Evolving Deep Multiple Kernel Learning Networks Through Genetic Algorithms

Wangbo Shen, Weiwei Lin, *Member, IEEE*, Yulei Wu, *Senior Member, IEEE*, Fang Shi, Wentai Wu, *Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Today's Industrial Internet of Things (IIoT) have achieved excellent manufacturing efficiency and automation results by leveraging machine learning (ML) and deep learning (DL). However, trustworthiness of ML/DL brings significant challenges to IIoT. This article proposes an evolving deep multiple kernel learning network through genetic algorithm (KNGA). Our KNGA method uses genetic algorithm (GA) to find the best deep multiple kernel learning structure, including the weights and the topology of the model. Compared with the current well-known models, KNGA has advantages in three aspects: 1) It can achieve good results without using many samples during model training; 2) the model can evolve in the process of training, including self-growth, and self-pruning; and 3) its trustworthiness and reliability can be guaranteed. Moreover, the whole model ensures excellent performance and requires manual adjustment of only a few parameters. Extensive experiments on the UCI, KEEL, Caltech256, and MNIST datasets demonstrate the effectiveness and trustworthiness of the proposed method.**

***Index Terms*—AutoML, evolution algorithm, kernel learning, neural networks, trustworthiness.**

Manuscript received 29 December 2021; revised 21 April 2022 and 26 July 2022; accepted 4 September 2022. Date of publication 15 September 2022; date of current version 13 December 2022. This work was supported in part by Key-Area Research and Development Program of Guangdong Province under Grant 2021B0101420002, in part by the National Natural Science Foundation of China under Grant 62072187 and Grant 61872084, in part by the Guangdong Major Project of Basic and Applied Basic Research under Grant 2019B030302002, in part by the Major Key Project of PCL under Grant PCL2021A09, and in part by the Guangzhou Development Zone Science and Technology Project under Grant 2021GH10 and Grant 2020GH10. Paper no. TII-21-5822. *(Corresponding author: Weiwei Lin.)*

Wangbo Shen and Fang Shi are with the School of Computer Science & Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: 202010107337@mail.scut.edu.cn; 978772638@qq.com).

Weiwei Lin is with the School of Computer Science & Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Pengcheng Laboratory, Shenzhen 518055, China (e-mail: linww@scut.edu.cn).

Yulei Wu is with the Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter EX4 4QF, U.K. (e-mail: y.l.wu@exeter.ac.uk).

Wentai Wu is with the Pengcheng Laboratory, Shenzhen 518055, China (e-mail: wentai.wu@warwick.ac.uk).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3206817.

Digital Object Identifier 10.1109/TII.2022.3206817

I. Introduction

RECENTLY, our society has developed and deployed a wide range of Internet of Things (IoT) applications, because it can provide automation systems and services not only for people but also for industrial sectors [1], [2]. The success of automation in IoT has facilitated the formation and vigorous development of the current Industrial IoT (IIoT). However, compared with IoT applications, today's IIoT focuses more on connecting machines and equipment in industries, such as oil and gas, electric utilities, and healthcare. As a result, IIoT is exposed to more life-threatening or high-risk situations due to system failures and outages, and faces more system security and model trustworthiness issues [3], [4]. IIoT technologies are supported by artificial intelligence (AI) approaches and machine learning/deep learning (ML/DL) models. Although many AI approaches and ML/DL models, such as recurrent neural networks (RNN) [5], convolutional neural networks (CNN) [6], and other neural networks show an excellent performance in many specific application scenarios, they still encounter several problems that need to be addressed urgentlyas follows.

1) This kind of feed-forward neural networks based on back-propagation (BP) algorithms belongs to black-box models. Therefore, its trustworthiness and reliability cannot be guaranteed.
2) Network models cannot recover from attacks, such as data poisoning, data collusion, security violation, and indiscriminate attacks due to the inexplicability of network models.
3) Most of these models rely on parameters tuning based on trial and error, which requires tremendous workforce and computing power.
4) The training of models requires a large number of samples.

In terms of the interpretability of the model, support vector machines (SVM) has been proved mathematically [7], and the multiple kernel learning (MKL) based on it has also shown good performance in classification and regression problems [8]. Nevertheless, MKL receives less attention as deep neural networks (DNN) become more powerful in the era of Big Data [9]. Interestingly, with the rapid development of DNN, many studies, such as [10] introduced the idea of DNN into MKL to form the deep multiple kernel learning (DMKL) structure. DMKL transforms the input data through multiple nonlinear processing layers to construct new features which can make a dramatic

improvement in pattern recognition. Meanwhile, each kernel function in DMKL can be regarded as a dimensional change of the current sample and can also form a single classifier, which makes the whole training process of DMKL observable and makes DMKL more trustworthy compared with the current neural network models [11].

Automation is also a crucial indicator in IIoT. Automatic algorithm tuning is essential for any models, including DMKL. In the course of network topology formation and kernel function parameters autotuning research, Ren et al. [12] used Rademacher chaos complexity to evaluate each kernel function in the DMKL network and remove the bad ones. Meanwhile, a grid search method was used to optimize the parameters of the kernel function. Thus, a self-adaptive deep multiple kernel learning (SA-DMKL) method was built. Liu et al. [13] presented a group-based local adaptive deep multiple kernel learning (GLDMKL) method with lp norm, and divided samples into multiple groups using the multiple kernel k-means clustering algorithm. However, there are also some limitations of DMKL models. For example, parameter adjustment of the main models is not flexible enough, and the structure of some models is not elastic enough.

Network equipment used in IIoT applications has various limitations related to energy, processing, and communication, making it hard to have full-scale AI/ML/DL approaches running on tiny devices. This is also why many excellent AI models need to be carefully deployed in IIoT.

To address the above issues, in this article we use ideas from the studies that optimize networks through evolutionary algorithms [14], [15], and develop DMKL with a genetic algorithm (GA) algorithm, hence the kernel learning network through GA (KNGA) model is proposed. By comparing with one traditional ML model and three advanced automatic ML models, it is shown that the training of KNGA model only needs very few hyperparameters. At the same time, after many times of GA optimization, the model can automatically reach the optimal network topology and weights.

In summary, we make the following key contributions.

1) A new KNGA model and its training methods are proposed, which can automatically adjust network parameters and topology adaptively.
2) A comparison of the proposed KNGA with some recent autotuning neural network models in terms of the number of hyperparameters is summarized. It is proved that the proposed model can achieve a good classification result only by manually adjusting a few hyperparameters.
3) An evaluation of KNGA is carried out in terms of the convergence time, classification ability, and the sensitivity to parameter settings. Compared with the state of the art and classical models, KNGA proves its excellent performance and trustworthiness.

The rest of this article is organized as follows. DMKL is introduced in Sections II and III as background, Sections IV and V present the proposed model. Section VI analyzes the model theoretically, and Section VII carries out performance analysis. Finally, Section VIII concludes this article.

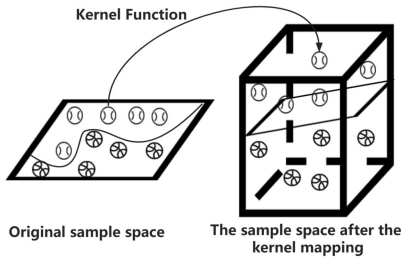


Fig. 1. Datasets that were previously linearly indivisible become linearly separable in higher dimensions.

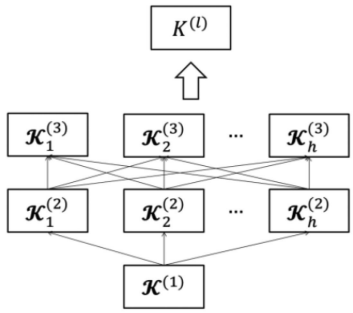


Fig. 2. Depiction of a deep multiple kernel architecture. Lines represent the weights for each connection.

II. Background

As the most crucial computing tool of SVM, its kernel function is to calculate the inner product distance between the input samples in higher dimensions, which can be expressed as

$$K(x, y) = \phi(x) \cdot \phi(y)$$

where $K^{(1)}(x, y)$ represents a kernel function, and different kernel functions like RBF, sigmoid, or Poly have different calculation formulas. Sometimes the sample is linearly indivisible in lower dimensions, but it may be linearly separable in higher dimensions, as shown in Fig. 1. Moreover, the kernel is used to simplify this process, and the theoretical proof and research on kernel function can be found in [16] and [17]. Then the DMKL model is a network formed by combining multiple kernel functions with weights at different layers shown in Fig. 2. From [11], DMKL can be defined as follows.

