electronics MDPI

Article

Evolutionary Convolutional Neural Network Optimization with Cross-Tasks Transfer Strategy

Zhao Wang [1,2], Di Lu [2], Huabing Wang [1], Tongfei Liu [2] and Peng Li [2,*]

[1] State Key Laboratory of Complex Electromagnetic Environment Effects on Electronics and Information System (CEMEE), Luoyang 471003, China; wangzhao@xidian.edu.cn (Z.W.); xueshan@ustc.edu (H.W.)
[2] Key Laboratory of Electronic Information Countermeasure and Simulation Technology of Ministry of Education, Xidian University, No. 2 South TaiBai Road, Xi'an 710071, China; Di_Lu@stu.xidian.edu.cn (D.L.); liutongfei_home@hotmail.com (T.L.)
* Correspondence: li_peng001@163.com; Tel.: +86-137-0919-3246

Abstract: Convolutional neural networks (CNNs) have shown great success in a variety of real-world applications and the outstanding performance of the state-of-the-art CNNs is primarily driven by the elaborate architecture. Evolutionary convolutional neural network (ECNN) is a promising approach to design the optimal CNN architecture automatically. Nevertheless, most of the existing ECNN methods only focus on improving the performance of the discovered CNN architectures without considering the relevance between different classification tasks. Transfer learning is a human-like learning approach and has been introduced to solve complex problems in the domain of evolutionary algorithms (EAs). In this paper, an effective ECNN optimization method with cross-tasks transfer strategy (CTS) is proposed to facilitate the evolution process. The proposed method is then evaluated on benchmark image classification datasets as a case study. The experimental results show that the proposed method can not only speed up the evolutionary process significantly but also achieve competitive classification accuracy. To be specific, our proposed method can reach the same accuracy at least 40 iterations early and an improvement of accuracy for 0.88% and 3.12% on MNIST-FASHION and CIFAR10 datasets compared with ECNN, respectively.

Keywords: evolutionary algorithm; convolutional neural network; transfer learning; image classification

check for updates

Citation: Wang, Z.; Lu, D.; Wang, H.; Liu, T.; Li, P. Evolutionary Convolutional Neural Network Optimization with Cross-Tasks Transfer Strategy. *Electronics* **2021**, *10*, 1857. https://doi.org/10.3390/electronics10151857

Academic Editor: Amir Mosavi

Received: 24 June 2021
Accepted: 29 July 2021
Published: 2 August 2021

1. Introduction

Machine learning has shown great success in various real-world applications. Among machine learning approaches, convolutional neural networks (CNNs), which show overwhelmingly superiority among machine learning approaches, have been widely used in various real-world applications, such as image processing [1], engineering [2], health care [3,4], and cognitive science [5], etc. Convolutional neural network commonly consists of convolution, pooling, and fully-connected layers and are trained on the source dataset and then applied to the target dataset. As is well-known, the success of CNNs mainly benefit from the improvement on fundamental CNN architectures, such as increasing the depth of neural networks, the employment of skip layers, and adding inner network structures, etc. However, the state-of-the-art CNN architectures with high performance are manually devised by experienced experts with trial-and-error. As designing efficient CNN architectures is a challenging process, researchers have developed algorithms to design the CNN architectures automatically, which aims to enhance the applicability and universality of CNNs.

Designing the best CNN architecture can be viewed as a neural architecture search (NAS) process on the given dataset, and the searching parameters include the number of convolutional layers, the configuration of different layers and the placement of skip layers, etc. In the domain of machine learning, reinforcement learning (RL) is an early method for NAS and is always collocated with recurrent neural network (RNN) to control the process of hyperparameter tuning [6]. In addition to reinforcement learning, Bayesian optimization [7] has also been widely used, which achieves global optimization by maintaining black-box functions that do not assume any specific forms. Some other machine learning approaches including grid search, random search and gradient search are explored in [8–10]. Nonetheless, with these methods remains the problem of taking up too much searching space, thus requiring excessive GPU memory.

Evolutionary algorithms (EAs) are optimization approaches which seek the optimal solution by simulating the natural evolution process inspired by Darwin's theory of evolution [11]. Evolutionary algorithm begins with generating individuals that undergo crossover, mutation, and selection to retain strong individuals and eliminate weak ones. After several generations, it can solve complex optimization problems and generate high-quality optimization schemes efficiently. Since evolutionary algorithm has flexible representations and strong searching capacity, it has been widely used to solve a variety of tasks, especially under the environment with non-convex or non-derivative functions [12–16]. In recent years, EAs have attracted great interest as they offer alternative methods to solve NAS problems. With flexible encoding strategy and strong searching capacity, EAs are becoming a promising approach to optimizing CNN architectures. The general framework of evolutionary convolutional neural network (ECNN) follows the evolution procedure, the performance of which mainly depends on the effective design of network architecture evolution strategy.

However, most of the ECNN methods focus on improving the performance of discovered CNN architectures without considering the relevance between different classification tasks. The evolution process has to start from scratch if the learning environment changes. In such a situation, evolutionary transfer learning which utilizes the knowledge from the previously solved tasks to facilitate solving target task, is promising for NAS problems. Evolutionary transfer learning is a human-like learning process, it has been frequently introduced to solve different but related problems for more effective evolutionary algorithms. In [17–22], EAs with transfer capacity are designed to solve extensive optimization problems like dynamic vehicle routing, heterogeneous and image classification problems, etc. With the assistance of knowledge from optimized solutions, the performance of evolutionary algorithms can be enhanced.

In this paper, an effective evolutionary convolutional neural network optimization method with cross-tasks transfer strategy (CTS-ECNN) is proposed to take full advantage of the knowledge extracted from the previously solved tasks when a new classification task is encountered. The main contributions of the proposed CTS-ECNN method are summarized as follows:

(1) We propose a simple and effective cross-tasks transfer strategy, which can select the valuable knowledge from the original task to transfer for improving the performance of the target task. Especially at the early generations, our method can increase the optimization speed significantly, which is important when the learning time or computing resource is limited;
(2) Within the case study of image classification tasks, it is demonstrated that the proposed CTS-ECNN can obtain better results than the ECNN that starts from scratch and some manually-designed state-of-the-art methods do;
(3) In the framework of the proposed CTS-ECNN, when a new task is encountered, we can extract knowledge from the optimized tasks. With more knowledge achieved from related tasks, the proposed method can be applied to more tasks rapidly without considering the sequence of tasks.

