameters of transferable blocks into a fixed-length vector, the standard PSO operations can be applied. However, there are a couple of points specifically related to the surrogate model and the surrogate data set that need to be explained. First, two hyperparameters are defined. The first is the threshold to control the activation of the surrogate model, which is activated only when the mean value of the *scores of the surrogate model* obtained in Section III-E is larger than the threshold. The second is the *feature-cutting epoch*, which is the same as the *feature-cutting epoch* in Section III-D and Algorithm 2. Furthermore, if the surrogate model is activated, before evaluating the particle's new position, the surrogate model will be used to predict whether the new position of the particle would outperform its best position. The new position is evaluated by the fitness evaluation function only when it is predicted to surpass its best position; otherwise, the fitness value of the new position is set to 0. Therefore, the proposed method can avoid unnecessary fitness evaluation for transferable blocks with poor performance to reduce the computational cost.

**Algorithm 4** Stack and Select the Best Candidate

**Input:** Evolved block $b$, the training set $d$, the maximum number of times to stack $s_{max}$;

1: $d_{train}, d_{test}$ $\leftarrow$ Randomly split $d$ into training part $d_{train}$ and test part $d_{test}$ by 80% and 20%;
2: $c_{set}, t$ $\leftarrow$ Empty set of candidates, 0 as the current number of times to stack;
3: **while** $t < s_{max}$ **do**
4: $t \leftarrow t + 1$;
5: $c_{set}$ $\leftarrow$ Stack $b$ for $t$ times to generate a candidate and append it to the candidate set;
6: **end while**
7: Use Adam optimization to train each candidate in $c_{set}$ on $d_{train}$ and evaluate it on $d_{test}$;
8: $c_{best}$ $\leftarrow$ Find the candidate with the best classification on $d_{test}$;
9: **return** $c_{best}$;

### *H. Stacking and Selecting the Best CNN*

Since the block obtained from Algorithm 3 is learned from the surrogate data set, it might not be trained sufficiently to capture the complexity of the whole data set. Therefore, a stacking approach depicted in Algorithm 4 is introduced to enhance the capacity of the final network. The stacking approach is also required to transfer the evolved block to other domains because the capacity of the transferable block might not fit other data sets either. There is a hyperparameter—the *maximum number of times* $s_{\max}$ to stack the transferable block, which is defined to restrict the maximum capacity of the final network. $s_{\max}$ is dependent on the complexity of the whole data set and the hardware resource. During the stacking process, a set of candidates are generated by stacking the learned block from once to $s_{\max}$ times, which are then sent to multiple GPU cards to be evaluated in parallel to speed up the stacking process. The evaluation of each candidate is based on the whole training data set, which is split into two parts: the training part and the test part. The candidate is trained on the training part and is evaluated on the test part. When transferring the evolved block to target domains, the training data set is the training data set from the target domain. After receiving the classification accuracies of all candidates, the best candidate is selected as the final solution.

## IV. Experiment Design

In this section, the detailed design of the experiments will be depicted. The four benchmark data sets, CIFAR-10, CIFAR-100, SVHN, and ImageNet, and the selected peer competitors will be discussed in Sections IV-A and IV-B. In addition, the parameter settings to run the experiments will be listed and explained in Section IV-C.

### *A. Benchmark Data Sets*

First of all, the data set taken by the proposed method to generate the surrogate data set needs to be selected. CIFAR-10 [33] can fit this purpose well because it is a widely- used benchmark data set to evaluate image classification, and it is a medium-scale data set of ten classes comprised of various images with decent complexity. Therefore, there are three benefits of using CIFAR-10 to generate the surrogate data set. The first benefit is that the surrogate data set sampled from CIFAR-10 can reflect various images. The second is that the training process of CNNs on the surrogate data set does not take too much computational resource. The third is that the reduced number of examples in the surrogate data set does not make the classification task too hard during the fitness evaluation due to the small number of classes. Conversely, if CIFAR-100 is chosen instead of CIFAR-10, the surrogate data set might not be sufficient to train an effective CNN to distinguish the images from 100 classes. There are 60 000 colored images of ten classes in the CIFAR-10 data set, which contains 50 000 training images and 10 000 test images. In addition, to assess the classification performance of the block, CIFAR-10 is used again to validate the effectiveness of the evolved block learned from a subset of itself. Last but not least, the transferability of the evolved block needs to be evaluated on other benchmark data sets, which are not seen by the proposed method during the evolutionary process. CIFAR-100 [33] is chosen because the image domain and the total number of images in CIFAR-100 are similar to those of CIFAR-10, but its number of classes is extended to 100, which results in a much more complex classification task. Besides CIFAR-100, the SVHN data set [34], which is comprised of digit images from 0 to 9 obtained from house numbers in Google Street View images, could test the transferability of the evolved block in a different domain because its images are disparate from those of CIFAR-10. To further test the evolved blocks, the ImageNet data set [3], which contains 1000 classes and 1.2 million images in the training set, is adopted to evaluate the transferability of the proposed method on this much larger data set. Hence, the evolved transferable block can be thoroughly verified in various domains and various scales.

