Intelligent Automation & Soft Computing

Tech Science Press

DOI: 10.32604/iasc.2023.040502
Article

A Novel Attack on Complex APUFs Using the Evolutionary Deep Convolutional Neural Network

Ali Ahmadi Shahrakht[1], **Parisa Hajirahimi**[2], **Omid Rostami**[3] **and Diego Martín**[4,*]

[1]Department of Industrial Engineering, Sharif University of Technology, Tehran, 14588-89694, Iran
[2]Department of Business Administration, Boston University, Boston, MA, 02215, USA
[3]Department of Industrial Engineering, University of Houston, Houston, TX, 77204, USA
[4]ETSI de Telecomunicación, Universidad Politécnica de Madrid, Madrid, 28040, Spain
*Corresponding Author: Diego Martín. Email: diego.martin.de.andres@upm.es
Received: 21 March 2023; Accepted: 12 June 2023; Published: 11 September 2023

Abstract: As the internet of things (IoT) continues to expand rapidly, the significance of its security concerns has grown in recent years. To address these concerns, physical unclonable functions (PUFs) have emerged as valuable tools for enhancing IoT security. PUFs leverage the inherent randomness found in the embedded hardware of IoT devices. However, it has been shown that some PUFs can be modeled by attackers using machine-learning-based approaches. In this paper, a new deep learning (DL)-based modeling attack is introduced to break the resistance of complex XAPUFs. Because training DL models is a problem that falls under the category of NP-hard problems, there has been a significant increase in the use of meta-heuristics (MH) to optimize DL parameters. Nevertheless, it is widely recognized that finding the right balance between exploration and exploitation when dealing with complex problems can pose a significant challenge. To address these challenges, a novel migration-based multi-parent genetic algorithm (MBMPGA) is developed to train the deep convolutional neural network (DCNN) in order to achieve a higher rate of accuracy and convergence speed while decreasing the run-time of the attack. In the proposed MBMPGA, a non-linear migration model of the biogeography-based optimization (BBO) is utilized to enhance the exploitation ability of GA. A new multi-parent crossover is then introduced to enhance the exploration ability of GA. The behavior of the proposed MBMPGA is examined on two real-world optimization problems. In benchmark problems, MBMPGA outperforms other MH algorithms in convergence rate. The proposed model are also compared with previous attacking models on several simulated challenge-response pairs (CRPs). The simulation results on the XAPUF datasets show that the introduced attack in this paper obtains more than 99% modeling accuracy even on 8-XAPUF. In addition, the proposed MBMPGA-DCNN outperforms the state-of-the-art modeling attacks in a reduced timeframe and with a smaller number of required sets of CRPs. The area under the curve (AUC) of MBMPGA-DCNN outperforms other architectures. MBMPGA-DCNN achieved sensitivities, specificities, and accuracies of 99.12%, 95.14%, and 98.21%, respectively, in the test datasets, establishing it as the most successful method.

Keywords: IoT security; PUFs; modeling attacks; evolutionary deep learning; migration-based multi-parent genetic algorithm

1 Introduction

The Internet of Things (IoT) is a wide network of different nodes that are connected using the Internet. Nowadays, the existence of many telecommunication networks with limited computing, power, and communication capacity has made the necessity of IoT inevitable in many infrastructures [1–3]. Security is one of the major concerns that needs to be considered for IoT to be practically available in the near future, as it is with all communication networks, especially given the unique features of IoT. Many efforts have been made so far to address security issues in the IoT, though providing a suitable security solution considering the limitations of the IoT has been an open problem [4–7]. Among all proposed security solutions for the IoT, physically unclonable functions (PUFs) [8] are among the most popular primitives because they provide lightweight authentication as well as security against physical attacks, which are among the most important attacks in IoT networks [9–14].

PUFs work as challenge-response black boxes and can be utilized to extract unique random bits by using the unique manufacturing-related physical properties of an electronic device. In this way, many challenge-response pairs (CRPs) can be generated as cryptographic keys for device authentication purposes in IoT networks using the embedded PUF in IoT devices [15–17]. These responses are typically extremely difficult for an adversary to replicate or predict. Several authentication schemes have been introduced to eliminate the practical restrictions where a server with high computational capacity authenticates a low-power device [18].

The Arbiter PUF (APUF) was proposed and implemented at the very beginning of PUFs as a low-power device-oriented secret key generator [19] with a simple structure and functionality and low implementation costs. However, it has been shown that APUFs are assailable to different modeling threats. For example, an adversary who can get access to a set of CRPs from a PUF instance attempts to build a mathematical framework using the gathered CRPs and predict a PUF response with a high probability. Among all modeling attack techniques, machine learning (ML) algorithms like support vector machine (SVM) and logistic regression (LR) have been the most favorite attacks that could have broken the resistance of APUF. More importantly, multiple-layer perceptron neural networks (MLPNN) have been recently employed to break the security of APUF [20–25]. On the other hand, XOR APUFs (XAPUFs) have been recently proposed to increase the security of APUFs against modeling attacks [26]. Although some efforts have shown that XAPUFs can be modeled by some ML-based approaches [27–31], breaking the security of the more complex XAPUF is still an open problem. Furthermore, the majority of the proposed XAPUF attacks place a high computational burden on the attacker's side. Therefore, another motivation for this paper is to propose more practical attacks for breaking the security of complex XAPUFs structures.

According to the literature [32–43], different ML algorithms have been proposed to break the security of XAPUFs, containing deep learning (DL), neural networks (NNs), and SVMs. Although the most effective approach is uncertain, since 2006, DL has gained popularity in the ML field. DL models have surpassed conventional ML models because of advancements in information availability.

The popularity of DL techniques has also gained due to improvements in decreasing execution time and enhancing the convergence rate. Therefore, this paper uses a deep convolutional neural network (DCNN) to bypass the security of XAPUFs. However, an effective DL training approach is known to be one of the major ML tasks. Gradient-based methods have significant limitations, such as becoming stuck at local extremums in multi-objective loss functions, considerable computational cost, and requiring continuous objective functions [44–46]. To address this challenge, researchers have delved into the application of MH algorithms for parameter optimization in DL. In this paper, a new migration-based multi-parent genetic algorithm (MBMPGA) has been developed to train DCNN. MH algorithms are able to greatly enhance the training of DL models by enhancing the learning process, leading to better accuracy and decreased computational complexity.