The rest of this paper is organized as follows: In Section 2, the related works are summarized. In Section 3, our method is introduced in detail. Section 4 gives the experimental settings and the analysis of the experimental results. Finally, the conclusions and future works are described in Section 5.

2. Related Works

2.1. Evolutionary CNN Optimization

Since the evolutionary algorithm has the advantage of gradient-free and being insensitive to local optimum, evolutionary deep learning has been an interesting domain recently. Among all of them, the neural network, especially CNN architecture optimization with the evolutionary algorithm, attracts much attention. The general framework of evolutionary convolutional neural network is concluded in Algorithm 1. As is shown in Algorithm 1, the whole process of evolutionary CNN follows the evolution procedure: initialization (step 1), evaluation (step 2), selection (step 4), mutation and crossover (step 5). Specially, the fitness evaluation in evolutionary CNN is executed by training the corresponding CNN architecture. The evaluation process is completed by optimizing the weights in CNN architecture to reach the maximal classification accuracy. To evaluate each individual's fitness accurately, a CNN is trained for several epochs by using the same initialization method, loss function and optimizer. The performance evaluation of CNN is provided in Algorithm 2.

Algorithm 1 Evolutionary convolutional neural network.

```
Input: $N$: the max number of generations, $k$: the size of each generation, $p_m$: the
mutation probability, $p_c$: the crossover probability.
Output: Individuals of the last generation with their fitness values.
1: Randomly initialize $k$ individuals and map them to the corresponding CNNs;
2: Compute the classification accuracy of each CNN to obtain the fitness value with
   Algorithm 2;
3: for $n = 1 : N$ do
4:     Generate a new generation with selecting method on parent individuals based on
       the fitness value;
5:     Produce offspring through the operator of mutation and crossover with probability
       $p_m$ and probability $p_c$;
6:     Compute the classification accuracy of each individual on offspring;
7: end for
```

Algorithm 2 Performance evaluation of CNN.

```
Input: $p$: the individual, $D_{train}$: the training dataset, $D_{valid}$: the validation dataset, $T$: the
epoch number, $B$: the training batch size, $L$: the loss function, $\eta$: the learning rate.
Output: The classification accuracy.
1: Map $p$ into the corresponding CNN architecture;
2: $\omega \leftarrow$ initialize the weights of CNN with predefined method;
3: for $t = 1 : T$ do
4:     $\eta \leftarrow$ update $\eta$ according to $t$;
5:     for each $B$ in $D_{train}$ do
6:         $\nabla\omega \leftarrow$ compute the gradient by $\partial L/\partial\omega$;
7:         $\omega \leftarrow \omega - \eta\nabla\omega$;
8:     end for
9:     Compute the classification accuracy on $D_{valid}$;
10: end for
```

In [23], Xie et al. introduced a binary encoding method to represent CNN and use GA to learn network architectures automatically. Concurrently, Cartesian genetic programming was used to design deep convolutional neural network architectures with a variable-length genotype-to-phenotype method in [24]. Real et al. [25] designed an image classifier named AmoebaNet which applies evolutionary algorithm to neural network topologies. Sun's work [26] gave a comprehensive comparison of manually designed and automatically designed neural networks, and then proposes CNN-GA which is competent for discovering deep neural networks. Lu and Whalen et al. [27] addressed multi-objective framework to neural architecture search, and the NSGANetV1 algorithm is demonstrated on new classification tasks, i.e., corrupted CIFAR-10, ImageNet-V2 and medical X-ray images.

2.2. Transfer Learning

In the domain of machine learning, transfer learning has received significant interest on solving different but related problems for better effective deep learning performance. Knowledge transfer of internal representations is an example of transfer learning in [28]. A trained deep convolutional neural network is demonstrated that its components can be transferred to another network to learn new information with smaller training sets. Terekhov et al. [29] applied knowledge transfer to deep neural networks by re-using block-modular architecture to solve new tasks. This architecture can outperform networks trained from scratch and has fewer weights to learn. Based on the learning features of each layer in neural networks, Yosinski et al. [30] proposed a method which can quantify the transferability of layer features. The results show that transferability is affected by difficulties of splitting network layers and the specificity of higher network layers.

There is also a growing interest in evolutionary transfer algorithms in recent years. Evolutionary transfer algorithms have been applied to solving different types of problems, such as multi-task optimization, multi-objective optimization, and complex optimization, etc. In multi-task environment, [31] transferred knowledge through crossover based on the theory that the solving of one problem may facilitate the solving of other related problems. In [32], Y. Ong et al. extended knowledge transfer by designing a explicit auto-encoder to transfer optimized solutions instead of genetic crossover. In the domain of multi-objective optimization, Liang et al. [18] devised a one-layer auto-encoder to enhance the performance of evolutionary algorithm by transferring knowledge across heterogeneous problems. Iqbal et al. [19] developed the GP-criptor to transfer learning GP-criptor which can reuse knowledge from past solved classification problems to improve image classification accuracy.

In this work, we focus on utilizing transfer learning to improve the performance and efficiency of neural architecture evolution. The proposed method can take full advantage of knowledge transferred from the previous solved tasks when a new classification task is encountered.

3. Materials and Method

As mentioned before, the key problem of transfer learning in EAs is how to utilize the knowledge from the source tasks efficiently. Which information to be transferred and how to transfer the knowledge determine whether the transfer process can facilitate the evolution of target task better than random initialization. In this section, we will introduce the CTS-ECNN method which constructs the suitable individuals for transference based on the multi-population framework [33]. Meanwhile, considering the high computation cost of evaluating the performance of a CNN architecture, we also use a clustering method to accelerate the transfer process. Figure 1 shows the workflow of the CTS, as well as the associated ECNN.

As shown in Figure 1, the proposed CTS-ECNN method is composed of three modules, i.e., classification tasks, evolutionary algorithm of CNN, and cross-tasks transfer. First, a set of individuals representing different CNN architectures are evolved on the first classification task with their performances (shown in Section 3.1). After evolution, the CNN architectures with top fitness values are encoded for clustering similar individuals with affinity propagation (AP) method (shown in Section 3.2). With clustered individuals, exemplars of each cluster are evaluated on the target classification task, and then their performances are used as the ranking index prepared for constructing the subpopulation (shown in Section 3.3). When the mixed subpopulation is constructed, we can transfer the useful knowledge to the target classification task through applying the selected individuals as the initial generation. To clarify the proposed method, we give a general framework of CTS-ECNN in Algorithm 3. Noting that our proposed method provides a sequential transfer framework applicable to different classification tasks. If we have discovered CNN architectures on the preceding (M-1) tasks, the M-*th* task can be facilitated by utilizing the knowledge obtained from the previously solved (M-1) tasks.

