electronics

MDPI

Article

The Tabu_Genetic Algorithm: A Novel Method for Hyper-Parameter Optimization of Learning Algorithms

Baosu Guo [1,2], Jingwen Hu [1], Wenwen Wu [1], Qingjin Peng [3] and Fenghe Wu [1,2,*]

1 School of Mechanical Engineering, Yanshan University, Qinhuangdao 066004, China; guobaosu@163.com (B.G.); hjw1028@163.com (J.H.); ahwuwen@126.com (W.W.)
2 Heavy-duty Intelligent Manufacturing Equipment Innovation Center of Hebei Province, Qinhuangdao 066004, China
3 Department of Mechanical Engineering, University of Manitoba, Winnipeg, MB R3T 5V6, Canada; Qingjin.Peng@umanitoba.ca
* Correspondence: risingwu@ysu.edu.cn

Received: 30 April 2019; Accepted: 24 May 2019; Published: 25 May 2019

Abstract: Machine learning algorithms have been widely used to deal with a variety of practical problems such as computer vision and speech processing. But the performance of machine learning algorithms is primarily affected by their hyper-parameters, as without good hyper-parameter values the performance of these algorithms will be very poor. Unfortunately, for complex machine learning models like deep neural networks, it is very difficult to determine their hyper-parameters. Therefore, it is of great significance to develop an efficient algorithm for hyper-parameter automatic optimization. In this paper, a novel hyper-parameter optimization methodology is presented to combine the advantages of a Genetic Algorithm and Tabu Search to achieve the efficient search for hyper-parameters of learning algorithms. This method is defined as the Tabu_Genetic Algorithm. In order to verify the performance of the proposed algorithm, two sets of contrast experiments are conducted. The Tabu_Genetic Algorithm and other four methods are simultaneously used to search for good values of hyper-parameters of deep convolutional neural networks. Experimental results show that, compared to Random Search and Bayesian optimization methods, the proposed Tabu_Genetic Algorithm finds a better model in less time. Whether in a low-dimensional or high-dimensional space, the Tabu_Genetic Algorithm has better search capabilities as an effective method for finding the hyper-parameters of learning algorithms. The presented method in this paper provides a new solution for solving the hyper-parameters optimization problem of complex machine learning models, which will provide machine learning algorithms with better performance when solving practical problems.

Keywords: genetic algorithms; machine learning algorithms; neural networks; optimization methods; hyper-parameter optimization

1. Introduction

Learning algorithms such as deep learning are making major advances in solving problems that have resisted different attempts by the artificial intelligence community for many years [1,2]. The advantages of learning algorithms in discovering the structure of complex high-dimensional data make them widely used in many fields such as engineering technology and scientific development. However, due to the complexity of learning algorithms such as deep neural networks, their applications face enormous challenges. Prior to the advent of new and more effective learning algorithms, improving the performance of existing learning algorithms is a major task in the field of artificial intelligence. How to improve the performance of the learning algorithm when solving practical problems is a

common challenge faced by the academic community. As a sub-topic of this issue, choosing good hyper-parameter values for learning algorithms is also an effective way to improve the performance of learning algorithms.

The process of searching for good values of hyper-parameters in machine learning algorithms can be considered as an optimization process that attempts to find hyper-parameters for the best objective performance such as verification errors. The hyper-parameters optimization problem can be abstracted into a mathematical model as shown in Equation (1) [3]. In this model, A is a learning algorithm and λ is the hyper-parameters of A. The ultimate goal of the hyper-parameters optimization is to find the hyper-parameter combination $\lambda^{(*)}$ to minimize the generalization error $E_{\chi\sim G_\chi}\left[L\left(\chi; A_\lambda(\chi^{(train)})\right)\right]$ of the learning algorithm A on dataset $\chi^{(train)}$ (a finite set of samples from G_χ).

$$\lambda^{(*)} = \underset{\lambda\in\Lambda}{\operatorname{argmin}} E_{\chi\sim G_\chi}\left[L\left(\chi; A_\lambda(\chi^{(train)})\right)\right] \tag{1}$$

In the early period, the development of learning algorithms was in its infancy, the algorithm model was relatively simple, and the number of main hyper-parameters in the model was small. The manual search [4] was an effective way to determine hyper-parameters. Manual search does not require the high computational overhead, but requires a high level of expertise. Manual search is mainly suitable for solving the hyper-parameter selection problem of early simple learning algorithms. As learning algorithms become complex, the manual search has been difficult to use to provide good performance. At present, the automatic hyper-parameter optimization algorithm is the main trend in development, especially with the development of computer technology.

Larochelle et al. proposed a Grid Search [5] method for hyper-parameter optimization. The Grid search is simple to implement and reliable in the low-dimensional space, but it becomes trapped in dimensionality disasters in the high-dimensional space. It can be seen from Equation (1) that when the number of $\lambda^{(*)}$ is large and the range of each λ is wide, the search process will become complicated. The amount of computation required will increase exponentially, and the search algorithm will easily fall into the dimension disaster. Grid search has been proven to be easily fall into dimensionality disasters in a high-dimensional hyper-parameter space [3]. Hinton et al. [6] combined the manual search and grid search to train restricted Boltzmann machines. But this approach cannot completely avoid the shortcomings of the manual search and grid search. Bergstra et al. proposed a Random Search method [3] for the hyper-parameter optimization. This method is proven to be superior to the grid search, and the method is simple and easy to parallelize. However, the random search is somewhat similar to the grid search, and there is no guarantee that the result is optimal, or that the random search is non-adaptive.

Sequential model-based optimization methods [7,8] and particularly Bayesian optimization methods [9–14] were also used in the hyper-parameter optimization. Bayesian search uses a completely different approach compared to the grid search and random search, as the latter ignores the information of the previous search point in the search process, but Bayesian search makes full use of the previous search point information. The Bayesian optimization method also has its limitations. Once it searches for a maximum value point, it will continue to sample continuously around the highest value point, which makes it easy to fall into a local optimum. In addition, evaluating the hyper-parameter settings of the learning algorithm requires fitting the model and verifying its performance on the dataset. The computational overhead of this process is so huge that makes a sequential hyper-parameter optimization on a single computational unit impractical. It is undeniable that the sequence model based optimization and Bayesian optimization methods are essentially sequential. Although a certain degree of parallelization can be achieved by computing the optimal solution of multiple acquisition functions [15,16], a perfect parallelization is difficult to achieve.

Meta-heuristic algorithms are widely used to solve various optimization problems. In recent years, they have also been used for hyper-parameters optimization problems. Francescomarino et al. [17] and Zhang et al. [18] used Genetic Algorithms to optimize the hyper-parameters of learning algorithms and

achieved good results in practical problems. Young et al. [19] and Furtuna et al. [20] used Evolutionary Algorithms to optimize hyper-parameters of learning algorithms. As an improved evolutionary algorithm, CMA-ES (Covariance Matrix Adaptation Evolution Strategy) was also used to optimize learning algorithms [21]. Soon et al. [22] and Lorenzo et al. [23] used Particle Swarm Optimization to optimize deep neural network models. However, a single meta-heuristic algorithm has a common shortcoming. When the objective function is non-convex, it is easy fall into the local optimal solution. With the absence of a reasonable strategy to jump out of the optimal solution, a single meta-heuristic algorithm is difficult to use to find the optimal hyper-parameter combination.

In addition to the methods mentioned above, there are several recently proposed methods. Tang et al. [24] proposed a novel methodology that uses the geometric characteristics of line-segment representations to optimize hyper-parameters for deep networks. Diaz et al. [25] addressed the problem of choosing appropriate parameters for neural networks by formulating it as a box-constrained mathematical optimization problem, and applying a derivative-free optimization tool that automatically and effectively searches the parameter space. Maclaurin et al. [26] proposed a gradient-based hyper-parameter optimization method. However, their research is limited to the use of their own methods to optimize a model to solve practical problems, and is not compared to several hyper-parameter optimization algorithms that are widely considered to be excellent. Their methods are less versatile for different machine learning algorithms.