TABLE I
PARAMETER SETTINGS

| Parameter | Value |
|---|---|
| **EffPNet hyper-parameters** | |
| maximum number of layers in Section III-B | 16 |
| range of growth rate in Section III-B | [11, 32] |
| threshold to activate surrogate in Section III-E | 90% |
| feature-cutting epochs in Section III-D | 10 |
| data reduction ratio in Section III-F | 10% |
| downsampling factor in Section III-F | 2 |
| the maximum number of times to stack the block in Section III-H | 5 |
| **PSO parameters** | |
| inertia weight $w$ | 0.7298 |
| acceleration coefficient $c1$ | 1.49618 |
| acceleration coefficient $c2$ | 1.49618 |
| velocity range | [-10.5, 10.5] |
| population size | 30 |
| number of generations | 50 |

### B. Peer Competitors

The peer competitors are selected mainly based on the availability of the performance reported on the above benchmark data sets and their relevance to the proposed method. First, the performance of the evolved block on CIFAR-10, where the block is learned from its own surrogate data set, needs to be assessed. There are three sets of competitors chosen for the comparison. The first set consists of a couple of the state-of-the-art CNNs designed manually, which are ResNet [5] and DenseNet [6]. The second set incorporates automatically designed CNNs by RL, which are PNASNet [24], BockQNN [25], EAS [35], NASNet-A (7 @ 2304) [11], NASH (ensemble across runs) [36], and NAS v3 max pooling [9]. The last set is comprised of CNNs designed automatically by EC methods, which are EIGEN [28], RENAS [37], AECNN [29], AmoebaNet-B (6,128) [14], Hier. repr-n, evolution (7000 samples), CGP-CNN(ResSet) [27], DENSER [38] GeNet from WRN [12], CoDeapNEAT [39], and LS-Evolution [26]. Since the proposed method is for automatically designing CNNs based on EC, the relevance of peer competitors increases from the first to the last set. Therefore, the number of competitors grows from the first to the last set as well. Furthermore, the performance of the evolved block transferred to the other two data sets needs to be verified. Based on the availability of the reported performance on CIFAR-100, SVHN, and ImageNet, the following peer competitors will be compared with the proposed method—Network in Network [40], CiCNet [41], Deeply Supervised Net [42], FractalNet [43], WideResNet [7], ResNet [44], and DenseNet($k = 12$) [6].

### C. Parameter Settings

The hyperparameters of the experiments are defined and listed in Table I. First, the maximum number of layers is set to 16 and the range of growth rate for the block to be evolved is from 11 to 32 due to the hardware limit of running the experiments. The experiments run on a distributed system designed for evolving CNNs [45] with the lowest GPU card of GeForce RTX 2070. In our trial experiments, it was noticed that if the above hyperparameters increased, the error of insufficient memory would occur more frequently. Admittedly, the above numbers are not meant to be the optimal values, but they are chosen based on the computational resources available to us and can show the effectiveness of the proposed method. If more powerful GPUs are available, their values can be improved to further enhance the performance. Second, a set of hyperparameters for the surrogate model and the surrogate data set are designed. A high threshold of activating the surrogate model is set to 90% accuracy, and the feature-cutting epochs of 10 are designed for the surrogate model to achieve high accuracy. The data reduction ratio and downsampling factor are 10% and 2 to achieve a smaller surrogate data set in terms of both the number of examples and the image size, respectively. Third, the maximum number of times to stack the evolved CNNs is set to 5 based on the complexity of the benchmark data sets. Finally, the PSO parameters are set to widely used values according to the community convention [46]–[48].

## V. RESULTS AND ANALYSIS

The section endeavors to analyze the results from the experiments. For the result analysis, first of all, the performance on the four benchmark data sets, CIFAR-10, CIFAR-100, SVHN, and ImageNet, will be shown and compared with peer competitors. Then, the convergence of the proposed method will be visualized and analyzed. Furthermore, the experimental results specifically related to the surrogate model will be analyzed to demonstrate the effectiveness and efficiency of the surrogate model. In the end, the growth rates for different layers in the evolved blocks will be analyzed to show the preference of growth rates for different layers in the evolved blocks.

### A. Performance Comparisons

*1) Performance Comparisons on CIFAR-10:* Table II lists the classification accuracy, number of parameters, and time taken to obtain the final CNNs of the proposed method and the selected peer competitors. In the columns of the *Error rate%* and the *Number of Parameters*, if the result is obtained from multiple runs, the mean value and the standard deviation will be written as $mean\ value \pm standard\ deviation$. For the classification accuracy, comparing the best classification accuracy achieved by the proposed method with those of the 18 peer competitors, the proposed method achieved the fifth best, i.e., underperforming four peer competitors, whose error rates are in bold font in the table, with a very small margin. By applying the Mann–Whitney–Wilcoxon (MWW) statistical test on the classification accuracies from the ten runs of the proposed method and the error rate of 3.74% from DenseNet, it shows that the proposed method is significantly better than the error rate of DenseNet. With regard to the number of parameters, the smallest CNN found by the proposed method has slightly more parameters than 3 of the 18 competitors. However, the three competitors with competitive sizes perform much worse than the proposed method in terms of classification

