electronics

Article

Quantum Dynamic Optimization Algorithm for Neural Architecture Search on Image Classification

Jin Jin [1], Qian Zhang [2], Jia He [3,*] and Hongnian Yu [4]

[1] School of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China
[2] Active Network (Chengdu) Co., Ltd., Chengdu 610021, China
[3] School of Computer Science, Chengdu University of Information Technology, Chengdu 610225, China
[4] School of Computing, School of Engineering and the Built Environment, Edinburgh Napier University, Edinburgh 16140, UK
* Correspondence: hejia@cuit.edu.cn

Abstract: Deep neural networks have proven to be effective in solving computer vision and natural language processing problems. To fully leverage its power, manually designed network templates, i.e., Residual Networks, are introduced to deal with various vision and natural language tasks. These hand-crafted neural networks rely on a large number of parameters, which are both data-dependent and laborious. On the other hand, architectures suitable for specific tasks have also grown exponentially with their size and topology, which prohibits brute force search. To address these challenges, this paper proposes a quantum dynamic optimization algorithm to find the optimal structure for a candidate network using Quantum Dynamic Neural Architecture Search (QDNAS). Specifically, the proposed quantum dynamics optimization algorithm is used to search for meaningful architectures for vision tasks and dedicated rules to express and explore the search space. The proposed quantum dynamics optimization algorithm treats the iterative evolution process of the optimization over time as a quantum dynamic process. The tunneling effect and potential barrier estimation in quantum mechanics can effectively promote the evolution of the optimization algorithm to the global optimum. Extensive experiments on four benchmarks demonstrate the effectiveness of QDNAS, which is consistently better than all baseline methods in image classification tasks. Furthermore, an in-depth analysis is conducted on the searchable networks that provide inspiration for the design of other image classification networks.

Keywords: quantum dynamics; global optimization; neural architecture search; image classification

Citation: Jin, J.; Zhang, Q.; He, J.; Yu, H. Quantum Dynamic Optimization Algorithm for Neural Architecture Search on Image Classification. *Electronics* **2022**, *11*, 3969. https://doi.org/10.3390/electronics11233969

Academic Editor: Dimitris Apostolou

Received: 4 November 2022
Accepted: 23 November 2022
Published: 30 November 2022

1. Introduction

Deep learning (DL) methods have shown great potential for such applications as computer vision and natural language processing [1]. Image classification is one of the four major tasks of computer vision. Given an input image, the image classification task aims to determine the category of the image [2].

To effectively deal with a classification task, multiple network architectures, i.e., ResNet [3], DensNet [4], and SENet [5], have been proposed. These new architectures have heuristic significance for designing neural networks, such as the residual module in ResNet, which has now become the basic module in many network architectures.

However, designing DL algorithms requires designers to have rich experiences. It is a challenging task to design neural network architectures due to the fact that little prior knowledge on architecture design is available and the designed structures are problem-dependent. In that case, the ability to automatically generate the correct network architecture for any given task has become a new requirement [6,7]. One way to generate these architectures is to use evolutionary algorithms (EA) [8]. Traditional topological neuroevolution research is the exploration of early neural network architecture searches [9,10]. EA uses neural networks to simplify search, weighting, structured search, and multi-objective search [11,12].

Google Research showed that Regular Evolutionary Algorithms (REA) [13] work well in neural network architecture.

Recent research on the neural network architecture search problem has brought a new trend in evolutionary neural network architecture search. However, there are two main challenges: (1) most well-designed new algorithms could not be used for neural network architecture search [14–16]; and (2) each search algorithm is only experimented on for a specific search space during the search process and is not verified on other, well-known search spaces [17,18].

The quantum dynamics optimization algorithm (QDO) is an iterative optimization algorithm [19] constructed by simulating the optimization process of the quantum dynamics equation. In the quantum dynamics optimization algorithm, the evolutionary process of the optimization algorithm over time is transformed into a quantum dynamics process. In the quantum dynamics optimization algorithm, the modulus of the wave function represents the distribution of the solution. Therefore, the evolution process is the evolution process of the optimization algorithm solution. The modulus of the quantum wave function can be obtained from the ensemble theory in physics, where the probability distribution represents the probability distribution of the quantum particles in a given state. The tunneling effect, potential barrier estimation, and other theories in quantum mechanics can effectively facilitate the optimization process of the optimization algorithm.

Here we explore an application of quantum dynamic optimization algorithms for a neural architecture search (NAS) problem. In the neural network architecture search problem, novelty search facilitates the discovery of excellent architectures [20]. The quantum dynamics optimization algorithm can effectively jump out of the local optimum and find the global optimum by using the tunnel effect. It is a well-designed intelligent optimization algorithm. The potential barrier estimation in quantum mechanics can make reasonable use of the information on non-optimal solutions in the process of algorithm optimization, thereby increasing the diversity of solutions. In the neural network architecture search problem, some non-optimal architectures may evolve into optimal architectures after iteration. The properties of these two aspects of the quantum dynamics optimization algorithm suggest that it may be a better solution to the neural network architecture search problem.

The proposed method is shown in Figure 1. Quantum dynamics optimization algorithms are competitive optimizations proposed in [19]. Recent research primarily focuses on improving quantum dynamics optimization algorithms [21]. By introducing different mechanisms, the optimization performance of the algorithm is further improved by improving the performance of the algorithm. Unlike previous studies, the method in [19] does not improve the performance of the algorithm for specific optimization tasks. Instead, it uses the most basic quantum dynamic optimization algorithm (QDO) to explore its application in neural network architecture research.

The NAS method relies on a search strategy to determine the next architecture to be evaluated, and a performance evaluation strategy to evaluate its performance [8]. This article will focus on search strategies. To evaluate the performance of the search algorithm more comprehensively, we use table-based NAS benchmarks as the benchmark dataset [22–24].