1.1 Paper Contributions

We intend to develop a deep architecture by proposing a novel MBMPGA to train the fully connected NN in the proposed deep structure for achieving higher attack accuracy. The major contributions of this paper can be summarized as follows:

- A new MH algorithm named MBMPGA is introduced to enhance the exploitation and exploration abilities of GA by using the non-linear migration operator in BBO and multi-parent crossover, respectively.
- The performance of a deep CNN as an attacking tool is improved by tuning the weights and biases in its NN using the proposed MBMPGA in order to reach a high rate of accuracy for modeling complex XAPUFs.
- The performance of the proposed MBMPGA-DCNN is compared against eight different models, namely GA-DCNN, particle swarm optimization (PSO)-DCNN, BBO-DCNN, improved crow search algorithm (I-CSA)-DCNN, black widow optimization (BWO)-DCNN, DCNN, MLPNN, and SVM.
- The performance of MBMPGA is evaluated using several real-word datasets, namely tension spring design, three-bar truss design, and complex XAPUFs problems. To compare the models, various metrics were utilized, including sensitivity, accuracy, specificity, mean square error (MSE), receiver operating characteristic (ROC) curve, convergence curve, execution time, best fitness function, mean fitness function, and standard deviation.
- Compared to previous attacking models, the simulation results of the proposed model on some simulated CRPs [33] show a significant improvement in computational cost and the number of CRPs needed for training the model.

1.2 Paper Organization

The remainder of this paper is formed as follows: Section 2 presents the related works. Section 3 presents the proposed MBMPGA algorithm. Section 4 elaborates on the DL model improved by MBMPGA. Section 5 first measures the performance of MBMPGA using different benchmark datasets, then compares the performance of the proposed DL architecture in modeling different complex XAPUFs, and finally draws a conclusion about this work in Section 6.

2 Related Works

In typical modeling attacks on PUFs, by using a CRP sub-related to a PUF instance, an adversary can drive a numerical PUF structure. Accordingly, many efforts have been made to study the resistance of XAPUFs against machine-learning-based attacks. These efforts have usually utilized various attack models. Ruhrmair et al. [27] employed LR to model 4-XAPUF, 5-XAPUF, and 6-XAPUF with 64-bit 128-bit challenges, and the attack models could predict the PUF responses with 99% accuracy. With the exception of 4-XAPUF, their attacks took some time to train their model. Tobisch et al. [30] suggested parallelizing the LR-based technique according to [27]. By having 2×10^7 and 1.5×10^8 CRPs, a better accuracy was achieved for 7-XAPUF and 8-XAPUF with 64-bit challenge bits. However, the suggested modeling attack needed a long-time training process.

For the first time, Hospodar et al. [32] used an MLP neural network to attack XAPUFs. They achieved approximately 90% accuracy when attacking the 2-XAPUF with a challenge-bit length of 64 bits. Santikellur et al. [28] introduced the earliest effective employment of DL in modeling different types of APUF. Their modeling results have shown 98% accuracy on 5-XAPUF and 6-XAPUF with 64-bit challenge-bit length. In [29], Aseeri et al. proposed an MLP NN for the 4to8-XOR-APUF for different challenge bits. By employing just 3×10^7 CRPs, they were able to predict the testing CRPs of the 8-XOR-APUF with 64-bit challenge-bit length with 99% accuracy.

Mursi et al. [33] have proposed MLP-based attacks on 5to9-XOR-APUFs. Their attack's structure followed the Aseeri techniques, though the number of used hidden layers in NN was 50% reduced. Shi et al. have proposed two novel modeling attacks, the logical approximation and the global approximation, which utilize an ANN for modeling the nonlinear nature of the XOR-APUFs [34]. In addition, an effective attack entitled CANDECOMP/PARAFAC-Tensor Regression Network (CP-TRN) has been introduced by Santikellur et al. [31] to reduce the computational complexity of the attack on XAPUFs. This is important to mention that CP-TRN is a type of CP-decomposition-based TRN. Cui et al. [35] proposed an ML-based attack on different types of strong PUFs. To achieve this, XORs in the XOR-based Multi-PUF were substituted with multiplexers [36]. According to the literature, different ML algorithms have been proposed to break the security of XAPUFs. However, these attacks have typically failed to model the more complex structure of XAPUF, necessitating a large set of CRPs as well as significant computational power.

Based on the review of the papers, it becomes evident that in order to achieve more precise modeling of complex XAPUFs problem, the utilization of more robust DL algorithms is necessary. The optimization of parameters in DLs is a challenging task owing to their NP-hard nature. To address this challenge, researchers have delved into the application of MH algorithms for parameter optimization in DL. Although MH-based methods have been effective in finding optimal solutions for complex problems, the use of MH for optimizing different DL structures is a relatively new research area. The first attempt to employ a GA to train a DCNN was proposed in [47]. This method used the advantages of various GA operators to train the DCNN by treating the parameters of the NN as chromosomes. Fan et al. proposed an approach using progressive unsupervised learning (PUL) to optimize the variables of a pre-trained DCNN [48]. Their scheme demonstrated efficient implementation and was employed as an optimized principle for unsupervised learning. Sun et al. presented a GA-trained DCNN in order to classify image features without requiring extra information about the used DCNN. However, the major drawback of the mentioned scheme was that the training algorithm becomes slow when the deep architecture is relatively large due to the large size of the GA's chromosomes [49]. In recent years, several other training algorithms have been introduced to enhance the performance of different DL models [44,50].