Definition 1: A deep multiple kernel architecture is an l-level multiple kernel architecture with h sets of m kernels at each layer

$$K^{(l)} = \left\{ \theta_{1,1}^{(l)} K_{1,1}^{(l)} \left(\theta_{1,1}^{(l-1)} K_{1,1}^{(l-1)} + \cdots \right) + \cdots \theta_{h,m}^{(l)} K_{h,m}^{(l)}(\ldots) \right\}$$

where $K_{h,m}^{(l)}$ represents the mth kernel function in set h at layer l with an connected weight $\theta_{h,m}^{(l)}$, and $K^{(l)}$ means the combined kernel at layer l. $\kappa^{(l)}$ represents all kernels in layer l. In general, compared with the previous single kernel learning, the purpose of DMKL is to combine the advantages of each kernel function and make the performance of the model better.

III. Related Work

After introducing the theoretical background of DMKL, this section will show the availability of kernel learning through investigating the adoption of MKL in the broad industry applications.

Kernel learning has been an active research topic in ML since its widespread use in 2004 [18]. These kernel methods have been successfully applied to various real-world applications and generally showed auspicious performance. Even now, kernel learning can perform very well in small data scenarios [19], [20], [21]. However, with the emergence of DNN algorithms that rely on mighty computing power and Big Data environment, the dominant position of kernel learning is gradually replaced. In recent years, as scholars began to combine multikernel learning with a deep learning framework and achieved excellent research results in some scenarios, even exceeding the performance of a DNN, kernel learning received attention again [22].

With the successful combination of kernel learning and deep learning, many people have applied kernel learning technology in practical applications in recent years. For example, Zhang et al. [23] proposed a multikernel ELM (MKELM)-based method to classify EEG images based on DMKL. In [24], the authors also proposed an DMKL method named MK-FSVM-SVDD for predicting DNA-binding proteins. Singh et al. [25] proposed an MKL model named SAMKL for lung cancer prediction. Guo et al. [26] and Shi et al. [27] assessed dry weight of hemodialysis patients and classified pre-microRNA using MKL.

Next, we briefly introduce the DMKL model proposed in this article.

IV. Structure Overview

This section introduces the constituent units of the proposed model, the way of coding, and how the model structure grows.

A. Basic Unit

The proposed model comprises L basic units or layers, and each layer contains n kernel functions. The output of each kernel function is connected with an activation function, such as the tanh, sigmoid, and ELU activation functions. In addition, the option of no activation function is also considered. Fig. 3 presents the basic unit's composition rules that combine multiple kernels and activation functions in each layer.

B. Genetic Encoding

This model's gene coding mimics the human gene coding. Each kernel function like the base pairs of A (adenine), T(thymine), C(cytosine), G(guanine), and U(uracil) in our gene sequence.

DMKL's network structure can be encoded by mimicking how DNA is compiled, where each layer corresponds to a base pair composed of n kernel functions like a base pair in a DNA strand. For example, assume the primary network structure of the proposed model is made up of L layers and its genetic

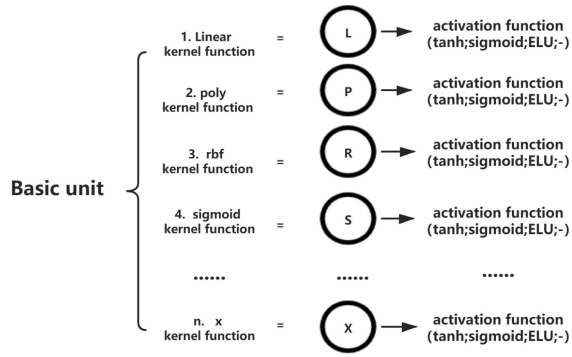


Fig. 3. Base unit contains n kernel functions.

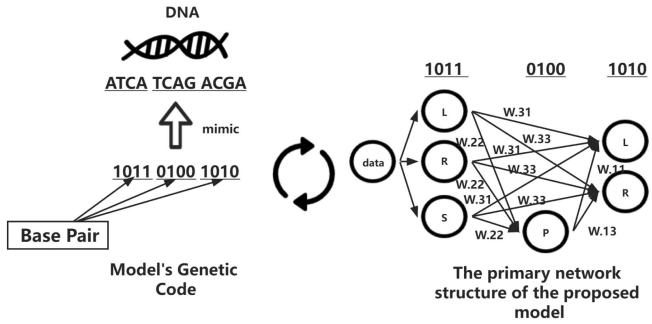


Fig. 4. Primary network structure of the proposed model is compiled in a four-digit two-level system.

code composed by L base-pair, as shown in Fig. 4. Then, the corresponding base pairs can be compiled in a n-digit two-level system, where 1 denotes the corresponding kernel function is active, and 0 represents the corresponding kernel function is deactivated.

In the early stages of model formation, m nonidentical candidates are randomly generated, namely, $M = G_1, G_2, \ldots, G_m$, where G represents a candidate. Each candidate represents a topology model of the DMKL network.

After generating m networks, model weighting is needed. In this model, each active kernel function needs to establish a weighted connection with others (the kernel functions at each layer are disconnected from each other). At the same time, $W.ij$ means the weight connection between the current kernel function and the jth kernel function at the ith layer after the current layer. Therefore, the model results in m weighted networks M^W. After that, the model adopts the optimization algorithm to optimize M^W to find M^{best_W} of every network.

C. Find the Optimal Topology

When the model forms m weighted networks, the topology structure is optimized. After that, the model adopts the optimization algorithm to optimize the network topology structured among m weighted networks to evolve to the best topology structure M_{best_C}, as shown in Fig. 5. Finally, the model gets an optimal solution of the model $M_{\text{best}_C}^{\text{best}_W}$.

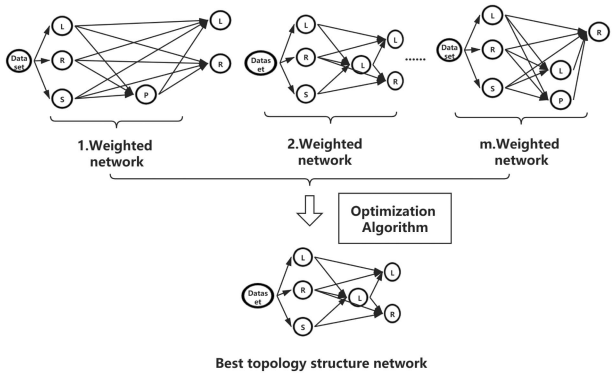


Fig. 5. Selecting candidate model topologies using GA.

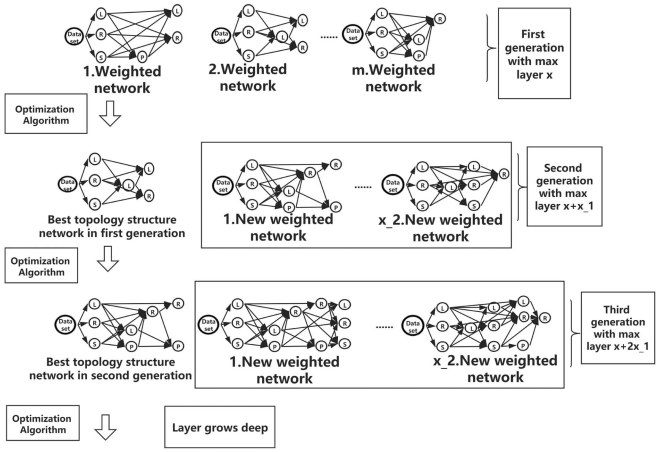


Fig. 6. Model layers growing.

D. Model Layers Growing

When the optimal network $M_{\text{best}_C}^{\text{best}W}$ obtained by the model that is trained following the above steps still fails to meet the required indicators, x_2 weighted networks with $x + x_1$ layers are generated based on the current network M. The above steps continue to process until the required indicators are met, as shown in Fig. 6. x represents the number of layers in M, and x_1 and x_2 are random integers greater than 0.

V. Optimization Algorithm

This section introduces how the proposed model KNGA can optimize its model parameters by GA.