The contributions of this work can be summarized as follows:

- In addition to conventional evolutionary algorithms, for the first time, this paper applies a quantum heuristic optimization algorithm as a search algorithm for a neural network architecture search problem. We transform the applicability of quantum dynamics optimization algorithms from traditional optimization problems to neural network architecture search problems. The designed algorithm does not depend on specific data and is a general neural network architecture search algorithm.
- Reduce the problem search space by defining reasonable discretization encoding methods, and quantum heuristic rules. The use of the quantum tunneling effect and barrier estimation principle makes the proposed algorithm more competitive with general evolutionary methods.

- Conduct extensive experiments on NAS-Benchmark to demonstrate the effectiveness of the proposed models.

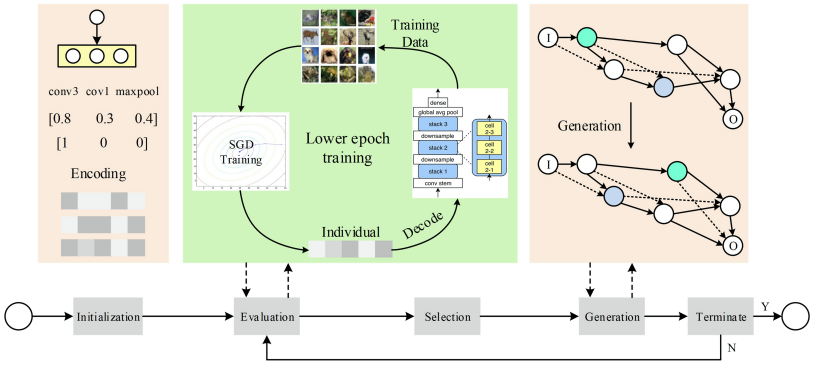

Figure 1. Pipeline of the QDO-NAS.

We first describe the quantum dynamic optimization algorithm (QDO; Section 2), then describe how to apply QDO to NAS (Section 3), and then Section 4 verifies the effectiveness of the search algorithm proposed for table-based benchmarks, such as NAS-Bench-101 [22], NAS-Bench-1Shot1 [23], NAS-Bench-201 [24], and NSATs-Bench [25].

2. Quantum Dynamic Optimization

The quantum dynamics optimization algorithm is an iterative optimization algorithm [19], in which the evolution of the optimization algorithm is transformed over time into a quantum dynamic process. The theories such as the tunneling effect and potential barrier estimation in quantum mechanics can effectively promote the optimization process of optimization algorithms.

According to the basic iterative operation of the optimization algorithm under the quantum dynamics model, the basic iterative process can be obtained in Algorithm 1.

Algorithm 1: Pseudocode of QDO.

Randomly generate k copies of free particle in the domain $[d_{min}, d_{max}]$, $\sigma_s = d_{max} - d_{min}$
while *(stop condition is not satisfied)* **do**
  Initialize $Ac = 0$
  while $(\sigma < \sigma_k)$ **do**
    for i=1 *to* k **do**
      Generate $x'[i] \sim N(x[i], \sigma^2)$
      if $(f(x'[i]) \leq f(x[i]))$ **then**
        $x[i] = x'[i]$, update the ith particle
      else
        $x[i] = x'[i]$, update the ith particle according to $T \propto e^{-\frac{\Delta x \sqrt{\Delta f}}{\sigma}}$
      end
    end
    $Ac = Ac + 1$
    Calculate the σ_k for k copies
  end
  $x_{\text{worse}}[i] = x_{\text{aver}}[i]$
  $\sigma = \sigma/2$
end
Output: $x_{best}[i]$

All operations of this basic iterative process are obtained by using the theoretical platform of the quantum dynamics of the optimization algorithm and the approximation and estimation of the objective function. The specific steps of QDO are as follows.

1. Generate k sampled individuals in the domain $[d_{min}, d_{max}]$.
2. The probability evolution of the location distribution of k sampled individuals can be considered as the evolution of the particle wave function modulus. The larger the value of k, the closer to the probability distribution of the wave function modulus. The initial mean square error σ takes the length of the domain. When the initial mean square error is large, the algorithm is not sensitive to the initial position of the sampled individual.
3. Generate new solutions with a normal distribution $x'[i] \sim N(x[i], \sigma^2)$, if the new solution $f(x'[i]) \leq f(x[i])$; that is, the new solution is better than the old solution, then the new solution is directly accepted; if the new solution is worse than the old solution, it can be considered from the physical image that the particle is blocked by the potential barrier, and the difference solution is accepted according to the probability that the barrier penetrates the transmission coefficient T.
4. This iterative process is repeated until the mean square error of the $x[i]$ positions of the k sampled individuals is less than or equal to the mean square error of the current normal sampling.
5. Replacing the worst position with the mean of the sampled individuals $x[i]$, $x_{\text{worse}}[i] = x_{\text{aver}}[i]$ reduces the mean square error of normal sampling and enters a smaller scale to perform the same iterative process.
6. If the algorithm meets the set maximum function evolution times $maxFE$, the entire iterative process ends, and the optimal solution $x_{\text{best}}[i]$ among the current k sampled individuals $x[i]$ is output.

3. Proposed Method

3.1. NAS Problem Black Box Modeling

The principle of NAS is to give a set of candidate neural network structures called the search space and use a certain strategy. During the search for the optimal network structure, the pros and cons of the neural network structure are measured via the performance of some indicators, such as accuracy and speed degree to measure, called performance evaluation.

In the NAS problem, the form of the fitness function is unknown; it belongs to the black-box optimization problem [26]. It has the characteristics of nonlinearity and non-convexity, and intelligent optimization algorithms have natural advantages for solving such problems.

In the neural network architecture search problem, the search space represents and defines the variables of the optimization problem; that is, it is the basic components of the problem that need to be optimized, such as convolution size, stride, what kind of pooling, and the number of layers of the network.