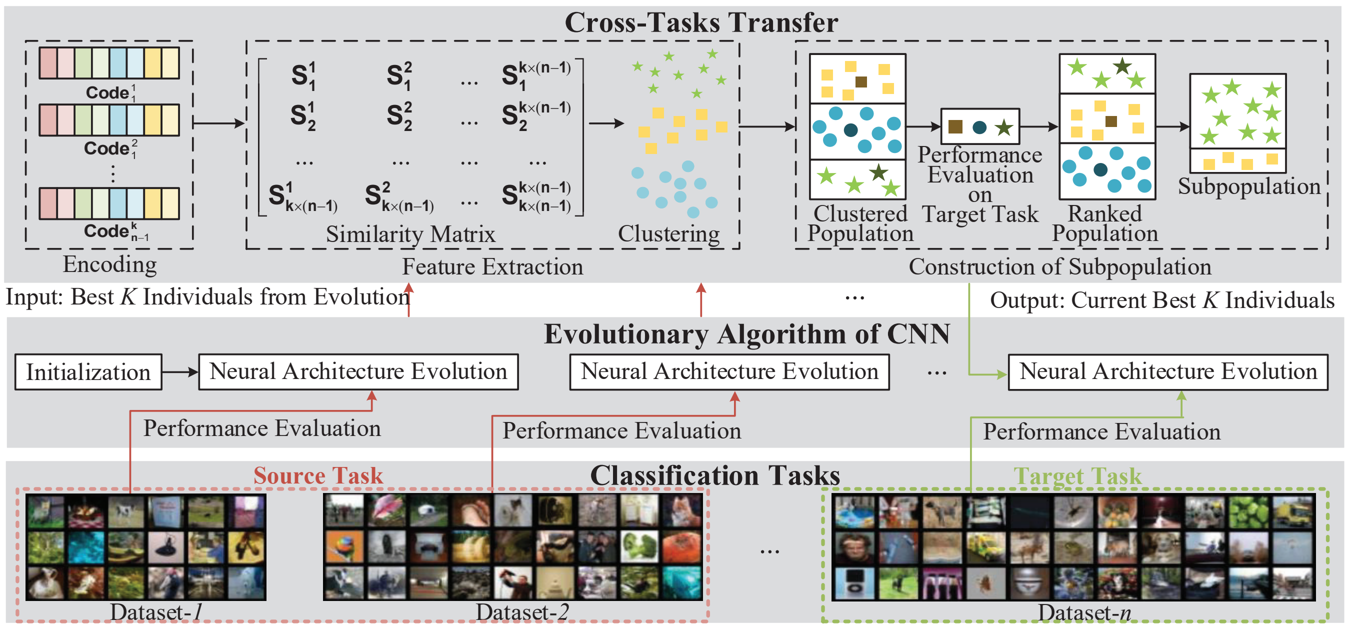

Figure 1. The workflow of the CTS-ECNN.

Algorithm 3 The pseudocode of CTS-ECNN.

Input: N: the max number of generation, k: the size of each generations, D_M: the target dataset.
Output: The best CNN architecture for D_M.

```
1:  for $m = 1 : M$ do
2:    if $m = 1$ then
3:      $\{O_P\}_k \leftarrow$ Randomly initialize $k$ individuals;
4:    else
5:      $\{O_P\}_k \leftarrow$ Transfer $k$ individuals based on Algorithm 4;
6:    end if
7:    Set $n \leftarrow 0$;
8:    while $n \leq N$ do
9:      Obtain offspring $\{O_C\}_k$ with crossover and mutation operators;
10:     $\{O_R\}_{2k} \leftarrow \{O_P\}_k \cup \{O_C\}_k$;
11:     Evaluate each individual in $\{O_R\}_{2k}$;
12:     $\{O_P\}_k \leftarrow$ Select the next generation;
13:     $n \leftarrow n + 1$;
14:   end while
15:   while $m < M$ do
16:     $\{O_r^c\} \leftarrow$ collect individuals with top fitness values for CTS in Algorithm 4;
17:   end while
18: end for
```

3.1. Neural Architecture Evolution

Generally, the whole process of ECNN follows the procedure in Algorithm 3 (steps 1–14). The first step is to design a proper genotype-to-phenotype mapping strategy. We provide a variable-length string representation to describe the CNN architecture. In our method, we prepare two types of convolutional and pooling operators for building a CNN architecture, i.e., the standard and residual convolutional operators, the max and average pooling operators. The max number of convolutional operators N_c and pooling operators

N_p are predefined. It is noted that the proposed method focuses on the optimization of neural network structure, so we select these building blocks as functional nodes to realize flexible genotype-to-phenotype mapping. For the first classification task, a set of individuals with predefined population size is randomly initialized based on the string representation. Accordingly, subsequent classification tasks use the assigned subpopulation from cross-tasks transfer as the initial generation (steps 2–6).

During evolution, mutation and crossover operator are implemented on each generation. When $\{O_R\}_{2k}$ is obtained, each produced individual is evaluated on the corresponding dataset to compute its classification accuracy by training the CNN architecture it represents through several epochs, and then these accuracies serve as fitness values to produce a new generation. When the max number of generations is reached, we not only obtain the optimized CNN architecture for the current classification task but also utilize individuals with top fitness values as the learning resource to facilitate posterior classification tasks.

3.2. Encoding and Extraction of Feature

The encoding operator acts on the optimized CNN architectures which have been evolved on previous classification tasks. To extract the features of each CNN architecture, these architectures have to be encoded into fixed-length strings which are suitable for similarity computation. Each convolutional operator is encoded into a quaternion as (*type*, *channel*, *filter*, *in*), where *type* for the standard and residual convolutional operator is set to 0 and 1, respectively. As for pooling operators, each of them is encoded into a pair as (*type*, *in*), *type* of the max and average pooling operators is denoted by 0 and 1. There is no need to encode the parameter of *out* because it can be deduced by the *in* of subsequent operator. When the number of convolutional operator is smaller than N_c or the number of pooling operators is smaller than N_p, the blank position will be set to zeros to keep the coding length invariable.

