A Hybrid GA-PSO Method for Evolving Architecture and Short Connections of Deep Convolutional Neural Networks

Bin Wang, Yanan Sun, Bing Xue, and Mengjie Zhang

School of Engineering and Computer Science
Victoria University of Wellington, PO Box 600, Wellington 6140, NEW ZEALAND
`bin.wang, bing.xue, yanan.sun, mengjie.zhang}@ecs.vuw.ac.nz`

Abstract. Image classification is a difficult machine learning task, where Convolutional Neural Networks (CNNs) have been applied for over 20 years in order to solve the problem. In recent years, instead of the traditional way of only connecting the current layer with its next layer, shortcut connections have been proposed to connect the current layer with its forward layers apart from its next layer, which has been proved to be able to facilitate the training process of deep CNNs. However, there are various ways to build the shortcut connections, it is hard to manually design the best shortcut connections when solving a particular problem, especially given the design of the network architecture is already very challenging. In this paper, a hybrid evolutionary computation (EC) method is proposed to *automatically* evolve both the architecture of deep CNNs and the shortcut connections. Three major contributions of this work are: Firstly, a new encoding strategy is proposed to encode a CNN, where the architecture and the shortcut connections are encoded separately; Secondly, a hybrid two-level EC method, which combines particle swarm optimisation and genetic algorithms, is developed to search for the optimal CNNs; Lastly, an adjustable learning rate is introduced for the fitness evaluations, which provides a better learning rate for the training process given a fixed number of epochs. The proposed algorithm is evaluated on three widely used benchmark datasets of image classification and compared with 12 peer Non-EC based competitors and one EC based competitor. The experimental results demonstrate that the proposed method outperforms all of the peer competitors in terms of classification accuracy.

Keywords: Evolutionary Computation · Image Classification · Convolutional Neural Networks · Shortcut Connections.

1 Introduction

Deep Convolutional Neural Networks (CNNs) have been the leading approach for solving image classifications tasks since it was introduced around 30 years ago [12]. Various CNN methods have been developed, e.g. VGGNet [16], Xception [2] and GoogLeNet [21]. Deep CNNs have achieved better and better accuracy on image classification tasks. However, the architectures of CNNs grow deeper and deeper (i.e. more and more layers), which makes the training of deep CNNs much harder due to the difficulty in the CNNs *architecture design* and *network training*.

Almost all of the state-of-the-art CNNs are with a manually designed architecture, which is very challenging to achieve without expertise both in CNNs and domain knowledge on the target problem. However, most real-world users often do not have such knowledge. In recent years, evolutionary computation (EC) has shown to be effective in *automatically* searching for the optimal architecture of CNNs [13] [24] [20].

Back-propagation with gradient descent optimisation is the most commonly used method for training CNNs, but the vanishing gradients problem often occurs when training a deep CNN [1] [18]. Recently, *shortcut connections* have been introduced and shown to be effective in dealing with this problem [15]. Shortcut connections add extra connections between the current layer and the forward layers. Typical examples are ResNet [6] as shown in Fig. 1 and the densely-connected shortcuts in DenseNet [7] as illustrated in Fig. 2. As can be seen from Fig. 1, in ResNet, along with the direct forward connections between the current layer and the next layer, there are also jump connections, which connect the current layer to the layer after the next layer. DenseNet divides the CNN architecture into a number of blocks. Each layer can be connected to all of the forward layers of the same block, which is called densely-connected structure. Such shortcut connections have been heavily investigated in recent years with different variants [15] [25]. However, such shortcut connections are manually designed and there still are a large number of open questions. For example, although the operations after shortcut connections are addition in ResNet and concatenation in DenseNet, it is unclear whether the shortcut connections in ResNet with the concatenation mechanism is better than DenseNet. Without rich expertise, it is still challenging to design the best shortcut connections to effectively and efficiently address a given problem. Therefore, it is needed to develop an approach to automatically searching for the shortcut connections.