The existing hyper-parameter optimization methods have made great contributions in solving the problem of selecting appropriate hyper-parameters for learning algorithms, but they still have some problems. One of them is that the computational overhead during the search process is huge. Both the sequence model-based optimization and single meta-heuristic algorithm optimization methods have this problem. It is worth noting that with development of the computer hardware, especially the development of GPU, the computational overhead has been greatly reduced compared to the past. But for complex learning algorithms like deep neural networks, the optimization process can still take several weeks. Although several new methods have been proposed in recent years with good performance, they are limited to several specific models, and their generalization ability is poor.

The hyper-parameter optimization process of learning algorithms often wastes a lot of unnecessary computational overhead due to repeated searches of hyper-parameter points. In the existing hyper-parameter optimization method, a grid search can effectively avoid repeated searches of hyper-parameter points. But an obvious problem with the grid search is that the computational overhead grows exponentially as the number of hyper-parameters increases. We want to use genetic algorithms to optimize the hyper-parameters of deep neural networks, and to establish a mechanism like grid search to avoid repeated searches. For genetic algorithms, repeated searches are an inevitable problem. The mechanism for setting up a Tabu list in the Tabu search algorithm can effectively avoid repeated searches. In other heuristic algorithms, there is no suitable way to avoid repeated searches. Based on this, we have come up with the idea of combining Tabu search and genetic algorithms.

Considering the problem that the current method is easily fall into the local optimal solution and the computational cost is huge, we use genetic algorithms as the basic algorithm to solve this problem, and find ways to avoid falling into local optimal solutions in order to improve search efficiency. Through our research, a new hyper-parameter optimization method is proposed, this method is defined as the Tabu_Genetic Algorithm (Tabu_GA). In order to avoid falling into local optimal solutions, methods of the crossover, mutation and selection in Tabu_GA have been improved. At the same time, in order to reduce the computational overhead during the whole search process, the idea of setting up a Tabu list in Tabu Search has also been incorporated into Tabu_GA. Compared to the existing methods, this method can significantly reduce the computational overhead during the search process with good adaptability to different learning algorithms. In later experiments, Tabu_GA has been proven to be superior to several existing hyper-parametric optimization methods.

The key contribution of our research is the development of a Tabu_GA algorithm and applying it to better solve the hyper-parameter optimization problem of learning algorithms. This makes the learning algorithms such as deep learning perform better when solving practical problems.

2. Proposed Approach

The key of our approach is the combination of the advantages of Genetic Algorithm and Tabu Search to achieve the efficient search for hyper-parameters of learning algorithms. In order to describe the workflow of Tabu_GA, the basic principle of Genetic Algorithms and Tabu Search is introduced first. Working steps and specific operations of Tabu_GA will then be described in detail, including the selection of fitness functions and process of crossover, mutation and selection. Search for hyper-parameters using Tabu_GA will also be described.

2.1. Genetic Algorithms and Tabu Search

A genetic algorithm is a meta-heuristic algorithm. A genetic algorithm (GA) [27,28] simulates the evolutionary theory of nature and screen population individuals by establishing natural mechanisms similar to the survival of the fittest, to find an optimal solution or suboptimal solution of the optimization problem. Figure 1 shows the workflow of a simple GA. GAs have an outstanding performance in solving optimization problems in science and engineering. However, GAs also have their own shortcomings, such as easily premature convergence and falling into local optimal solutions.

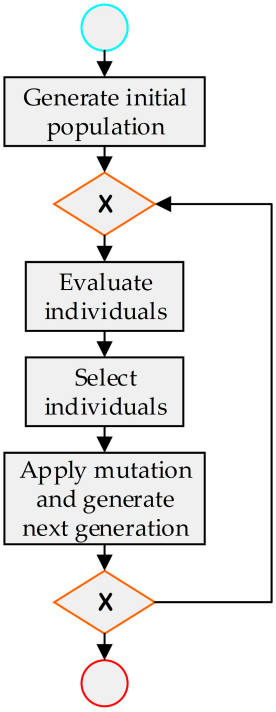

Figure 1. A simple GA workflow.

Tabu Search (TS) [29–31] algorithm is also a meta-heuristic algorithm. A single TS does not perform a global optimal search well. TS is better at solving local optimization problems compared to genetic algorithms. It uses history of the search to implement search strategies and avoid local minima. Its basic principle is modeled on the memory function of human beings. TS avoids repeated searches by setting up a Tabu list during the search process. A Tabu list is able to remember the information of solutions that has been searched to avoid them in subsequent searches. The use of this list prevents the return of the most recently accessed solution, effectively avoiding an endless loop. A simple TS algorithm is presented in Algorithm 1. [32].

Algorithm 1 the simple TS method

1: Choose, at random, an initial solution s in the search space
2: Tabu List ← ∅
3: **while** the stopping criterion is not satisfied do
4: Select the best solution $s' \in N(s) \setminus TabuList$
5: $s \leftarrow s'$
6: Update *TabuList*
7: **end**
8: **return** the best solution met

2.2. The Tabu_Genetic Algorithm

The optimization process by meta-heuristic algorithms can be considered as a black-box optimization, no derivative and gradient changes are involved during the whole process. In a sense, without considering the influence of some minor factors, the hyper-parameters optimization problem can also be regarded as a black-box optimization problem. Some scholars have carried out research on this area as mentioned in the previous section. It is therefore feasible to use the meta-heuristic algorithm while searching for the hyper-parameters of learning algorithms. However, GA will easily fall into local optimal solutions and continue to oscillate around this point. This will not only waste the computational overhead but also result in the final searched hyper-parameter values for learning algorithms not being optimal.

In order to prevent the GA from falling into the local optimal solution prematurely in the optimization process, we try to improve the genetic operation of the GA. Our starting point is to control the direction of population evolution by controlling the genetic operation of the GA to prevent pre-convergence. The improvement mainly includes methods of crossover, mutation and selection in Tabu_GA. At the same time, in order to reduce the computational overhead during the search process, the idea of the Tabu list in TS has also been incorporated into Tabu_GA. In the optimization process, it is inevitable to search for a combination of hyper-parameters with the same value, which will waste search time. By setting up a Tabu list, we can save the searched points into the list to avoid repeated searches and reduce the computation time. The basic workflow of Tabu_GA is shown in Figure 2.

Tabu_GA mainly consists of two parts including the GA part and TS part. As shown in Figure 2, on the left side of the flowchart is the GA part, and the right part is the TS Part. Firstly, GA is used to search the whole space to find the optimal solution or the sub-optimal solution. Then, TS is used to perform a local search on nearby points of the sub-optimal solution searched by GA to find the optimal solution. Tabu_GA still follows the basic steps of GA, including crossover, mutation, and so on.

First, hyper-parameters that need to be optimized are discretized and encoded and then an initial population is randomly generated. A fitness function is determined to evaluate the quality of individuals. Termination criteria are established for screening unqualified individuals. Crossover and mutation operations are performed on individuals who do not meet the termination criteria to generate new individuals. The above steps are repeated until an individual who meets the termination criteria is searched. The individual that satisfies the termination criterion will be considered as the optimal solution searched by the genetic algorithm part of Tabu_GA. However, the GA part of Tabu_GA may not generate the optimal solution. In order to obtain the optimal solution, a local search will be performed by the TS part of Tabu_GA.

In order to ensure that individuals from Tabu_GA are globally optimal, the Tabu list of TS and the local neighborhood search of Hill-Climbing Algorithm are combined in Tabu_GA to avoid falling into a local optimal solution. The TS part of Tabu_GA regards the optimal solution obtained by the GA as the initial point and compares it with neighboring nodes. If the neighboring nodes are much better, they will be used as the initial points to continue the search until the optimal solution is found.