In the context of complex APUFs, several technical gaps have been identified, leading to the development of the proposed methodology utilizing MBMPGA for training DCNNs. The following technical gaps have motivated the design of this novel approach:

- Existing methodologies lack effective attack strategies specifically tailored for complex APUFs. This gap highlights the need for an innovative algorithm that can successfully exploit vulnerabilities in these systems.
- Optimizing the parameters of DL models is a difficult task. To tackle this challenge, researchers have explored the use of MHs for DL parameter optimization. Over the years, a plethora of MH algorithms have been proposed and shown promising results in solving various engineering problems. As problems become more intricate, the demand for new MH algorithms becomes more apparent than ever. There are five main motivations that drive this need a) flexibility; b) derivation-free mechanisms; c) simple and effective implementation; d) simple structure and concepts; e) local optima avoidance. Addressing the exploration and exploitation aspects of complex optimization problems is recognized as a challenging endeavor. To overcome these difficulties, in this paper, a new MBMPGA algorithm has been developed to train DCNN.
- Traditional genetic algorithms often rely on a single parent population, limiting the exploration and exploitation of diverse solutions. The proposed MBMPGA addresses this gap by incorporating multiple parent populations and migration strategies to enhance diversity and convergence speed.

3 Migration-Based Multi-Parent Genetic Algorithm (MBMPGA)

GA is a type of search algorithm that takes inspiration from Charles Darwin's theory of natural evolution. First presented by Holland in 1975 [37], the aim of GA is to produce offspring that are biologically superior to their parents. The algorithm works by selecting the most competent individuals from the population and allowing them to reproduce by producing offspring. During the reproduction process, the genes from both parents' crossover, leading to a genetic mutation. The offspring then reproduce, and the cycle continues, resulting in the production of healthier generations. GA consists of four distinct phases [38].

At the beginning of the genetic algorithm, a collection of solutions called the "initial population" is generated. Each solution is represented by a chromosome, consisting of a set of genes that define the variables of the problem. The next step involves selecting two pairs of chromosomes (parents) based on their fitness values using a selection method such as the roulette wheel. The most important step in the algorithm is the crossover operator, which involves randomly selecting a crossover point within the genes and exchanging them between the parents to create offspring. This operator is used to enhance the exploitation of GA by searching the space around a chromosome. The mutation is also introduced to enhance exploration by randomly modifying the genes of some newly formed offspring.

There are several drawbacks known for the original GA e.g., weaker exploitation ability and getting stuck in local extremums. However, if chromosomes suddenly and rapidly change their motion space to move in the desired direction, the GA algorithm converges faster. Fig. 1 indicates the single-point crossover operator of the normal GA. As it is observed, in this method, only two parents are mixed, thus the chromosomes generated (offspring) are not very distinct from the parents [39].

Another disadvantage of standard single-point crossover is that it only searches the space between two parents. If a better global search is done in GA, in addition to diversifying the answers, the convergence curve of the GA will also improve. In this article, a novel MBMPGA algorithm is presented as an innovative operator to enhance the exploitation and exploration of GA. In MBMPGA, the migration operator of the BBO is employed to enhance the exploitation of GA. Multi-parent crossover (as a new operator) is then introduced to develop the exploration ability of GA. Considering the comparison of the performances of different migration models, in this paper, the generalized sinusoidal model has been used. Emigration ($\mu_{k(j)}$) and immigration rates ($\lambda_{k(j)}$) are defined as Eqs. (1) and (2).

$$\mu_{k(j)} = \frac{E}{2} \times \left(-cos\ cos \left(\frac{k(j)\pi}{N} + \varphi\right) + 1\right) \tag{1}$$

$$\lambda_{k(j)} = \frac{I}{2} \times \left(cos\ cos \left(\frac{k(j)\pi}{N} + \varphi\right) + 1\right) \tag{2}$$

where $k(j)$ denotes the rank of species in the j^{th} habitat. E and I show the maximum rates of emigration and immigration, respectively. Figs. 2 and 3 show examples of multi-parent crossover and migration-based multi-parent crossover in the MBMPGA, respectively. As is observed, the value of RMSE is like the objective function, and the lower the RMSE, the better the approach. In MBMPGA, when several of the best parents are selected to generate a new offspring simultaneously, the resulting offspring bears less resemblance to one parent, i.e., the offspring is more diverse (improving exploration). However, since the offspring have obtained all of their genes from multiple chromosomes, the exploitation of the algorithm improves. Fig. 4 shows the flowchart of the MBMPGA. The pseudo-codes of the MBMPGA are also shown in Algorithm 1.

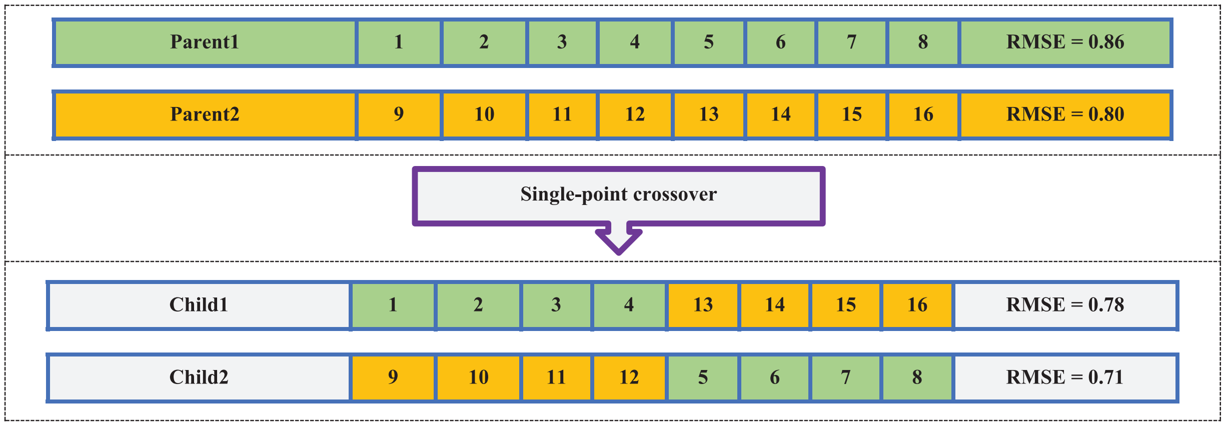


Figure 1: An example of a single-point crossover

4 Training DCNN Architecture

A DCNN is built upon four main layers: convolution, pooling, activation, and fully connected layers. In the convolution layer, a convolution filter is applied to the input data to extract features. This involves multiplying the input data with a set of values determined by the filter. The kernel then bypasses the image multiple times, with the weights being multiplied by the input data at each position. The kernel moves from top to bottom and from left to right to mask all the input data before an operation is applied. The results of this process are summed up to produce a distinctive value for each kernel position.