TABLE II
PERFORMANCE COMPARISON WITH PEER COMPETITORS ON CIFAR-10

| Method | Error rate% | Number of Parameters | Computational Cost |
|---|---|---|---|
| ResNet-110 [5] | 6.43 | **1.7M** | – |
| DenseNet(k = 40) [6] | 3.74 | 27.2M | – |
| PNASNet [24] | **3.41 ± 0.09** | 3.2M | 225 GPU-days |
| BockQNN [25] | 3.54 | 39.8M | 96 GPU-days |
| EAS [35] | 4.23 | 23.4M | <10 GPU-days |
| NASNet-A (7 @ 2304) [11] | **2.97** | 27.6M | 2,000 GPU-days |
| NASH (ensemble across runs) [36] | 4.40 | 88M | 4 GPU-days |
| NAS v3 max pooling [9] | 4.47 | 7.1M | 22,400 GPU-days |
| EIGEN [28] | 5.4 | *2.6M* | **2 GPU-days** |
| RENAS [37] | **2.88** | 3.5M | 6 GPU-days |
| AECNN [29] | 4.3 | **2.0M** | 27 GPU-days |
| AmoebaNet-B (6,128) [14] | **2.98** | 34.9M | 3150 GPU-days |
| Hier. repr-n, evolution (7000 samples) [49] | 3.75 | – | 300 GPU-days |
| CGP-CNN(ResSet) [27] | 5.98 | **1.68M** | 29.8 GPU-days |
| DENSER [38] | 5.87 | 10.81M | – |
| GeNet from WRN [12] | 5.39 | – | 100 GPU-days |
| CoDeapNEAT [39] | 7.3 | – | – |
| LS-Evolution [26] | 4.4 | 40.4M | >2,730 GPU-days |
| **EffPnet (Best classification accuracy)** | 3.49 | 2.54M | <3 GPU-days |
| **EffPnet (10 runs)** | 3.576±0.0078 | 2.68M±0.015M | <3 GPU-days |

accuracy. In respect of the computational cost, only EIGEN took less time than the proposed method by 1 GPU-day, but its classification accuracy is almost 2% worse than that of the proposed method. Overall, the proposed method demonstrates its strong competitiveness with regard to the performance of all three measurements compared to the 18 peer competitors—the fifth best in the error rate, the fourth best in the number of parameters, and the second best in the computational cost.

*2) Transferability on CIFAR-100, SVHN, and ImageNet:* To verify the effectiveness of the block's transferability, the transferable block learned from CIFAR-10 is stacked and the best CNN architecture is selected based on the CIFAR-100, SVHN, and ImageNet data sets by following the method described in Section III-H. The classification error rates on CIFAR-100, SVHN, and ImageNet are listed in Table III along with those of its peer competitors. It can be observed that on the CIFAR-100 data set, the proposed method statistically significantly outperforms all the compared methods according to the MWW test. For SVHN, the best error rate achieved by the proposed method exceeds the peer competitors apart from DenseNet. However, after applying the MWW statistical test, the proposed method is not significantly better than Wide ResNet, so it shares the second place on SVHN. On the ImagetNet data set, due to tremendous computational cost, the error rate was achieved from one run instead of ten runs. It can be noticed that the proposed method ranks in the middle, i.e., three out of the six competitors achieved better performance. Therefore, the proposed method shows its competitiveness even on the large-scale data set. In addition, the margins between the proposed method and the best competitor are fairly small, which are less than 3% for the top-1 error rate and less than 2% for the top-5 error rate. To sum up, the proposed method has demonstrated its transferability by achieving very competitive classification accuracy in three different benchmark data sets.

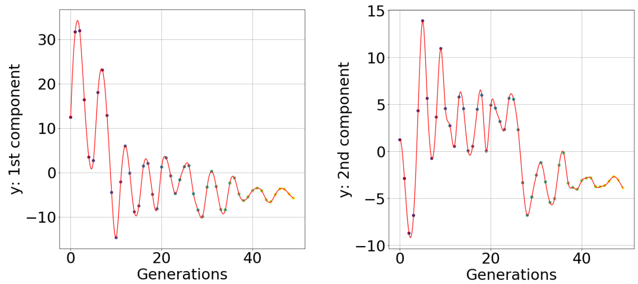
TABLE III
ERROR RATE COMPARISON WITH PEER COMPETITORS ON CIFAR-100, SVHN, AND IMAGENET. FOR IMAGENET, BOTH THE TOP-1 AND TOP-5 ERROR RATES ARE REPORTED

| Method | CIFAR-100 | SVHN | ImageNet (top-1/top-5) |
|---|---|---|---|
| Network in Network [40] | 35.68 | 2.35 | – |
| Deeply Supervised Net [42] | 34.57 | 1.92 | 33.7/13.1 |
| CiCNet [41] | 24.82 | – | – |
| FractalNet [43] | 23.30 | 2.01 | **24.12/7.39** |
| Wide ResNet [7] | 22.07 | **1.85** | 30.4/10.93 |
| ResNet [44] | 27.22 | 2.01 | **25.03/7.76** |
| DenseNet(k=12) [6] | 20.20 | **1.67** | **25.02/7.71** |
| **EffPNet (Best)** | 18.49 | 1.82 | 27.01/9.28 |
| **EffPNet (10 runs)** | 18.70±0.1620 | 1.85±0.0273 | – |

The experimental results indicate the possibility of transferring a searched block from one data set to another data set, which is consistent with NASNet [11] that inspires this research. There could be two parts of the proposed method that contribute to the transferability. First, the block evolved from the original data set was the best one selected from hundreds of blocks, which could have the ability to capture important features from images. Second, the stacking process described in Section III-H stacked the evolved block various times to produce a set of CNNs with various model sizes, which were then evaluated based on the training set of the target domain. Only the best CNN was selected, so the stacking process could benefit the transferability of the evolved block by selecting a proper model size on the target training data set.