The search strategy specifies the algorithm used to search for the optimal architecture. These algorithms include: random search [27], Bayesian optimization [28], evolutionary algorithms [26], reinforcement learning [29], and gradient-based algorithms [30]. Among them, Google's reinforcement learning search method was an earlier exploration in 2017. This paper made architecture search more popular [31], and later research institutions, such as Uber, OpenAI, and Deepmind, began to apply evolutionary algorithms to this field. NAS has become a key application of evolutionary computing, and many domestic companies have also begun the same attempt.

Formally, NAS can be modeled as a black-box optimization problem, as shown in Equation (1):

$$\begin{cases} \arg\min_A = \mathcal{L}(A, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{fitness}}) \\ \text{s.t. } A \in \mathcal{A} \end{cases} \tag{1}$$

where $\mathcal{A}$ represents the search space of the potential neural architecture, and $\mathcal{L}(\cdot)$ measures the fitness evaluation $\mathcal{D}_{fitness}$ on the dataset D_{train}. $\mathcal{L}(\cdot)$ is usually non-convex and non-differentiable. *s.t.* is the abbreviation of subject to (such that), which means to be bound. In principle, NAS is a complex optimization problem with a series of challenges, such as complex constraints, discrete representations, two-layer structures, a high computational cost, and multiple conflicting criteria. A NAS algorithm refers to an optimization algorithm specially designed to efficiently and efficiently solve the problem represented by Equation (1). The following section will explore the application of the quantum dynamics optimization algorithm (QDO) in neural network architecture search.

3.2. QDNAS

Recent NAS methods and benchmarks parameterize the unit structure of deep neural networks into directed graphs. The realization of the unit structure can be seen as assigning related operations from a set of choices or values, such as selecting the predecessor and successor of a node in a directed graph or an operator that selects a node.

The selection of the candidate unit structure belongs to the discrete optimization problem. It can be seen from the basic iterative process of QDO that the basic operation of QDO is Gaussian sampling in continuous space.

We discretize it, that is, set a function as Equation (2). For example, the value obtained by sampling [cov3,cov1,maxpool] is [0.8,0.3,0.4], then the discretized value is [1,0,0].

$$f(x) = \begin{cases} 1, & x \geqslant 0.5 \\ 0, & \text{else} \end{cases} \tag{2}$$

The algorithm involves the problem of replacing the difference solution with the mean value, which is explained here with the solution search matrix of NAS-Bench-101. When NAS-Bench-101 searches, the adjacency matrix is used to encode the network architecture; that is, the sampled particles are the adjacency matrix. Suppose the two sampled particles are x_1= $\begin{bmatrix} 0.3 & 0.2 & 0.4 \\ 0.1 & 0.6 & 0.3 \\ 0.3 & 0.7 & 0.2 \end{bmatrix}$ and x_2= $\begin{bmatrix} 0.2 & 0.8 & 0.3 \\ 0.9 & 0.1 & 0.4 \\ 0.6 & 0.2 & 0.1 \end{bmatrix}$, then x_{aver}= $\begin{bmatrix} 0.25 & 0.5 & 0.35 \\ 0.5 & 0.35 & 0.35 \\ 0.45 & 0.45 & 0.5 \end{bmatrix}$. The final architectural adjacency matrix obtained by the function $discrete(x)$ is X= $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ QDNAS is shown in Algorithm 2. Figure 1 shows the framework of the algorithm. To demonstrate the performance of the framework, several state-of-the-art NAS methods are compared in the simulation experiments section.

The specific steps of QDONAS are:

1. Initialize the population, specifying the dataset D to use.
2. Randomly sample the architecture in the search space and assign it to a queue pop_i.
3. The particles are discretized according to Equation (2).
4. Generate new particle according to $POP_i' = regularized(POP_i + \sigma N(0,1))$.
5. If $f(POP_i) < f(POP_i')$, then POP_i' is assigned to POP_i. Otherwise, the poor solution is accepted with a certain probability. In this part, the probability is 0.1. This probability is selected on the basis of many trials.
6. Replace the worst position with the mean of the sampled individuals $pop_{worst} = pop_{aver}$, and discretize the sampled individuals again.
7. Keep repeating lines 2 to 12 in QDNAS until the maximum number of iterations is reached.

QDO is a sampling-based method, but the difference from random sampling is that QDO can effectively use the information from the previous generation of individuals. QDO introduces a Gaussian distribution in the sampling process. The probability of a Gaussian

distribution in the range of σ is 65.26%, and the probability of falling into the range of 3σ is 99.74%. In other words, the particles will move to the vicinity of the better solution with a small step length, which ensures the mining of the algorithm. At the same time, in order to ensure the diversity of the population, the difference is accepted with a certain probability to ensure the diversity of the population. At the end of the iteration of each group, a certain perturbation mechanism is introduced through mean replacement to avoid premature stagnation of the algorithm.

Algorithm 2: Pseudocode of QDNAS.

Input: f:NAS problem with evaluation,D:The reference dataset,k:Population size
Output: Best architecture.
while *(stop condition is not satisfied)* **do**
  for $i = 1$ *to* k **do**
    $pop_i \leftarrow random\_configuration()$
    $POP_i \leftarrow discretized\_architecture()$
    $f_i \leftarrow evaluate_architecture(POP_i)$
    Generate POP_i' for POP_i to $POP_i' = regularized(POP_i + \sigma N(0,1))$
    if ($f(POP_i') < f(POP_i)$) **then**
      $POP_i = POP_i'$, update the ith particle
    else
      $POP_i = POP_i'$, update the ith particle according to T
    end
  end
end
POP_{worst}=POP_{aver}
Output:POP_{best}

The pipeline of our method is shown in Figure 1. Initialization is performed first, the initial population is uniformly sampled, and the initial population is discretized. That is, discretization is performed with 0.5 as the threshold. Each individual obtains an initial structure through decoding. We evaluate these structures and record the evaluation results as the fitness value of the individual. We choose the better individual as the next generation and accept the difference with a certain probability. We generate new individuals with a Gaussian distribution around the current individual. We judge whether the termination condition is met; if it is met, the loop ends; if it is not met, the loop will continue.