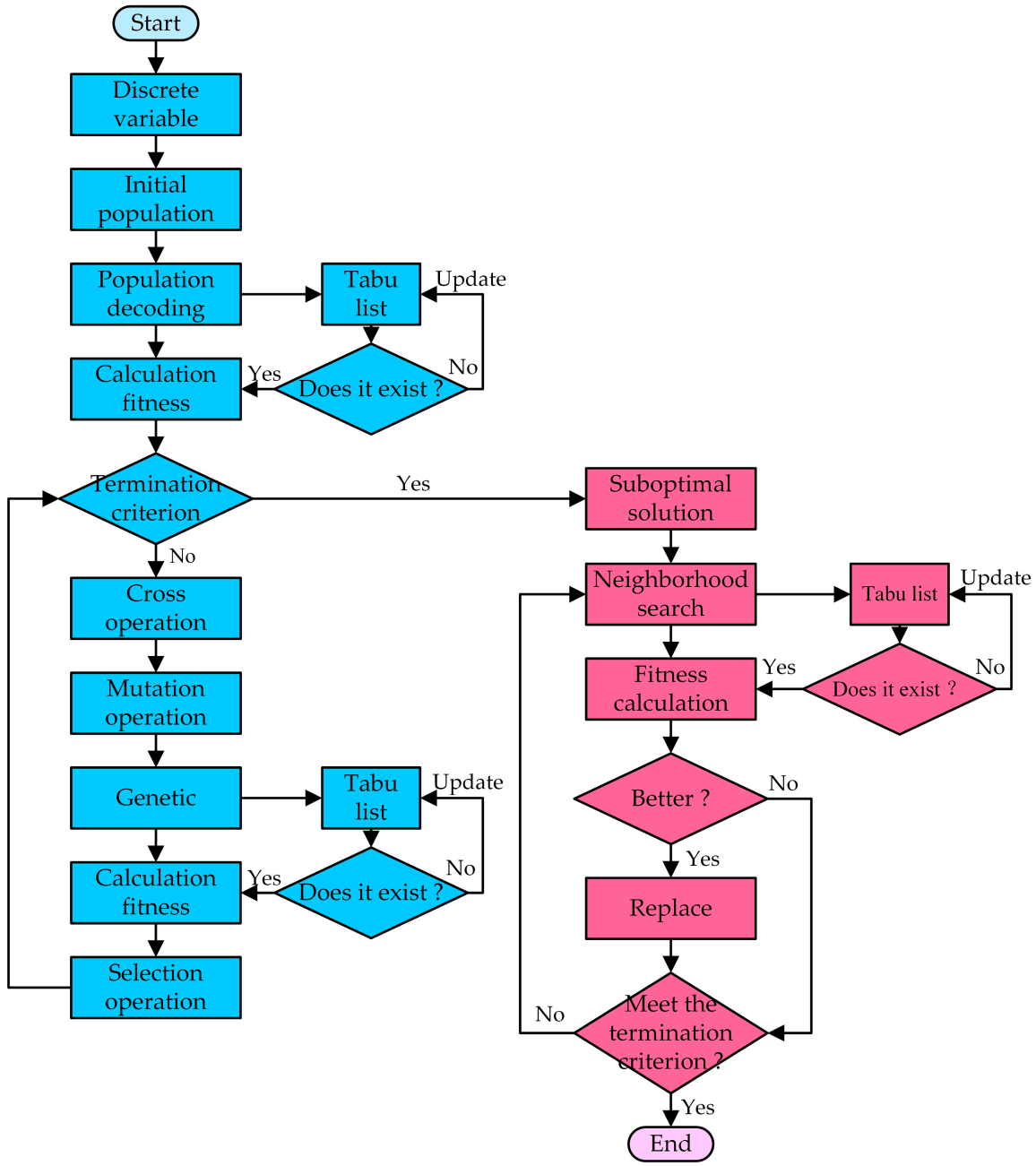

Figure 2. Tabu_GA workflow.

2.2.1. Fitness Function

In order to avoid over-fitting, the average accuracy of a validation set is used as a criterion for evaluating whether the training of learning algorithms is terminated [2]. In all experiments, training is stopped early if at any training epoch after first 100 iterations, the best accuracy on the validation set is achieved before the current number of epochs divided by two. The fitness function is defined in Equation (2). Here i represents the number of iterations on the validation set. This fitness function will be used as a criterion for the Tabu_GA to evaluation of the searched hyper-parameter points. The higher the value of the fitness function, the better the searched hyper-parameter point.

$$Fitness = mean \sum_{i=1}^{n} (accuracy\ of\ verification\ set)_i \quad (2)$$

2.2.2. Hyper-Parameters Discretization

The raw value of the hyper-parameter is a real number, which is not convenient for the input and calculation in the optimization process, so the hyper-parameter needs to be discretized. The discrete method of hyper-parameters mainly adopts the following two processes: (1) For the hyper-parameter with a large range of values that must be an integer, it is uniformly valued within its range of values; (2) For the hyper-parameter with a small range of values, it is sampled with a logarithmic scale. For example, the base learning rate and weight decay rate are on a log scale. In other words, the corresponding decision variables in Tabu_GA represent the log10 of values used to train the learning algorithm, rather than the values themselves: this can ensure more effective exploration of the hyper-parameter space. In the literature [3], the logarithmic scale is used to sample hyper-parameters in Random Search. We have adopted the same approach here.

In order to facilitate the cross-variation of the hyper-parameter individuals in subsequent experiments, the discrete hyper-parameters are encoded using binary.

2.2.3. Crossover, Mutation, and Selection Operation

A single-point crossover operation is used as the cross-operation of Tabu_GA. The basic principle of single point crossing is shown in Figure 3. Two different parent individuals, through a single-point crossover operation, produce two different offspring individuals. In order to ensure that the superior traits of parent individuals can be inherited to offspring individuals, some small changes follow. First, individuals are arranged according to values of the fitness function from small to large, and individuals with the highest fitness function value are then selected to cross-operate with remaining individuals to generate new individuals.

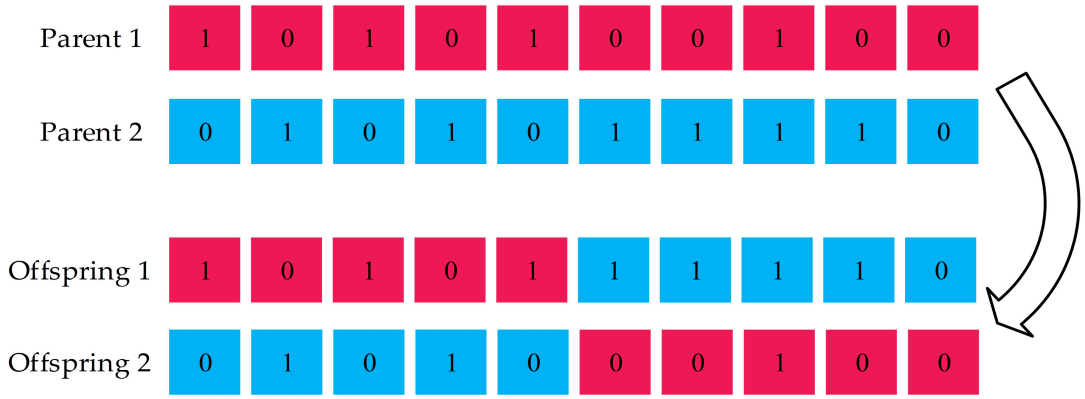

Figure 3. A single point crossover operation.