## B. Convergence Analysis

The convergence of the particle's position is investigated. An example of the particle's position convergence is shown in Fig. 5(a) and (b). Two principal components are extracted by applying principal component analysis (PCA) [50], [51] on




(a) (b)

Fig. 5. Position convergence analysis. (a) First component of converged particle. (b) Second component of converged particle.

the data comprised of the position vectors of all the particles. The movements of the particle's position during the whole evolutionary process are drawn based on the first and second principal components, respectively. It is clear that the particle fluctuates at the beginning, but the fluctuation becomes smaller and smaller for both of the principal components. In the end, the fluctuation curve is almost flattened, which demonstrates that the particle's position converges gradually during the evolutionary process.

### *C. Analysis on Surrogate Assistance*

*1) Surrogate Model Performance:* Section III-D describes that the task of predicting the performance of CNNs has been transformed into a binary classification task. The performance of the surrogate model can be measured by the classification accuracy, which reflects how accurate the predictions of the surrogate model are. The promising performance is supported by plotting the classification accuracy of the surrogate model across the generations of the evolutionary process, as shown in Fig. 6. The high accuracy of more than 90% demonstrates that the surrogate model can distinguish the better CNN between a pair of given CNNs, which can efficiently and effectively assist the fitness evaluation. The very small standard deviation shows that the surrogate model consistently performs well across all of the generations, which proves its great stability during the whole evolutionary process. Therefore, the surrogate model can assist the fitness evaluation effectively during the entire search process. Another observation is that the surrogate model performs well even at the first generation with the constructed data set from training only 30 blocks, which indicates that a small number of training examples can fit the SVM classifier in the surrogate model reasonably well.

Besides the accuracy, another key metric of assessing the surrogate model is the number of particles filtered by the surrogate model. Before discussing the experimental results in this metric, two terms need to be defined first. First, a *filtered* particle is defined as the particle predicted by the surrogate model to underperform the particle's best-so-far position, which does not need to undergo the time-consuming CNN training process. The *trained* particle means the particle predicted to outperform its best-so-far position, which, therefore, has to be accessed by training the CNN. Fig. 7 shows the numbers of the filtered and trained particles at each generation. It can be seen that the number of trained particles exceeds the number of filtered particles only at the first several generations,

TABLE IV

SURROGATE MODEL ACCURACIES WITH VARIOUS COMBINATIONS OF FEATURES

| Features | Accuracy |
|---|---|
| losses | 82.02% |
| accuracies | 86.27% |
| block parameters | 70.60% |
| losses + accuracies | 86.96% |
| losses + accuracies + block parameters | 91.13% |

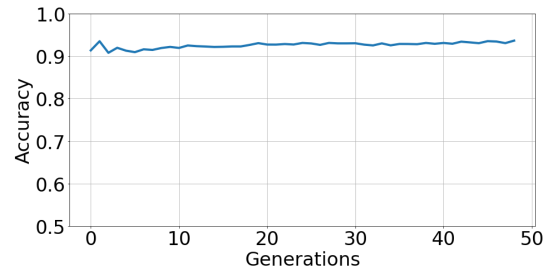



Fig. 6. Classification accuracies of the surrogate model across 50 generations. The statistical details of the classification accuracies of 50 generations are as follows—the mean accuracy is 92.63%, the standard deviation is 0.66%, the minimum accuracy is 90.81%, and the maximum accuracy is 93.68%.

but for the majority of the generations, the filtered particles significantly outnumber the trained particles. Fig. 8 shows the percentages of filtered and trained particles, respectively, in total, which shows that 80.1% of the particles have been filtered by the surrogate model. Therefore, the surrogate model has successfully sped up the fitness evaluation by preventing 80.1% of the particles from being trained by the time-consuming CNN training process.

Another simple approach of accelerating the fitness evaluation is to train the CNNs represented by the particles for a small number of epochs, e.g., ten epochs, and use the accuracy at the tenth epoch as the fitness value [52], [53]. Based on the above-constructed data set, an evaluation of using the accuracy at the tenth epoch as the sole indicator of the CNN's performance is done, which only achieves an accuracy of 69.94%. Considering that this is a binary classification problem, the method of solely utilizing the accuracy of the tenth epoch may be able to indicate the final classification accuracy, but the accuracy is not good. Instead, the surrogate model has consistently achieved an accuracy of more than 90% during the whole evolutionary process, so the surrogate-assisted fitness evaluation can be deemed as a reliable approach to speed up the fitness evaluation process.