A. Model Weighting of Networks With Different Topologies Through GA

Each deep network structure from M randomly generates k initial weighted populations, and the random generation method uses the traditional artificial neural network (ANN) method to initialize the network weight. Then, the initial connection weight population W corresponding to m deep network structures can be represented as $W = P_1, P_2, P_3, P_4, \ldots, P_i, \ldots, P_m$, where P_i represents the initial connection weight population corresponding to the ith deep network and $P_i = \omega_1, \omega_2, \omega_3, \omega_4, \ldots, \omega_j, \ldots, \omega_k$, where ω_j denotes the jth individual in the population of k initial weights generated randomly.

1) Gene Coding: Assuming that each generated network has L layers, each of which consists of n essential kernel functions. The set of connection weights corresponding to the network topology can be expressed as a $(L \cdot n) \times (L \cdot n)$ 2-D matrix ω, as shown in Fig. 7. A matrix ω corresponds to a genetic code of connection weights. The number of weighted connections N in the network is shown in the following formula:

$$N = \frac{((L-1) \cdot L)}{2} \times n^2. \tag{1}$$

If the deactivated kernel function is at the ith kernel function of layer j, the position $(j-1-k) \cdot n + i$ in the connection array of the k layers' every kernel function (a column in matrix ω) at the previous layer will be set to 0, where n is the number of essential kernel function.

2) Objective Function: The minimum value of network error $E(\omega)$ in the current evolution band is taken as the objective function for the fitness function. Then, the fitness function is proposed while a coefficient of positive number δ is added. Finally, the objective function of the genetic algorithm is obtained, as

$$F(\omega) = \frac{\delta}{E(\omega_i)}, \quad i = 1, 2, 3 \ldots, k. \tag{2}$$

3) Selection: According to the fitness of the contemporary gene group, the gene with the highest fitness level is directly entered into the next generation, and the gene with the lowest fitness level is directly eliminated. The remaining genes are selected into the next generation according to the order of fitness. The target is to retain 80% of the genes into the next generation for selection, and the remaining 20% of individuals are generated through crossover and mutation operation. Thus, P_{select} represents the probability that the individual I is selected after the state changes

$$P_{\text{select}} = \begin{cases} 1, & f(\omega') \geq f(\omega) \\ 0, & f(\omega') < f(\omega). \end{cases} \tag{3}$$

4) Mutation: In the mutation step, $P_{(\omega_1 \to v_1)}$ represents the probability of individual ω_1 mutation, and it is calculated by the following formula:

$$P_{(\omega_1 \to v_1)} = \begin{cases} \frac{k_1(f_{\max} - f)}{f_{\max} - f_{\text{avg}}}, & f \geq f_{\text{avg}} \\ k_2, & f < f_{\text{avg}} \end{cases} \tag{4}$$

where $f_{\max}$ represents the maximum fitness value in the population; f denotes the individual fitness value that requires variation; f_{avg} is the average fitness value in the population; k_1 and k_2 are two constants, both of which are set to 0.5.

There are two specific steps in the process of individual ω_1 mutation. First, the q-dimension of an individual ω_1 must mutate. In the second step, $p \in (1, \frac{(L \cdot (L-1))}{2} \cdot n^2)$ except for q would have a 50% chance to mutate, and the individual ω_1 would mutate between 0.01 and 0.1 plus or minus.

$$\begin{array}{c} \begin{array}{cccccccccc} 1 & 2 & \cdots & n & n+1 & \cdots & L\times(n-1) & \cdots & L\times n & \end{array} \\ \left(\begin{array}{cccccccccc} - & - & \cdots & - & - & \cdots & - & \cdots & - \\ - & - & \cdots & - & - & \cdots & - & \cdots & - \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ - & - & \cdots & - & - & \cdots & - & \cdots & - \\ a_{11} & a_{21} & \cdots & a_{n1} & - & \cdots & - & \cdots & - \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{1(n\times(n-2))} & a_{2(n\times(n-2))} & \cdots & a_{n(n\times(n-2))} & a_{(n+1)(n\times(n-3))} & \cdots & - & \cdots & - \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{1(n\times(n-1))} & a_{2(n\times(n-1))} & \cdots & a_{n(n\times(n-1))} & a_{(n+1)(n\times(n-2))} & \cdots & a_{(L\times(n-1))n} & \cdots & - \end{array}\right) \end{array} \begin{array}{c} 1 \\ 2 \\ \cdots \\ n \\ n+1 \\ \cdots \\ L\times(n-1) \\ \cdots \\ L\times n \end{array}$$

Fig. 7. $(L\cdot n)\times(L\cdot n)$ 2-D matrix ω (the kernel functions at each layer are disconnected from each other).

5) Crossover: In the cross step, $P_{((\omega_1,\omega_3)\to v_2)}$ represents the crossover probability, and it is calculated by the following formula:

$$P_{((\omega_1,\omega_3)\to v_2)} = \prod_{i=1}^{\frac{(L\cdot(L-1))}{2}\cdot n^2} K(i) \tag{5}$$

where $K(i)$ represents the probability of each dimension crossing which is calculated by the following formula:

$$K(i) = \begin{cases} 1, (\omega_1 == v_2) \quad or \quad (i == q) \\ (1-P_{\text{cross}})\cdot \text{Sel}\left(\text{rand}(i) > P_{\text{cross}}\right) \\ +P_{\text{cross}}\cdot \text{Sel}\left(\text{rand}(i) \le P_{\text{cross}}\right), \text{otherwise} \end{cases} \tag{6}$$

where q denotes that the q-dimension of the two populations must crossover; P_{cross} represents preset crossover probability; $\text{Sel}(x) = \begin{cases} 1, x \quad is \quad \text{true} \\ 0, x \quad is \quad \text{false} \end{cases}$; $\text{rand}(i)$ is a random floating-point number from 0 to 1. The heuristic crossover is adopted. If A in father A and father B has good fitness, the generation of child individual C can be expressed by the following formula:

$$C_{\text{child}} = P_{\text{better}} + \lambda(P_{\text{better}} - P_{\text{less}}) \tag{7}$$

where P_{better} represents fathers with better fitness and P_{less} fathers with poor fitness. λ is a default parameter that specifies how far the children are from the parent with better fitness, usually set to 1.5.

Algorithm 1: KNGA Algorithms.

Data: Training dataset D_{train} and Test dataset D_{test}; The number of kernel functions n corresponding several kernel function; Number of topology network structures m; The number of weight populations corresponding to each network k; Number of the initialtopology network structures's layer x; Network default target value V_{target} ; Maximum fitness of the model V_{best}; The preset constant x_1; The preset constant x_2; The preset constant x_3.
Result: Solution of the model $M_{\text{best}_C}^{\text{best}_W}$.
$V_{\text{best}} = 0$;
while ($V_{\text{best}} \ge V_{\text{target}}$) *or (The model deepened* x_1 *layers for* x_3 *times while* V_{best} *did not change and* $V_{\text{best}} \le V_{\text{target}}$*)* **do**
  if $V_{\text{best}} == 0$ **then**
    Generate m nonidentical deep multiple kernel learning topology network structures M;
    Generate k initial connection weight populations corresponding to each network W;
    $populations = m$
  else
    Add x_2 nonidentical deep multiple kernel learning topology network with $x + x_1 \times generations$ layers to $M_{best_C}^{best_W}$;
    $populations = x_2 + 1$
  while $f(M_{best_C}^{best_W})$ *did not change for* x_3 *times evolutions* **do**
    for *i=0;i<populations;i++* **do**
      while $f(M_i^{best_W})$ *did not change for* x_3 *times evolutions* **do**
        $M_i^{best_W}$ = GA(M_i^W);
    $M_{best_C}^{best_W}$ = GA(M^{best_W});
    if $V_{best} < f(M_{best_C}^{best_W})$ **then**
      V_{best}=$f(M_{best_C}^{best_W})$;
return $M_{\text{best}_C}^{\text{best}_W}$;

B. Find the Optimal Topology Through GA

After Section V-A, m DMKL network M^{best_W} with optimal connection weight is obtained. Then, the network topology optimization of M^{best_W} is carried out again through the genetic algorithm to find out the optimal network topology M_{best_C}. When two networks with different layers need to crossover, the extra dimensions are filled up with 0. Finally, we obtain the optimal solution of the model $M_{\text{best}_C}^{\text{best}_W}$. The process can be represented by Algorithm 1.