As the encoding of CNN architectures has been finished, the codes containing structure information can be used directly by similarity computation. Algorithm 4 shows the framework of our proposed CTS, and steps 2–7 give a general process of constructing a similarity matrix. We use Euclidean distance as the similarity of CNN architectures:

$$s(i,j) = -\|x_i - x_j\|^2 \tag{1}$$

where x_i and x_j are two different codes, and the setting of a negative squared error is for convenient calculation. Each code of its corresponding CNN architecture is compared with all the other codes string-to-string to work out the similarity *s(i, j)*, and then the similarity *s(i, i)* for each code is set to a shared value, for which we choose the median of the input similarity.

Considering the demand of reducing transfer computation and utilizing knowledge from previous tasks efficiently, clustering is a necessary preprocessing for the construction of subpopulation. Of all the clustering methods, AP has the advantages of good robustness and accuracy over other clustering methods [34]. It does not need to determine the number of clusters before running the algorithm and is based on the concept of "message passing" between data points, which updates two matrices:

$$r(i,j) \leftarrow s(i,j) - \max_{j' \neq j}\{a(i,j') + s(i,j')\} \tag{2}$$

$$a(i,j) \leftarrow \min\left\{0, r(j,j) + \sum_{i' \notin \{i,j\}} \max\{0, r(i',j)\}\right\} \tag{3}$$

where *r(i,j)* is the value of responsibility matrix and reflects the fitness that *j* serves as the exemplar for *i* on account of the other potential exemplars. *a(i,j)* is the value of availability matrix and reflects the appropriateness that *i* chooses *j* as its exemplar on account of the other potential exemplars. To be specific, *a(i,j′)* represents the belongingness of other points to *i* except *j*. *s(i,j′)* represents the attraction of other points to *i* except *j*. If the value of *r(i,j)* is greater than 0, it means that *j* has better chance to become the exemplar; *r(i,j′)* represents the similarity that *j* becomes the exemplar of other points except *i*. Taking all the attraction values greater than or equal to 0 and the possibility that *j* is the exemplar into consideration, *a(i,j)* represents the cumulative proof that *i* chooses *j* as the exemplar.

Both matrices are initialized to all zeroes. Iterations proceed until cluster boundaries remain unchanged over several iterations. The *j* with the maximum value of *a(i,j)* + *r(i,j)* will be chosen as the exemplar of its corresponding cluster. When AP terminates, the number of clusters and the exemplar of each cluster are obtained.

Algorithm 4 Cross-task transfer strategy (CTS).

```
Input: $\{O_r^c | r = 1, 2, ..., k, c = 1, 2, ..., M-1\}$: individuals with top fitness values from the preceding $(M-1)$ datasets, $q_m$: the mutation probability, $TP$: the transfer parameter.
Output: The parent $P_{Trans}$ for $D_M$.
1: $E_t \leftarrow$ encode $O_r^c$ independently;
2: if $i \neq j$ then
3:     $s(i, j) \leftarrow$ compute the similarity between $E_i$ and $E_j$;
4: end if
5: if $i = j$ then
6:     $s(i, i) \leftarrow$ input the median of the acquired similarities;
7: end if
8: Cluster CNN architectures with $AP$ algorithm;
9: $p \leftarrow$ the number of clusters;
10: $\{M\}_p \leftarrow$ choose the exemplar of each cluster;
11: Approximate the classification accuracies of $\{M\}_p$;
12: $P_{opt} \leftarrow$ choose the optimal cluster;
13: $N_{sub} \leftarrow$ determine the size of suboptimal subpopulation with $TP$;
14: $P_{sub} \leftarrow$ randomly choose $N_{sub}$ individuals from the suboptimal cluster;
15: if $N_{opt} + N_{sub} < k$ then
16:     $P_{ext} \leftarrow$ perform mutation operator with probability $q_m$ on $P_{opt}$;
17:     $P_{Trans} \leftarrow P_{opt} \cup P_{sub} \cup P_{ext}$;
18: else if $N_{opt} + N_{sub} > k$ then
19:     $P_{opt} \leftarrow$ choose $(k - N_{sub})$ individuals from $P_{opt}$ randomly;
20:     $P_{Trans} \leftarrow P_{opt} \cup P_{sub}$;
21: else
22:     $P_{Trans} \leftarrow P_{opt} \cup P_{sub}$;
23: end if
```

3.3. Construction of Subpopulation

In original transfer learning strategies, the entire knowledge extracted from previous tasks is evaluated on the target task to sort out the best solution. However, considering the high computational cost of evaluating the performance of a CNN architecture, it is computationally cumbersome to evaluate each individual especially when we have numerous source tasks for transfer learning. In our method, we only evaluate the exemplar of each cluster on the target task to represent its corresponding cluster. The proposed CTS is a straightforward way to utilize the ranked clusters of alternative CNN architectures based on exemplars' performance. To enable efficient transfer, we construct the subpopulation by exploiting high-quality individuals and exploring new individuals.

The efficacy of the proposed CTS-ECNN depends on how the subpopulation of each target task is constructed. On account of the complementarity of individuals, we adopt the multi-population framework to construct subpopulation which includes the optimal cluster P_{opt} and the suboptimal cluster P_{sub}. Specifically, P_{opt} can be viewed as the individuals which carry most problem-solving knowledge and thus can be viewed as the individuals belonging to target task, P_{sub} can be viewed as the extra individuals selected from related tasks accordingly. When the mixed subpopulation is constructed, it can collect inter-task knowledge and inner-task knowledge to generate offspring for the target task. N_{sub}

is determined by the cross-task transfer parameter TP, thus the number of suboptimal individuals N_{sub} is denoted by

$$N_{sub} = \min\{[TP \times k], N_{sub,max}\} \tag{4}$$

where k represents the size of each generation and $N_{sub,max}$ is the size of suboptimal cluster which guarantees that cross-task knowledge will not overflow. TP is the transfer parameter which controls the degree of inter-task knowledge transfer thus maintaining the balance of subpopulation.

It is an important issue to set the transfer parameter TP thoughtfully as it controls the amount of inner-task and inter-task knowledge transferred into the target task. To be specific, if TP is large, the more extra individuals from related tasks are collected, therefore the more inter-task knowledge can be transferred to target task. Correspondingly, if TP is small, the more inner-task knowledge will be extracted. To conclude, a large TP is suitable for a compact searching space where individuals have small divergence, while a small TP is suitable for individuals with significantly different performances.

With N_{sub} determined by TP, we randomly choose N_{sub} individuals in the suboptimal cluster as a part of subpopulation. In Algorithm 4, the construction of subpopulation is executed in step 15–23. As shown in Algorithm 4, the rest part of the subpopulation are produced by the optimal cluster. When N_{opt} is insufficient, some extra individuals are produced by operating mutation on the individuals of optimal cluster with probability q_m to keep the size of generation invariable. On the contrary, if (N_{opt} + N_{sub}) exceeds the size of generation, we randomly choose ($k - N_{sub}$) individuals from the optimal cluster for the same purpose.

