Quantum-Inspired Evolutionary Algorithm for Convolutional Neural Networks Architecture Search

Weiliang Ye
School of Artificial Intelligence, Xidian University
Xi'an, Shanxi Province, 710071, China
weiliangye@stu.xidian.edu.cn

Ruijiao Liu
School of Artificial Intelligence, Xidian University
Xi'an, Shanxi Province, 710071, China
rj_liu@stu.xidian.edu.cn

Yangyang Li
Senior Member, IEEE
School of Artificial Intelligence, Xidian University
Xi'an, Shanxi Province, 710071, China
yyli@xidian.edu.cn

Licheng Jiao
Fellow, IEEE
School of Artificial Intelligence, Xidian University
Xi'an, Shanxi Province, 710071, China
lchjiao@mail.xidian.edu.cn

***Abstract*—Convolutional neural networks (CNN) are widely used and effective deep learning methods for image classification tasks. But the architecture of CNN such as LetNet and AlexNet were designed elaborately by experts because designing the neural networks is time-consuming and requires expert knowledge. This paper proposed a quantum-inspired evolutionary algorithm to search the neural architectures. First, we encode CNNs into quantum chromosomes and distinguish these chromosomes from the Convolutional Layer, Pooling Layer, Fully-connected Layer and Disabled Layer with its range. Second, quantum chromosomes are updated by applying quantum gates and find the best individual with quantum genetic algorithm. Third, we can predict the network performance after a few steps of stochastic gradient descent by means of evaluation estimate strategy so that we can stop training the bad networks early, which can speed up evolutionary process. The proposed algorithm is examined and compared with some state-of-art methods for image classification in three benchmark datasets. The experimental results prove the proposed algorithm can search a strong classifier robustly. In addition, it performs better than the general evolutionary algorithm. More importantly, with the help of evaluation estimate strategy, it is substantially faster than the algorithms without evaluation estimate strategy which means we can take less time to search a good network for the given task.**

***Index Terms*—genetic algorithm, quantum-inspired, neural architecture search, evaluation estimate**

I. Introduction

Image classification is a fundamental task in computer vision, implying a wide range of applications. Recently, the state-of-the-art algorithms on image classification are mostly based on the Deep Convolutional Neural Network (CNN). Such as LeNet-5 [1], AlexNet [2], VGGNet [3] and GoogLeNet [4]. Deep Neural Network have demonstrated its successful applications in computer vision [2], speech recognition [5] , and natural language processing [6]. In most cases, these neural networks are still designed by hand, which is a time-consuming process. Additionally, the vast amount of possible architectures of CNN requires expert knowledge to restrict the search. Therefore, the need of designing automatic methods for determining the hyper-parameters and architectures is especially important for increasingly complex CNN architectures.

Neural architecture search (NAS) methods [7] attempt to automate the process of finding optimal neural networks architectures for any given task. NAS methods generally are designed as an optimization problem [8], which objective function is validation loss for a given task. In image classification task, NAS methods have been demonstrated to be able to discover neural networks architecture which performance is comparable or superior to human-designed neural networks [9], [10], [11], [12] and [13].

Many different search strategies can be used to discover neural network architectures, including random search [14], [8], bayesian optimization [15], reinforcement learning (RL) [16] and evolution algorithm [9], [17]. However, these methods need huge computing resources. Real et al. [9] proposed an evolutionary approach to search networks for image classification problem using a parallel system executed on 250 computers.

In order to reduce the consumption of computer resources and increase the speed of evolution process,we propose Quantum-Inspired Evolutionary Algorithm (QIEA), a neuro-evolutionary algorithm that combines the strengths of evolu-

This work was supported by the National Natural Science Foundation of China under Grant 61772399, Grant U1701267, Grant 61773304, Grant 61672405 and Grant 61772400, the Key Research and Development Plan of Innovation Chain of Industries in Shaanxi Province under Grant 2019ZDLGY09-05, the Technology Foundation for Selected Overseas Chinese Scholar in Shaanxi (Nos. 2017021 and 2018021), the Program for Cheung Kong Scholars and Innovative Research Team in University Grant IRT_15R53, and the Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project) Grant B07048.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

tionary algorithms and quantum-behaved algorithms to search the architectures of convolutional neural networks for image classification. Evolution algorithm is always used for multi-objective optimization problem [18] [19] and K-traveling repairman problem [20]. Historically, evolutionary algorithms were already used by many researchers to evolve neural architectures (and often also their weights) decades ago [21], [22], [23] and [24] . Yao [25] provides a literature review of work earlier than 2000. The evolutionary algorithm involves constructing an initial population of individuals and performing evolutionary operations. It is worth emphasizing that the evolutionary process is computationally expensive, as we need to undergo a complete network training process for each generation individual.