VI. Theoretical Analysis

A. Analysis of Model Convergence

The whole model M contains m topological structures of deep kernel learning networks expressed as $M = \{P_1, P_2, P_3, P_4, \ldots, P_i, \ldots, P_m\}$, where P_i denotes the initial connection weight population corresponding to the ith topological structure of deep kernel learning networks, and $P_i = \{\omega_1, \omega_2, \omega_3, \omega_4, \ldots, \omega_j, \ldots, \omega_k\}$, where ω_j denotes randomly generating the jth individual in k initial weighted population.

This article designs a set of optimization algorithms suitable for DMKL. The optimization algorithms mainly include two layers of genetic algorithms. The first layer of the genetic algorithm is to optimize P_i, and the second layer of the genetic algorithm is to optimize M.

Lemma 1: In the current genetic algorithm, the state of individual I transforms from I_t to I_{t+1}, and the probability of $I_t \to I_{t+1}$ is $P_{(I_t\to I_{t+1})} = P_{(\omega_1\to v_1)}\cdot P_{((\omega_1,\omega_2)\to v_2)}\cdot P_{\text{select}}$, where I_t represents the state of individual I at time t, $P_{(\omega_1\to v_1)}$ denotes the probability of mutation, and $P_{((\omega_1,\omega_2)\to v_2)}$ represents the probability of crossover.

Lemma 2: In the iterative process of genetic algorithm of this model, the probability of population state S_i and one-step jump to state S_j is $P_{(S_i\to S_j)} = \prod_{x=1}^{k} P_{(I_{xi}\to I_{xj})}$.

It can be concluded from Lemmas 1 and 2 that the nested two-layer genetic algorithm in the algorithm belong to the differential evolution algorithm (DEA). Therefore, according to [28], the algorithm of this article must converge. However, the genetic

TABLE I
HYPERPARAMETERS COMPARISON OF FOUR MODELS

Model	Hyperparameters introduce	Symbol	Number of parameters
1.KNGA(Proposed)	Number of initial network structures**(important)**	m	**8**
	Number of initial connection weight populations**(important)**	k	
	Number of the initialtopology network structures's layer	x	
	Parameters that control the iterations of population evolution and network growing	x_1, x_2, x_3	
	Number of basic kernel functions **(important)**	n	
	Network default target value	V_{target}	
2.An automatic CNN architecture(2020) [29]	The probabilities of crossover and mutation	P_C, P_M	**9**
	Learning rate	α	
	Epochs	$Epochs$	
	The number of epochs that the learning rate is decayed	$Epochs - decayed$	
	The available numbers of feature maps	$Feature - maps$	
	The normalized probability of increasing the depth	$Normalized - depth$	
	The population size**(important)**	$Loss$	
	The number of generations**(important)**	$Generations$	
3.CIFAR-CNN(2021) [30]	Sampled DNN architectures(generate network parameters casually)	D_A	**10**
	Four regression models Selection	$Regression$	
	The minimum number of samples required to split an internal node**(important)**	$min - samples - split$	
	The minimum number of samples required to be a leaf node**(important)**	$min - samples - leaf$	
	The maximum depth of the tree**(important)**	$max - depth$	
	The number of features to consider for the best split**(important)**	$max - features$	
	The criterion to evaluate the model**(important)**	$criterion$	
	The parameters tuned for the RForest	$RForest$	
	The parameter of the learning rate regarding the corresponding gradient-based optimization	α	
	The parameters to be tuned are the kernel type, the kernel coefficient, and the regularization parameter	$Kernel$	
4.BNAS-CCE(2021) [31]	Number of convolution layers	C_L	**11**
	Number of pooling layers	P_L	
	Number of convolution kernels	K_N	
	Size of convolution kernels	K_S	
	Scanning step	$Step$	
	Weight initialization method	$Weight - Method$	
	The number of enhancement block**(important)**	v_s	
	The number of enhancement block in broad scalable architectures**(important)**	v_d	
	The number of deep Cell**(important)**	k_d	
	Minibatch set	$batch$	
	Epochs	$Epochs$	

algorithm of the model cannot guarantee that global convergence can be achieved by training on each generated population.

Although the optimization algorithm of the model cannot guarantee global convergence, the experiments and the actual application show that the optimization algorithm has a robust global search ability like the BP algorithm and particle swarm optimization (PSO) algorithm. Essentially, these methods cannot guarantee global convergence. They may easily fall into a local optimum, but the search capability still has strong stability and other advantages in industrial applications and experiments. Meanwhile, in model training, the maximum number of network layers of the whole network's population is constantly increasing, and new networks are constantly added, which can avoid the model falling into local optimum.

B. Hyperparameters Analysis

In this section, KNGA is compared with some cutting-edge models, an automatic CNN architecture [29], CIFAR-CNN [30], and BNAS-CCE [31], to show and analyze the number of hyperparameters of these models, as shown in Table I.

As we all know that traditional neural network models CNN have a large number of hyperparameters. In addition, almost every hyperparameter of the traditional neural network has a significant impact on the model's performance, which increases the complexity of artificial parameter adjustment. To solve this problem, the authors in [29] proposed an automatic optimization architecture of CNN based on genetic algorithms designed for image recognition. Furthermore, in [30], four kinds of regression models were used to optimize the network hyperparameters, and the authors in [31] proposed a broad neural architecture search called BNAS which can not only save time but also automatically optimize the hyperparameters.

This article proposes the model KNGA adopted two layers of genetic algorithms to optimize topology and hyperparameters. Compared with the recent representative neural architecture search (NAS) methods, KNGA has fewer hyperparameters and can essentially optimize network topology. At the same time, the subsequent experiments prove that KNGA also has good performance.

TABLE II
CLASSIFICATION RESULTS ON UCI

Dataset	Algorithms							
	SKSVM	L2MKL	SM1MKL	DMKL	MLMKL	SA-DMKL	ANN	**KNGA(proposed)**
liver	63.66%	67.50%	68.83%	69.01%	71.80%	75.65%	70.01%	**76.06%**
breast	94.25%	96.18%	96.36%	96.59%	**97.21%**	94.25%	96.49%	**97.21%**
sonar	50.29%	84.51%	85.00%	83.94%	83.84%	84.51%	82.61%	**85.79%**
Austrila	71.54%	81.64%	82.89%	84.40%	85.42%	82.03%	84.40%	**86.66%**
German	70.18%	69.98%	70.14%	72.02%	75.06%	72.52%	73.04%	**78.82%**
Monk	90.32%	97.22%	96.66%	96.62%	96.89%	**97.55%**	97.02%	97.22%

VII. EXPERIMENTAL RESULTS AND ANALYSIS

A. Experiment Settings

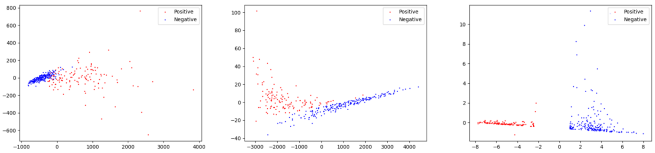
In this section, we select four base kernel functions, i.e., Linear, RBF, Poly, and Sigmoid, and their initial parameter settings are autoset by the sklearn package.

In order to simplify the experiments, the number of initial DMKL topology network structures is $m = 10$. The number of initial connection weight populations corresponding to each network is $k = 10$. The number of initial topology network structure's layer is $x = 4$. The parameters are set to $x_1 = 1$, $x_2 = 5$, $x_3 = 4$, and $V_{\text{target}} = 1.0$.

B. Datasets

1) UCI Datasets: This article selects six datasets from the UCI database, such as breast cancer [32] to evaluate the performance of the KNGA method for classification tasks that have samples and dimensions with small sizes.

In this article we present an exhaustive comparative study using the following algorithms: SKSVM (SVM algorithm with a single RBF kernel), L2MKL [33], SM1MKL [34], DMKL [11], and MLMKL [10], SA-DMKL [12], and an ANN method.



(a) (b) (c)

Fig. 8. The change of sample space mapped by the kernel function. (a) The original. (b) First layer. (c) Second layer.