Goals: we aim to develop a novel EC based approach that can automatically find the optimal CNN architecture and decide whether there should be shortcut connection(s) between one layer and its forward layer(s). A two-level encoding strategy is proposed, which is then used by a hybrid EC method of a genetic algorithm (GA) and particle swarm optimisation (PSO) to evolve both the network architecture and shortcut connections. Since both the architecture and the shortcut connections are dynamically decided during the evolutionary process without any human interference, the proposed method is named *DynamicNet*. The proposed method will be examined and compared with one EC based method and 12 state-of-the-art non-EC based methods on three of the widely-used datasets having different levels of difficulties. The specific objectives and contributions are:

- Design a new encoding strategy that includes both the CNN architecture and the shortcut connections. Since the CNN architecture is decisive to the classification accuracy and the shortcut connections impact how well the

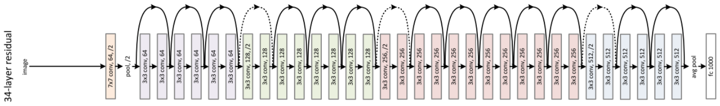


Fig. 1: ResNet architecture (image taken from [6])

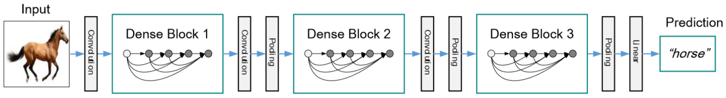


Fig. 2: DenseNet architecture (image taken from [7])

CNN can be trained, a two-level encoding is proposed with the first level representing the CNN architecture and the second level representing the shortcut connections. These two levels are encoded as a vector with decimal values and a vector of binary values, respectively;

- Develop a hybrid algorithm that can work with the two-level encoding. A variable-length PSO algorithm is proposed to evolve the CNN architectures due to PSO's promising performance on continuous optimisation while GA is used to evolve the shortcut connections since it works well on optimisation tasks with binary values;
- Propose a new fitness evaluation method to improve the effectiveness and efficiency of the encoded CNN. Classification accuracy is used as the fitness value of the proposed method. Each evaluation requires to train the encoded CNN, which is an expensive process. Motivated by previous work [19], a small number of training epochs is used to speed up the training. Furthermore, an automation method is developed to search for the best learning rate among a sequence of learning rates to improve the classification accuracy.

2 Background

2.1 ResNet

As shown in Fig. 1, the architecture is built on a plain CNN architecture called VGG nets [17], which mostly contains convolutional layers with 3×3 filters; while by inserting shortcut connections, the plain architecture is turned into the recently proposed ResNet. The output is calculated based on Equation (1), where x is the input, $\mathcal{F}(x, W_i)$ represents the output of the convolutional layer with the weights W_i, and W_s can be a constant of $\mathbf{1}$ if the dimension of the input is identical to that of the output of the convolutional layer; otherwise it will be a linear projection of the input in order to match the dimension of the output of the convolutional layer.

$$y = \mathcal{F}(x, W_i) + W_s x \tag{1}$$

2.2 DenseNet

DenseNet is a newly proposed CNN architecture in image classification tasks. As shown in Fig. 2, a DenseNet is composed of several dense blocks, and the convolutional layer and the pooling layer between the dense blocks which are referred to as the transition layer. To be more specific with the dense block, suppose a single image x_0 is passed to a dense block, which is composed of L layers. Each of the L layers implements a non-linear transformation $H_l(\cdot)$, and the output of the l^{th} layer is denoted as x_l. As the output of the l^{th} layer receives all of the feature maps of all preceding layers, the output x_l can be calculated according to Formula (2), where $[x_0, x_1, ..., x_{l-1}]$ refers to the concatenation of the feature maps obtained from layer 0, 1, ..., $l-1$, and H_l represents a composite function of three consecutive operations, which are batch normalization (BN) [8], a rectified linear unit (ReLU) [5] and 3 × 3 convolution (Conv).