*2) Surrogate Model With Different Feature Combinations:* To analyze the features of the data used to train the surrogate model, the performances of the surrogate model with various combinations of features are evaluated and listed in Table IV. A few interesting points can be observed from the table. First, the surrogate model with only the first ten losses or only the first ten accuracies achieves more than 10% accuracy than only using the block parameters. This indicates that the losses and accuracies are more important features than the block parameters. Second, despite good accuracies achieved by using

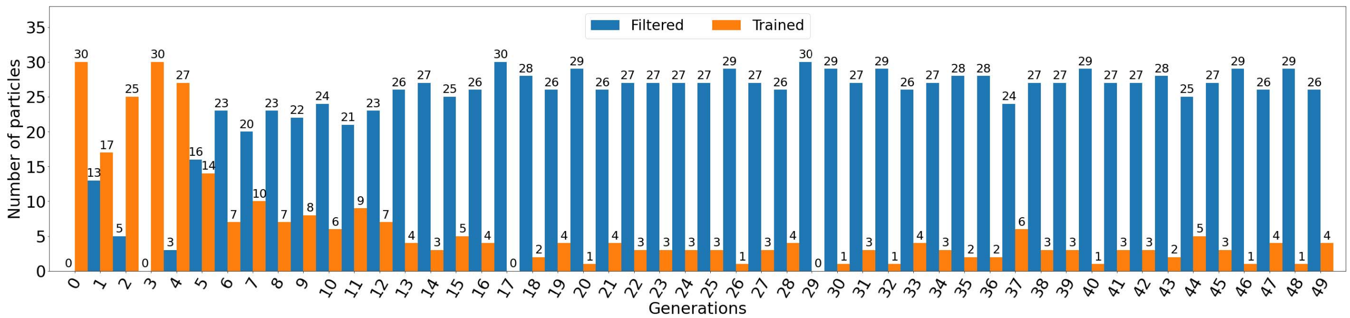



Fig. 7. Surrogate-assisted fitness evaluation across generations. The blue bars represent the number of particles filtered by the surrogate model in each generation, wheresas the orange bars show the number of particles trained by SGD.

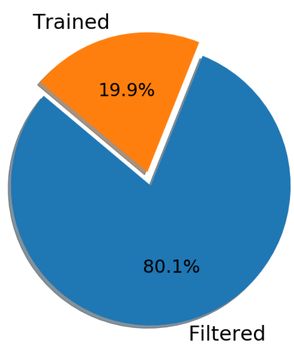



Fig. 8. Surrogate-assisted fitness evaluation summary. The blue pie represents the percentage of particles filtered by the surrogate model in total, whereas the orange pie indicates the percentage of particles trained by SGD in total.

the losses and accuracies individually, when combining the losses and accuracies as the features for the surrogate model, the improvement is tiny with only 0.69% more accurate than solely taking the accuracies as the features. It implies that these two sets of features may be redundant. Finally, it can be found that there is a decent improvement of the accuracy by combining the block parameters with the losses and accuracies, which is almost 4.17% more accurate than using the features without the block parameters. Since the combination of using all features shows the best performance, the surrogate model in the proposed method has chosen the proper features in terms of achieving the best classification accuracy.

*3) Data Analysis of Block Training History:* To further obtain an insight on how the features used for the surrogate model affect the classification accuracy of the dense block, the parameters of the evolved block, the losses and accuracies of the first ten epochs, and the best accuracy during the whole training process are extracted. Therefore, it is feasible to discover the patterns of how the best classification accuracy is influenced by the block parameters, and the performance trend including the losses and the accuracies of the first ten epochs. The best classification accuracy is the dependent variable and the remaining features are the explanatory variables. The most straightforward method of analyzing data is to visualize them, so the dimensionality of the explanatory variables needs to be reduced to 2-D or 3-D. As the purpose of the analysis is to find the importance of the explanatory variables, the variance is important for keeping the feature information during the dimensionality reduction. Thus, PCA [50], [51] is adopted because it is the most popular multivariate statistical technique, which was designed to reduce the dimensionality by extracting the most important information. Once the visualization is done, it would be more explainable to detect the pattern of the data points. A simple support vector regression (SVR) method [20], [54] is chosen to discover the pattern, i.e., in Fig. 9, an SVR model is fit by the blue points in each of the subfigures and the trained SVR model draws the red line as the discovered pattern.

Fig. 9 plots the first principal component (PC1) versus the classification accuracy. It can be observed from Fig. 9(a), a general pattern for the PC1 of the first ten losses is presented because the blue points gather near the red line. A kind of correlation between the $x$-axis and the $y$-axis can be observed, where the points closer to the left side of the $x$-axis tend to achieve higher $y$ values. By examining Fig. 9(b), similarly, a general pattern of the PC1 is also detected. It can be seen that the similar patterns are found between Fig. 9(a) and (b), which makes sense because the accuracy increases, while the training losses decreases during the neural network training process by SGD algorithms [30], [31], especially at the early stage without overfitting issues. Fig. 9(c) does not show a clear pattern for the PC1 because the red line cannot match the data points well. To summarize, the performance trend carried by the losses and accuracies from the first ten epochs is crucial features to predict the final performance of the dense block, while the block parameters are less important features, but they may be used as assistant features to improve the accuracy of predicting the final performance. This is also consistent with the performance of the surrogate model with various feature combinations discussed in Section V-C2, where it showed the losses and the accuracies were more important features than the block parameters.