TABLE III
FOUR SETS OF KERNEL FUNCTION PARAMETERS

Kernel type	Set 1	Set 2	Set 3	Set 4
Linear				
RBF	automatic	$\delta = 1.0$	$\delta = 10.0$	$\delta = 100.0$
Poly	automatic	$a = 2.0, b = 1.5, c = 2.0$	$a = 10.0, b = 20.0, c = 3.0$	$a = 100.0, b = 200.0, c = 4.0$
Sigmod	automatic	$a = 2.0, c = 1.5$	$a = 10.0, c = 15.0$	$a = 100.0, c = 150.0$

Table II shows the accuracy results of the classification with the above algorithms.

To show how DMKL works and how to enhance the model generalization ability through the continuous increase of layers, this article chooses the breast cancer dataset, which is one of the most famous datasets in ML, to conduct experiments. Furthermore, to better show the mapping of the sample space image and the space reasons, we present the sample space by the PCA method to draw dimension reduction to 2-D data. Thus, the three breast cancer datasets' sample space includes the original sample space, sample space after the first layer of kernel function mapping, and sample space after the second layer of kernel function mapping, as shown in Fig. 8.

2) KEEL Datasets: To further evaluate the generalization and self-evolution capability of the proposed model KNGA, we select six datasets Appendicitis (106), Balance (625), Heart (270), Haberman (306), Titanic (2201), Twonorm (7400) from the KEEL [35]. The number in parentheses represents the number of samples in its dataset [35].

In this article, the results obtained after automatic training of these six datasets in KEEL through KNGA are shown in Fig. 9. Each polyline represents the evolution of each initial population. The solid green line represents the final optimized candidate evolved from the ten initial populations, while the dotted green line denotes the optimal fitness of the current model.

To show recognition ability in the process of evolution more intuitively, the fitness recorded in Fig. 9 and the figures after that is denoted as: Fitness $=$ Accuracy $+ 0.0001 \times$ layers(added), rather than the value of the objective function, so that we can better observe the evolution of the model.

At the same time, an experiment is set to verify if the KNGA model is sensitive to the initial parameter setting of kernel functions, which is shown in Table III. We set up the four sets of different orders of magnitude of kernel function parameters on Appendicitis, Balance, Heart, and Haberman in the KEEL dataset and carry on the comparison experiment shown in Table IV.

3) Caltech 256 Datasets: The Caltech 256 dataset is a Caltech collated dataset from the Google Image dataset, and the images that do not fit its category are manually removed. In this dataset, images are divided into 256 categories, with more than 80 images in each category.

To verify whether the KNGA model is effective in the image dataset, we select Backpack, Butterfly, Knife, and Treadmill from this dataset. In addition, these four datasets are classified among each other, and the accuracy is shown in Table V.

4) MNIST: The DNN is well known because of its superior recognition performance (often better than humans). However, as DNN is getting thorough research, it has been found that DNN achieves excellent performance by finding many subtle features in some samples and then expanding these features by mapping the multilayer network structure. In other words, we can easily make the neural network's recognition result remarkably different by slightly modifying some parts of the sample without destroying the characteristics of the sample itself [36], which puts a big question mark on the trustworthiness of DNN. The methods described above can be collectively referred to as adversarial attacks, which can be roughly divided into two categories, i.e., white-box attacks and black-box attacks, which are essential criteria for verifying the trustworthiness of AI models [37]. This article applies five typical white-box attacks and one black-box attack on CNN, KNGA, two advanced defense models wBT [38] and RN [39] and some models based on a single kernel on the MNIST dataset, respectively, to test the trustworthiness of these models, as shown in Table VI. Since most current attacks are aimed at the network design of DNN, samples generated by the attacks will all be based on an independent CNN and are saved as test samples in the experimental design; some examples, as shown in Fig. 10.

TABLE IV
CLASSIFICATION RESULTS WITH FOUR SETS OF KERNEL FUNCTION PARAMETERS ON KEEL

Dataset	Parameter			
	Set 1	Set 2	Set 3	Set 4
Appendicitis	**0.8837→ 0.9533**	0.8837→ 0.9533	0.9069→ 0.9069	0.9069→ 0.9069
Balance	**0.9360→ 0.9679**	0.9360→ 0.9526	0.9360→ 0.9360	0.9360→ 0.9360
Heart	**0.7963→ 0.8330**	0.7963→ 0.8241	0.7963→ 0.7963	0.7963→ 0.7963
Haberman	**0.7154→ 0.8107**	0.7398→ 0.7967	0.7398→ 0.7561	0.7236→ 0.7480

TABLE V
CLASSIFICATION RESULTS ON CALTECH 256

Dataset	First layer's accuracy	Best accuracy
Backpack	0.9170	0.9414
Butterfly	0.8098	0.8244
Knife	0.8195	0.8537
Treadmill	0.8780	0.9024

TABLE VI
CLASSIFICATION RESULTS UNDER ADVERSARIAL ATTACKS

Test Samples	Models							
	CNN(Atk)	CNN	RBF	Poly	Sigmoid	wBT [38] (2021)	RN [39] (2021)	KNGA
Original	96.29%	94.17%	98.33%	97.87%	97.87%	96.92%	95.04%	**99.25%**
FGSM [36](White)	63.49%	74.98%	64.90%	83.93%	83.93%	88.67%	**89.03%**	**89.03%**
PGD [40](White)	45.94%	56.21%	72.14%	87.30%	87.30%	92.52%	89.03%	**93.33%**
CW [41](White)	88.97%	93.83%	**97.95%**	97.60%	97.60%	90.22%	94.91%	**97.95%**
Spsa [42](White)	96.10%	92.37%	93.94%	**97.04%**	**97.04%**	95.04%	92.42%	**97.04%**
Sparse [43](White)	25.90%	53.22%	51.17%	66.06%	66.06%	73.12%	**78.32%**	66.59%
Noise [44](Black)	78.91%	78.91%	58.56%	77.81%	77.81%	69.78%	**87.30%**	78.91%

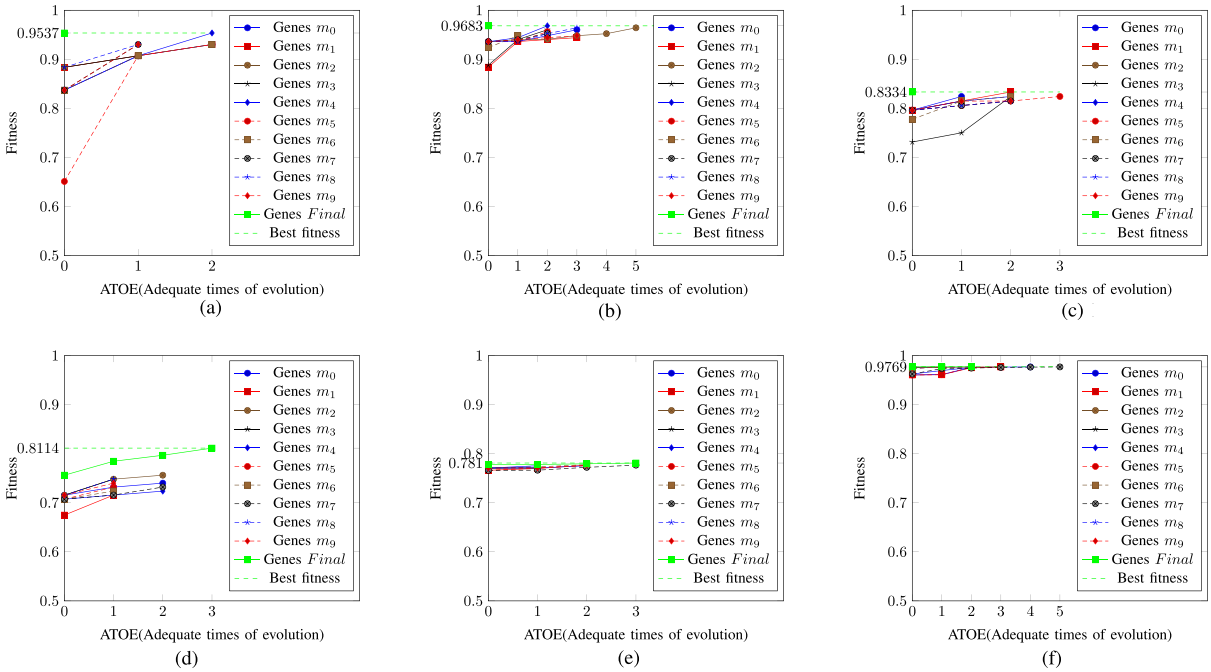


Fig. 9. Experiments on KEEL. (a) Appendicitis. (b) Balance. (c) Heart. (d) Haberman. (e) Tianic. (f) Twonorm.