4. Experiments

We verified the performance of QDNAS in four recent NAS benchmark tests, NAS-Bench-101, NATs-Bench, NAS-Bench-1shot1, and NAS-Bench-201. Different articles use different hyperparameters/data enhancement/regularization/etc. when retraining the searched network structure. Using NAS-Bench can make a fair comparison of each NAS algorithm.

For the image classification task, this paper chooses the default dataset Cifar-10 of NAS-Bench. The CIFAR-10 dataset has a total of 6×10^4 color images, and the size of these images is 32×32, divided into 10 non-overlapping classes. During an architecture search, the training dataset uses CIFAR-10, and the final search network is a network suitable for image classification.

The benchmark test algorithm is Random Search (RS) [27], Tree-Structured Parzen Estimator (TPE) [8], and Regularized Evolution Algorithm (REA) [32]. The experimental parameters are set to NP = 40 and the transmission coefficient is 0.1. Among these algorithms, REA is the preferred benchmarking algorithm, first because REA and QDO are both heuristic algorithms and secondly, because REA has demonstrated excellent performance in past work. For each algorithm, we conduct 500 independent experiments and record the mean performance of the immediate validation regret.

4.1. Nas-Bench-101

The NAS-Bench-101 dataset contains 423k samples, mapping the model structure to the corresponding index (run time and accuracy) traverses the entire search space, making it possible to perform complex analysis on the entire search space.

NAS-Bench-101: The dataset table contains the CNN structure and corresponding training/evaluation indicators using Cell coding. The dataset is Cifar-10 (40k training/10k verification/10k test). Each model was repeatedly trained and evaluated three times under four types of Epochs $E_{\text{stop}} \in \left\{ \frac{E_{\max}}{3^3}, \frac{E_{\max}}{3^2}, \frac{E_{\max}}{3^1}, E_{\max} \right\} = \{4, 12, 36, 108\}$. The indicators used in NASBench101 are: training accuracy, validation accuracy, testing accuracy, number of parameters, and training time.

Figures 2 and 3 show the performance of the search algorithm QDO. Figure 2 shows the trajectory of test accuracy and verification accuracy in 10 tests. Red represents the verification accuracy, and blue represents the test accuracy. It can be seen from the figure that for Random search, the curve is more scattered, which means that the results of each run are quite different, indicating that the randomness is strong. Regarding the regular evolutionary algorithm, this problem has been improved to a certain extent, but it still has a certain degree of randomness. The QDO algorithm verification accuracy rate is relatively concentrated, indicating that the algorithm is robust. However, only two test accuracy rates have large deviations. Furthermore, in the visualization of Figure 2, the comparison of the three can be seen.

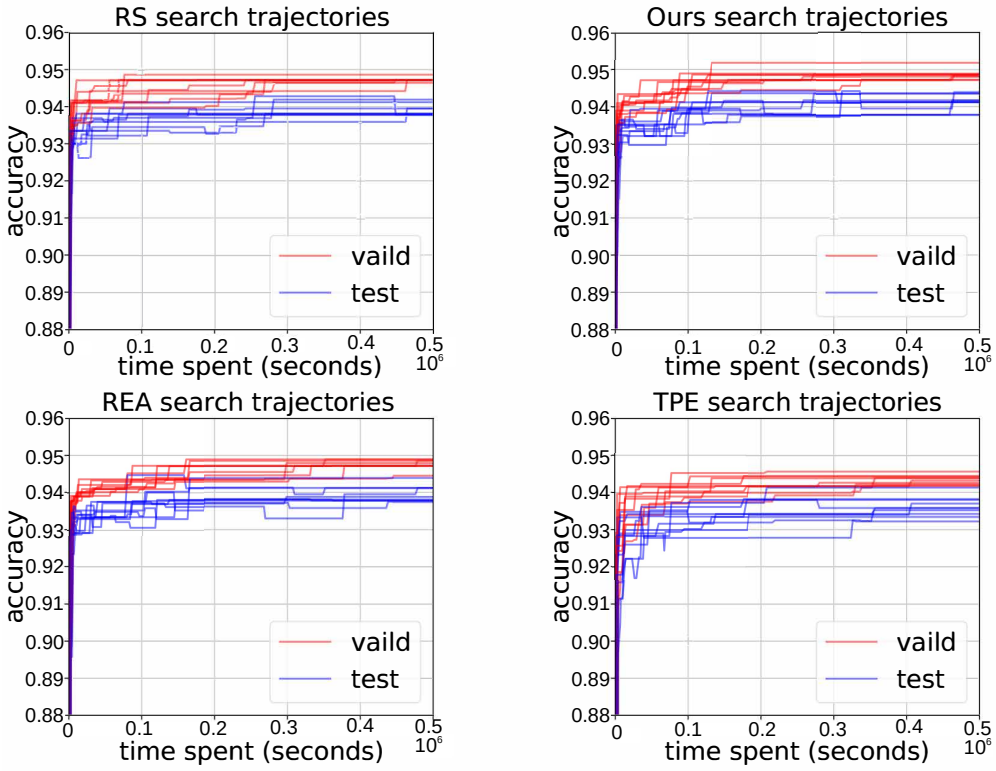

Figure 2. Search trajectories of Random search, REA, and QDO on NAS-Bench-101.

4.2. Nas-Bench-201

NAS-Bench-201 has trained more than 15,000 neural networks on three datasets (CIFAR-10, CIFAR-100, and ImageNet-16-120) based on different random number seeds and different hyperparameters many times. It provides the training and testing time after each training epoch, the loss function and accuracy of the model in the training set/validation set/test set, model parameters after training, model size, model calculation amount, and other important information. With NAS-Bench-201, every NAS algorithm can be compared fairly. Different articles use different hyperparameters/data enhancement/regulations/etc.

when retraining the searched network structure. Using the NAS-Bench-201 API, each researcher can fairly compare the searched network structure.