QIEA use a qubit representation (A qubit may be in the '1' state, in the '0' state, or in any superposition of the two) instead of binary numeric, or symbolic representations so that it can imitate parallel computation in classical computers. Under the quantum characteristics and evaluation estimate strategy, we can complete this method with only a single GPU.

II. Related Works

A. Convolutional Neural Networks

Recent years have witnessed a revolution in visual recognition. Conventional classification tasks are extended into large-scale environments. AlexNet trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes [2].

Starting with LeNet-5 [1], CNN have typically had a standard structure – convolutional layers (optionally followed by batch normalization and max-pooling) are followed by one or more fully-connected layers.(see Fig.1) These layers extract features by convolving the input image or—for deeper layers—the output of the previous layer with a set of filters. Convolutional layers are regularly followed by pooling steps which reduce the spatial dimensionality of the feature map.

It is based on the observation that a network with enough neurons is able to fit any complicated data distribution. A CNN can be considered as a composite function, which is trained by back-propagating error signals defined by the difference between the supervision and prediction at the top layer.

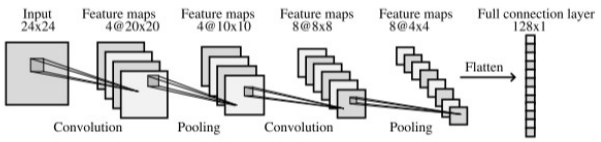

Fig. 1. A general architecture of the convolutional neural networks.

B. Quantum Evolutionary Algorithm

Evolutionary Algorithms(EAs) [26] do not require rich domain knowledge in the process of neural architecture search. The genetic operators commonly used in EAs are crossover, mutation and selection. Mutation is the simulation of the genetic material mutation of organisms in nature. The crossover is the simulation of chromosome exchange process in sexual reproduction. The selection is the process of finding the fittest chromosome.

EAs can be applied to solve optimization problems. A population of chromosomes is randomly initialized. Each chromosome represents a candidate solution to the optimization problem. A fitness function is used to evaluate each chromosome to determine which the chromosome can solve the problem. In a generational model, each chromosome is made up of several genes, and these genes are altered using a genetic operator. The resulting chromosome after the application of a genetic operator is known as an offspring. The number of offspring is the population size. EAs provide a further key advantage over other optimization algorithms: they fluently handle complex combinations of discrete and continuous search spaces, making them ideal for neuro-evolutionary studies.

Quantum gennetic algorithm(QGA) is characterized by principles of quantum computing including concepts of qubits and superposition of states [27]. QGA uses a qubit representation instead of binary, numeric, or symbolic representations and it can imitate parallel computation in classical computers [28]. The smallest unit of information stored in a two-state quantum computer is called a quantum bit or qubit. A qubit may be in the '1' state, in the '0' state, or in any superposition of the two. The state of a qubit can be represented as

$$|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1}$$

$|\alpha|^2$ gives the probability that the qubit will be found in '0' state and $|\beta|^2$ gives the probability that the qubit will be found in the '1' state. Normalization of the state to unity guarantees

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2}$$

If there is a system of m-qubits, the system can represent 2^m states at the same time. However, in the act of observing a quantum, it collapses to a single state.

C. Learning curves

The learning curve is defined as a function of the performance of an iterative model and its training time or number of iterations. Learning curves are very popular for visualizing overfitting and it can help researchers stop training bad models early.

We fit the learning curve by a set of parametric model families so that we can predict validation performance and stop training process when it is unlikely to beat the performance of the best model we have encountered so far. Swersky et al. [29] devised a GP-based Bayesian optimization method that includes a learning curve model. But it not works well for deep neural networks

III. Our Approach

This section presents the quantum evolutionary algorithm for searching competitive convolutional neural network structures. First, we propose a way of encoding a network structure into a quantum chromosome. Second, we find the best chromosome by quantum-inspired evolutionary algorithm.

A. Algorithm Overview

Algorithm 1 outlines the framework of the Quantum-Inspired Evolutionary Algorithm. The method to initialize the population will be described in section *Initialization*. The observation and evaluation method with evaluation estimate strategy will list in Algorithm 2 and Algorithm 3.