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]) \tag{2}$$

2.3 GAs and PSO

GAs As an EC approach, GAs are inspired by the process of natural selection. The bio-inspired operators, such as mutation, crossover and selection, are utilised to evolve the population in order to obtain a high-quality solution [14]. The procedure of GA is composed of five parts: initialisation, selection, fitness evaluation, mutation, and crossover. At the stage of initialisation, a population of random vectors with a fixed dimension is generated; Next, the selection is performed by using a selection algorithm to select the individuals into a mating pool; After that, mutation is performed by selecting one individual from the mating pool and the value of each dimension is randomly chosen to be changed to evolve a new individual; Crossover is performed by selecting two individuals in the mating pool and combining a part of one individual's vector with that of the other. By iterating the fitness evaluation, selection, mutation, and crossover, the new population can be filled with new individuals with hopefully better solutions. The whole process terminates when the stopping criteria are met, and the best individual of all generations is reported as the evolved solution.

$$v_{id}(t+1) = w * v_{id}(t) + c_1 * r_1 * (P_{id} - x_{id}(t)) + c_2 * r_2 * (P_{gd} - x_{id}(t)) \tag{3}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{4}$$

PSO As one of the EC approaches, PSO is motivated by the social behaviour of fish schooling or bird flocking [9] [4]. In PSO, there is a population consisting of a number of candidate solutions also called particles, and each particle has a position and a velocity. The representation of the position is $\mathbf{x_i} = (x_{i1}, x_{i2}, ...x_{id})$, where $\mathbf{x_i}$ is a vector of a fixed dimension representing the position of the ith particle in the population and x_{id} means the dth dimension of the ith particle's position. $\mathbf{v_i} = (v_{i1}, v_{i2}, ...v_{id})$ illustrates the velocity of a particle, where $\mathbf{v_i}$ is a fix-length vector expressing the velocity of the ith particle and v_{id} means the dth dimension of the ith particle's velocity. The way that PSO solves the optimisation problems is to keep moving the particle to a new position in the search space until the stopping criteria are met. The position of the particle is updated according to the update equation which incorporates two equations - the velocity update equation 3 and the position update Equation (4). In Formula (3), $v_{id}(t+1)$ indicates the updated dth dimension of the ith particle's velocity, r_1 and r_2 carry random numbers between 0 and 1, w, c_1 and c_2 are PSO parameters that are used to fine-tune the performance of PSO, and P_{id} and P_{gd} bear the dth

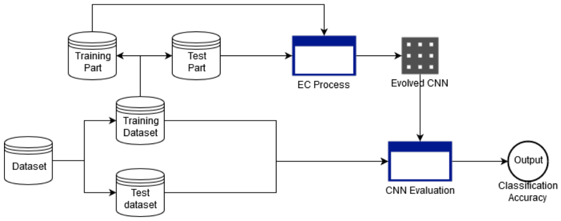


Fig. 3: The flowchart of the experimental process

dimension of the local best and the global best, respectively. After updating the velocity of the particle, the new position can be achieved by applying Formula (4).

3 The Proposed Method

3.1 Overall Structure of the System

Fig. 3 shows the overall structure of the system (this structure is actually used by all of the experiments in this paper). The dataset is split into a training set and a test set, and the training set is further divided into a training part and a test part. The training part and the test part are passed to the EC process, which is the HGAPSO algorithm. During the fitness evaluations, the training part is used to train the neural network, and the test part is used to obtain the test accuracy of the trained neural network, which is used as the fitness value. EC produces the evolved CNN architecture, which is the best individual. Lastly, in the CNN evaluation procedure, the produced CNN architecture is trained on the whole training set, and the test accuracy of the trained CNN model is obtained, which is the final output of the system.

3.2 DynamicNet - The Evolved CNN Architecture

By comparing the figures of ResNet and DenseNet, it can be observed that in ResNet, each layer has at most two connections from previous layers. However, in DenseNet, the connections of each layer coming from previous layers are the number of previous layers due to the densely-connected structure. Therefore, the number of input feature maps is the sum of the numbers of feature maps of all previous layers, which results in the exploding growth of the number of feature maps, particularly for the layers near the output layer. The solution introduced in DenseNet is to divide the whole CNN into multiple blocks called Dense Block. Each block is followed by a transition layer, which comprises a convolutional layer and a pooling layer, to reduce the number of feature maps to half the number of input feature maps. The hyperparameters of the convolutional layer are fixed, which are 3 as the filter size, 1 as the stride size, and half the number of input feature maps as the number of feature maps; The pooling layer also has fixed hyperparameters, which are 2×2 as the kernel size and 2 as the stride size. As the proposed DynamicNet may be densely-connected, it might have the same exploding growth issue of the number of feature maps. Therefore, DynamicNet adopts the block mechanism of DenseNet.