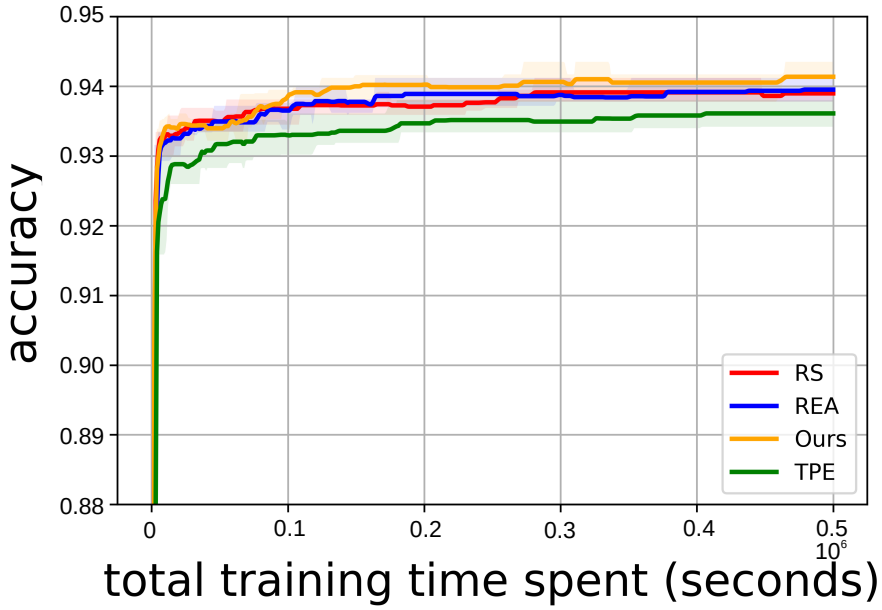

Figure 3. Comparison of the mean test accuracy along with error bars on NAS-Bench-101.

Figures 4 and 5 show the comparative performance of the algorithms. From the comparative performance analysis of the four algorithms, it can be seen that in 10 test experiments, the random search algorithm is more random, and the accuracy of each search changes greatly.

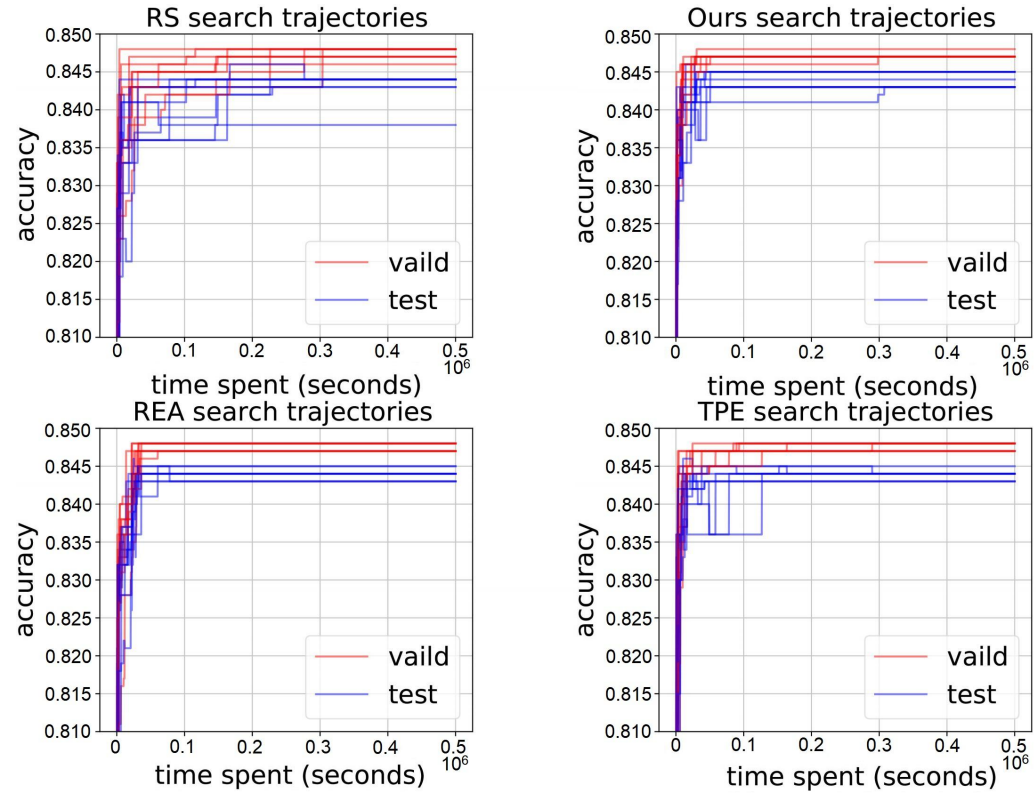

Figure 4. Search trajectories of Random search, REA, and QDO on NAS-Bench-201.

Figure 6 shows the instant validation regret after 500 independent runs. From the results, we can see that for Cifar10, we conclude that even though TPE is better than other algorithms at the beginning it is much slower when approaching the global optimum. The test regrets of DE and RE are almost the same, while RS has shown excellent convergence performance after recovering from the misleading early assessment, and its convergence speed is faster than other algorithms.