Algorithm 1 Framework of QIEA

```
Q ← Initialize the population with the proposed encoding strategy
q ← individual in Q
q_best ← the best individual in Q
while termination criterion is not satised do
    for q in Q do
        observe and evaluate q;
        update q with rotation U;
    end for
    update rotation angle θ;
    update q_best;
end while
```

B. Encoding Strategy

This section introduces the strategy about how to encode three types of layers into binary string and represent this string by qubits [30]. QIEA uses a novel representation that is based on the concept of qubits. An m-qubits representation is defined as

$$\begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \end{pmatrix} \tag{3}$$

where$|\alpha_i|^2+|\beta_i|^2=1, i=1,2,...,m$. This representation has the advantage that it is able to represent any superposition of states[19]. For instance, if there is a three-qubits system with three pairs of amplitudes such as

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & 1 & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & 0 & \frac{\sqrt{3}}{2} \end{pmatrix} \tag{4}$$

the state of the system can be represented as

$$\frac{1}{2\sqrt{2}}|000\rangle+\frac{\sqrt{3}}{2\sqrt{2}}|001\rangle+\frac{1}{2\sqrt{2}}|100\rangle+\frac{\sqrt{3}}{2\sqrt{2}}|101\rangle \tag{5}$$

The above result means that the probabilities to represent the state $|000\rangle, |001\rangle, |100\rangle$,and$|101\rangle$are$\frac{1}{8}$, $\frac{3}{8}$, $\frac{1}{8}$,and$\frac{3}{8}$, respectively. By consequence, the three-qubits system of (4) has four states information at the same time. Evolutionary computing with the qubit representation has a better characteristic of diversity than classical approaches, since it can represent superposition of states. Only one quantum chromosome such as (4) is enough to represent four states, but in classical representation at least four chromosomes, $|000\rangle, |001\rangle, |100\rangle$and$|101\rangle$ are needed.

QIEA maintains a population of quantum chromosomes, $Q(t) = \{q_1^t, q_2^t, \cdots, q_n^t\}$at generation t, where n is the size of population, and q_j^t is a quantum chromosome defined as

$$q_j^t = \begin{bmatrix} \alpha_{j1}^t & \alpha_{j2}^t & \cdots & \alpha_{jm}^t \\ \beta_{j1}^t & \beta_{j2}^t & \cdots & \beta_{jm}^t \end{bmatrix} \tag{6}$$

where m is the string length of the quantum chromosome, and$j = 1, 2, \cdots, n$

TABLE I
THE PARAMETERS AND THEIR RANGE OF DIFFERENT TYPES OF CNN LAYERS WITH AN EXAMPLE IN THE EXAMPLE COLUMN

Layer Type	Parameters	Range	Number of bits	Example
Convolution	Filter size	[1,8]	3	2(001)
	Feature maps	[1,128]	7	16(000 1111)
	Stride size	[1,4]	2	2(01)
	Summary		12	001 000 1111 01
Pooling	Kernel size	[1,4]	2	2(01)
	Stride size	[1,4]	2	2(01)
	Type: 1(maximal), 2(average)	[1,2]	1	2(1)
	Place holder	[1,64]	6	16(00 1111)
	Summary		11	01 01 1 00 1111
Fully-Connected	Neurons	[1,2048]	11	512(01111 111111)
Disabled	Place holder	[1,2048]	11	512(01111 111111)

In this work, the CNN architecture is comprised of three types of layers – Convolutional Layer, Pooling Layer, and Fully-Connected Layer. In order to deal with the variable-length of the architectures of CNN, Disabled Layer, a special layer of disabling some of the layers will be used.

As there are several attributes in the configuration of each type of CNN layers, each of which is an integer value within a range, so each value of the attribute can be smoothly converted to several binary strings. As for Convolutional Layer, there are three key parameters, the filter size, the number of feature maps and the stride size. Secondly, in terms of Pooling Layer, the key parameters are the kernel size, the stride size and the pooling type. Lastly, Fully-Connected Layer only has one parameter, the number of neurons. This binary strings can be concatenated to a large binary string, which is called chromosome, to represent the whole configuration of CNN. The information of different types of layers in terms of both the number of parameters and the range in each parameter is shown in table I. First of all, the length of the binary string of different types of layers is various, so we need to design the length of the string. With regard to Convolutional Layer, the length of string is 12 and the range of the parameters are set to [1,8], [1,128] and [1,4]. For instance, a filter size with the string of 000 means the size is 1 (filter size cannot set to 0). Form table I, the Pooling Layer, Fully-Connected Layer and Disabled Layer have the same length of string with 11. As the largest number of bits to represent a layer is 12, we can

represent this binary string by 2 bytes (one bytes is comprised of 8 bits) and transfer it to dot-decimal notation.