The constructed subpopulation is used as the current best generation for the target classification task to accelerate the evolution of CNN architectures. Particularly, the more source tasks we have trained, the more transfer knowledge we can utilize to facilitate the subsequent tasks. More importantly, as the knowledge from all the previous tasks participates in our cross-task transfer process, the sequential transfer process can keep going without deploying specified order of tasks.

3.4. Training and Prediction

Based on conventional settings in machine learning community, stochastic gradient descent (SGD) with a batch size of 128 is used to train the CNN architectures, whose weights are initialized by He's method [35]. For fear of over-fitting, the weight decay is set to 5×10^{-4}. The learning rate is initialized to 10^{-2} for the first 30 epochs, followed by 10^{-3} for 120 epochs, 10^{-4} for 90 epochs, and 10^{-5} for 30 epochs. Each CNN architecture is trained for 50 epochs in the phase of fitness evaluation to reduce computation time. All of the CNN architectures are trained on two NVIDIA 1080TI GPUs. Noting that these parameter settings are applied in both experimental scenarios.

For the parameter settings of neural architecture evolution, the max number of convolutional operators N_c and pooling operators N_p are set to 10 and 5, respectively. We set the mutation probability p_m and the crossover probability p_c to 0.8 and 0.2, respectively, to accelerate the evolution of new CNN architectures. The number of generations for the target classification dataset is set to 50 and each generation contains 20 individuals. Note that the max generation for the source classification dataset which is prepared for the CTS is set as 20.

4. Experimental Results and Discussion

4.1. Datasets

In the case study, the proposed CTS-ECNN method, which is based on transfer learning, is evaluated using two benchmark image classification datasets. Results are compared with the ECNN that starts from scratch and some state-of-the-art methods.

As mentioned in Section 3, the proposed method is a sequential transfer framework, so we design two experimental scenarios to test its applicability for different classification tasks. For the first scenario, our method is tested on the MNIST-FASHION dataset and the transfer process is based on the MNIST classification task which has been optimized before. For the second scenario, our method is tested on the CIFAR10 dataset and the transfer process is based on the MNIST and MNIST-FASHION classification task which have been optimized before.

The MNIST dataset is a large database of handwritten digits, which consists of 60,000 grayscale images for training and 10,000 grayscale images for testing, and each of them has the dimension of 28 × 28. There are 10 categories, i.e., digits from 0 to 9, which have the equal number of samples for both the training and testing set. The MNIST-FASHION dataset shares the same image size and structure of training and testing splits with the MNIST dataset. The only difference between them is that there are 10 categories of commodity in MNIST-FASHION. The CIFAR10 dataset is a subset of 80 million tiny images, which consists of 50,000 color images for training and 10,000 color images for testing, and each of them has the dimension of 32 × 32. There are also 10 categories, which have the equal number of samples for both the training and testing set. As there are no validation sets in these benchmark datasets, 10% images of the training sets will be randomly selected as the validation sets to attain the fitness value.

4.2. Results of the First Experimental Scenario

We first compare the classification accuracy of each evolution step for the proposed CTS-ECNN method and the ECNN that starts from scratch on the MNIST-FASHION testing dataset. The maximum classification accuracy of each evolution process is visualized in Figure 2. It is obvious that the max classification accuracy is improved significantly in the first few generations with the transferred knowledge from the MNIST classification task. As shown in Figure 2, the proposed CTS-ECNN can achieve the same accuracy about 40 iterations early compared with ECNN that starts from scratch. When the evolution terminates, our method still obtains better classification accuracy than the original ECNN method does. The above result shows that the proposed CTS-ECNN can achieve the valuable knowledge from the MNIST classification task to help the neural network optimization of MNIST-FASHION classification task.

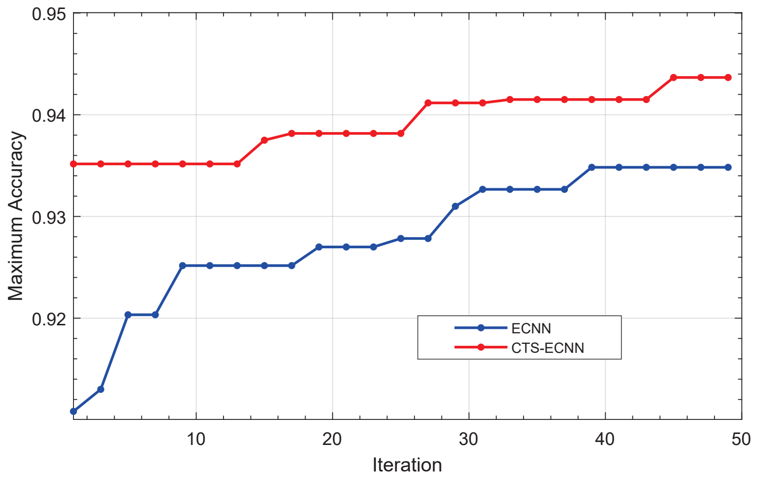

Figure 2. The maximum classification accuracy of each evolution step on the MNIST-FASHION testing dataset.

To make a comprehensive comparison of the two methods, classification performance of all the individuals in each evolution step are reported in Figure 3. As shown in Figure 3, the red and blue points represent the individuals in CTS-ECNN method and the original ECNN method that starts from scratch, respectively. As can be seen in each iteration, most of the individuals of our method can obtain better performance, which means with transferred knowledge the excellent parent generation is more likely to generate good offspring via evolution process. There are also a few red points lying below all the blue points in the figure, especially at the early iterations of the evolution process. As evolutionary algorithm is a heuristic search method that uses crossover and mutation operators to generate new populations, good individuals may also produce poor offspring. However, the historically best individuals are always maintained, so the evolutionary process can keep a steady improvement. With the evolution proceeding, individuals of our method perform better aggregation index than the original ECNN method does. We argue that the evolutionary algorithm tends to preserve the useful transferred knowledge, so that the excellent individuals are more easily to generate.