Inside each block, there are a number of convolutional layers with a fixed filter size of 3×3 and a fixed stride size of 1. After each layer, the total number of input feature maps grows by the number of feature maps of the convolutional layer, which is called the *growth rate* of the block. In DenseNet, *the number of blocks*, *the number of convolutional layers* and *t*he growth rate are manually designed, which requires good domain knowledge and a lot of manual trials to find a good architecture. In the proposed HGAPSO algorithm, these three hyperparameters will be also automatically designed.

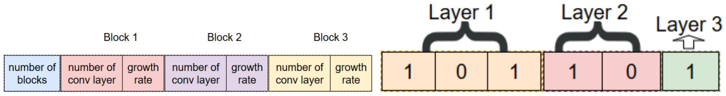
3.3 HGAPSO Encoding Strategy

DynamicNet is comprised of a number of blocks which are connected by transition layers, and the shortcut connections are built between layers inside the block. Based on the construction pattern of the network, the hyperparameters of the architecture can be split into the architecture and the shortcut connections. Regarding the architecture of the network, there are various hyperparameters including the number of blocks, the number of convolutional layers in each block and the growth rate of the convolutional layer in the block, which need to be evolved. In addition to the densely-connected structure in DenseNet, different topologies of shortcut connections, i.e. the different combination of partial shortcut connections in each block, will be explored by the proposed HGAPSO method in order to keep the meaningful features and remove the unmeaningful features learned by previous layers.

Based on the analysis of the architecture and hyperparameters, the encoding process can be divided into two steps. The first step is to encode the hyperparameters of the CNN architecture. Each of the hyperparameters is a dimension of the architecture encoding, which is shown in Fig. 4a. The first dimension is the number of blocks, and the two hyperparameters of each block, the number of convolutional layers and the growth rate, as two dimensions are appended to the vector. The first step of the encoding is named the first-level encoding, which will be used by the first-level evolution. Based on the results of the first-level encoding, the shortcut connections can be encoded into a binary vector at the second-level encoding. An example of one block with 5 layers is illustrated in Fig. 4b. Each of the dimensions represents a shortcut connection between two layers that are not next to each other, and the two layers next to each other are always connected. Taking the first layer as an example, the three binary digits - [101] represents the shortcut connections between the first layer to the third, to the fourth, and to the fifth layer, respectively, where 1 means the connection exists and 0 means there is no connection. A number of similar binary vectors shown in Fig. 4b constitute the whole vector that represents the shortcut connections of the whole block.

3.4 HGAPSO Search

Overview Based on the two-level encoding strategy, the algorithm is composed of two levels of evolution, which are described in Algorithm 1. The first-level evolution is designed to evolve the architecture of the CNNs encoded by the first-level encoding, and the second-level evolution is performed to search for the



(a) First-level encoding (b) Second-level encoding

Fig. 4: HGAPSO encoding

best combination of shortcut connections. There are a couple of reasons to separate the architecture evolution from the evolution of the shortcut-connection combination. First of all, since the architecture and the shortcut connections play different roles in the performance of CNNs, which are that the architecture including the depth and the width of the CNNs represents the capacity of network and the shortcut connections are to facilitate the training process of the network, the training process is only comparable when the architecture is fixed, which inspires the idea of splitting the evolution to two levels. Secondly, if the hyperparameters of both the architecture and the shortcut connections are combined into one encoded vector, it will bring some uncertainties to the search space, which, therefore, may deteriorate the complex search space by introducing more disturbance to the search space.