The mutation operation in Tabu_GA is different from traditional GA with following improvements. In this research, a new operation of mutation is proposed to define an arithmetic mutation operation. The arithmetic mutation operation lets each individual in the population have different mutation probabilities to present an arrangement of arithmetic progressions. The mathematical representation is described in Equation (3). Where n represents the number of individuals in the population, i represents the position of individuals after sorting, and P_{mi} represents probability of the mutation for individuals. It can be seen from Equation (3) that for individuals with less fitness, the probability of genetic mutation is greater. This mutation operation mainly includes following two advantages: (1) maintaining diversity of the population and preventing premature convergence; (2) jumping out of the local optimal solution effectively.

$$P_{mi} = 1 - \frac{i-1}{n} \ (1 \leq i \leq n) \tag{3}$$

The selection operation of Tabu_GA is also different from traditional GA. Selection operations of an evolutionary algorithm include $(\mu + \lambda)$ and (μ, λ) [32]. For the selection operation (μ, λ), although the pre-convergence at the local best point can be effectively avoided, the selection operation can only select the offspring generation without the parent information. This will result in the best individuals of the previous generation not being well preserved to the next generation. Relatively speaking, the selection operation $(\mu + \lambda)$ can preserve outstanding individuals for both parents and offspring individuals much better. Therefore, the selection operation $(\mu + \lambda)$ of the evolutionary algorithm is used in Tabu_GA. The selection operation is combined with above-mentioned operations of crossover and mutation to ensure correcting the wrong evolution direction in time and retaining the optimal individual in the previous generation.

3. Experiments

In order to verify the performance of Tabu_GA and prove that our improvement is meaningful, we have designed two sets of experiments, including main experiments and additional experiments. In the main experiments, we compared the performance of Tabu_GA with several algorithms that are currently considered to be excellent. In the additional experiments, we compared the optimization performance of Tabu_GA with GA (without Tabu Search) and Tabu Search (TS).

The experiments were performed on a computer equipped with NVIDIA GTX1080Ti GPU and Intel Core i7 CPU. The software platform is based on Tensorflow. Tabu_GA, GA and TS are programmed using Python 3. The other four methods are called from Hyperopt [33,34]. The experimental details are described below.

3.1. Main Experiments

In the main experiments, two sets of experiments were performed to verify the reliability and adaptability of Tabu_GA. In order to verify the performance of Tabu_GA on different datasets, experiments were conducted on two sets of datasets, one experiment on the MNIST dataset and the other on Flower-5 dataset. Convolutional neural networks were used as the optimization object in both experiments. In order to verify the search effect of Tabu_GA in the low-dimensional space and high-dimensional space, two different convolutional neural network models were selected. The experiment on the MNIST dataset selected the classic LetNet-5 convolutional neural network as the optimized object. The LetNet-5 convolutional neural network has a simple structure and a relatively small number of hyper-parameters. The experiment on the MNIST dataset simulates the hyper-parametric search process of relatively simple learning algorithms, which can also be considered as the search process in a low-dimensional space. The experiment on the Flower-5 dataset selected a more complex deep convolutional neural network model as the optimized object. The experiment on the Flower-5 dataset simulates the hyper-parametric search process of relatively complex learning algorithms, which can also be considered as the search process in a high-dimensional space. Both results of experiments are used to compare the performance of Tabu_GA with the Bayesian optimization method, Simulated Annealing Algorithm (SA), CMA-ES and Random Search. Bayesian optimization methods, simulated annealing, CMA-ES and random search are several hyper-parametric optimization algorithms that generally considered to be excellent methods applied in the field. Therefore, we chose these four methods to compare with Tabu_GA in the experiment. The Bayesian optimization method chosen for this experiment is Tree-structured Parzen Estimator Approach (TPE) [10].

3.1.1. Convolutional Neural Network

Convolutional Neural Network (CNN) is a well-known deep learning architecture inspired by the natural visual perception mechanism of living creatures [35]. The convolutional neural network relies on convolution and pooling operations to identify information. It has been widely used in target detection, image classification [36] and other fields [37]. There are many types of convolutional neural networks, such as LetNet-5 [38,39] and AlexNet [40]. The experiment on the MNIST dataset selected

the classic LetNet-5 convolutional neural network as the optimized object. The LetNet-5 convolutional neural network is an early convolutional neural network model with a simple structure and is also trained through back-propagation. The experiment on the Flower-5 dataset selected a more complex deep convolutional neural network model as the optimized object. The model can be found on GitHub (https://github.com/deep-diver/CIFAR10-img-classification-tensorflow) and its structure is shown in Figure 4, which consists of four convolutional layers, four pooling layers, and four hidden layers. It is a deep convolutional neural network model.

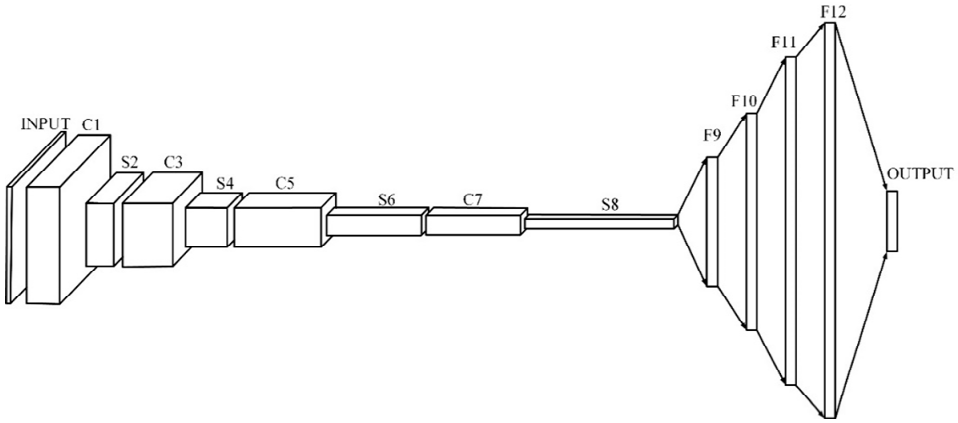

Figure 4. Structure of the convolutional neural network for experiments on the Flower-5 dataset.

3.1.2. Experiment on MNIST Dataset

The MNIST dataset (http://yann.lecun.com/exdb/mnist/) is a classic dataset used in classification problems of machine learning. The dataset consists of 60,000 of grayscale handwritten digits, which are divided into 10 categories, i.e. numbers 0-9. The 60000 images are divided into verification sets and training sets. In many papers and practical questions, the MNIST data set is used to evaluate the performance of the model. Figure 5 are some examples of handwriting in the MNIST data set.

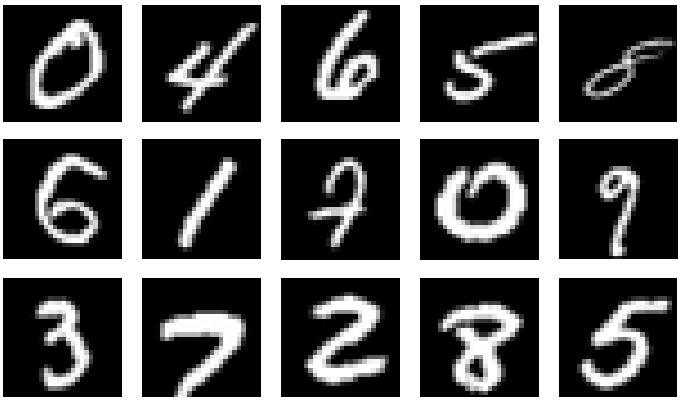

Figure 5. MNIST handwriting data set.

Hutter et al. proposed an efficient approach for assessing the hyper-parameter importance [41] and proved that even in very high dimensional cases—the most performance variation is attributable to just a few hyper-parameters. Five important hyper-parameters of the LetNet-5 convolutional neural network are selected as objects to search, including learning rate, batch size, number of full-connected layer1 (F1) units, dropout rate and L2 weight decay. For the learning rate, the dropout rate and L2 weight decay will use the log scale sampling mentioned in Section 2, while the batch size and Number of F1 units use uniform sampling. The range of values for each hyper-parameter is shown in Table 1.