Secondly, in order to define a specific type of layers, the string of dot-decimal with a fixed length of enough capacity can be designed to accommodate all the types of CNN layers and then the dot-decimal string can be divided into numerous sublayers, which can define a specific type of layers. The range of sublayers is listed in table II. For Convolution Layer, 0.0 is designed as the starting; in addition, the total length of the designed dot-decimal string is 16 and the total number of bits required is 12, so the total number of bits subtract from the length of the dot-decimal string is 4, which brings the sublayer representation to 0.0/4 with the range from 0.0 to 15.255. Regarding the Pooling layer, the total number of bits is 11, which results in 16.0/5 with the range from 16.0 to 23.255 as the sublayer representation of the Pooling layer. Similarly, the sublayer 24.0/5 with the range from 24.0 to 31.255 is designed as the sublayer of the Fully-connected layer.

TABLE II
THE RANGE OF BINARY STRING TO DIFFERENT LAYER

Layer type	Sublayer (Dot-decimal)	Range(Dot-decimal)
Convolutional Layer	0.0/4	0.0-15.255
Pooling Layer	16.0/5	16.0-23.255
Fully-Connected Layer	24.0/5	24.0-31.255
Disabled Layer	32.0/5	32.0-39.255

TABLE III
AN EXAMPLE OF BINARY STRING TO DIFFERENT LAYER

Layer type	Binary	Dot-decimal
Convolutional Layer	(0000)001 0001111 01	2.61
Pooling Layer	(00010)01 01 0 001111	18.143
Fully-Connected Layer	(00011)01111 111111	27.255
Disabled Layer	(00100)01111 111111	35.255

TABLE IV
AN EXAMPLE OF DOT-DECIMAL IN A CHROMOSOME CONTAINING 5 CNN LAYERS

2.61(C)	18.143(P)	2.61(C)	35.255(D)	27.255(F)

table IV shows an example of a chromosome to explain how the CNN architecture is encoded. Assume the length of layer is 5, the length of quantum chromosome will be 80, after observation, this quantum chromosome collapsed to a determined string. In this example, assume the quantum chromosome collapsed to the sample string in table III, where C represents a Convolution Layer, P represents a Pooling Layer, F represents a Fully-connected Layer, and D represents a Disabled Layer. Since there is a Disabled Layer, the actual number of layers is 4. However, after a few population updates, the Disabled Layer may be become 18.143, which turns the Disabled Layer to a Pooling Layer. All in all, the Disabled Layer should be disable when decode the binary string to CNN, so this encoding strategy can represent variable-length architectures of CNN.

C. Initialization

In terms of the population initialization, there are a few parameters that need to be determined, which are listed in table VI. A matrix is used to represent the individual of quantum population. Each dimensional of the matrix represent α and β (both equal to $\frac{1}{\sqrt{2}}$ at initialization stage)of qubit in the quantum chromosome.

In order to ensure the availability of each searched network architecture (the output label matches the classification label) and the rationality of the network architecture (without a large number of duplicate network layer stacks), the population initialization should follow the rules. L represents maximum length of CNN layers and F represents maximum number of Fully-connected Layers. The first element in the chromosome will always be a Convolution Layer; From the second to (L–F) layer, each element can be filled with a Convolutional Layer, Pooling Layer or Disabled Layer; From (L-F) to (L-1) layer, it can be filled with any of the four types of layers until the first Fully-connected is added, and after that only Fully-connected layers or Disabled layers are allowed; The last element will always be a Fully-connected layer with the size the same as the number of classes.

D. Quantum Observation

Before train and evaluate the CNN, the CNN architecture should accord with the initial rules and the quantum chromosome must collapse to a determined binary string. The quantum chromosome is comprised of m-qubits. As is shown in Algorithm 2, we observe the chromosome with a given random number and then get the binary string.

Algorithm 2 Quantum chromosome observation

Input:
Q_{in} with quantum chromosome
$m \leftarrow$ length of quantum chromosome
Output: Q_{out} with determined binary string

```
for individual q in Q_in do
    i ← 1
    while i <= m do
        r ← random(0, 1)
        if r > α_i^2 then
            q_i = 1
        else
            q_i = 0
        end if
    end while
end for
return Q_out
```

E. Evaluation Estimate Strategy

In this section, we explain how we predict the performance of networks from the initial portion of a learning curve. The performance of networks is denoted by validation dataset accuracy in the iteration process. Let y_n denote the validation

TABLE V
11 DIFFERENT PARAMETRIC LEARNING CURVE MODELS

Name	Formula
vapor pressure	$\exp\left(a+\frac{b}{x}+c\log(x)\right)$
pow3	$c-ax^{-\alpha}$
log log linear	$\log(a\log(x)+b)$
Hill3	$\frac{y_{\max}x^{\eta}}{\kappa^{\eta}+x^{\eta}}$
log power	$\frac{a}{1+\left(\frac{x}{e^{b}}\right)^{c}}$
pow4	$c-(ax+b)^{-\alpha}$
MMF	$\alpha-\frac{\alpha-\beta}{1+(\kappa x)^{\delta}}$
exp4	$c-e^{-ax^{\alpha}+b}$
Janoschek	$\alpha-(\alpha-\beta)e^{-\kappa x^{\delta}}$
Weibull	$\alpha-(\alpha-\beta)e^{-(\kappa x)^{\delta}}$
ilog2	$c-\frac{a}{\log x}$

dataset accuracy for the first n iterations and y_m denote the validation dataset accuracy after a large number of iterations.