Algorithm 1: Framework of HGAPSO

```
$P \leftarrow$ Initialize the population with first-level encoding elaborated in Section 3.3;
$P_{best} \leftarrow$ Empty PSO Personal Best;
$G_{best} \leftarrow$ Empty PSO global best;
while first-level termination criterion is not satisfied do
    $P \leftarrow$ Update the population with first-level PSO evolution described in Section 3.4;
    for particle $ind$ in population $P$ do
        $P\_sub \leftarrow$ Initialize the population with second-level encoding based on the value of $ind$ illustrated in Section 3.3;
        while second-level termination criterion is not satisfied do
            $P\_sub \leftarrow$ Update the population with second-level GA evolution described in Section 3.4;
            evaluate the fitness value of each individual;
            $P\_sub_{best} \leftarrow$ retrieve the best individual in $P\_sub$;
        end while
        Update $P_{best}$ if $P\_sub_{best}$ is better than $P_{best}$;
    end for
    $G_{best} \leftarrow$ retrieve the best individual in $P$;
end while
```

It is arguable that the computational cost of the two-level evolution may be high, but the two-level encoding strategy divides the complex search space into two smaller search spaces and reduces the disturbance in the search space, so the two-level evolution we believe will not perform worse than searching for the optima in a much more complex search space. Other than that, as the second-level evolution of searching for the best combination of shortcut connections only depends on the specific architecture evolved in the first-level evolution, the second-level evolution can be done in parallel for each of the individual of the first-level evolution, which can dramatically speed up the process if sufficient hardware is available.

HGAPSO First-level PSO evolution Algorithm 2 shows the pseudo code of the PSO evolutionary process. Based on the encoded vector from the first-level encoding, the value of each dimension is a decimal value, and PSO has been proved to be effective and efficient in solving optimisation problems with decimal values, so PSO is chosen to perform the first-level search. However, the dimensionality of the encoded vector is not fixed, so an adapted variable-length PSO is proposed to solve this variable-length problem. Since the size of the input feature maps to each block is different and the specific block is trained and designed to learn meaningful features given the size of the input feature maps, when applying EC operators on two individuals, it is important to find the matched blocks which have the same size of input feature maps and apply the operators on the matched blocks. To be specific with the PSO evolution in HGAPSO, the length of the particle may be different from the length of the personal best and global best, so based on the blocks of the individual, the corresponding blocks in the personal best and the global best need to be matched by selecting the blocks with the same size of the output feature and the PSO algorithm is only applied on the matched blocks.

Algorithm 2: HGAPSO first-level PSO evolution

```
Input: The current particle $ind$, the personal best $P_{best}$, the global best $G_{best}$, the rate of
  changing the number of blocks $r_{cb}$;
  $rnd \leftarrow$ Generate a random number from a uniform distribution;
  Find the matched blocks of the particle $ind$ by comparing the feature map size;
  Update the velocity and position of the matched blocks of the particle $ind$ according to Equation
  3 and 4;
  if $rnd < r_{cb}$ then
      Update the velocity and position of the dimension of number of blocks of the particle $ind$
      according to Equation 3 and 4;
      Randomly cut or generate the blocks to the value of the number of blocks.
  end if
```

The first dimension of the vector represents the number of blocks. When the number of blocks changes, the depth of the CNN architectures changes, which achieves the ability to evolve the depth of the CNN architecture and keeping the diversity of the PSO population. However, the change of the number of blocks incurs a dramatic change to the CNN architecture, and if it changes too often, each CNN architecture evolution might be too short to achieve good performance, so it is better to leave the evolution some time to optimise other hyperparameters given the specific number of blocks. In order to keep the diversity of the number of blocks and reduce the disturbance caused by frequently changing the number of blocks, the rate of changing the number of blocks in the vector is introduced, which is a real value between [0, 1]. Therefore, the preference for diversity or stability depending on specific tasks can be controlled by tweaking the rate of changing the number of blocks.

When the number of blocks is changed, some blocks need to be randomly cut or randomly generated in order to meet the requirement of the number of blocks in the first dimension. For example, suppose the number of blocks is increased from 3 to 4, the hyperparameters of the fourth block need to be randomly generated based on the first-level encoding strategy, which then are appended to the vector of 3 blocks; On the other way around, assuming the number of blocks is decreased from 4 to 3, the last block is removed. In the proposed HGAPSO method, whenever removing a block(s), it always starts from the last layer because it does not affect the feature map sizes of the other blocks.

HGAPSO Second-level GA evolution According to the second-level encoding depicted in Section 3.3, once the CNN architecture is obtained from the first-level evolution, the dimensionality of the second-level encoding will be fixed, so the encoded vector can be represented by a fixed-length binary vector. Since GAs are good at optimising binary problems, a GA is chosen as the algorithm to perform the second-level evolution.