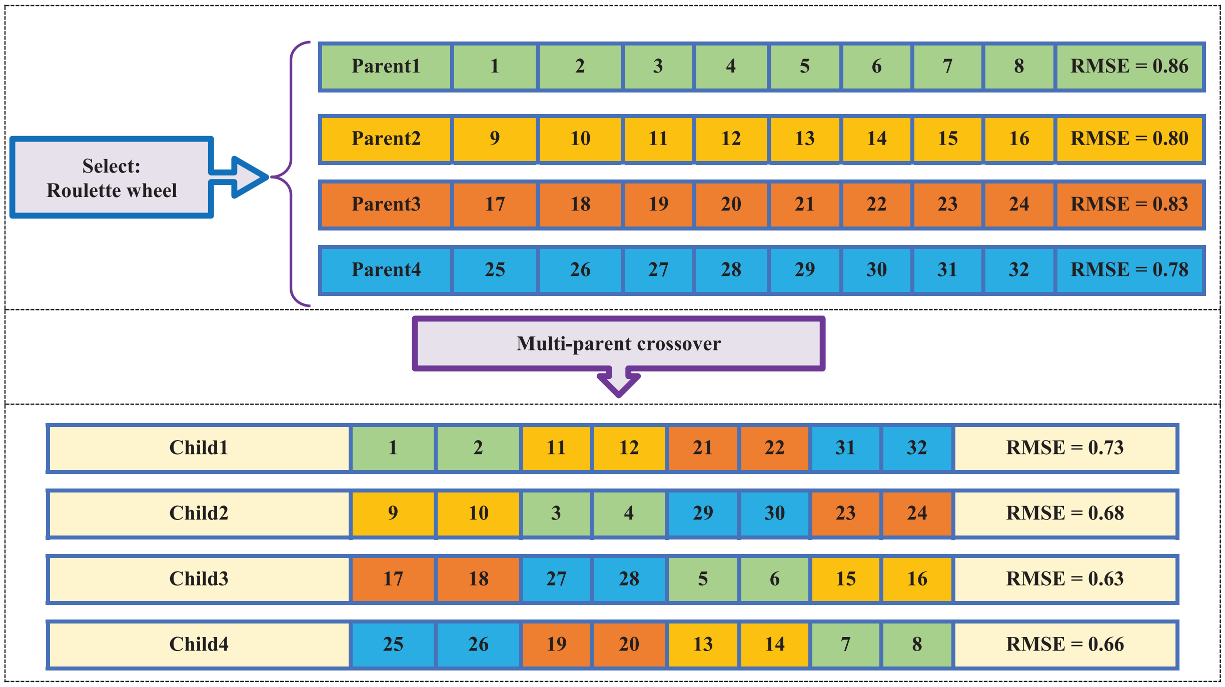

Select:
Roulette wheel
Parent1 1 2 3 4 5 6 7 8 RMSE = 0.86
Parent2 9 10 11 12 13 14 15 16 RMSE = 0.80
Parent3 17 18 19 20 21 22 23 24 RMSE = 0.83
Parent4 25 26 27 28 29 30 31 32 RMSE = 0.78
Multi-parent crossover
Child1 1 2 11 12 21 22 31 32 RMSE = 0.73
Child2 9 10 3 4 29 30 23 24 RMSE = 0.68
Child3 17 18 27 28 5 6 15 16 RMSE = 0.63
Child4 25 26 19 20 13 14 7 8 RMSE = 0.66