Our basic idea is to model the learning curve by a set of parametric models $\{f_1,\ldots,f_K\}$. With the help of observing y_n, we aim to predict probability of $P\left(y_{\mathrm{m}}\geq\hat{y}|y_n\right)($ $\hat{y}$ is the best performance at this generation) by using Markov Chain Monte Carlo (MCMC) approach and decide whether to terminate training process which is time-consuming. Each of these functions is described through a set of parameters $\boldsymbol{\theta}_k$ and the probability of y_t under model f_k is given as

$$p\left(y_t|\boldsymbol{\theta}_k,\sigma^2\right)=\mathcal{N}\left(y_t;f_k\left(t|\boldsymbol{\theta}_k\right),\sigma^2\right) \tag{7}$$

We chose a large set of parametric curve models whose shape coincides with our prior knowledge about the form of learning curves: They are typically increasing, saturating functions; for example, functions from the power law or the sigmoidal family. In total we considered K = 11 different model families, which are shown in table V. We note that all of these models capture certain aspects of learning curves, but that no single model can describe all learning curves by itself, motivating us to combine the models in a probabilistic framework.

We combine all K models into a model which is given as:

$$f(t|\boldsymbol{\xi})=\sum_{k=1}^{K}w_k f_k\left(t|\boldsymbol{\theta}_k\right) \tag{8}$$

where the parameter vector is:

$$\boldsymbol{\xi}=\left(w_1,\ldots,w_K,\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_K,\sigma^2\right) \tag{9}$$

In order to use MCMC approach ,we have to know the distribution function of $\boldsymbol{\xi}$ with the observing accuracy y_n. According to Bayes theorem, $P\left(\xi|y_n\right)$ can be given as:

$$P\left(\boldsymbol{\xi}|y_n\right)\propto P\left(y_n|\boldsymbol{\xi}\right)P(\boldsymbol{\xi}) \tag{10}$$

Where $P\left(y_n|\xi\right)$ for the model is given as $P(y_n|\boldsymbol{\xi})=\Pi_{t=1}^{n}\mathcal{N}(y_t;f(t|\boldsymbol{\xi}),\sigma^2)$ and $P(\boldsymbol{\xi})$ is the prior distribution of $\boldsymbol{\xi}$ which is decided by $P(w_K)$ and $P(\boldsymbol{\theta}_K)$.

Then we can sample over the joint parameter and weight space $\boldsymbol{\xi}$ and get S samples $\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_S$ so that a sample approximation for y_m can be formed as

$$\mathbb{E}\left[y_m|y_n\right]\approx\frac{1}{S}\sum_{s=1}^{S}f\left(m|\boldsymbol{\xi}_s\right) \tag{11}$$

We have an estimate of the parameter and the predictive distribution $P\left(y_m|y_{1:n},\boldsymbol{\xi}\right)$ is a Gaussian for each fixed $\boldsymbol{\xi}$, so we can estimate the probability that y_m exceeds a certain value $\hat{y}$ as

$$\begin{aligned}P\left(y_{\mathrm{m}}\geq\hat{y}|y_n\right)&\approx\frac{1}{S}\sum_{s=1}^{S}P\left(y_m>\hat{y}|\boldsymbol{\xi}_s\right)\\&=\frac{1}{S}\sum_{s=1}^{S}\left(1-\Phi\left(\hat{y};f\left(m|\boldsymbol{\xi}_s\right),\sigma_s^2\right)\right)\end{aligned} \tag{12}$$

where $\Phi\left(\cdot;\mu,\sigma^2\right)$ is the cumulative distribution function of the Gaussian with mean μ and variance σ^2.