This gets about 1.96 million hyper-parameter combinations after discretizing the five hyper-parameters. Table 2 shows some parameters and settings of the LetNet-5 convolutional neural network that do not change during the experiment.

Table 1. The range of values for the five hyper-parameters of LetNet-5 convolutional neural network participating in the optimization.

Name	Ranges
Learning rate	[0.0001,0.1]
Batch size	[16,128]
Number of F1 units	[128,1024]
Dropout rate	[0.1,0.5]
L2 weight decay	[0.0001,0.01]

Table 2. Some parameters and settings of the LetNet-5 convolutional neural network that do not change in the experiment.

Name	Value
Convolution layer 1	Filter:2, strides:1, padding:SAME
Pooling layer 1(max)	ksize:2, strides:1
Convolution layer 2	Filter:5, strides:1, padding:SAME
Pooling layer 2(max)	ksize:2, strides:1
Full-connected layer 1	Dropout
Number of iterations	10,000
Learning rate decay policy	Step (gamma = 0.99, step size = 100)
Activation function	Relu

A standard for evaluating algorithms in experiments is the smallest validation error found in all epochs when the training time (including the time spent on model building) is limited [21]. The generalization error of learning algorithms on the verification set is one of the important indicators to measure its performance. This can indirectly reflect the performance of the optimization method. The relevant settings during the experiment are shown in Table 3.

Table 3. Relevant settings in the experiments on MNIST.

Method	Experiment Setup
Random search	Conduct 610 assessments
Simulated Annealing	Conduct 610 assessments
CMA-ES	Conduct 610 assessments (61 generations, 10 individuals per generation), $\mathcal{N}$ (0.5, 0.2^2)
TPE	Conduct 610 assessments
Tabu_GA	The genetic algorithm part was carried out for 59 generations, with 10 individuals per generation, for a total of 600 assessments; The Tabu search section was evaluated 10 times; For a total of 610 evaluations.

The experimental results on the MNIST dataset are shown in Figure 6. Figure 6 depicts the verification set error curve of the convolutional neural network model optimized by the five optimization methods in the experiment on the MNIST data set. It can be seen from Figure 6 that the convolutional neural network model optimized by Random Search performs poorly. In contrast, the performance of the convolutional neural network model optimized by the other four methods is basically consistent. The model optimized by CMA-ES finally achieved the best performance. But that was after nearly 250 evaluation searches. Tabu_GA found the sub-optimal model with less than 100 searches. Tabu_GA finds a better model in less time, which is preferable. Tabu_GA is superior to the Bayesian optimization

and random search. Experiments on the MNIST dataset demonstrate that Tabu_GA is efficient for searching good values of hyper-parameters for simple learning algorithms, to reach or surpass the level of several well-known methods.

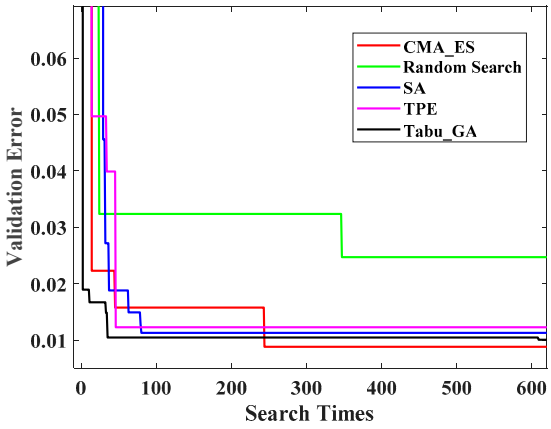

Figure 6. Average verification error curves for five methods in experiments on the MNIST dataset.

3.1.3. Experiment on Flower-5 Dataset

Another set of experiments was performed on the Flower-5 dataset. Flower-5 dataset (https://www.kaggle.com/alxmamaev/flowers-recognition) includes approximately 4000 images of five type flowers, includes daisy, dandelion, rose, sunflower, and tulip, as shown in Figure 7. There are about 800 images for each type of flower. The Flower-5 dataset is more complex than the MNIST dataset, and is very sensitive to the setting of hyper-parameters. The difficulty of performing classification tasks on the Flower-5 dataset is greater than on the MNIST dataset.

Figure 7. Flower-5 Dataset.

The experiment on the Flower-5 dataset selected a more complex deep convolutional neural network model as the optimized object. The model can be found on GitHub (https://github.com/deep-diver/CIFAR10-img-classification-tensorflow) and its structure is shown in Figure 4, which consists of four convolutional layers, four pooling layers, and four hidden layers. It is a deep convolutional neural network model. The eleven hyper-parameters of this deep convolutional neural network model were selected as search objects for this experiment, as shown in Table 4. The experiment steps performed on the Flower-5 data set were identical to the experimental performed on the MNIST data set. Only some specific details are different.

Table 4. The range of values for eleven hyper-parameters of the deep convolutional neural network in the optimization.

Name	Ranges
Learning rate	[0.0001,0.1]
Batch size	[16,128]
Number of F1 units	[128,1024]
Number of F2 units	[128,1024]
Number of F3 units	[128,1024]
Number of F4 units	[128,1024]
L2 weight decay	[0.0001,0.1]
Dropout rate1	[0.1,0.5]
Dropout rate2	[0.1,0.5]
Dropout rate3	[0.1,0.5]
Dropout rate4	[0.1,0.5]

Experiments performed on the Flower-5 dataset still use the method described in Section 2 to discretize the hyper-parameters with the same coding method. There are about 5×10^{12} pairs of hyper-parameter combinations after eleven hyper-parameters are discretely completed. Table 5 shows parameters and settings of the deep convolutional neural network that do not change during the experiment. Relevant settings in the experiment are shown in Table 6. The evaluation criteria for the method in the experiment are same as the experiment on the MNIST dataset.

Table 5. Parameters and settings of the deep convolutional neural network that do not change in the experiment.

Name	Value
Convolution layer 1	Filter:3, strides:1, padding:SAME
Pooling layer 1(max)	ksize:2, strides:2
Convolution layer 2	Filter:3, strides:1, padding:SAME
Pooling layer 2(max)	ksize:2, strides:2
Convolution layer 3	Filter:5, strides:1, padding:SAME
Pooling layer 3(max)	ksize:2, strides:2
Convolution layer 4	Filter:5, strides:1, padding:SAME
Pooling layer 4(max)	ksize:2, strides:2
Full-connected layer 1	Dropout
Full-connected layer 2	Dropout
Full-connected layer 3	Dropout
Full-connected layer 4	Dropout
Number of iterations	20,000
Learning rate decay policy	Step (gamma = 0.99, step size = 100)
Activation function	Relu

Table 6. Relevant settings in experiments on Flower-5.

Method	Experiment Setup
Random search	Conduct 622 assessments
Simulated Annealing	Conduct 622 assessments
CMA-ES	Conduct 620 assessments (62 generations, 10 individuals per generation), $\mathcal{N}$ $(0.5, 0.2^2)$ [21]
TPE	Conduct 610 assessments
Tabu_GA	The genetic algorithm part was carried out for 59 generations, with 10 individuals per generation, for a total of 600 assessments; The Tabu search section was evaluated 22 times; For a total of 622 evaluations.