Fig. 10. Adversarial examples on MNIST.

In contrast, training samples remain unchanged and then trained with the above models (a different CNN model from the one being attacked) to ensure the fairness of experiments.

The hyperparameters of the CNN set above are: two convolutional layers, two pooling layers, one full connection layer, and two residual blocks [3 kernels (3*3)] added to the RN model. Moreover, the implementation code of the attack algorithm mentioned above can be found in CleverHans [45], which has been updated to version 4.0 on GitHub.

C. Results and Analysis

1) Convergence and Performance Analysis of the Model: According to the results in Fig. 9, it is evident that after the evolution of the model, the recognition performance has been significantly improved compared to the original single kernel function and shallow multiple kernel learning networks. In particular, it can be seen from the results in Fig. 9(d) that the evolution of the topology structure of the model by the second genetic algorithm has a massive impact on the model recognition performance. It can be seen from Table II that, compared with MKL, DMKL, and the neural network model, which are currently popular in the industry, KNGA shows excellent performance in such small sample sets as UCI. The main reason is that compared with most other kernel learning models, as long as KNGA's training lasts sufficiently long, it can always have the same or even exceed the performance of kernel learning models through self-optimization of model parameters and topology structure. As for the neural network, the UCI sample size is not large enough to give full play to its maximum potential, so the neural network's performance on a small sample set is worse than that of the MKL. At the same time, we can see that in the Monk dataset in Table II, the performance of the SA-DMKL model exceeds KNGA. The main reason for this experimental result may be caused by the insufficient training time of our model or the local optimum. Theoretically, as long as we train it for a sufficiently long time, we can always get a model that is as good as or even better than the model generated by SA-DMKL. It can be seen from Table VI that in the MNIST dataset with less complex images, KNGA can also show as good performance as the current well-trained CNN model.

2) Influence of Different Parameter Settings of Kernel Functions: It can be seen from Tables III and IV that when the parameter setting of the kernel function is relatively small, it does have a significant impact on the performance of the model. In contrast, the model is tough to evolve when the order of magnitude of parameter setting is large. This phenomenon maybe because of the kernel function's performance degradation by parameters setting is too large. When the performance of each network unit is reduced, the performance of the whole network is also degraded.

3) Influence of the Size of the Dataset: From the comparison between Fig. 9(a)–(d) and (e)–(f), it can be seen that the optimization ability of the model on the dataset with a small order of magnitude is more significant than that on the dataset with a large order of magnitude. The model can improve the recognition ability by 10–12 percentage points in the training process on a small dataset. The model can only improve the recognition ability by 2–4 percentage points on a large dataset. This result does not exclude the fact that the dataset itself is

Genes	Sequence
0	[[[1], [0], [1], [1]], [[1], [1], [0], [0]], [[0], [0], [1], [0]], [[0], [1], [1], [0]]]
1	[[[1], [1], [0], [0]], [[0], [1], [0], [1]], [[1], [0], [0], [0]], [[0], [1], [0], [0]]]
2	[[[1], [1], [1], [0]], [[1], [0], [1], [0]], [[1], [1], [0], [1]], [[0], [1], [0], [0]]]
3	[[[1], [0], [1], [0]], [[1], [1], [1], [1]], [[0], [1], [1], [0]], [[0], [0], [1], [0]]]
4	[[[1], [1], [1], [1]], [[0], [1], [0], [0]], [[1], [0], [0], [1]], [[0], [1], [1], [0]]]
5	[[[0], [0], [1], [1]], [[0], [0], [0], [0]], [[0], [0], [0], [0]], [[1], [1], [1], [0]]]
6	[[[1], [1], [1], [1]], [[1], [1], [1], [0]], [[0], [0], [0], [0]], [[0], [0], [0], [1]]]
7	[[[0], [1], [0], [0]], [[1], [1], [0], [0]], [[0], [1], [0], [0]], [[1], [1], [1], [1]]]
8	[[[1], [1], [0], [0]], [[0], [0], [0], [1]], [[1], [1], [1], [0]], [[0], [0], [0], [1]]]
9	[[[1], [0], [0], [0]], [[1], [1], [0], [0]], [[0], [0], [1], [1]], [[0], [1], [0], [0]]]

(a) First population's Gene Sequenc of the primary network structure

Genes	Sequence
0	[[[1], [1], [1], [1]], [[0], [0], [0], [1]], [[0], [0], [1], [0]], [[1], [1], [0], [0]]]
1	[[[1], [0], [1], [0]], [[1], [0], [0], [1]], [[1], [1], [1], [0]], [[0], [1], [0], [0]]]
2	[[[0], [1], [0], [0]], [[0], [1], [0], [1]], [[1], [0], [0], [0]], [[1], [0], [1], [1]]]
3	[[[1], [0], [1], [0]], [[1], [0], [1], [0]], [[1], [0], [1], [0]], [[0], [0], [1], [0]]]
4	[[[1], [0], [0], [1]], [[0], [1], [1], [0]], [[0], [1], [1], [1]], [[0], [0], [0], [0]]]
5	[[[1], [1], [0], [1]], [[1], [0], [0], [1]], [[0], [0], [0], [1]], [[0], [0], [1], [1]]]
6	[[[0], [1], [1], [1]], [[0], [0], [1], [1]], [[0], [1], [0], [0]], [[0], [0], [0], [1]]]
7	[[[1], [0], [1], [0]], [[1], [1], [1], [1]], [[0], [0], [1], [1]], [[0], [1], [0], [1]]]
8	[[[0], [0], [1], [0]], [[1], [1], [1], [0]], [[1], [1], [0], [0]], [[1], [0], [0], [1]]]
9	[[[0], [0], [0], [1]], [[1], [0], [1], [0]], [[1], [1], [1], [1]], [[1], [1], [0], [0]]]

(b) Second population's Gene Sequence of the primary network structure

Fig. 11. Two initial populations' Gene Sequence of the primary network structure tested on the breast cancer dataset.

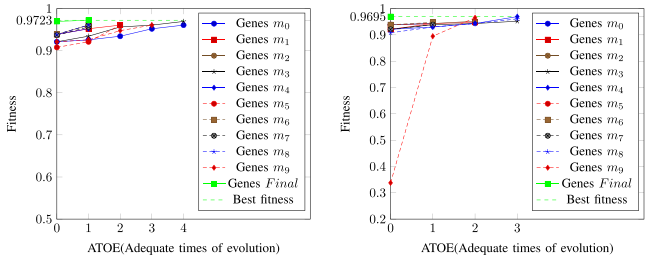


Fig. 12. Two model's fitness change figures tested on the breast cancer dataset. (a) First model's fitness change figure. (b) Second model's fitness change figure.

indivisible. Given this situation, the optimization ability of the model can be improved to a large extent only by adding several essential kernel functions.

4) Influence of Initial Population on Experimental Results: To understand this problem, a comparative experiment was set up in this article. Two experiments were conducted on the breast cancer dataset, each of which generates an initial group of populations and contains ten different topological structures. Then, the models were trained separately. The whole process is shown in Figs. 11 and 12.

From the above comparative experiments, we can see two points: 1) the difference in the initial population has a specific influence on the final recognition rate of the model, but it is not very significant. This difference can be eliminated by increasing the number of the initial population. 2) No matter whether every essential kernel function is activated in the first layer of the model, the final recognition rate of the model will slowly cluster in a small interval. Finally, the small interval can be cluster to one point by relaxing the termination conditions and variation range of the model evolution.

5) Ability of the Model to Recognize Different Types of Datasets: All the experiments from the above show that the recognition ability of KNGA in various fields is superior compared with the current neural networks and new DMKL models. However, the recognition rate of various models for these datasets has been relatively high. It seems that the recognition of image data is not as effective as it is on other types of datasets. The first reason for this phenomenon is that images, such as Caltech 256 have extensive sample feature information, which puts a significant burden on computing the inner product distance between samples by the kernel function and leads to the nonrepresentative features extracted by calculation. However, in the case of some single-channel image datasets with relatively few samples, the KNGA model can achieve excellent results, as shown in Table VI. Generally, recognizing complex image datasets using kernel functions requires some image processing and feature extraction methods to achieve outstanding results. However, in the experiment of Caltech 256, we expect to know more about the effect of performance improvement of KNGA on traditional DMKL in pure image data. Second, the high performance of the KNGA model is mainly based on the ability of each kernel function itself, and several kernel functions adopted in this experiment may be not good for the corresponding dataset, resulting in such results. Finally, these experiments conducted are intended to prove that the recognition and generalization ability of the model is as good as those AI models in their respective fields, with almost no need for manually adjusting parameters.