Before performing the network evaluation, a proper weight initialization method has to be chosen, and MSRA Filler weight initialization [31] is chosen as it keep the variance of each layer stable, and has been implemented in most of Deep Learning networks. As is shown in Algorithm 3, each quantum chromosome is observed and will be decoded to a CNN architecture with its settings. After training for each n iterations, we predict $P\left(y_{\mathrm{m}}\geq\hat{y}|y_n\right)$ by (12). If this probability is above a threshold δ then training continues as usual until the next n iterations. Otherwise, training is terminated and we return the expected validation accuracy. Finally, we return the accuracy for each individual, which will be stored as the individual fitness in Q_{out}

F. Update Networks with Quantum Gates

After quantum observation and network evaluation, we can update the network with quantum gates which is the important method for searching better individuals in the population. q_j^t is updated by applying some appropriate quantum gates $U(t)$. The appropriate quantum gates can be designed as rotation gates, such as

$$U(\phi)=\begin{bmatrix}\cos(\phi) & -\sin(\phi)\\ \sin(\phi) & \cos(\phi)\end{bmatrix} \tag{13}$$

where ϕ is a rotation angle, $s\left(\alpha_i\beta_i\right)$ is the rotation angle direction, which can be got from table VI. This step makes the quantum chromosomes converge to the fitter states. It should be noted that some genetic operators can be applied, such as mutation which creates new individuals by a small change in a single individual, and crossover which creates new individuals by combining parts from two or more individuals. Mutation and crossover can make the probability of linear superposition of states change. But because QIEA is a kind of probabilistic representation, the mutation and crossover are equivalent, so we only use rotation gate instead of mutation operation on the quantum chromosome.

Algorithm 3 Network Evaluation with Evaluation Estimate strategy

Require: The quantum population Q_{in}; the training dataset D_{train}; the size of dataset D_{size} the batch size s; the iteration intervals n; the best performance at that generation $\hat{y}$; the maximum iteration for each generation MAX;
Ensure: The population Q_{out} with with fitness;

```
for individual q in Q_in do
    i ← 1
    while i <= MAX do
        Decode q to CNN;
        Train the CNN on the D_train with s;
        if i = kn, k = 1, 2...MAX then
            if P(y_m ≥ ŷ|y_n) > δ then
                Training continues
            else
                Terminate training and return accuracy
            end if
        end if
        Return accuracy of q
        Update the fitness of q in the population Q_out;
    end while
end for
return Q_out
```

TABLE VI
ROTATION ANGLE OF QUANTUM GATES

x_i	$best_i$	$f(x) \geq f(best)$	ϕ	$s(\alpha_i\beta_i)$			
				$\alpha_i\beta_i > 0$	$\alpha_i\beta_i < 0$	$\alpha_i = 0$	$\beta_i = 0$
0	0	F	0	0	0	0	0
0	0	T	0	0	0	0	0
0	1	F	0	0	0	0	0
0	1	T	0.05π	-1	+1	±1	0
1	0	F	0.01π	-1	+1	±1	0
1	0	T	0.025π	+1	-1	0	±1
1	1	F	0.005π	+1	-1	0	±1
1	1	T	0.025π	+1	-1	0	±1

x_i is the i-th position of the current chromosome, $best_i$ is the i-th position of the best chromosome, $f(x)$ is the fitness function of the problem. QIEA mutates individuals through quantum rotation gates to create new individuals and the mutation rules also ensure the convergence of the algorithm. For example, $x_i = 0, best_i = 1, f(x) \geq f(best)$ indicates that the i-th position of the current chromosome is the '0' state, the i-th position of the best chromosome is the '1' state and the current chromosome fitness is better than local optimal chromosomes. In order to make the new individuals more likely to be observed as individuals with higher fitness,The mutation process should increase the probability that the i-th position of current chromosome takes the '0' state, that is to increase $|\alpha|^2$

The rotation angle in the quantum rotation gate determines the degree of mutation of the quantum chromosomes. The direction of rotation is determined by the relationship between the performance of the network architecture and the performance of the optimal network architecture in the current evolutionary iteration. The specific rotation direction and rotation angle can be obtained by looking up table VI.

TABLE VII
PARAMETERS LIST

Parameters	Value
Population size	30
Evaluation generation	30
The training epoch for evaluation	10
The training intervals n	60
threshold δ	0.05
The training epoch for q_{best}	30
Maximum length of CNN layers (L)	10
Maximum number of Fully-connected Layers (F)	3
Batch size	128
Activation function	ReLU
Optimizer	Adam

IV. EXPERIMENTS

A. Experimental Setup

For each dataset, we executed **10** times for QIEA and QIEA with evaluation estimate strategy(EESQIEA) respectively and averaged the results. QIEA and EESQIEA were evaluated on a single machine with a Tesla P100 and 16GB of CPU RAM. During the evolutionary process, the GPU utilization varied from 50% to 99% based on the dateset. The algorithm was developed in Python 3.6.4 and Pytorch 1.1.0. In the experiments, the peer competitors on the benchmarks are CAE-2 [32],RandNet-2 [33],PCANet-2(softmax) [33],LDANet-2 [33], Deep Belief Network (denoted DBN-3) [34], Stacked Auto Associators (denoted SAA-3) models [34], a single hidden-layer neural network (NNet) [34], SVM models with Gaussian (SVMrbf) and polynomial (SVMpoly) kernels [34] and Neural search by particle swarm optimization(IPPSO) [35]