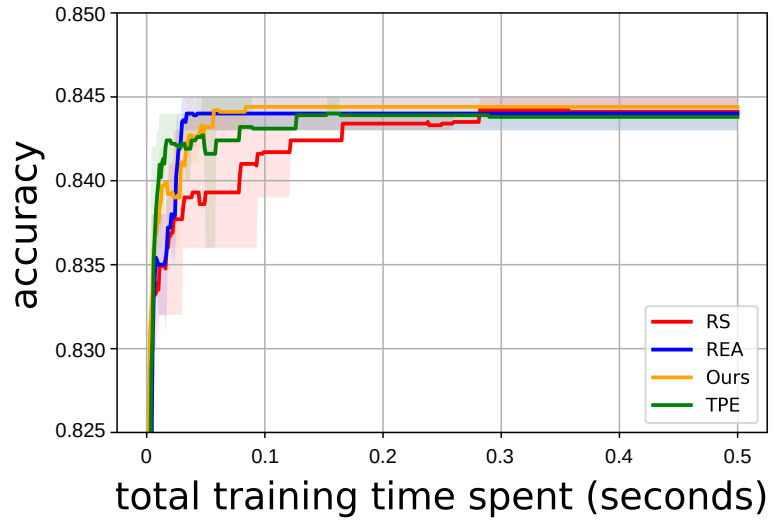

Figure 5. Comparison of the mean test accuracy along with error bars on NAS-Bench-201.

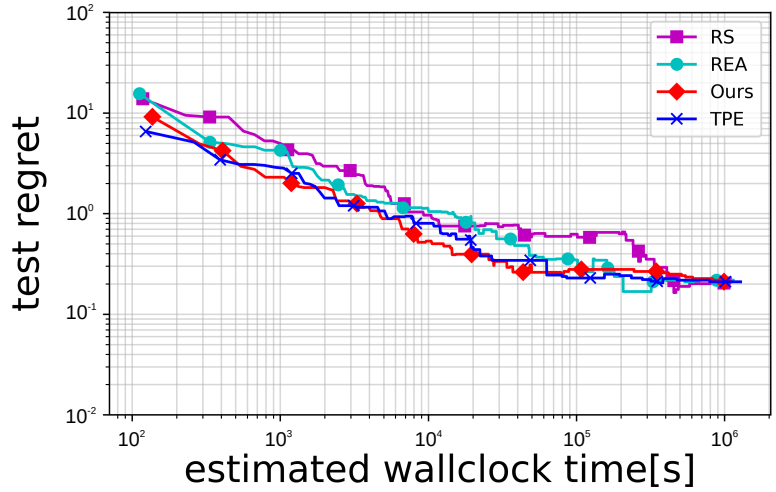

Figure 6. A comparison of the mean test regret performance of 500 independent runs as a function of estimated training time for NAS-Bench-201 on Cifar10

4.3. Nas-Bench-1shot1

NAS-Bench-1shot1 modifies the cell-level topology based on NAS-Bench-101 while keeping the network-level topology unchanged. NAS-Bench-1shot1 makes the NAS approach more practical. It defines three search spaces that are convenient for the weight-sharing algorithm to use: search space 1, search space 2, and search space 3. The number of schemas available for searching are 6240, 29160, and 363648.

It can be seen from Figure 7 that RS has better performance in the initial search stage, the reason may be that a better architecture is randomly searched, and when the iteration time is around the point of 2500, REA and QDO are better due to the algorithm itself having a better search mechanism, so it quickly locks in a better search area. When the time is 2700, QDO shows an overwhelming advantage, and the accuracy of the searched

architecture is higher. As the iteration progresses, the performance of several algorithms on the NAS-Bench-1Shot1 test set gradually tends to be the same.

Figure 8 shows the immediate test regret after 500 independent runs. It can be seen from the results that both RS and REA performed better in the initial stage, but the QDO algorithm performed better in the later stage, and TPE performed better in the middle stage, but there was premature stagnation. The performance of the QDO algorithm is average in the early stage, but there is a rapid convergence in the later stage. The REA algorithm outperforms other algorithms in the later architecture search.

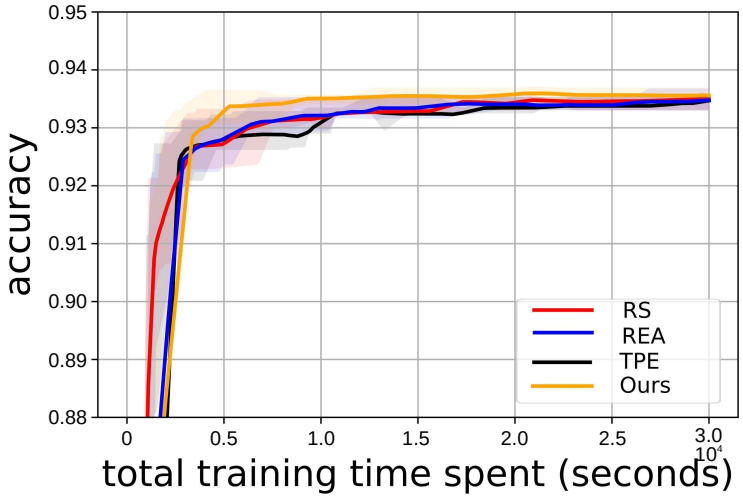

Figure 7. A comparison of the mean test regret performance of 500 independent runs as a function of estimated training time for NAS-Bench-1Shot1 on Cifar10

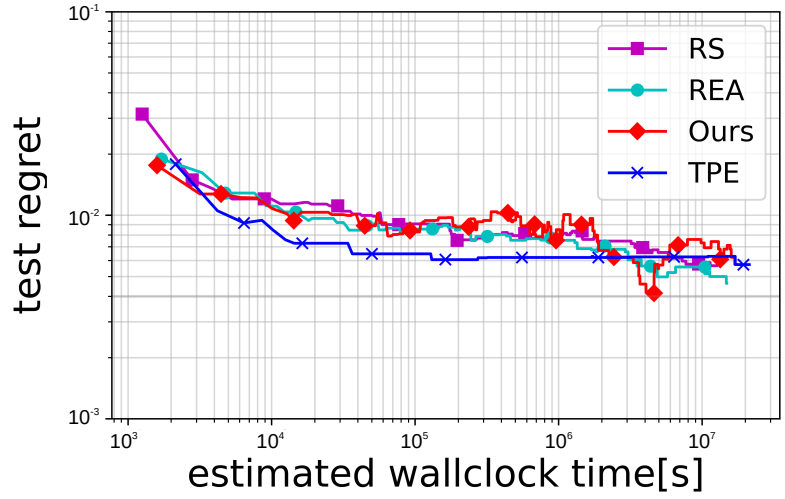

Figure 8. Search trajectories of Random search, REA, and QDO on NAS-Bench-1Shot1.