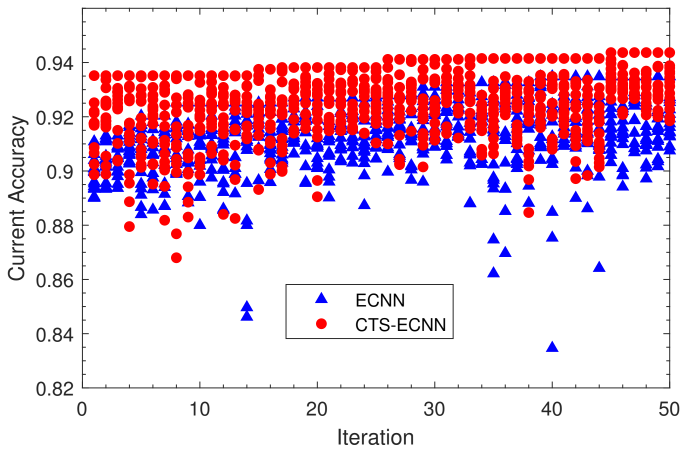

Figure 3. The current classification accuracy of each evolution step on the MNIST-FASHION testing dataset.

In addition to reporting the overall results of each iteration, we complete quantitative analysis on these two methods to obtain precise statistics. Results are summarized in Table 1. With the help of CTS, we can always find better network architectures which can achieve better classification accuracy. After the 10th generation, classification accuracies of the two methods keep a similar rate of growth and the accuracy improvement becomes smaller than the previous generations. To be specific, among the 10th and 50th generation, the proposed CTS-ECNN and original ECNN attain the improvement of 0.85% and 0.87% on maximum classification accuracy, respectively. Although during the period of evolution the average and medium classification accuracy are fluctuating a little, on the whole they gradually get higher. According to three kinds of difference value, i.e., the maximum, the average and the medium accuracy difference, the CTS-ECNN method shows significant advantage over the ECNN algorithm. This is important, because it means the CTS-ECNN algorithm can guarantee the overall improvement on original ECNN algorithms. After 50 generations, the maximum classification accuracy of the CTS-ECNN method reaches 94.37% and keeps a leading margin of 0.88% over the original ECNN method.

4.3. Results of the Second Experimental Scenario

As mentioned before, we have two optimized classification tasks, i.e., the MNIST and MNIST-FASHION dataset, which means when a new task is encountered, we can extract knowledge from the above three tasks. We will first compare the CTS-ECNN method based on the two classification tasks with the original ECNN method. The maximum classification accuracy of the two methods on the CIFAR10 testing dataset is shown in Figure 4. As in the MNIST-FASHION experiment, the maximum classification accuracy grows from generation to generation. Compared with the first experiment on MNIS-FASHION dataset, with more knowledge extracted from the previously solved two classification tasks, the initial generation of CTS-ECNN can outperform even the last generation of the original ECNN. This means our method can not only skip the process of random initialization but also generate better individuals. In the end of the evolution process, our proposed method obtains the improvement of about 3% for the maximum classification accuracy, which can demonstrate the scalability of our proposed method. As we all know that the iteration of the neural network optimization is time-consuming, the experiment with 50 generations can demonstrate that our method has the ability to reduce the computational cost.

Table 1. Classification accuracy on the MNIST-FASHION testing dataset. Diff is the difference of classification accuracy with ECNN that starts from scratch. Gen represents different evolution steps.

Gen	Max %	Diff %	Avg %	Diff %	Med %	Diff %
01	93.52	2.43	91.49	1.72	91.73	2.13
05	93.52	1.48	91.76	1.26	91.67	0.73
10	93.52	1.00	91.72	1.17	91.95	1.03
15	93.75	1.23	91.93	0.37	91.93	0.38
20	93.81	1.12	92.12	1.06	92.45	1.52
25	93.81	1.03	92.70	1.10	92.89	1.33
30	94.12	1.01	92.58	0.60	92.42	0.30
35	94.15	0.88	92.20	1.61	92.13	0.72
40	94.15	0.67	92.35	1.79	92.50	1.33
45	94.37	0.88	93.38	1.78	93.28	1.73
50	94.37	0.88	93.10	1.22	93.13	1.38

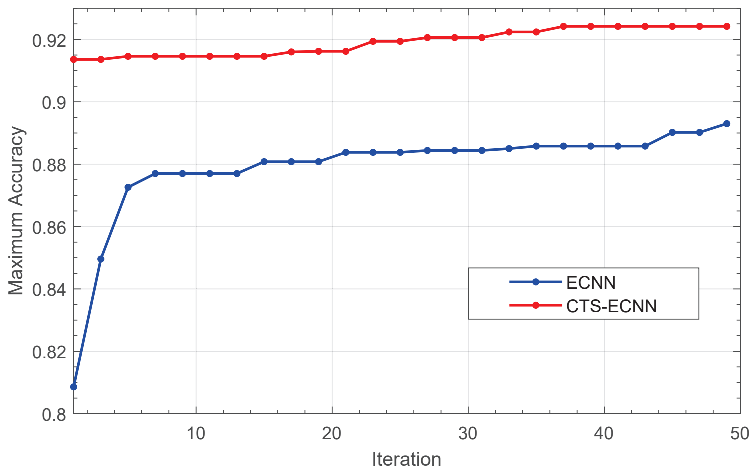

Figure 4. The maximum classification accuracy of each evolution step on the CIFAR10 validation dataset.

In order to better understand the details of the proposed method on the CIFAR10 classification task, we draw box plots in Figure 5. As is shown in Figure 5, both of the two methods show increase for the maximum classification accuracy and fluctuation for the median classification accuracy. However, our proposed method shows smaller fluctuation and better median classification accuracy. By investigating the height of each box, it can also be observed that the variation of the classification accuracy during each generation of our proposed method is much smaller than the original ECNN, which implies the evolution processes towards a more steady state on the CIFAR10 classification task with transferred knowledge.

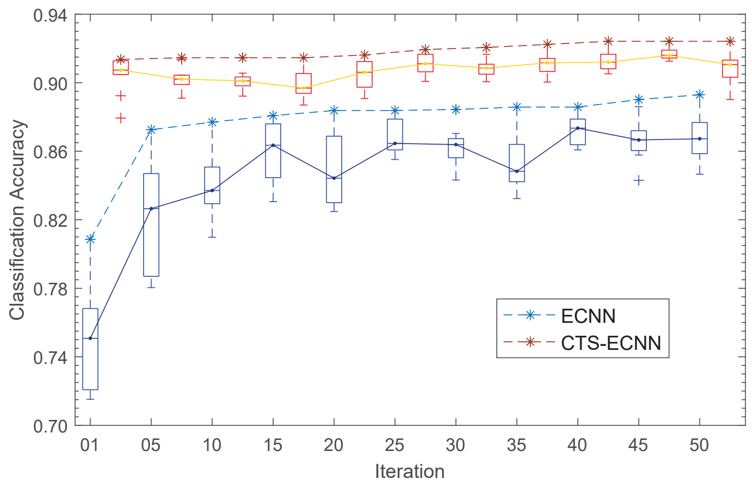

Figure 5. The classification accuracy of different evolution steps on the CIFAR10 testing dataset. The red box represents CTS-ECNN and blue box represents the original ECNN. The max and median classification accuracy of different evolution steps are connected by a dashed line and solid line.