B. Benchmark Datasets

The datasets of these experiments are three widely used image classification benchmark datasets. There are the MNIST digit recognition dataset [1], the MNIST with rotated digits plus background images(MNISTRB) [34] and the Convex Sets [34]. MNISTRB is a dataset with a random patch from a black and white image was used as the background for the digit image and the digits were rotated by an angle generated uniformly between 0 and 2π radians. The Convex Sets is for recognizing the shapes of objects and is a two-class classification problem. Each image in these datasets is with the size . There are the example of these datasets in Fig. 2.

C. Parameters

The parameters setting is listed in table VII. Each individual in population will train 10 epochs with batch size to 128. After evaluating for 30 generations, we train the best individual with 30 epochs and return the best network.

Fig. 2. Examples of the three datasets. From left to right, each two images as a group are from one benchmark, and each group is from MNIST, MNISTRB, and CS, respectively.

TABLE VIII
THE CLASSIFICATION ACCURACY OF QIEA AND EESQIEA AGAINST THE PEER COMPETITORS ON THE MNIST, MNISTRB AND CS BENCHMARK DATASETS

Classifier	MNIST	MNISTRB	CS
CAE-2	97.52 ↓	54.77 ↓	-
RandNet-2	98.75 ↓	56.31 ↓	94.55 ↑
PCANet-2	98.60 ↓	64.14 ↓	**95.81** ↑
LDANet-2	**98.95** ↑	61.46 ↓	92.78 ↓
DBN-3	96.89 ↓	52.61 ↓	81.37 ↓
SAA-3	96.54 ↓	48.07 ↓	81.59 ↓
NNet	95.31 ↓	37.84 ↓	67.75 ↓
SVM-RBF	96.97 ↓	44.82 ↓	80.87 ↓
SVM-Poly	96.31 ↓	43.59 ↓	80.18 ↓
QIEA(mean)	98.82	**79.80**	93.31
EESQIEA(mean)	98.81	79.77	93.30
QIEA(standard deviation)	0.097	1.54	1.63
EESQIEA(standard deviation)	0.098	0.67	1.06

TABLE IX
THE TIME OF QIEA AND EESQIEA

Datasets	Time(s)		Improvement
	EESQIEA	QIEA	
MNIST	**4985**	7120	↑29.98%
MNISTRB	**12421**	19342	↑35.78%
CS	**6213**	9921	↑37.37%

D. Result and Analysis

the results of experiments are shown in tab VIII It is obvious that our approach is better than all the peer competitors in MNISTRB dataset which is the most complicated dataset among these three and almost the greatest approach in the Convex Set. According to tab X, the performance of architecture search by QIEA is better than general evolutionary algorithm IPPSO in all datasets. In addition, the standard deviation of QIEA is steady in all the datasets, which means our approach is not easy to fall into the local optimum. More importantly, EESQIEA is faster than QIEA in all datesets, which is shown in tab IX

QIEA combines evolutionary algorithms and quantum characteristics and uses the representation of a quantum chromosome. Due to this representation of probability amplitudes, a quantum chromosome carries information about multiple states. Generating new individuals from the quantum chromosome can bring about a rich population, thereby maintaining the diversity of the population and overcoming precocity.

TABLE X
THE PERFORMANCE OF IPPSO AND QIEA

Datasets	best		mean		standard deviation	
	IPPSO	**QIEA**	IPPSO	**QIEA**	IPPSO	**QIEA**
MNIST	98.95	**98.97**	98.79	**98.82**	0.103	**0.097**
MNISTRB	67.50	**80.19**	65.20	**79.80**	2.96	**1.54**
CS	91.52	**94.52**	87.94	**93.31**	2.25	**1.63**

E. Visualization

In order to explain the advantage of EESQIEA, we visualise the performance for each generation and its time consumption on MNISTRB and CS.

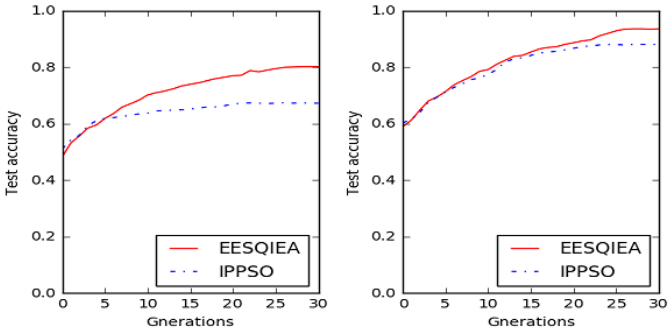

Fig. 3. Comparison of learning curves for EESQIEA and IPPSO on MNISTRB and CS with evaluation estimate strategy

The best result of each generation on MNISTRB and CS are plotted in solid and dashed lines, respectively, in Fig 3a and 3b. After 10 generations, the evolution of IPPSO have stop but EESQIEA can search the better individual until the 28rd generations on MNISTRB.