## D. Growth Rate Analysis

To explore the impact of various growth rates for each layer, the distribution of the accuracy and the growth rates of the evaluated blocks represented by the particles is drawn in Fig. 10. Fig. 10(a) outlines the accuracy distribution of the evaluated blocks. It is obvious that the area with higher accuracies is explored much more than other space, especially the search space where the accuracy is around 0.6. This indicates

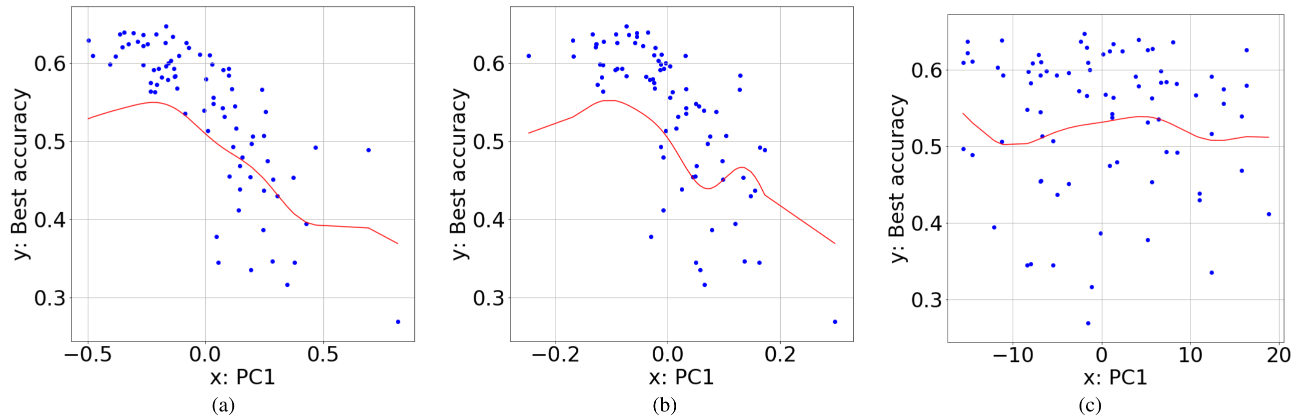



Fig. 9. $x$-axis is the PC1 and the $y$-axis is the best classification accuracy. The red line is the pattern discovered by the SVR model. (a). Extract the first ten losses from the *block training history* and apply PCA on the extracted data to achieve the PC1. (b). Extract the first ten accuracies from the *block training history* and apply PCA on the extracted data to achieve the PC1. (c). Extract the 16 growth rates from the *block training history* and apply PCA on the extracted data to achieve the PC1.

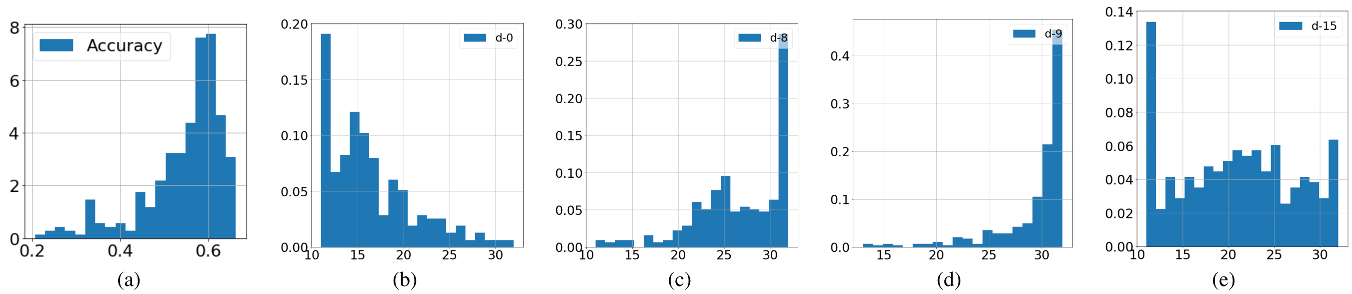



Fig. 10. Distribution of accuracy and growth rates. All of the subfigures are histogram plots with 20 bins. (a) $x$-axis is the accuracy and $y$-axis is the number of evaluated blocks, whose accuracies fall into the range of the accuracy bin. (b)–(e) $x$-axis is the growth rate and $y$-axis is the number of evaluated blocks, whose growth rates fall into the growth rate bin.

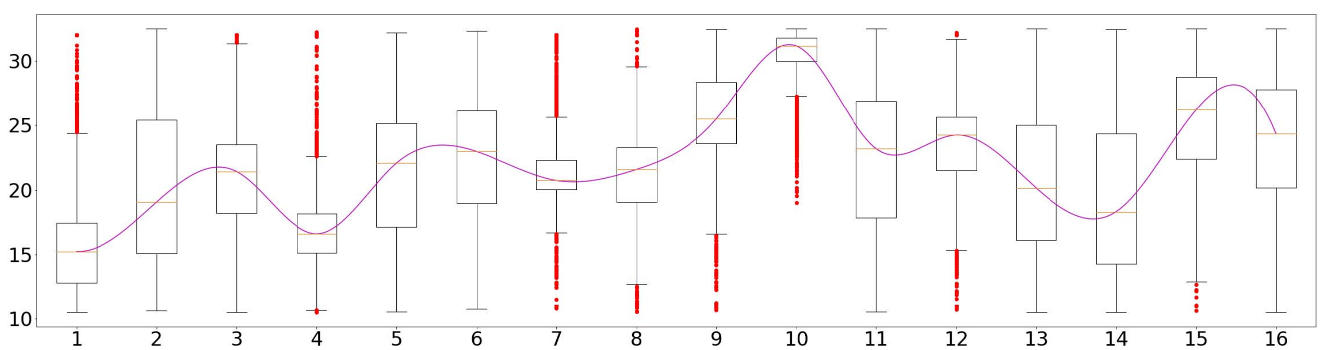


Fig. 11. Box plot of the growth rates for each layer of all of the evaluated blocks. The $x$-axis exhibits the layers in the dense block from the first layer to the 16th layer. The $y$-axis indicates the growth rates of the corresponding layers. The curved line connects the median values of the growth rates of each of the layers.