4.4. NATs-Bench

NATs-Bench is based on NAS-Bench201, which expands the NAS-Bench201 dataset into three, namely CIFAR10, CIFAR100, and ImageNet-16-120. NATS Bench includes 15,625 candidate neurons in the three datasets. Among them, the topological search space S_t is applicable to all NAS methods and the size of the search space S_s complements the lack of architecture size analysis. The average convergence curves of the four algorithms on the NATs-Bench test set are shown in Figure 9. From the visual analysis of the average convergence curve, it is known that QDO and REA have better robustness.

Figure 9. Comparison of the mean test accuracy along with error bars.

4.5. Results Discussion

We record the statistical results of the experimental data of the QDO algorithm, as shown in the table, in which Table 1 records the experiments of the benchmark algorithm for the NAS-Bench-101, NAS-Bench-201, and NAS-Bench-1Shot1 test sets results. The bold words in the table indicate the top ranking. From the experimental results, on the Cifar10 classification dataset; the optimal architecture searched by the QDO algorithm on NAS-Bench-101 is 0.003 higher than the accuracy rate of RS and REA, while in NAS, excellent results were also obtained on Bench-210 and NAS-Bench-1Shot1. REA is a baseline algorithm proposed by the Google AI research team. It is proven that QDO is competitive in architecture search problems.

Table 1. Statistical experimental results for NAS-Bench on the Cifar10 dataset.

	NAS-101	**NAS-201**	**NAS-1Shot1**
RS	0.940	0.939	0.946
TPE	0.933	0.936	0.947
RE	0.940	0.940	0.947
Ours	**0.943**	**0.942**	0.947

In addition to the optimization performance, robustness is also an important factor. Whether the algorithm is sensitive to randomness during training and searching is also a measure of whether the NAS algorithm is good. Since REA and QDO performed better in the previous experiments, this part of the experiment only compares the REA and QDO algorithms. Figure 10 is the empirical cumulative distribution of the final test regret after 500 runs of REA and QDO. Based on the robust performance ratios of REA and DE on different test sets in the figure, it can be seen that the robustness of the QDO algorithm on NAS-Bench-101 is significantly better than that of the REA algorithm, while on the other three datasets, the two algorithms' robustness differs little.

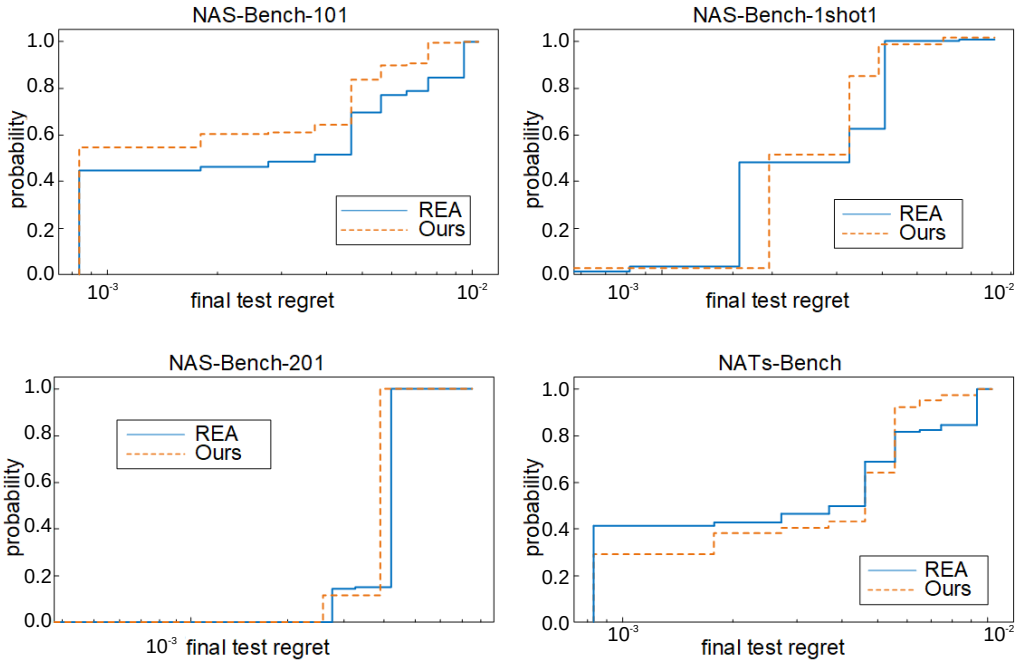

Figure 10. Empirical cumulative distribution of the final test regret after 500 runs of REA and QDNAS.

5. Conclusions

We proved that the quantum dynamics optimization algorithm can be used for a neural network architecture search. The quantum dynamics optimization algorithm is a sampling-based algorithm. Due to the quantum tunneling effect, it has advantages in dealing with mixed data types and high-dimensional optimization problems. Therefore, QDO may be a good candidate for NAS, which may help discover novel but unknown architectures. Since the quantum dynamics optimization algorithm has a natural parallelism, we will explore the parallel implementation of the algorithm in the architecture search in the future.

First, we performed classification recognition on the CIFAR-10 image classification dataset. It should be noted here that by adjusting the core size and number of channels of the convolutional and pooling layers, the algorithm can be easily applied to other fields.

Author Contributions: Conceptualization, Q.Z. and H.Y.; methodology, J.J.; formal analysis, J.H. All authors have read and agreed to the published version of the manuscript.