3.5 HGAPSO Fitness Evaluations

It can be observed from Algorithm 1 that fitness evaluation only takes place inside the second-level GA evolution, and the fitness of the best GA individual is used as the fitness of its corresponding particle of first-level PSO evolution. Backpropagation with Adam Optimiser [10] is used to train the network for a number of epochs on part of the training data. The accuracy of the trained CNN on the test part of the training data as the fitness value of the individual.

There are two hyperparameters for the fitness evaluations, which are the number of epochs and the initial learning rate of Adam Optimiser. In the experiment, 5 epochs are used by considering the hardware available and a fairly-short experimental time. After the number of epochs is chosen, DenseNet is used as a baseline to determine an initial learning rate for optimising a CNN with the given depth and width, i.e. after the architecture of the CNN determined, the network with fully-connected blocks are used to find a best initial learning rate among 0.9, 0.1 and 0.01.

In order to speed up the evolution process, a part of the training dataset is used for the second-level evolution because the second-level evolution consumes the most computation. While for the first-level evolution, as the computational cost is not that high, and in order to achieve a more stable performance given the architecture of a CNN, the full training dataset is used.

4 Experiment Design

4.1 Benchmark Datasets and State-of-the-art Competitors

Due to our limited hardware resource, the DECNN method proposed in [23], which only requires a few days running of the experiment on a single GPU, is chosen as the peer EC competitor instead of the method proposed in [26], which takes 28 days on 500 GPUs to obtain the final result. The state-of-the-art machine learning algorithms used to compare with DECNN are also used as the peer Non-EC competitors. As DECNN did not perform well on CONVEX benchmark dataset [11], CONVEX dataset is selected as one of the benchmark datasets to see if the proposed HGAPSO algorithm able to achieve better performance. Apart from the CONVEX dataset, the MB and MDRBI datasets [11] are also used as benchmark datasets to evaluate the proposed algorithm across different complexities, as MB is the simplest dataset among the MB variants, and MDRBI is the most complicated variant of the MB datasets. On MB, the images represent the handwritten digits from 0 to 9, and there are 12,000 training images and 5,000 test images; MDRBI contains the same amount of training and test images, but some noises are added to the original MB dataset. The CON-

Table 1: Parameter Settings

Parameter	Value
HGAPSO parameters	
the range of # of layers in each block	[4, 8]
the range of growth rate in each block	[8, 32]
population size	20
generation	10
PSO	

Parameter	Value
c_1, c_2	1.49618
w	0.7298
GA	
mutation rate	0.01
cross over rate	0.9
elitism rate	0.1

VEX dataset contains images with the shape of convex or non-convex, which are divided into the training dataset of 8,000 images and the test dataset of 5,000 images. Since EC methods are stochastic, the experiment will be run 30 times and statistical tests will be performed when comparing the proposed algorithm with its peer competitors.

As it would be more convincing to evaluate the proposed HGAPSO algorithm on larger datasets such as CIFAR-10, but the computational cost is too high, e.g. one run of HGAPSO on CIFAR-10 takes more than a week. Therefore, the experiment on CIFAR-10 will not be run for 30 times due to the very high computational cost, our limited GPU resource and the time constraint. Instead, only one run of the experiment will be performed in order to obtain an initial result, which gives suggestions on whether it is worth continuing the experiments for 30 runs in the future when more GPU resources are ready.

4.2 Parameter settings

All of the parameters are configured according to the conventions in the communities of PSO [22] and GAs [3] along with taking into account the computational cost and the complexity of the search space. The values of the parameters of the proposed algorithm are listed in Table 1.

5 Results and Discussions

5.1 HGAPSO vs. State-of-the-Art methods

The experimental results and the comparison between HGAPSO and the state-of-the-art methods are shown in Table 2. In order to clearly show the comparison results, the terms (+) and (-) are provided to indicate the result of HGAPSO is significantly better or worse than the best result obtained by the corresponding peer competitor. The term (–) means there are no available results reported from the provider or cannot be counted.