Experimental results on the Flower-5 dataset are shown in Figure 8. Figure 8 depicts verification set error curves of the deep convolutional neural network model optimized by five optimization

methods in the experiment on the Flower-5 dataset. It can be seen from Figure 8 clearly that the deep convolutional neural network model optimized by SA has an extremely poor performance. In order to prove that this situation is not caused by mistakes in experimental operations, a second identical experiment was performed according to the same criteria. In the end, we found that this situation was not caused by mistakes in experimental operations. This phenomenon is caused by defects of SA itself. The search process of SA falls into a local optimal solution and continues to oscillate around it. This shows that SA is not suitable for use in the high-dimensional space. The performance of the model optimized by the other four methods is basically the same. However, Tabu_GA is still the best. Compared to the Bayesian optimization method, CMA-ES and Random Search, the model searched by Tabu_GA performs better. Tabu_GA finds a better model in less time. Experiments on the Flower-5 dataset demonstrate that Tabu_GA has good search capabilities even in high-dimensional cases and is superior to Random Search and Bayesian optimization methods.

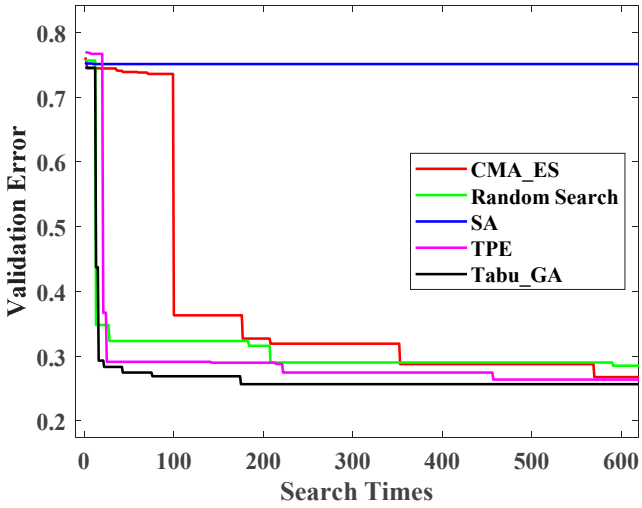

Figure 8. Average verification error curves for five methods in experiments on the Flower-5 dataset.

3.1.4. Summary of the Main Experiment

In order to further compare the five methods involved in the experiment, the experimental results are specifically shown in Tables 7 and 8, including the maximum number of assessments that reach the optimal model, the maximum accuracy that can be achieved, and the average accuracy. Both results of experiments on MNIST and Flower-5 show that Tabu_GA is more preferable. As shown in Table 7, Tabu_GA does not significantly improve the classification accuracy on the MNIST dataset. The MNIST data set is easy to be classified, and the convolutional neural network can achieve a high classification accuracy. However, based on solutions in Table 7, Tabu_GA reduces the number of iterations required for the model to achieve the highest classification accuracy, and improve the classification efficiency. In addition, Table 8 shows that for a more complex data set such as Flower-5, Tabu_GA improves the classification accuracy of the model for more than 1% by optimizing the hyper-parameter of the model. This means that Tabu_GA can bring great benefits in the face of complex data sets and models. The above analysis results also show that Tabu_GA is feasible and effective for the hyper-parameter optimization of learning algorithms.

Table 7. Performance comparisons of five optimization methods on the MNIST dataset experiment.

Algorithm	MNIST		
	Average Accuracy	**Maximum Precision**	**Number of Evaluations**
Random search	0.9753	0.9849	610
Simulated annealing	0.9812	0.9858	610
TPE	0.9877	0.9932	610
CMA_ES	**0.9912**	0.9942	610
Tabu_GA	0.9899	**0.9944**	**313**

Table 8. Performance comparisons of five optimization methods on the Flower-5 dataset experiment.

Algorithm	Flower-5		
	Average Accuracy	**Maximum Precision**	**Number of Evaluations**
Random search	0.7150	0.7624	622
Simulated annealing	0.2488	0.2488	622
TPE	0.7364	0.7738	622
CMA_ES	0.7325	0.7735	620
Tabu_GA	**0.7434**	**0.7881**	**609**

3.2. The Additional Experiment

The purpose of our additional experiments was to compare the performance of Tabu_GA, GA and Tabu Search in optimization, and to analyze whether our improvements are meaningful. Although the previous experimental results have shown that Tabu_GA is better than several excellent algorithms, this is not enough strong evidence showing the improvement. In order to further verify the experimental solutions, we conducted additional experiments.

The additional experiments used the same data set and convolutional neural network model as the previous experiments did, and were conducted under the same experimental environment on the MNIST dataset and Flower-5 dataset, respectively. The same GA (without tabu search) and TS were chosen as the comparison object for Tabu_GA.

From experimental results shown in Figures 9 and 10, Tabu_GA performs better than GA and TS, both on the experiment of MNIST dataset and Flower-5 dataset. We also found that TS can achieve better results in optimizing hyper-parameters of the model. According to our survey, no scholars have used TS to optimize the hyper-parameters in deep neural networks or other machine learning algorithms. In addition, the experimental results also show that advantages of Tabu_GA compared to GA and TS become more apparent as the complexity of data sets and models increases. This can be seen from Figures 9 and 10. The verification error of the deep convolutional neural network model optimized by Tabu_GA on the Flower-5 dataset is about 3.7% lower than that of GA. Results of additional experiments show that our improvement is correct and meaningful. Tabu_GA has better performance than GA and TS. Compared to GA and TS, Tabu_GA has obtained a better convolutional neural network model with less time.

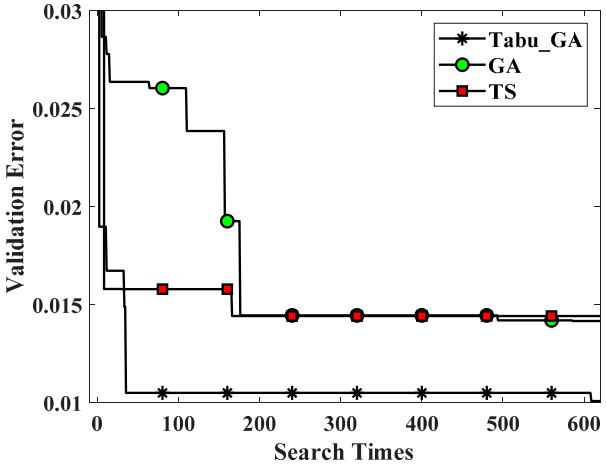

Figure 9. Average verification error curves for GA, TS, and Tabu_GA in additional experiments on the MNIST dataset.

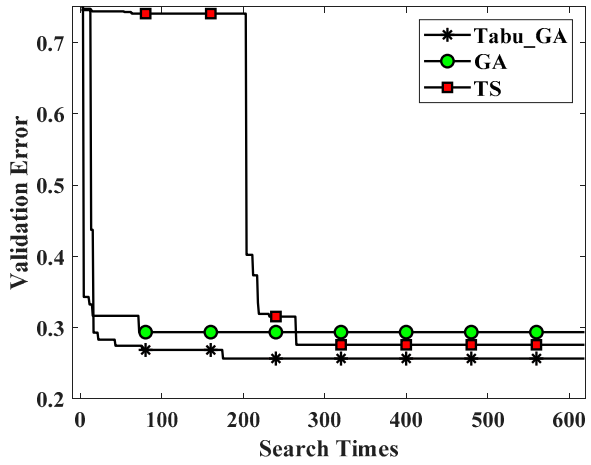

Figure 10. Average verification error curves for GA, TS, and Tabu_GA in additional experiments on the Flower-5 dataset.

3.3. Stability Analysis of Tabu_GA

In the above experiment, the hyper-parameter value was first generated by a set of random seeds and different random seeds may cause different experimental results. In order to analyze whether the optimized performance of Tabu_GA is affected by this randomness, we performed five additional experiments (different random seeds for each set of experiments). It should be noted that there are no other differences between the five sets of experiments except for the random seeds. The experimental setup is the same as the experiment in Section 3.1. We use the standard deviation and mean of experimental results to show the effect of random seeds on Tabu_GA. We separately consider the mean and standard deviation of the maximum and average accuracy that the model can achieve in each set of experiments although getting the best model is our ultimate goal.