In addition, our proposed method is compared with some state-of-the-art methods in Table 2. We group these methods into two different categories, namely manually-designed methods and automatically-designed methods. For the first category including ResNet (depth = 1.10), ResNet (depth = 1.202) [36], Maxout [37], Network in Network [38] and Highway Network [39], we mainly compare the classification accuracy. Among these manually-designed peer competitors, after the 50th generation, our proposed method can obtain higher classification accuracy than most of the state-of-the-art CNN architectures but lower than ResNet (depth = 101). We note that ResNet (depth = 101) is much deeper, i.e., ResNet (depth = 101) has 101 layers while the proposed CTS-ECNN has less than 15 layers (10 convolutional layers and 5 pooling layers). Even at the 10th generation, our method can outperform Maxout and Network in Network. It can be demonstrated that when compared with the state-of-the-art manually designed methods, the proposed method can design competitive CNN architecture automatically with limit computation cost. For the second category including hierarchical evolution [40], CGP-CNN [24], genetic CNN [23], and the proposed CTS-ECNN, we compare the classification accuracy and the number of iterations that each method costs, which represent the efficiency of each method. Among these automatically-designed peer competitors, hierarchical evolution and CGP-CNN obtain 3.96% and 1.6% improvements on the CIFAR10 testing dataset over our proposed method. However, both of the methods consume much more iterations to obtain the best classification accuracy and hierarchical evolution focus on convolutional cells, rather than the entire neural network architecture [40]. It is noted that the classification accuracy of genetic CNN is slightly higher than CTS-ECNN, but it still requires the manual tuning based on expertise. It is demonstrated that our method could find competitive CNN architecture with limited computational resources. It makes sense, as someone with little knowledge of neural network architecture can design a competent neural network to solve the certain task easily.

Table 2. The comparisons between the proposed method and the state-of-the-art methods in terms of the classification accuracy (%) on the CIFAR10 testing dataset. Gen represents the evolution steps each method takes.

	Method	Acc %	Gen
Manually Designed	ResNet (depth = 101)	93.57	–
	ResNet (depth = 1202)	92.07	–
	Maxout	90.70	–
	Network in Network	91.19	–
	Highway Network	92.40	–
Automatically Designed	Hierarchical Evolution	96.37	7000
	CGP-CNN	94.02	300
	Genetic CNN	92.90	50
	CTS-ECNN (G-10)	91.46	10
	CTS-ECNN (G-30)	92.06	30
	CTS-ECNN (G-50)	92.42	50

5. Conclusions

In this paper, we apply the transfer learning to facilitate the evolutionary CNN architecture optimization. We propose an effective ECNN method with cross-task transfer strategy named CTS-ECNN which constructs the suitable individuals to transfer without taking up too much computational resource. For the case study, our proposed method is compared with the original ECNN and some state-of-the-art methods on benchmark image classification datasets. The results show that our method can not only accelerate the evolution process significantly but also find competitive CNN architectures.

However, our method still suffers from several drawbacks. First, although we attempt to reduce the computational cost of transfer strategy, the process of neural architecture evolution on source tasks also requires computational resource. Second, in this work, transfer strategy is only applied to the initial generation. It would be interesting to transfer knowledge among each generation. The above directions are left for future work.

Author Contributions: Conceptualization, Z.W. and D.L.; methodology, Z.W.; validation, Z.W., D.L., and H.W.; investigation, T.L.; writing—original draft preparation, Z.W. and D.L.; writing—review and editing, H.W., T.L., and P.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of Shaanxi Province (grant no. 2021JQ-210), the Fundamental Research Funds for the Central Universities (Grant no. XJS200216, JB210202) and the National Natural Science Foundation of China (Grant no. 62036006).

Data Availability Statement: The MNIST dataset can be downloaded from http://yann.lecun.com/exdb/mnist/ (accessed on 3 July 2021). The MNIST-FASHION dataset can be downloaded from https://github.com/zalandoresearch/fashion-mnist (accessed on 3 July 2021). The CIFAR10 dataset can be downloaded from https://www.kaggle.com/c/cifar-10 (accessed on 3 July 2021).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CNNs	Convolutional neural networks
ECNN	Evolutionary convolutional neural network
EAs	Evolutionary algorithms
CTS	Cross-tasks transfer strategy
NAS	Neural architecture search
RL	Reinforcement learning
RNN	Recurrent neural network
AP	Affinity propagation
SGD	Stochastic gradient descent