Figure 2: An example of multi-parent crossover

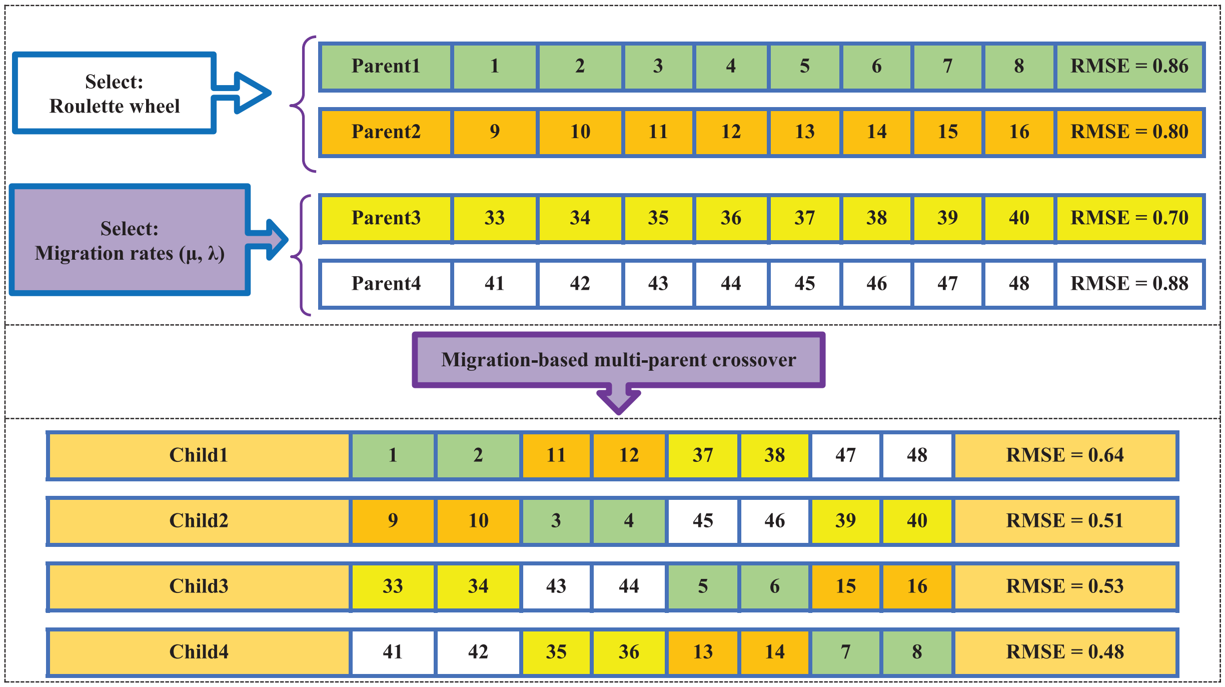

Select:
Roulette wheel
Parent1 1 2 3 4 5 6 7 8 RMSE = 0.86
Parent2 9 10 11 12 13 14 15 16 RMSE = 0.80
Select:
Migration rates (μ, λ)
Parent3 33 34 35 36 37 38 39 40 RMSE = 0.70
Parent4 41 42 43 44 45 46 47 48 RMSE = 0.88
Migration-based multi-parent crossover
Child1 1 2 11 12 37 38 47 48 RMSE = 0.64
Child2 9 10 3 4 45 46 39 40 RMSE = 0.51
Child3 33 34 43 44 5 6 15 16 RMSE = 0.53
Child4 41 42 35 36 13 14 7 8 RMSE = 0.48

Figure 3: An example of multi-parent crossover

In the subsequent step, the output of the convolution operation is passed through a nonlinear activation function known as the Rectified Linear Unit (ReLU). This function replaces all negative values with zeros, which helps to further process the data. The pooling layer then performs the task of reducing the input data size while retaining the most useful information. For instance, in a group of eight pixels, max pooling selects the most noteworthy ones. This layer plays a vital role in deep learning as it helps to reduce computational costs and minimize over fitting. Following the convolution and pooling stages, the fully connected layer comes into play, which mainly consists of an MLP NN.

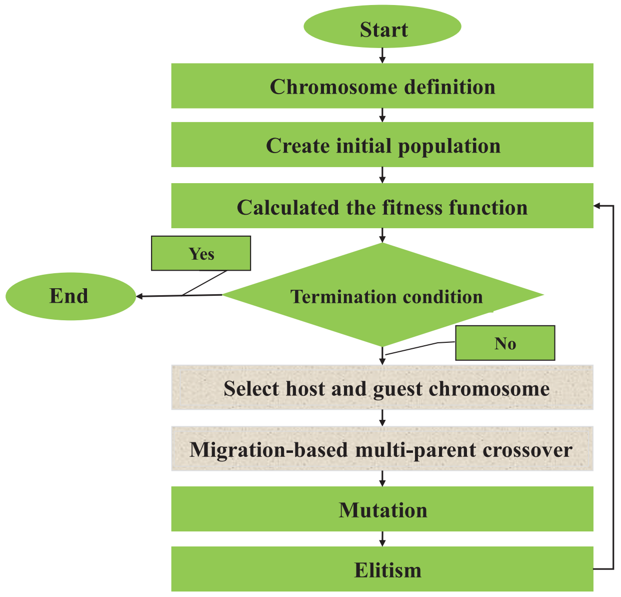

Start
Chromosome definition
Create initial population
Calculated the fitness function
Yes
End
Termination condition
No
Select host and guest chromosome
Migration-based multi-parent crossover
Mutation
Elitism

Figure 4: The flowchart of the proposed MBMPGA

Algorithm 1: Pseudo-codes of MBMPGA

```
1   %% Chromosomes initialization;
2   for i = 1 to N do
3       Create initial population;
4       Evaluate fitness function;
5       Set parameters;
6       Calculate the immigration and emigration rates of chromosomes;
7   end
8   %%Main loop;
9   While (termination criteria is not met) do
10      for i = 1 to N do
11          Select parents (Roulette wheel);
12          Select parents (migration rates (μ, λ));
13          Migration-based multi-parent crossover;
14          Mutation;
15          Elitism;
16      end
```

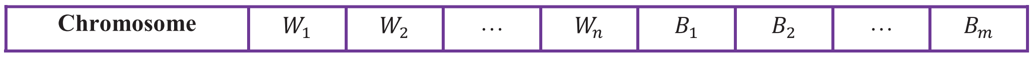
(Continued)

Algorithm 1 (continued)

```
17          Calculate the fitness and sort the population from the best to the worst;
18      end
```

In our experiment setup, the deep architecture has four 1D convolutional layers with increasing numbers of filters (32, 64, 128, and 256, respectively) with a kernel size of 5, followed by max-pooling layers with pool sizes of 2 to reduce the dimensionality of the feature maps. Each convolutional layer is followed by a dropout layer (0.25) to prevent overfitting. The output of the fourth convolutional layer is flattened and passed through two fully connected layers (with 512 and 256 units, respectively) each of which is also followed by a dropout layer (0.5). The output layer consists of a single Sigmoid unit representing the predicted output of the XOR Arbiter PUF. The model uses the Adam optimizer and binary cross-entropy loss function, and accuracy is used as the evaluation metric. We have used 10^6 CRPs of which 80% have been used for training and 20% have been used for testing.

In this paper, MBMPGA is used to train DCNN. In the suggested approach, MBMPGA tunes the weights and biases of the DCNN. When it comes to MBMPGA modeling, a crucial objective is to establish a chromosome-based solution. Fig. 5 demonstrates the structure of a chromosome in MBMPGA. The cost function of the algorithm can be shown in Eq. (3).



Chromosome	W_1	W_2	$\cdots$	W_n	B_1	B_2	$\cdots$	B_m

Figure 5: Chromosome definition in proposed MBMPGA-DCNN

$$Mean\ Square\ Error\ (MSE) = \frac{1}{k}\sum\nolimits_{i=1}^{k}(O_i - D_i)^2 \tag{3}$$

where, k shows the number of samples, O_i shows the output, and D_i shows the desired value. Many chromosomes are randomly produced to build an initial population. The roulette wheel method (Eq. (4)) and Migration rates (μ, λ) are also used to select parent chromosomes.

$$P_r = \frac{F(X_r)}{\sum\nolimits_{k=1}^{n} F(X_k)} \tag{4}$$

where, P_r is the probability of choosing chromosome r, $F\,(X_r)$is the cost function, and n is the total number of the initial population. The migration-based multi-parent crossover operator is the same as Fig. 3. Finally, Fig. 6 shows the mutation operator of MBMPGA.