Funding: Project of Sichuan Science and Technology Department (2021Z005).

Data Availability Statement: Not applicable.

Acknowledgments: Thanks to Sichuan Intelligent Tolerance Design and Testing Engineering Research Center.

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. Yin, D.; Gontijo Lopes, R.; Shlens, J.; Cubuk, E.D.; Gilmer, J. A fourier perspective on model robustness in computer vision. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 13276–13286.
3. 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.
4. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
5. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7132–7141.
6. Guo, Y.; Luo, Y.; He, Z.; Huang, J.; Chen, J. Hierarchical neural architecture search for single image super-resolution. *IEEE Signal Process. Lett.* **2020**, *27*, 1255–1259. [CrossRef]

7. Wang, Y.; Liu, Y.; Dai, W.; Li, C.; Zou, J.; Xiong, H. Learning Latent Architectural Distribution in Differentiable Neural Architecture Search via Variational Information Maximization. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 12312–12321.
8. Elsken, T.; Metzen, J.H.; Hutter, F. Neural architecture search: A survey. *J. Mach. Learn. Res.* **2019**, *20*, 1997–2017.
9. Sun, Y.; Xue, B.; Zhang, M.; Yen, G.G. Evolving deep convolutional neural networks for image classification. *IEEE Trans. Evol. Comput.* **2019**, *24*, 394–407. [CrossRef]
10. Stanley, K.O.; Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evol. Comput.* **2002**, *10*, 99–127. [CrossRef] [PubMed]
11. Sun, J.D.; Yao, C.; Liu, J.; Liu, W.; Yu, Z.K. GNAS-U 2 Net: A New Optic Cup and Optic Disc Segmentation Architecture With Genetic Neural Architecture Search. *IEEE Signal Process. Lett.* **2022**, *29*, 697–701. [CrossRef]
12. Gong, M.; Liu, J.; Qin, A.K.; Zhao, K.; Tan, K.C. Evolving deep neural networks via cooperative coevolution with backpropagation. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 420–434. [CrossRef]
13. Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y.L.; Tan, J.; Le, Q.V.; Kurakin, A. Large-scale evolution of image classifiers. In Proceedings of the International Conference on Machine Learning. PMLR, Sydney, NSW, Australia, 6–11 August 2017; pp. 2902–2911.
14. Niu, R.; Li, H.; Zhang, Y.; Kang, Y. Neural Architecture Search Based on Particle Swarm Optimization. In Proceedings of the 2019 3rd International Conference on Data Science and Business Analytics (ICDSBA), Istanbul, Turkey, 11–12 October 2019; pp. 319–324.
15. Xie, L.; Yuille, A. Genetic cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 1379–1388.
16. Junior, F.E.F.; Yen, G.G. Particle swarm optimization of deep neural networks architectures for image classification. *Swarm Evol. Comput.* **2019**, *49*, 62–74. [CrossRef]
17. Sun, Y.; Xue, B.; Zhang, M.; Yen, G.G.; Lv, J. Automatically designing CNN architectures using the genetic algorithm for image classification. *IEEE Trans. Cybern.* **2020**, *50*, 3840–3854. [CrossRef] [PubMed]
18. Xue, Y.; Wang, Y.; Liang, J.; Slowik, A. A self-adaptive mutation neural architecture search algorithm based on blocks. *IEEE Comput. Intell. Mag.* **2021**, *16*, 67–78. [CrossRef]
19. Wang, P.; Xin, G.; Jiao, Y. Quantum Dynamics Interpretation of Black-box Optimization. *arXiv***2021**, arXiv:2106.13927.
20. Zhang, M.; Li, H.; Pan, S.; Liu, T.; Su, S.W. One-Shot Neural Architecture Search via Novelty Driven Sampling. In Proceedings of the IJCAI, Yokohama, Japan, 11–17 July 2020; pp. 3188–3194.
21. Jin, J.; Wang, P. Multiscale Quantum Harmonic Oscillator Algorithm with Guiding Information for Single Objective Optimization. *Swarm Evol. Comput.* **2021**, *65*, 100916. [CrossRef]
22. Ying, C.; Klein, A.; Christiansen, E.; Real, E.; Murphy, K.; Hutter, F. Nas-bench-101: Towards reproducible neural architecture search. In Proceedings of the International Conference on Machine Learning. PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 7105–7114.
23. Zela, A.; Siems, J.; Hutter, F. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. *arXiv* **2020**, arXiv:2001.10422.
24. Dong, X.; Yang, Y. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv* **2020**, arXiv:2001.00326.
25. Dong, X.; Liu, L.; Musial, K.; Gabrys, B. Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 3634–3646. [CrossRef] [PubMed]
26. Liu, Y.; Sun, Y.; Xue, B.; Zhang, M.; Yen, G.G.; Tan, K.C. A survey on evolutionary neural architecture search. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**. 1–21. [CrossRef] [PubMed]
27. Li, L.; Talwalkar, A. Random search and reproducibility for neural architecture search. In Proceedings of the Uncertainty in Artificial Intelligence. PMLR, virtual online, 3–6 August 2020; pp. 367–377.
28. Kandasamy, K.; Neiswanger, W.; Schneider, J.; Poczos, B.; Xing, E.P. Neural architecture search with bayesian optimisation and optimal transport. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, Canada, 3–8 December 2018; pp. 2020–2029
29. Chen, Y.; Meng, G.; Zhang, Q.; Xiang, S.; Huang, C.; Mu, L.; Wang, X. Renas: Reinforced evolutionary neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 4787–4796.
30. Santra, S.; Hsieh, J.W.; Lin, C.F. Gradient descent effects on differential neural architecture search: A survey. *IEEE Access* **2021**, *9*, 89602–89618. [CrossRef]
31. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv* **2016**, arXiv:1611.01578.
32. Real, E.; Aggarwal, A.; Huang, Y.; Le, Q.V. 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.