that the blocks with higher accuracies are well explored by the proposed method. Looking at the distribution of the growth rates for the first and last layers, respectively, as shown in Fig. 10(b) and (e), the bar with the smallest growth rate is longer than others, while, for the two middle layers shown in Fig. 10(c) and (d), the largest growth rate significantly outnumbers others. Therefore, for the blocks with higher accuracies dominating the distribution, different layers tend to prefer different growth rates. In addition to the distribution, the shape of the curved line drawn in Fig. 11 shows that different growth rates have been chosen for different layers. The first layer has the smallest median value and the tenth layer obtains the largest median value, which implies that the middle layer needs a larger growth rate than the other layers toward the first or the last layer. To conclude, the proposed method has searched the areas of the search space, where the classification accuracy tends to be better, and it is meaningful to optimize the growth rates for each layer in the dense blocks instead of using a fixed growth rate for all as proposed in the original DenseNet paper.

## VI. Conclusion and Future Work

In conclusion, a new surrogate-assisted PSO method has been proposed to effectively and efficiently evolve CNN blocks that are transferable to different domains in an automatic manner. This is supported by the experimental results, which have demonstrated that the proposed method can efficiently learn an effective block from the CIFAR-10 data set by achieving promising performance on CIFAR-10. Furthermore, the evolved blocks have exhibited their transferability by achieving competitive classification accuracies on the CIFAR-100, SVHN, and ImageNet data sets. To achieve the promising performance in terms of both classification accuracy and computational cost, first, a surrogate model and a surrogate data set were proposed to significantly accelerate the fitness evaluations. The reliability of the surrogate model and the surrogate data set has been upheld by visualizing and analyzing the accuracy and decision boundary of the surrogate model. Second, the surrogate model and surrogate data set were integrated into the PSO algorithm to form a surrogate-assisted PSO to achieve the goal of efficiently searching for optimal blocks. Finally, an encoding strategy to accommodate various growth rates of different layers in variable-length blocks was proposed, and the growth rate of each layer in the block was analyzed to show the necessity of having different growth rates at different layers of the block. Overall, the goal of this article and the specific objectives has been fulfilled.

This work demonstrates the potential of efficiently learning a dense block, which could be effectively transferred to other domains. However, the prior knowledge of DenseNet is utilized and the search space is restricted to search for hyperparameters of a dense block. It would be interesting to explore CNN architectures in a more flexible search space without any prior knowledge. For example, the search space could be composed of only the basic convolutional and pooling layers and the connections in the block could be sparsely or densely connected. In addition, the proposed method used a feed-forward fashion to stack the evolved block at the last step, but the shortcut connections could also be used. By loosening the restrictions of the search space, the major advantage is to enable the possibility of discovering unknown CNN architectures, which could outperform the state-of-the-art CNNs. However, the biggest challenge is the efficiency of the search algorithms. Developing an effective and efficient method to explore CNN architectures in a large search space could be very attractive research in the future.

## References


[1] Y. Zhao, K. Hao, H. He, X. Tang, and B. Wei, "A visual long-short-term memory based integrated CNN model for fabric defect image classification," *Neurocomputing*, vol. 380, pp. 259–270, Mar. 2020.

[2] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015. [Online]. Available: http://arxiv.org/abs/1409.1556

[5] 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.

[6] 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, pp. 4700–4708.

[7] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2016, p. 87.

[8] D. Han, J. Kim, and J. Kim, "Deep pyramidal residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5927–5935.

[9] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2017. [Online]. Available: https://arxiv.org/abs/1611.01578

[10] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through hypernetworks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018. [Online]. Available: http://arxiv.org/abs/1708.05344

[11] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710.

[12] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1379–1388.

[13] B. Wang, Y. Sun, B. Xue, and M. Zhang, "A hybrid differential evolution approach to designing deep convolutional neural networks for image classification," in *Proc. Australas. Joint Conf. Artif. Intell.* Wellington, New Zealand: Springer, 2018, pp. 237–250.

[14] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2019, pp. 4780–4789.

[15] B. Wang, B. Xue, and M. Zhang, "Particle swarm optimisation for evolving deep neural networks for image classification by evolving and stacking transferable blocks," 2019, *arXiv:1907.12659*. [Online]. Available: http://arxiv.org/abs/1907.12659

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.

[17] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, 1995, pp. 39–43.

[18] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, May 1998, pp. 69–73.

[19] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[20] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.

[21] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6550

[22] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, vol. 37, 2015, pp. 448–456.

[23] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc. IEEE Swarm Intell. Symp.*, Apr. 2007, pp. 120–127.

[24] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 19–34.

[25] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2423–2432.

[26] E. Real *et al.*, "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 2902–2911.

[27] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2017, pp. 497–504.