6) Trustworthiness Analysis of the Model: First of all, the following points can be seen from the experimental results: 1) From Table VI, we can see that the current model attack algorithm has a powerful influence on the AI model. Moreover, the sample's formation shown in Fig. 10 has not changed its original core characteristics. 2) The black-box attack has a similar impact on the neural network and kernel learning model compared with a white-box attack. In contrast, the white-box attack has a much more significant impact on neural networks than the KNGA model and some individual single kernel learning models. 3) The cutting-edge defense algorithm based on the neural network has a specific defense ability for the current attack. Meanwhile, it has a better effect than KNGA in the face of an attack that seriously damages the image's overall structure (sparse and noise).

The above three phenomena illustrate that most current white-box attack algorithms change some subtle features along the opposite direction of the neural network's optimization gradient to cause model lying, so the effect of attacks on the neural network is significant. However, the negative effect of the same attack algorithm on the kernel learning model is mainly due to the different classification mechanisms between the neural network model and the kernel learning model. The neural network is mainly based on taking good features and expanding their influence to obtain good classification results. The kernel learning calculates the inner product distance between samples and extracts the most representative samples from the training set to form the classification hyperplane. With only a tiny change of some eigenvalues of the samples, the inner product distance

between the samples will not have a significant deviation, leading to the classification failure of the kernel learning model. For example, a neural network is through a person's specific characteristics, such as eyes, nose, and so on, to distinguish the difference between people. However, kernel learning is to judge the differences between people by comparing the similarity between persons, which is the main reason why most white-box attacks are not adequate for the kernel learning model. The sparse and noise algorithms still have a considerable influence on KNGA. The reason is mainly due to its kernel learning algorithm because its image recognition performance is not good without any image preprocessing. At the same time, the above two algorithms generate tremendous noise for samples, which is not available in the training set. This is one of the main reasons for low performance of kernel learning and CNN, and can be solved by adding noise samples in the training set, but this has nothing to do with the model's trustworthiness.

If we suppose there is an attack algorithm, there must be a defense method for the model. As summarized in [37], existing defense methods can be divided into three main directions: 1) modifying training samples or the input form of test samples. 2) Modifying the structure of the networks, such as adding more layers/sub networks, changing loss/activation functions, etc. 3) Using external models as network add-on when classifying unseen examples. For example, Wang et al. [38] studied the binary threshold value of the input image preprocessing to defend against attack, PGD. The authors [46] modified the dynamic change model parameters and input data to increase the model's toughness. Zhang et al. [39] put forward a network of image reconstruction to reconstruct an example of the input to defend against attacks.

It can be seen from the results shown in Table VI that the defense model designed in wBT [38] and RN [39] has a good effect in the face of multiple attacks, especially the defense ability in the face of brutal attacks is even better than KNGA. The reason is that these two methods carry out image optimization and image reconstruction for the attacked samples, thus improving the recognition ability of CNN. However, while these methods improve the model's defense, they significantly increase the network structure's complexity and computation and are vulnerable to design-specific attacks. In particular, although the model proposed in [39] improves the defense ability of the model, it also dramatically increases the complexity of network structure, which is more challenging to implement and also becomes very difficult to tune the model. In addition, it can be seen from Table VI that the generalization performance of the image correction method by binarization proposed in [38] drops. In the face of a large number of discrete noises generated by the noise [44], the recognition effect is not good.

Thus, these defenses do not prove the trustworthiness of neural networks, but only that they are techniques to enhance some aspects of their performance. From the perspective of trustworthiness, these methods do not fundamentally solve the credibility problem of the neural network, and they are more like a remedial way to make up for the temporary deficiency of neural networks in the arms race with attack algorithms. In contrast, this article shows kernel learning has higher interpretability and related security mechanisms, as proved by the experiments and shown in Fig. 8 and Table VI. Furthermore, this article argues that kernel learning is more reliable than neural networks.

VIII. Conclusion

In this article, a new multilayer multiple kernel learning method named KNGA was proposed. It adopted GA to find the best DMKL structures, including the weights and the model's topology. Although the optimization algorithm of the model cannot guarantee global convergence, the experiments and the actual application showed that the optimization algorithm has robust global searchability. Through continuous self-optimization through training, KNGA has shown excellent recognition ability and generalization ability on small-scale datasets. The accuracy rate can be increased by 10% to 20%. In future work, we plan to continue the investigation from two aspects: 1) apply the KNGA model in more complex industrial scenes, including model training in cloud-edge integration environment and more complex image recognition. 2) More techniques related to model parameter initialization will be applied in this model, making the model more elastic and less likely to fall into the optimal local situation.

Appendix

Proof of Lemma 1: It is known that the mutation, crossover, and selection operations of the genetic algorithm are all independent of each other, and the selection process of the three random individuals in the mutation operation is also independent of each other. Thus, in the current genetic algorithm, the state of individual I transforms from I_t to I_{t+1}, and the probability of $I_t \to I_{t+1}$ is $P_{(I_t \to I_{t+1})} = P_{(\omega_1 \to v_1)} \cdot P_{((\omega_1,\omega_2)\to v_2)} \cdot P_{\text{select}}$, where I_t represents the state of individual I at time t, $P_{(\omega_1 \to v_1)}$ represents the probability of mutation, and $P_{((\omega_1,\omega_2)\to v_2)}$ represents the probability of crossover. Thus Lemma 1 is proved. ■

Proof of Lemma 2: $S_i \to S_j$ represents the population state that all the individual states in S_i simultaneously transfer to the corresponding individual states in the population state in S_j, as $I_{xi} \to I_{xj}, x = 1, 2, 3, \ldots, k$, and all individual state transition is mutually independent events, so the probability of population state S_i and one-step jump to state S_j is $P_{(S_i \to S_j)} = \prod_{x=1}^{k} P_{(I_{xi} \to I_{xj})}$. Thus, Lemma 2 is proved. ■

References

[1] L. D. Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Inform.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.
[2] Y. Wu, H.-N. Dai, and H. Wang, "Convergence of blockchain and edge computing for secure and scalable IIoT critical infrastructures in industry 4.0," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2300–2317, Feb. 2021.
[3] J. Sengupta, S. Ruj, and S. Dasbit, "A comprehensive survey on attacks, security issues and blockchain solutions for IoT and IIoT," *J. Netw. Comput. Appl.*, vol. 149, 2020, Art. no. 102481.
[4] Y. Wu, H.-N. Dai, and H. Tang, "Graph neural networks for anomaly detection in industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9214–9231, Jun. 2022.
[5] J. L. Elman, "Finding Structure in Time," *Cogn. Sci. J.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.
[6] N. Ketkar, "Convolutional neural networks," in *Deep Learning With Python*. Berlin, Germany: Springer, 2017.