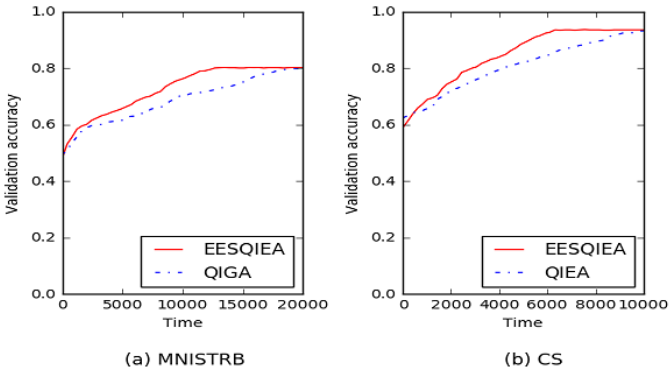

Fig. 4. Results for EESQIEA on on MNISTRB and CS with and without evaluation estimate strategy

The performance of evolution process on MNISTRB and CS are plotted in solid and dashed lines, respectively, in Fig 4a and 4b. EESQIEA with evaluation estimate strategy takes less time to reach the same performance as QIEA because it can stop bad runs early.

V. CONCLUSIONS

The paper proposed a quantum-inspired evolutionary algorithm to search the architectures of convolutional neural

networks for image classification tasks. QIEA approach can search for classifiers that perform better than traditional machine learning algorithms on the proposed benchmark datasets automatically.

More importantly, QIEA performs better than traditional evolution algorithm like IPPSO because it can maintain population diversity and overcome premature convergence phenomenon with the quantum characteristics during the process of searching. The decision variable is no longer a fixed information in a sense, but becomes a kind of information carrying different superimposed state information when a new individual is generated by the quantum probability amplitude, so it can bring a richer population than simply using genetic operations.

As the networks evaluation process is time-consuming, we predict the network performance from the initial portion of a learning curve and stop the bad runs early so that it can avoid the unnecessary training. Our approach with evaluation estimate strategy EESQIEA is almost 40% faster than QIEA with little performance loss.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[5] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal processing magazine*, vol. 29, 2012.

[6] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[7] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *arXiv preprint arXiv:1808.05377*, 2018.

[8] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," *arXiv preprint arXiv:1812.09926*, 2018.

[9] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2902–2911.

[10] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[11] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *arXiv preprint arXiv:1802.01548*, 2018.

[12] L. Xie and A. Yuille, "Genetic cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1379–1388.

[13] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," *arXiv preprint arXiv:1711.00436*, 2017.

[14] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[15] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[16] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[17] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 497–504.

[18] R. Shang, L. Jiao, Y. Ren, L. Li, and L. Wang, "Quantum immune clonal coevolutionary algorithm for dynamic multiobjective optimization," *Soft Computing*, vol. 18, no. 4, pp. 743–756, apr 2014.

[19] Y. Wang, Y. Li, and L. Jiao, "Quantum-inspired multi-objective optimization evolutionary algorithm based on decomposition," *Soft Computing*, vol. 20, no. 8, pp. 3257–3272, aug 2016.

[20] S. Jmal, B. Haddar, and H. Chabchoub, "Apply the quantum particle swarm optimization for the k-traveling repairman problem," *Soft Computing*, pp. 1–14, 2019.

[21] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

[22] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[23] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial life*, vol. 15, no. 2, pp. 185–212, 2009.

[24] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[25] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[26] D. Ashlock, *Evolutionary computation for modeling and optimization*. Springer Science & Business Media, 2006.

[27] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, p. 15023, 2016.

[28] K.-H. Han and J.-H. Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem," in *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, vol. 2. IEEE, 2000, pp. 1354–1360.

[29] K. Swersky, J. Snoek, and R. P. Adams, "Freeze-thaw bayesian optimization," *arXiv preprint arXiv:1406.3896*, 2014.

[30] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[32] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 2011, pp. 833–840.

[33] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, "Pcanet: A simple deep learning baseline for image classification?" *IEEE transactions on image processing*, vol. 24, no. 12, pp. 5017–5032, 2015.

[34] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 473–480.

[35] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.