[28] J. Ren, Z. Li, J. Yang, N. Xu, T. Yang, and D. J. Foran, "EIGEN: Ecologically-inspired GENetic approach for neural network structure searching from scratch," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9059–9068.

[29] 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.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[31] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. ICML*, 2013, vol. 28, no. 5, pp. 1139–1147.

[32] B. Wang, B. Xue, and M. Zhang, "Particle swarm optimization for evolving deep convolutional neural networks for image classification: Single- and multi-objective approaches," in *Deep Neural Evolution*. Singapore: Springer, 2020, pp. 155–184.

[33] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," M.S. thesis, Citeseer, Toronto Univ., Toronto, ON, Canada, 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[34] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," Stanford Univ., Stanford, CA, USA, Tech. Rep., 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[35] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2787–2794.

[36] T. Elsken, J. H. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1711.04528

[37] Y. Chen *et al.*, "RENAS: Reinforced evolutionary neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4787–4796.

[38] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "Evolving the topology of large scale deep neural networks," in *Proc. Eur. Conf. Genetic Program.* Parma, Italy: Springer, 2018, pp. 19–34.

[39] R. Miikkulainen *et al.*, "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Amsterdam, The Netherlands: Elsevier, 2019, pp. 293–312.

[40] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, 2014. [Online]. Available: http://arxiv.org/abs/1312.4400

[41] Y. Pang, M. Sun, X. Jiang, and X. Li, "Convolution in convolution for network in network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1587–1597, May 2018.

[42] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. Artif. Intell. Statist.*, 2015, pp. 562–570.

[43] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR)*, 2017. [Online]. Available: https://arxiv.org/abs/1605.07648

[44] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.* Amsterdam, The Netherlands: Springer, 2016, pp. 646–661.

[45] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep neural networks by multi-objective particle swarm optimization for image classification," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2019, pp. 490–498.

[46] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proc. Int. Conf. Evol. Program.* San Diego, CA, USA: Springer, 1998, pp. 591–600.

[47] I. C. Trelea, "The particle swarm optimization algorithm: Convergence analysis and parameter selection," *Inf. Process. Lett.*, vol. 85, no. 6, pp. 317–325, Mar. 2003.

[48] F. Vandenbergh and A. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Inf. Sci.*, vol. 176, no. 8, pp. 937–971, Apr. 2006.

[49] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1711.00436

[50] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 4, pp. 433–459, 2010.

[51] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.

[52] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8.

[53] B. Wang, B. Xue, and M. Zhang, "A hybrid GA-PSO method for evolving architecture and short connections of deep convolutional neural networks," in *PRICAI 2019: Trends in Artificial Intelligence*. Cuvu, Fiji: Springer, 2019, pp. 650–663.

[54] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004.


**Bin Wang** (Student Member, IEEE) received the B.E. degree from the Heilongjiang University of Science and Technology, Harbin, Heilongjiang, China, in 2008, and the B.Sc. degree (Hons.) from the Victoria University of Wellington, Wellington, New Zealand, in 2019, where he is currently pursuing the Ph.D. degree in computer science with the School of Engineering and Computer Science.

Mr. Wang has been severing as a Reviewer for top international journals, such as the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and *IEEE Computational Intelligence Magazine*.

**Bing Xue** (Member, IEEE) received the B.Sc. degree from the Henan University of Economics and Law, Zhengzhou, China, in 2007, the M.Sc. degree in management from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree in computer science from the Victoria University of Wellington (VUW), Wellington, New Zealand, in 2014.

She is currently a Professor in computer science and the Program Director of Science with the School of Engineering and Computer Science, VUW. She has over 200 papers published in fully refereed international journals and conferences. Her research focuses mainly on evolutionary computation, machine learning, classification, symbolic regression, feature selection, evolving deep neural networks, image analysis, transfer learning, and multiobjective machine learning.

Prof. Xue is the Vice-Chair of the IEEE Computational Intelligence Society (CIS) Evolutionary Computation Technical Committee, the IEEE Task Force on Evolutionary Feature Selection and Construction, the IEEE CIS Task Force on Transfer Learning and Transfer Optimization, and IEEE CIS Task Force on Evolutionary Deep Learning and Applications. She was the Chair of the Data Mining and Big Data Analytics Technical Committee. She has also served as an Associate Editor for several international journals, such as *IEEE Computational Intelligence Magazine* and IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

**Mengjie Zhang** (Fellow, IEEE) received the B.E. and M.E. degrees from the Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, Hebei, China, in 1989 and 1992, respectively, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 2000.

He is currently a Professor of computer science, the Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) of the Faculty of Engineering. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multiobjective optimization, feature selection and reduction, job shop scheduling, and transfer learning. He has published over 500 research papers in refereed international journals and conferences.

Prof. Zhang is a fellow of the Royal Society of New Zealand and has been a Panel Member of the Marsden Fund (New Zealand Government Funding), and a member of the Association for Computing Machinery. He was the Chair of the IEEE CIS Intelligent Systems and Applications Technical Committee and the IEEE CIS Emergent Technologies Technical Committee and the Evolutionary Computation Technical Committee and a member of the IEEE CIS Award Committee. He is also a Vice-Chair of the Task Force on Evolutionary Computer Vision and Image Processing and the Founding Chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a Committee Member of the IEEE NZ Central Section.