[7] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Netw.*, vol. 12, no. 2, pp. 181–201, Mar. 2001.
[8] Y. Lei and L. Ding, "Refined Rademacher chaos complexity bounds with applications to the multikernel learning problem," *Neural Comput.*, vol. 26, no. 4, pp. 739–760, 2014.
[9] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, 2015, Art. no. 436.
[10] I. Rebai, Y. Benayed, and W. Mahdi, "Deep multilayer multiple kernel learning," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 1–10, 2015.
[11] E. V. Strobl and S. Visweswaran, "Deep multiple kernel learning," in *Proc. 12th Int. Conf. Mach. Learn. Appl.*, 2013, pp. 414–417, doi: 10.1109/ICMLA.2013.84.
[12] S. Ren, W. Shen, C. N. Siddique, and Y. Li, "Self-adaptive deep multiple kernel learning based on Rademacher complexity," *Symmetry*, vol. 11, no. 3, 2019, Art. no. 325.
[13] S. Ren, F. Liu, W. Zhou, X. Feng, and C. N. Siddique, "Group-based local adaptive deep multiple kernel learning with lp norm," *PLoS One*, vol. 15, 2020, Art. no. e0238535.
[14] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, "Evolutionary multiobjective optimization driven by generative adversarial networks (GANs)," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3129–3142, Jun. 2021.
[15] J. Liang, E. Meyerson, B. Hodjat, D. Fink, K. Mutch, and R. Miikkulainen, "Evolutionary neural AutoML for deep learning," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 401–409.
[16] B. Schölkopf, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2003.
[17] S. O. Cheng, A. J. Smola, and R. C. Williamson, "Learning the kernel with hyperkernels," *J. Mach. Learn. Res.*, vol. 6, pp. 1043–1071, 2005.
[18] N. Cristianini, "Kernel methods for pattern analysis," in *Proc. 15th IEEE Int. Conf. Tools Artif. Intell.*, 2003, p. 21.
[19] L. Ding et al., "Approximate kernel selection via matrix approximation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4881–4891, Nov. 2020.
[20] M. Schuld, "Supervised quantum machine learning models are kernel methods," 2021, doi: 10.48550/ARXIV.2101.11020.
[21] S. Bubeck, R. Eldan, and T. L. Yin, "Kernel-based methods for bandit convex optimization," *J. ACM*, vol. 68, 2021, Art. no. 25.
[22] T. Wang, L. Zhang, and W. Hu, "Bridging deep and multiple kernel learning: A review," *Inf. Fusion*, vol. 67, pp. 3–13, 2021.
[23] Y. Zhang et al., "Multi-kernel extreme learning machine for EEG classification in brain-computer interfaces," *Expert Syst. Appl.*, vol. 96, pp. 302–310, 2018.
[24] Y. Zou, H. Wu, X. Guo, L. Peng, and F. Guo, "MK-FSVM-SVDD: A multiple kernel-based fuzzy SVM model for predicting DNA-binding proteins via support vector data description," *Curr. Bioinf.*, vol. 16, pp. 274–283, 2021.
[25] A. Singh, A. Dhillon, and J. K. Thind, "SAMKL: Sample adaptive multiple kernel learning framework for lung cancer prediction," in *Proc. Int. Conf. Inf. Commun. Technol. Intell. Syst.*, 2020, pp. 31–44.
[26] X. Guo, W. Zhou, B. Shi, X. Wang, and F. Guo, "An efficient multiple kernel support vector regression model for assessing dry weight of hemodialysis patients," *Curr. Bioinf.*, vol. 15, no. 2, pp. 284–293, 2021.
[27] H. Shi, W. Wang, P. Wu, and D. Wang, "Support vector machine based on localized multiple kernel learning in pre-microRNA classification," in *Proc. Int. Conf. Elect. Commun. Comput. Eng.*, 2020, pp. 1–5.
[28] C. F. Sun, J. Y. Zhao, and J. H. Chen, "Analysis of differential evolution's Markov chain model and convergence," *Comput. Technol. Develop.*, vol. 23, no. 8, pp. 60–65, 2013.
[29] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020.
[30] Y. Sun, X. Sun, Y. Fang, G. Yen, and Y. Liu, "A novel training protocol for performance predictors of evolutionary neural architecture search algorithms," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 524–536, Jun. 2021.
[31] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, and C. L. P. Chen, "BNAS: Efficient neural architecture search using broad scalable architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 5004–5018, Sep. 2022.
[32] K. Bache and M. Lichman, "UCI machine learning repository," 2013.
[33] A. Niculescu-Mizil, A. Kumar, and K. Kavukcuoglu, "Two-stage multiple kernel learning method," U.S. Patent 8 838 508, 2014.
[34] X. Xu, I. W. Tsang, and D. Xu, "Soft margin multiple kernel learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 5, pp. 749–761, May 2013.
[35] J. Alcala-Fdez et al., "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Log. Soft Comput.*, vol. 17, no. 2/3, pp. 255–287, 2011.
[36] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, doi: 10.48550/ARXIV.1412.6572.
[37] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.
[38] Y. Wang, W. Zhang, T. Shen, H. Yu, and F.-Y. Wang, "Binary thresholding defense against adversarial attacks," *Neurocomputing*, vol. 445, pp. 61–71, 2021.
[39] S. Zhang, H. Gao, and Q. Rao, "Defense against adversarial attacks by reconstructing images," *IEEE Trans. Image Process.*, vol. 30, pp. 6117–6129, 2021.
[40] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, doi: 10.48550/ARXIV.1706.06083.
[41] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 39–57.
[42] J. Uesato, B. O'Donoghue, A. V. D. Oord, and P. Kohli, "Adversarial risk and the dangers of evaluating against weak attacks," 2018, doi: 10.48550/ARXIV.1802.05666.
[43] F. Tramer and D. Boneh, "Adversarial training and robustness for multiple perturbations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5858–5868.
[44] I. Goodfellow, Y. Qin, and D. Berthelot, "Evaluation methodology for attacks against confidence thresholding models," 2019. [Online]. Available: https://openreview.net/forum?id=H1g0piA9tQ
[45] N. Papernot et al., "Technical report on the cleverhans v2.1.0 adversarial examples library," 2016, doi: 10.48550/ARXIV.1610.00768.
[46] D. Wang, A. Ju, E. Shelhamer, D. Wagner, and T. Darrell, "Fighting gradients with gradients: Dynamic defenses against adversarial attacks," 2021, doi: 10.48550/ARXIV.2105.08714.

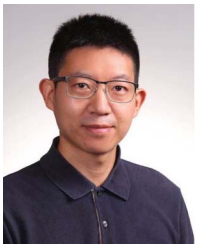
Wangbo Shen received the B.S. degrees from Changsha University, Changsha, China, in 2012 and 2016, and the M.S. degrees from Central South University, Changsha, China in 2016 and 2019, respectively. He is currently working toward the Ph.D. degree in computer science with the Department of Computer application, from the South China University of Technology, Guangzhou, China, supervised by Dr. Weiwei Lin.

His research interests include kernel learning, AutoML, and edge computing.

Weiwei Lin (Member, IEEE) received the B.S. and M.S. degrees in computer science from Nanchang University, Nanchang, China, in 2001 and 2004, respectively, and the Ph.D. degree in computer application from the South China University of Technology, Guangzhou, China, in 2007.

He has been serving as a Visiting Scholar with Clemson University, Clemson, SC, USA, from 2016 to 2017. He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology. He has authored or coauthored more than 150 papers in refereed journals and conference proceedings. His research interests include distributed systems, cloud computing, Big Data computing, and AI application technologies.

Dr. Lin is a Senior Member of China Computer Federation (CCF). He has been the Reviewer for many international journals, including IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON CLOUD COMPUTING, etc.



Yulei Wu (Senior Member, IEEE) received the B.Sc. degree (First Class Honours) in computer science and the Ph.D. degree in computing and mathematics from the University of Bradford, Bradford, U.K., in 2006 and 2010, respectively.

He is a Senior Lecturer with the Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter, U.K. His research interests include networking, Internet of Things, edge intelligence, AI and ethics, as well as privacy and trust.

Dr. Wu is a Senior Member of the ACM, and a Fellow of the Higher Education Academy. He is an Associate Editor for IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT and IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, as well as an Editorial Board Member of Computer Networks and Future Generation Computer Systems.

Fang Shi is currently working toward the Ph.D. degree in computer science with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China.

Her research interests include cache storage, distributed systems, physical-layer secure communications, and wireless cooperative communications.

Wentai Wu (Member, IEEE) received the bachelor's and master's degrees in computer science from the South China University of Technology, Guangzhou, China, in 2015 and 2018, respectively, and the Ph.D. degree in computer science from the University of Warwick, Coventry, U.K., in 2022, sponsored by CSC.

His research interests include distributed systems, federated learning, machine learning, and sustainable computing.

Keqin Li (Fellow, IEEE) received the Ph.D. degree from the University of Houston, Houston, TX, USA, in 1990.

He is a SUNY Distinguished Professor of Computer Science with the State University of New York, Albany, NY, USA. He is also a National Distinguished Professor with Hunan University, Changsha, China. His research interests include fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, Big Data computing, high performance computing, computer architectures and systems, computer networking, ML, and intelligent and soft computing.

Dr. Li is an AAIA Fellow. He is also a Member of Academia Europaea (Academician of the Academy of Europe). He is currently an Associate Editor of the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He has served on the editorial boards of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, and IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.