The experimental results are shown in Tables 9 and 10 for the mean and standard deviation of multiple sets of experimental results on MNIST and Flower-5. As can be seen from Table 9, results of five sets experiments show that the model optimized by Tabu_GA can achieve a high classification accuracy on the MNIST dataset. The mean of both the average accuracy and maximum accuracy remains at a higher value. Moreover, the standard deviation of the maximum accuracy and average accuracy is also in a very small range. This shows the fact that the final experimental results of the five sets of experiments are almost identical, and that Tabu_GA is stable in optimizing the LetNet-5

convolutional neural network model. Similarly, experimental results on the Flower-5 data set can be seen in Table 10. The mean of the average accuracy is smaller than the mean of the maximum accuracy. The main reason of this phenomenon is that Flower-5 is a relatively complex data set, and the convolutional neural network cannot achieve a good classification accuracy in the initial stage of optimization. Therefore, when calculating the average accuracy of a set of experimental results, it is bound to occur that the final mean is lower due to the low classification accuracy in the initial stage of optimization. This is different from the experiment on the MNIST dataset. Because the MNIST data set is easier to classify, and the convolutional neural network model can achieve the high classification accuracy with only a small number of optimization iterations. In Table 10, the maximum accuracy of the five sets of experiments is at a higher level than the maximum accuracy achieved by the other four methods as shown in Table 8. In addition, based on data in Table 10, the standard deviation of the average accuracy and maximum accuracy of the five sets of experiments on the Flower-5 data set is also in a small range. This also shows that the optimization performance of Tabu_GA is not affected by randomness. Therefore, Tabu_GA is sufficiently stable.

Table 9. Mean and standard deviation of five sets of experimental results on MNIST.

	Average (Accuracy)	**Max (Accuracy)**
Experiment 1	0.9897	0.9940
Experiment 2	0.9914	0.9944
Experiment 3	0.9897	0.9931
Experiment 4	0.9899	0.9931
Experiment 5	0.9899	0.9944
Mean	**0.99012**	**0.99380**
Standard Deviation	**0.000646**	**0.000590**

Table 10. Mean and standard deviation of five sets of experimental results on Flower-5.

	Average (Accuracy)	**Max (Accuracy)**
Experiment 1	0.7381	0.7851
Experiment 2	0.7393	0.7850
Experiment 3	0.7401	0.7825
Experiment 4	0.7457	0.7824
Experiment 5	0.7434	0.7881
Mean	**0.74132**	**0.78462**
Standard Deviation	**0.002809**	**0.002093**

By analyzing experimental results of the above 10 sets of experiments, it can be concluded that the optimization performance of Tabu_GA is not affected by randomness. This also verifies that Tabu_GA has good stability and adaptability.

4. Discussion

The hyper-parameter optimization problem of learning algorithms is currently the main challenge for its application. Before the advent of better learning algorithms, the key to solving this problem is to find better hyper-parametric optimization algorithms. Excellent hyper-parameter optimization methods will make the current learning algorithm obtain better performance in solving practical problems.

For a long time, Gird Search, Random Search and Bayesian optimization methods have been considered as the most effective way to solve the hyper-parameter optimization problem of learning algorithms. They are commonly used as natural benchmarks. The optimization method based on heuristic algorithms and other methods proposed in recent years are also used in the hyper-parameter optimization problem of learning algorithms, but the performance is general, and most of them are only suitable for solving a specific problem without universality. Heuristic algorithms are not the first choice among the popular hyper-parameter optimization methods. However, our results suggest that heuristic algorithms have great potential in solving the hyper-parameter optimization

problem of learning algorithms. The proposed Tabu_GA is superior to the existing popular methods in performance. This provides a new research idea for solving the problem of learning algorithms for the hyper-parameter optimization, which is to study the hyper-parameter optimization method based on heuristic algorithms. Although we did not verify the performance of Tabu_GA through a real data set, we still recommend Tabu_GA as a method to optimize the hyper-parameter of the learning algorithm in practical applications. The performance of Tabu_GA on the MNIST dataset and Flower-5 dataset can be representative of the actual application to a certain degree.

The results of this work also suggest that the research on the hybrid meta-heuristic algorithm is meaningful. Hybrid optimization methods will be a major trend in the future. This is not limited to the hybridization between different meta-heuristics, but also the hybrid use of meta-heuristics and other algorithms. The combined use of two different methods provides the original strengths of both, and the two methods complement each other. Tabu_GA is also a hybrid meta-heuristic. The performance of Tabu_GA also shows the value of the above development direction.

We found the Tabu_GA when solving the hyper-parameter optimization problem of convolutional neural networks, and achieved good results. Tabu_GA is essentially an optimization method. In a sense, Tabu_GA can be used to solve other optimization problems. As it is limited to the specific problems discussed in this article, we have not proved its application in other problems, but this is an area worthy of study, and may achieve unexpected results.

5. Conclusions

In this work, a new method of Tabu_GA is proposed for the hyper-parameter optimization of learning algorithms such as deep neural networks.

The experiments prove that Tabu_GA is an excellent hyper-parameter optimization method. Experiments on both the MNIST and Flower-5 data sets show that the effectiveness of Tabu_GA. In dealing with the hyper-parameter optimization problem of simple learning algorithm, it is verified by experiments on the MNIST dataset. Moreover, experiments on the MNIST dataset also show that Tabu_GA can find a better solution while reducing the computation time by 50% compared to the current mainstream optimization methods. Another set of experiments on the Flower-5 dataset shows that Tabu_GA is also excellent for the optimization of complex learning algorithms like deep neural networks. Although this advantage is not obvious, it exists objectively. Through these two sets of experiments, we can conclude that Tabu_GA can be used as a hyper-parameter optimization method for learning algorithms. Tabu_GA is equally applicable for both simple and complex learning algorithms.

The additional experimental results also prove that the proposed improvement method for GA is effective. Improving methods of the crossover, mutation and selection enable Tabu_GA avoid falling into local optimal solutions effectively. At the same time, the idea of setting up a Tabu list in Tabu_GA clearly reduces the computational overhead during the whole search process. Moreover, we analyzed the mean and standard deviation of the results in multiple sets experiments, and found that the optimization performance of Tabu_GA is stable and not affected by randomness.

In fact, based on the experimental results, we can conclude that the Tabu_GA can be used as a method to optimize the hyper-parameter of learning algorithms in a good adaptability. However, the Tabu list in our algorithm needs the extra memory during the search process, the memory demand will be increased as the depth of the search increases. We will continue to optimize Tabu_GA in the future work. In addition, Tabu_GA seems more suitable for optimizing complex models with more parameters. For a simple model with a small number of parameters, although Tabu_GA can achieve better optimization results, the cost performance will be not as good as some simple traditional methods.

Author Contributions: Conceptualization, B.G., J.H. and W.W.; Funding acquisition, B.G. and F.W.; Investigation, J.H.; Methodology, J.H. and W.W.; Resources, B.G. and F.W.; Software, J.H. and W.W.; Supervision, F.W.; Validation, B.G., Q.P. and F.W.; Visualization, W.W.; Writing—original draft, W.Wu; Writing—review & editing, B.Guo, J.Hu and Q.P.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant (51890881, 51605422), in part by the Natural Science Foundation of Hebei Province under Grant (E2017203156,

E2017203372), in part by the Science and Technology Projects of Universities in Hebei Province under Grant QN2016004, in part by the Chinese National Key Research and Development Program under Grant 2016YFC0802900, in part by the Open Project Program of Jiangsu Key Laboratory of Precision and Micro-Manufacturing Technology.