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
2. Jamshidi, M.; Lalbakhsh, A.; Lotfi, S.; Siahkamari, H.; Mohamadzade, B.; Jalilian, J. A neuro-based approach to designing a wilkinson power divider. *Int. J. Microw. Comput. Aided Eng.* **2020**, *30*, e22091. [CrossRef]
3. Roshani, S.; Jamshidi, M.B.; Mohebi, F.; Roshani, S. Design and modeling of a compact power divider with squared resonators using artificial intelligence. *Wirel. Pers. Commun.* **2021**, *117*, 2085–2096. [CrossRef]
4. Jamshidi, M.B.; Lalbakhsh, A.; Talla, J.; Peroutka, Z.; Roshani, S.; Matousek, V.; Roshani, S.; Mirmozafari, M.; Malek, Z.; Spada, L.L.; et al. Deep learning techniques and covid-19 drug discovery: Fundamentals, state-of-the-art and future directions. *Emerg. Technol. During Era COVID Pandemic* **2021**, *348*, 9.
5. Jamshidi, M.B.; Alibeigi, N.; Rabbani, N.; Oryani, B.; Lalbakhsh, A. Artificial neural networks: A powerful tool for cognitive science. In Proceedings of the 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 1–3 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 674–679.
6. Bello, I.; Zoph, B.; Vasudevan, V.; Le, Q.V. Neural optimizer search with reinforcement learning. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 459–468.
7. Snoek, J.; Larochelle, H.; Adams, R.P. Practical bayesian optimization of machine learning algorithms. In Proceedings of the Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012.
8. Pontes, F.; Amorim, G.; Balestrassi, P.; Paiva, A.; Ferreira, J. Design of experiments and focused grid search for neural network parameter optimization. *Neurocomputing* **2016**, *186*, 22–34. [CrossRef]
9. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
10. Bengio, Y. Gradient-based optimization of hyperparameters. *Neural Comput.* **2000**, *12*, 1889–1900. [CrossRef] [PubMed]
11. Bäck, T.; Schwefel, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.* **1993**, *1*, 1–23. [CrossRef]
12. Ramakurthi, V.B.; Manupati, V.; Machado, J.; Varela, L. A hybrid multi-objective evolutionary algorithm-based semantic foundation for sustainable distributed manufacturing systems. *Appl. Sci.* **2021**, *11*, 6314. [CrossRef]
13. Abualigah, L.; Diabat, A.; Sumari, P.; Gandomi, A.H. A novel evolutionary arithmetic optimization algorithm for multilevel thresholding segmentation of covid-19 ct images. *Processes* **2021**, *9*, 1155. [CrossRef]
14. Guerraiche, K.; Dekhici, L.; Chatelet, E.; Zeblah, A. Multi-objective electrical power system design optimization using a modified bat algorithm. *Energies* **2021**, *14*, 3956. [CrossRef]
15. Yılmaz, E.M.; Güntert, P.; Etaner-Uyar, Ş. Evaluation of multi-objective optimization algorithms for nmr chemical shift assignment. *Molecules* **2021**, *26*, 3699. [CrossRef]
16. Ponti, A.; Candelieri, A.; Archetti, F. A new evolutionary approach to optimal sensor placement in water distribution networks. *Water* **2021**, *13*, 1625. [CrossRef]
17. Zhou, L.; Feng, L.; Gupta, A.; Ong, Y.; Liu, K.; Chen, C.; Sha, E.; Yang, B.; Yan, B.W. Solving dynamic vehicle routing problem via evolutionary search with learning capability. In Proceedings of the IEEE Congress on Evolutionary Computation, San Sebastián, Spain, 5–8 June 2017; pp. 890–896.
18. Feng, L.; Ong, Y.; Jiang, S.; Gupta, A. Autoencoding evolutionary search with learning across heterogeneous problems. *IEEE Trans. Evol. Comput.* **2017**, *21*, 760–772. [CrossRef]
19. Iqbal, M.; Xue, B.; Al-Sahaf, H.; Zhang, M. Cross-domain reuse of extracted knowledge in genetic programming for image classification. *IEEE Trans. Evol. Comput.* **2017**, *21*, 569–587. [CrossRef]
20. Xu, Q.; Wang, N.; Wang, L.; Li, W.; Sun, Q. Multi-task optimization and multi-task evolutionary computation in the past five years: A brief review. *Mathematics* **2021**, *9*, 864. [CrossRef]
21. Dumitru, D.; Dioșan, L.; Andreica, A.; Bálint, Z. A transfer learning approach on the optimization of edge detectors for medical images using particle swarm optimization. *Entropy* **2021**, *23*, 414. [CrossRef]
22. Chu, S.-C.; Zhuang, Z.; Li, J.; Pan, J.-S. A novel binary quasi-affine transformation evolutionary (quatre) algorithm. *Appl. Sci.* **2021**, *11*, 2251. [CrossRef]

23. Xie, L.; Yuille, A. Genetic CNN. In Proceedings of the IEEE International Conference on Computer Vision ICCV, Venice, Italy, 22–29 October 2017; pp. 1388–1397.
24. Suganuma, M.; Shirakawa, S.; Nagao, T. A genetic programming approach to designing convolutional neural network architectures. In Proceedings of the Genetic and Evolutionary Computation Conference, Berlin, Germany, 15–19 July 2017; pp. 497–504.
25. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q. Regularized evolution for image classifier architecture search. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 4780–4789.
26. Sun, Y.; Xue, B.; Zhang, M.; Yen, G.; Lv, J. Automatically designing cnn architectures using the genetic algorithm for image classification. *IEEE Trans. Cybern.* **2020**, *50*, 3840–3854. [CrossRef] [PubMed]
27. Lu, Z.; Whalen, I.; Boddeti, V.; Dhebar, Y.; Deb, K.; Goodman, E.; Banzhaf, W. Nsga-net: Neural architecture search using multi-objective genetic algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference, Prague, Czech Republic, 13–17 July 2019; pp. 419–427.
28. Gutstein, O.F.S.; Freudenthal, E. Knowledge transfer in deep convolutional neural nets. *Int. J. Artif. Intell. Tools* **2008**, *17*, 555–567. [CrossRef]
29. Terekhov, A.V.; Montone, G.; O'Regan, J. Knowledge transfer in deep block-modular neural networks. *arXiv* **2015**, arXiv:abs/1908.08017.
30. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How Transferable Are Features in Deep Neural Networks? In *Advances in Neural Information Processing Systems 27*; Curran Associates, Inc.: New York, NY, USA, 2014; pp. 3320–3328.
31. Gupta, A.; Ong, Y.; Feng, L. Multifactorial evolution: Toward evolutionary multitasking. *IEEE Trans. Evol. Comput.* **2016**, *20*, 343–357. [CrossRef]
32. Feng, L.; Zhou, L.; Zhong, J.; Gupta, A.; Ong, Y.; Tan, K.; Qin, A.K. Evolutionary multitasking via explicit autoencoding. *IEEE Trans. Cybern.* **2019**, *49*, 3457–3470. [CrossRef]
33. Gong, M.; Tang, Z.; Li, H.; Zhang, J. Evolutionary multitasking with dynamic resource allocating strategy. *IEEE Trans. Evol. Comput.* **2019**, *23*, 858–869. [CrossRef]
34. Frey, B.J.; Dueck, D. Clustering by passing messages between data points. *Science* **2007**, *315*, 972–976. [CrossRef] [PubMed]
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Boston, MA, USA, 7–12 June 2015; pp. 1026–1034.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
37. Goodfellow, I.; Warde-Farley, D.; Mirza, M.; Courville, A.; Bengio, Y. Maxout networks. In Proceedings of the 30th International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 1319–1327.
38. Lin, M.; Chen, Q.; Yan, S. Network in network. In Proceedings of the International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014; pp. 1–10.
39. Srivastava, R.K.; Greff, K.; Schmidhuber, J. Highway networks. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015; pp. 1–6.
40. Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; Kavukcuoglu, K. Hierarchical representations for efficient architecture search. *arXiv* **2017**, arXiv:1711.00436.