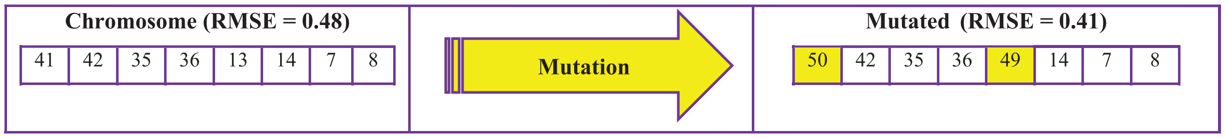


Figure 6: Example of the mutation operator in MBMPGA-DCNN

5 MBMPGA-DCNN Performance Evaluation

In this section, the performance of the proposed MBMPGA on two benchmark engineering problems is measured in compared to some MHs, including GA, PSO, BBO, I-CSA, and BWO. In the following subsection, the performance of the proposed MBMPGA-DCNN for modeling complex XAPUFs is evaluated. All algorithms were coded in MATLAB software and the calibration variables of the MHs are demonstrated in Table 1. In this paper trial and error method has been used. The objective function defines the main criterion for resetting the parameters.

Table 1: Calibration of algorithm parameters by trial and error method

Algorithm	Parameter	Value
MBMPGA	Elitism percent	8%
	Mutation rate	0.09
	Crossover rate	0.90
	The probability range for migrating into for each gene	[0,1]
	Maximum emigration (I) and immigration (E) coefficient	1
	Population size	120
	Iteration	300
I-CSA	Flight length (fl)	2
	Awareness probability (AP)	0.1
	Population size	120
	Iteration	300
BWO	Procreate rate (PP)	0.62
	Mutation rate (P_M)	0.23
	Cannibalism rate (CR)	0.46
	Population size	120
	Iteration	300
GA	Elitism percent	13%
	Mutation rate	0.11
	Crossover rate	0.94
	Population size	120
	Iteration	300
BBO	The probability range for migrating into for each gene	[0,1]
	Maximum emigration (I) and immigration (E) coefficient	1
	Elitism percent	10%
	Mutation rate	0.12
	Population size	120
	Iteration	300
PSO	The inertial movement rate (α)	0.15
	The movement toward the best personal experience rate (Φ1)	0.68
	The movement toward the best global experience rate (Φ2)	0.83
	Population size	120
	Iteration	300

5.1 Tension Spring Design

In this part, the performance of the proposed MBMPGA on a tension spring design is evaluated. Fig. 7 provides a schematic of the problem, and its formulation can be expressed through Eqs. (5)–(10) [46].

$$f(X) = (x_3 + 2)x_2x_1^2 \tag{5}$$

Subjected to

$$g_1(X) = 1 - \frac{x_3x_2^3}{71785x_1^4} \leq 0 \tag{6}$$

$$g_2(X) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0 \tag{7}$$

$$g_3(X) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0 \tag{8}$$

$$g_4(X) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \tag{9}$$

$$0.05 \leq x_1 \leq 2.00, 0.25 \leq x_2 \leq 1.30, 2 \leq x_3 \leq 15 \tag{10}$$

where, x_1 = Wire diameter (d), x_2 = Mean coil diameter (D), x_3 = number of active coils (N).

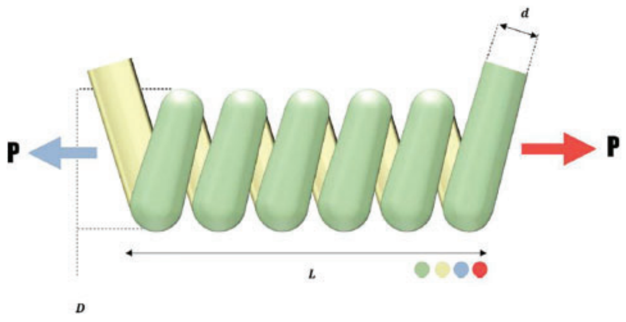


Figure 7: A schematic view of the design of the tension spring

Table 2 indicates the results of the algorithms on the tension spring problem. The table consists of four columns: Best of fitness, mean of fitness, standard deviation, and iteration. As is observed, MBMPGA achieved the best objective function value (Best fitness). The best solution (by MBMPGA) for this problem is 0.012666. MBMPGA also has the best standard deviation, as seen in the table. Fig. 8 shows the convergence curve of MBMPGA for this problem. The x-axis represents the iterations, while the y-axis shows the value of best fitness function. According to Fig. 8, MBMPGA has the best convergence rate. MBMPGA, in its 75th iteration, has achieved significant advancements and emerged with the most optimal solutions.

Table 2: The results of proposed algorithms for a spring system problem

Algorithm	Best fitness	Mean fitness	Standard deviation	Iteration
MBMPGA	1.26660E-2	1.29025E-2	0.0000356	300
I-CSA	1.26789E-2	1.38512E-2	0.0005296	300
BWO	1.26868E-2	1.45321E-2	0.0258563	300
BBO	1.26919E-2	1.48652E-2	0.7541239	300
PSO	1.27368E-2	1.89652E-2	1.9856321	300
GA	1.27396E-2	1.98632E-2	2.5232149	300

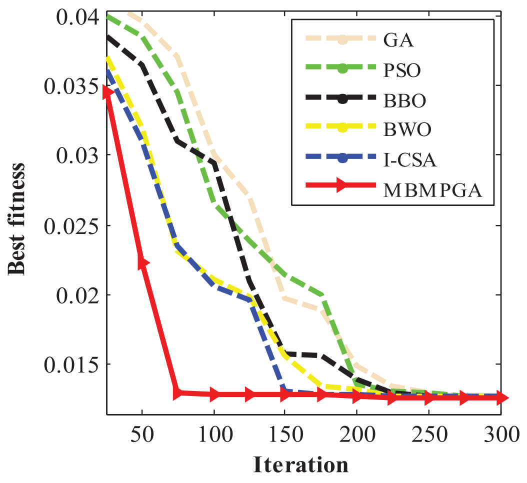


Figure 8: The convergence curve of algorithms for spring system design problem

5.2 Three-Bar Truss Design Problem

Here, we aim to obtain the minimum weight of the three-bar truss. Fig. 9 shows an overview of this problem and can be formulated as Eqs. (11)–(15).