It can be observed that the proposed HGAPSO method achieves a significant improvement in terms of the error rates shown in Table 2. HGAPSO significantly outperforms the other peer competitors across all the three benchmark datasets. To be specific, it further reduces the error rate over the best competitor by 5%, 1% and 10% on the CONVEX, MB and MDRBI datasets, respectively.

5.2 HGAPSO vs. DECNN

In Table 2, it can be observed that by comparing the results between HGAPSO and DECNN, both the mean error rate and the standard deviation of HGAPSO

Table 2: Classification errors of HGAPSO and Competitors

Method	CONVEX		MB		MDRBI	
CAE-2		–	2.48	(+)	45.23	(+)
TIRBM		–		–	35.50	(+)
PGBM+DN-1		–		–	36.76	
ScatNet-2	6.50	(+)	1.27	(+)	50.48	(+)
RandNet-2	5.45	(+)	1.25	(+)	43.69	(+)
PCANet-2 (softmax)	4.19	(+)	1.40	(+)	35.86	(+)
LDANet-2	7.22	(+)	1.05	(+)	38.54	(+)
SVM+RBF	19.13	(+)	30.03	(+)	55.18	(+)
SVM+Poly	19.82	(+)	3.69	(+)	54.41	(+)
NNet	32.25	(+)	4.69	(+)	62.16	(+)
SAA-3	18.41	(+)	3.46	(+)	51.93	(+)
DBN-3	18.63	(+)	3.11	(+)	47.39	(+)
HGAPSO(best)	1.03		0.74		10.53	
HGAPSO(mean)	1.24		0.84		12.23	
HGAPSO(standard deviation)	0.10		0.07		0.86	
DECNN(mean)	11.19		1.46		37.55	
DECNN(standard deviation)	1.94		0.11		2.45	
P-value	**0.0001**		**0.0001**		**0.0001**	

are smaller than that of DECNN, and from the statistical point of view, HGAPSO has a significant improvement in terms of the classification accuracy.

5.3 Evolved CNN Architecture

After investigating the evolved CNN architectures, it is found that HGAPSO demonstrates its capability of evolving both the architecture of CNNs and the shortcut connections between layers. By looking into the evolved CNN architectures, it can be observed that not only the CNN architectures with various number of layers but also different topologies of shortcut connections are evolved. For example, one evolved CNN architecture has 3 blocks. In the first block, there are 4 convolutional layers, and [0, 0, 0, 0, 1], [0, 1, 0, 1], [0, 0, 1], [0, 0] and [1] represent the connections from the input, the first layer, the second layer, the third layer to the following layers, where 1 indicates the connection exists, and 0 means no connection; The second block is composed of 8 layers with the growth rate of 34, and the corresponding connections are [1, 0, 1, 0, 1, 0, 1, 0], [0, 1, 1, 1, 1, 0, 1], [1, 1, 1, 1, 1, 0], [1, 1, 1, 0, 1], [1, 0, 0, 0], [0, 0, 0], [1, 1] and [0]; In the third block, there are 5 layers with the corresponding connections of [0, 0, 1, 1, 0], [0, 0, 0, 0], [1, 0, 0], [0, 1] and [0], and the growth rate is 39.

5.4 One-run Result on CIFAR-10 dataset

As mentioned earlier, the computational cost of testing HGAPSO is extremely high. For one run of the experiment using one GPU card, it takes more than a week to evolve the CNN architecture, and it took almost 12 hours to train the evolved CNN architecture. The classification accuracy of the specific run is 95.75%, which ranks the second among the state-of-the-art deep neural networks ranging from 75.86% to 96.53% that are collected by the rodrigob website [1]; However, all of the state-of-the-art deep neural networks require very highly specialised domain knowledge and tremendous experiments to manually fine-tune the performance, while HGAPSO has the ability of automatically evolving the CNN architecture without any human interference, which is considered as the biggest advantage.