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

References

1. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
2. Alom, Z.M.; Taha, M.T.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, S.M.; Hasan, M.; Van Essen, C.B.; Awwal, A.A.; Asari, K.V. A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics* **2019**, *8*, 292. [CrossRef]
3. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
4. LeCun, Y.; Bottou, L.; Orr, G.B.; Müller, K.R. *Efficient BackProp*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 9–50.
5. 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, Corvalis, OR, USA, 20–24 June 2007; pp. 473–480.
6. Hinton, G.E. *A Practical Guide to Training Restricted Boltzmann Machines*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 599–619.
7. Talathi, S.S. Hyper-parameter optimization of deep convolutional networks for object recognition. In Proceedings of the 2015 IEEE International Conference on Image Processing (ICIP), Quebec City, QC, Canada, 27–30 September 2015; pp. 3982–3986.
8. Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. In Proceedings of the International Conference on Learning and Intelligent Optimization, Rome, Italy, 17–21 January 2011; pp. 507–523.
9. Swersky, K.; Snoek, J.; Adams, R. Freeze-Thaw Bayesian Optimization. *arXiv* **2014**. Available online: https://arxiv.org/abs/1406.3896 (accessed on 16 June 2014).
10. Bergstra, J.S.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for hyper-parameter optimization. In Proceedings of the Advances in neural information processing systems, Granada, Spain, 12–15 December 2011; pp. 2546–2554.
11. Snoek, J.; Larochelle, H.; Adams, R.P. Practical bayesian optimization of machine learning algorithms. In Proceedings of the 26th Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 Decmber 2012; pp. 2951–2959.
12. Zeng, X.; Luo, G. Progressive sampling-based Bayesian optimization for efficient and automatic machine learning model selection. *Health Inf. Sci. Syst.* **2017**, *5*. [CrossRef] [PubMed]
13. Eggensperger, K.; Feurer, M.; Hutter, F.; Bergstra, J.; Snoek, J.; Hoos, H.; Leyton-Brown, K. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In Proceedings of the NIPS workshop on Bayesian Optimization in Theory and Practice, Lake Tahoe, NV, USA, 10 Decmber 2013.
14. Bergstra, J.; Yamins, D.; Cox, D.D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the 30th International Conference on International Conference on Machine Learning, Atlanta, GA, USA, 16–21 June 2013; pp. 115–123.
15. Chevalier, C.; Ginsbourger, D. Fast computation of the multi-points expected improvement with applications in batch selection. In Proceedings of the International Conference on Learning and Intelligent Optimization, Catania, Italy, 7–11 January 2013; pp. 59–69.
16. Desautels, T.; Krause, A.; Burdick, J.W. Parallelizing Exploration-Exploitation Tradeoffs in Gaussian Process Bandit Optimization. *J. Mach. Learn. Res.* **2014**, *15*, 3873–3923.
17. Di Francescomarino, C.; Dumas, M.; Federici, M.; Ghidini, C.; Maggi, F.M.; Rizzi, W.; Simonetto, L. Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Inf. Syst.* **2018**, *74*, 67–83. [CrossRef]
18. Zhang, Y.Y.; Huang, G. traffic flow prediction model based on deep belief network and genetic algorithm. *IET Intell. Transp. Syst.* **2018**, *12*, 533–541. [CrossRef]
19. Young, S.R.; Rose, D.C.; Karnowski, T.P.; Lim, S.-H.; Patton, R.M. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, Austin, TX, USA, 15 November 2015.

20. Furtuna, R.; Curteanu, S.; Leon, F. Multi-objective optimization of a stacked neural network using an evolutionary hyper-heuristic. *Appl. Soft Comput.* **2012**, *12*, 133–144. [CrossRef]
21. Loshchilov, I.; Hutter, F. CMA-ES for hyperparameter optimization of deep neural networks. *arXiv* **2016**. Available online: https://arxiv.org/abs/1604.07269 (accessed on 25 April 2016).
22. Soon, F.C.; Khaw, H.Y.; Chuah, J.H.; Kanesan, J. Hyper-parameters optimisation of deep CNN architecture for vehicle logo recognition. *IET Intell. Transp. Syst.* **2018**, *12*, 939–946. [CrossRef]
23. Lorenzo, P.R.; Nalepa, J.; Ramos, L.S.; Pastor, J.R. Hyper-parameter selection in deep neural networks using parallel particle swarm optimization. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin, Germany, 15–19 July 2017; pp. 1864–1871.
24. Tang, X.S.; Ding, Y.S.; Hao, K.R. A Novel Method Based on Line-Segment Visualizations for Hyper-Parameter Optimization in Deep Networks. *Int. J. Pattern Recognit. Artif. Intell.* **2018**, *32*. [CrossRef]
25. Diaz, G.I.; Fokoue-Nkoutche, A.; Nannicini, G.; Samulowitz, H. An effective algorithm for hyperparameter optimization of neural networks. *IBM J. Res. Dev.* **2017**, *61*, 1–11. [CrossRef]
26. Maclaurin, D.; Duvenaud, D.; Adams, R. Gradient-based hyperparameter optimization through reversible learning. In Proceedings of the 32th International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 2113–2122.
27. Sastry, K.; Goldberg, D.; Kendall, G. Genetic Algorithms. In *Search Methodologies*; Springer: Boston, MA, USA, 2005; pp. 97–125.
28. Mitchell, M. Genetic algorithms: An overview. *Complexity* **1995**, *1*, 31–39. [CrossRef]
29. Gendreau, M.; Potvin, J.-Y. Tabu Search. In *Handbook of Metaheuristics*; Springer: Boston, MA, USA, 2010; pp. 41–59.
30. Glover, F. Tabu Search—Part I. *ORSA J. Comput.* **1989**, *1*, 190–206. [CrossRef]
31. Glover, F. Tabu Search—Part II. *ORSA J. Comput.* **1990**, *2*, 4–32. [CrossRef]
32. Boussaid, I.; Lepagnot, J.; Siarry, P. A survey on optimization metaheuristics. *Inf. Sci.* **2013**, *237*, 82–117. [CrossRef]
33. Komer, B.; Bergstra, J.; Eliasmith, C. Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn. In Proceedings of the 13th Annual Scientific Computing with Python Conference, Austin, TX, USA, 6–12 July 2014; pp. 2825–2830.
34. Bergstra, J.; Bardenet, R.; Kégl, B.; Bengio, Y. Implementations of algorithms for hyper-parameter optimization. In Proceedings of the 24th International Conference on Neural Information Processing Systems, Granada, Spain, 12–15 December 2011.
35. Gu, J.X.; Wang, Z.H.; Kuen, J.; Ma, L.Y.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.X.; Wang, G.; Cai, J.F.; et al. Recent advances in convolutional neural networks. *Pattern Recognit.* **2018**, *77*, 354–377. [CrossRef]
36. Kwan, C.; Chou, B.; Bell, F.J., III. Comparison of Deep Learning and Conventional Demosaicing Algorithms for Mastcam Images. *Electronics* **2019**, *8*, 308. [CrossRef]
37. Zahid, M.; Ahmed, F.; Javaid, N.; Abbasi, A.R.; Zainab Kazmi, S.H.; Javaid, A.; Bilal, M.; Akbar, M.; Ilahi, M. Electricity Price and Load Forecasting using Enhanced Convolutional Neural Network and Enhanced Support Vector Regression in Smart Grids. *Electronics* **2019**, *8*, 122. [CrossRef]
38. LeCun, Y.; Boser, B.E.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.E.; Jackel, L.D. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems 2*; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1990; pp. 396–404.
39. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
40. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the 26th Conference on Neural Information Processing Systems (NIPS 2012), Lake Tahoe, NV, USA, 3–8 December 2012; pp. 1097–1105.
41. Hoos, H.; Leyton-Brown, K. An efficient approach for assessing hyperparameter importance. In Proceedings of the 31th International Conference on Machine Learning, Beijing, China, 22–26 June 2014; pp. 754–762.