$$f(X) = (2\sqrt{2x_1} + x_2) \times l \tag{11}$$

Subjected to

$$g_1(X) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0 \tag{12}$$

$$g_2(X) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0 \tag{13}$$

$$g_3(X) = \frac{1}{\sqrt{2}x_2 + x_1}P - \sigma \leq 0 \tag{14}$$

$$0 \leq x_1, x_2 \leq 1 \tag{15}$$

where, $l = 100\ cm$, $P = 2\ KN/cm^2$, $\sigma = 2\ KN/cm^2$

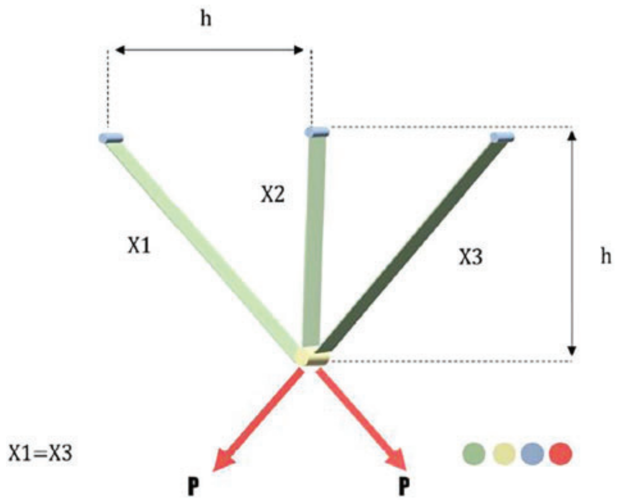


Figure 9: A schematic view of the truss design problem

The outcomes of various algorithms are displayed in Table 3. The table consists of four columns: Best of fitness, mean of fitness, standard deviation, and iteration. It is evident that MBMPGA has the best value for the cost function, resulting in a solution of 263.921453. Furthermore, MBMPGA has the best standard deviation, as seen in the table. Fig. 10 depicts the convergence rate for MBMPGA and other MHs. The x-axis represents the iterations, while the y-axis shows the value of best fitness function. MBMPGA's convergence rate is better than those of other MHs. MBMPGA, in its 100th iteration, has achieved significant advancements and emerged with the most optimal solutions.

Table 3: The results of algorithms for the three-bar truss design problem

Algorithm	Best fitness	Mean fitness	Standard deviation	Iteration
MBMPGA	263.921453	265.965214	0.0000483	300
I-CSA	263.974256	272.745268	0.0053269	300
BWO	264.195214	274.875632	0.0896521	300
BBO	264.865231	278.854123	1.0145893	300
PSO	265.562489	279.456328	2.7596321	300
GA	266.147023	280.214589	3.4789632	300

5.3 *Simulation Results for Modeling the Complex XAPUFs*

In this section, the performance of the MBMPGA-DCNN and other ML architectures for XAPUFs is evaluated. To do so, we exploited the generated CRPs from [33] to model complex XAPUFs. The CRPs were generated in an additive delay model (ADM)-based simulator [40]. The APUFs used in XAPUFs are 64-bit APUFs, and the corresponding challenges within each simulation were randomly determined from 0 to 264. Gaussian distribution with a standard deviation of $\sigma = 40$ and a mean of $\mu = 300$ has also been utilized to uniformly determine the MUX delays in APUFs. For this comparison, sensitivity, accuracy, and specificity analyses are utilized. These analyses are derived from the confusion matrix and are computable as Eqs. (16)–(18).

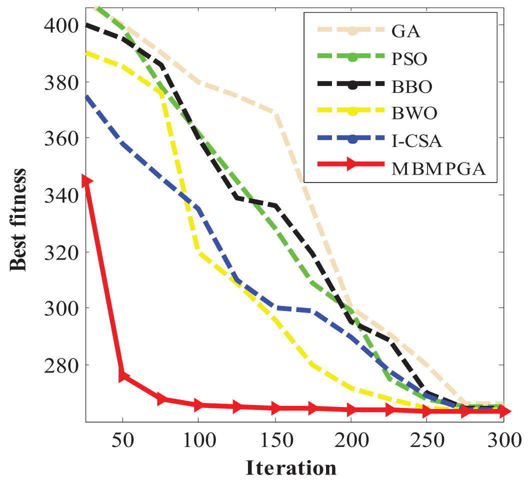


Figure 10: The convergence curve of algorithms for spring system design problem

$$Sensitivity = \frac{TP}{TP + FN} \tag{16}$$

$$Specificity = \frac{TN}{TN + FP} \tag{17}$$

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{18}$$

where, TP = True positive, TN = True negative, FN = False negative, FP = False positive.

Table 4 indicates the sensitivity, specificity, and accuracy of different evolutionary DL structures for complex XAPUF. As is observed, the MBMPGA-DCNN architecture shows the best performance in sensitivity, specificity, and accuracy for training and validation datasets. SVM also shows the worst performance in training and validation datasets. MBMPGA-DCNN reached 99.41% and 98.86% accuracy in the test and training datasets, respectively. Figs. 11 and 12 indicate the comparison of proposed structures in training and validation datasets, respectively (According to Table 4).