---

[1] http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html#43494641522d3130

6 Conclusions

This paper developed an EC based method for automatically evolving both the architecture of CNNs and shortcut connections, without human intervention or domain knowledge in either CNNs or the target problem. The proposed method outperforms both the EC competitor and the Non-EC competitors on commonly used benchmark datasets. The first reason is that by evolving shortcut connections, the feature maps learned in previous layers can be reused in further layers, which amplifies the leverage of useful knowledge; Secondly, the shortcut connections make the training of very deep neural networks more effectively by passing the gradients through shortcut connections, which has been proven by DenseNet [7]. Furthermore, the classification accuracy of HGAPSO on CIFAR-10 is promising as it is very competitive with the state-of-the-art deep neural networks. In addition, the most advantage of HGAPSO is that it does not require any human efforts to design the architecture of CNNs, which is usually required for the peer state-of-the-art competitors.

In regard to the future work, firstly, due to the hardware limitation, the proposed algorithm has been tested on relatively small datasets. It would be more convincing if the algorithms could be tested on other larger datasets such as ImageNet dataset; secondly, as there are more and more new CNN architectures proposed with better performance, it would be helpful to investigate more recent CNN architectures, based on which EC methods can be applied to automatically evolve more advanced CNN architectures.

References

1. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks **5**(2), 157–166 (1994)
2. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. arXiv preprint pp. 1610–02357 (2017)
3. Digalakis, J., Margaritis, K.: An experimental study of benchmarking functions for genetic algorithms. Proceedings of 2000 IEEE International Conference on Systems, Man and Cybernetics. (2000). https://doi.org/10.1109/icsmc.2000.886604
4. Eberhart, Shi, Y.: Particle swarm optimization: developments, applications and resources. In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546). vol. 1, pp. 81–86 vol. 1 (May 2001). https://doi.org/10.1109/CEC.2001.934374
5. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 315–323 (2011)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015), `http://arxiv.org/abs/1512.03385`
7. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. CoRR **abs/1608.06993** (2016), `http://arxiv.org/abs/1608.06993`
8. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Neural Networks, 1995. Proceedings., IEEE International Conference on. vol. 4, pp. 1942–1948 vol.4 (Nov 1995). https://doi.org/10.1109/ICNN.1995.488968

10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
11. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: Proceedings of the 24th international conference on Machine learning. pp. 473–480. ACM (2007)
12. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation **1**(4), 541–551 (1989)
13. Miller, J., Turner, A.: Cartesian genetic programming. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation. pp. 179–198. GECCO Companion '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2739482.2756571, `http://doi.acm.org/10.1145/2739482.2756571`
14. MITCHELL, M.: An introduction to genetic algorithms. MIT Press (1996)
15. Orhan, E., Pitkow, X.: Skip connections eliminate singularities. In: International Conference on Learning Representations (2018), `https://openreview.net/forum?id=HkwBEMWCZ`
16. Simonyan, Karen, Zisserman, Andrew: Very deep convolutional networks for large-scale image recognition (Apr 2015), `https://arxiv.org/abs/1409.1556`
17. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2014), `http://arxiv.org/abs/1409.1556`
18. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. CoRR **abs/1507.06228** (2015), `http://arxiv.org/abs/1507.06228`
19. Sun, Y., Xue, B., Zhang, M.: Evolving deep convolutional neural networks for image classification. CoRR **abs/1710.10741** (2017), `http://arxiv.org/abs/1710.10741`
20. Sun, Y., Xue, B., Zhang, M., Yen, G.G.: Automatically designing CNN architectures using genetic algorithm for image classification. CoRR **abs/1808.03818** (2018), `http://arxiv.org/abs/1808.03818`
21. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–9 (June 2015)
22. Vandenbergh, F., Engelbrecht, A.: A study of particle swarm optimization particle trajectories. Information Sciences **176**(8), 937–971 (2006). https://doi.org/10.1016/j.ins.2005.02.003
23. Wang, B., Sun, Y., Xue, B., Zhang, M.: A hybrid differential evolution approach to designing deep convolutional neural networks for image classification. In: Australasian Joint Conference on Artificial Intelligence. pp. 237–250. Springer (2018)
24. Xie, L., Yuille, A.: Genetic cnn. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 1388–1397 (Oct 2017). https://doi.org/10.1109/ICCV.2017.154
25. Yamanaka, J., Kuwashima, S., Kurita, T.: Fast and accurate image super resolution by deep cnn with skip connection and network in network. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.S.M. (eds.) Neural Information Processing. pp. 217–225. Springer International Publishing, Cham (2017)
26. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)