Table 4: Algorithms results for complex XAPUFs

Proposed architectures	Training datasets			Validation datasets		
	Sensitivity (%)	Specificity (%)	Accuracy (%)	Sensitivity (%)	Specificity (%)	Accuracy (%)
MBMPGA-DCNN	99.73	96.53	99.53	99.12	95.14	98.21
I-CSA-DCNN	99.12	95.91	98.62	98.20	94.73	97.83
BWO-DCNN	98.72	95.17	98.19	97.65	94.77	97.51
BBO-DCNN	98.14	95.20	97.75	97.10	94.13	96.76
PSO-DCNN	97.83	94.63	96.60	96.44	93.86	95.49
GA-DCNN	97.18	93.88	96.73	96.30	93.22	95.23

(Continued)

Table 4 (continued)

Proposed architectures	Training datasets			Validation datasets		
	Sensitivity (%)	Specificity (%)	Accuracy (%)	Sensitivity (%)	Specificity (%)	Accuracy (%)
Standard DCNN	95.08	92.50	94.06	94.04	91.16	93.19
MLPNN	94.11	91.29	93.74	93.49	90.86	92.65
SVM	92.56	90.14	92.04	91.75	89.25	91.24

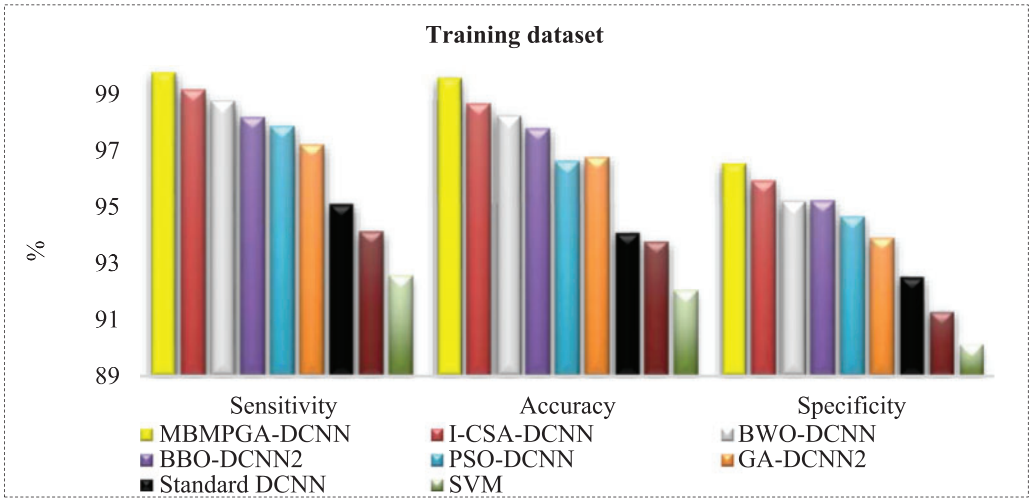


Figure 11: Comparison of ML approaches in training datasets

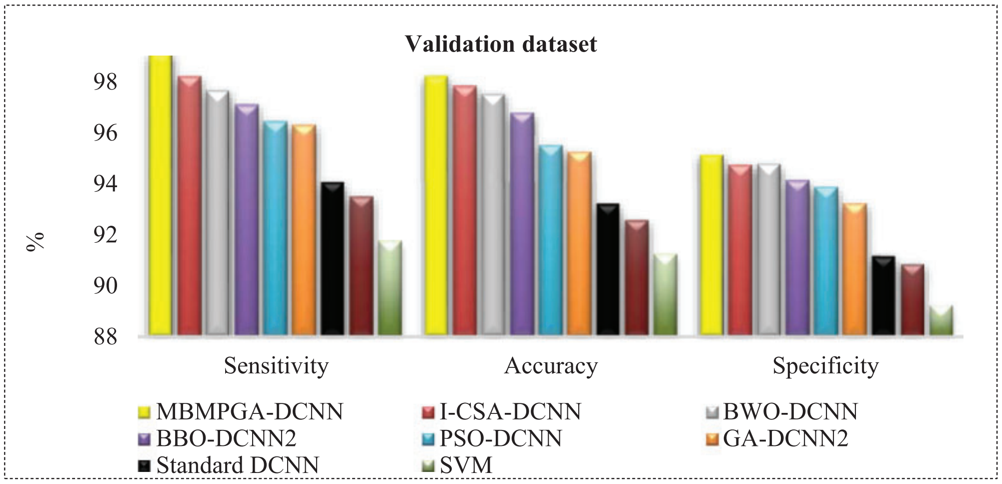


Figure 12: Comparison of ML approaches in validation datasets

Fig. 13 shows the ROC curve of architectures. The ROC Curve is valuable not only because it gives us an overview of our model's performance, but because it also gives us an easy visual to compare the performance of different classifiers to one another. As can be seen, the area under the curve (AUC) of MBMPGA-DCNN is better than other architectures. Table 5 shows a comparison of the proposed models using MSE criteria. MSE is lower in the MBMPGA-DCNN architecture than in other architectures. Thus, the introduced approach has been useful for solving this problem.

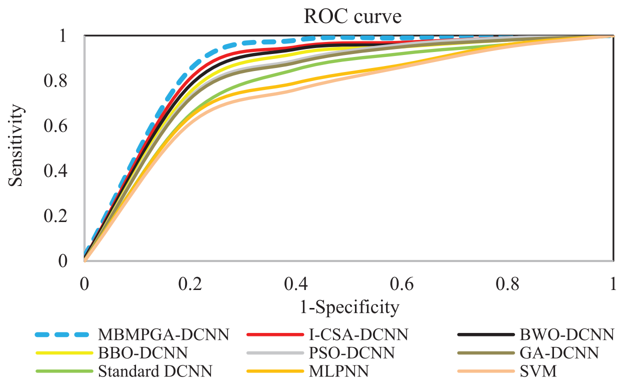


Figure 13: The ROC curve of architectures

Table 5: Comparison of the proposed models in MSE criteria

Proposed architectures	Mean Square Error (MSE)	
	Training datasets	Validation datasets
MBMPGA-DCNN	0.00005	0.00048
I-CSA-DCNN	0.00098	0.02156
BWO-DCNN	0.01181	0.07456
BBO-DCNN	0.08452	0.20156
PSO-DCNN	0.31530	0.50256
GA-DCNN	0.44215	0.63254
Standard DCNN	0.71865	0.92561
MLPNN	0.85632	0.99856
SVM	1.15638	1.35266

Fig. 14 indicates the convergence of MBMPGA-DCNN and other architectures in the accuracy criterion. The x-axis represents the epochs, while the y-axis shows the value of MSE. In MBMPGA, when several of the best chromosomes are selected to generate a new offspring, the resulting offspring bears less resemblance to one parent, i.e., the offspring is more diverse. However, since the offspring have obtained all of their genes from multiple chromosomes, the exploitation of the algorithm improves.

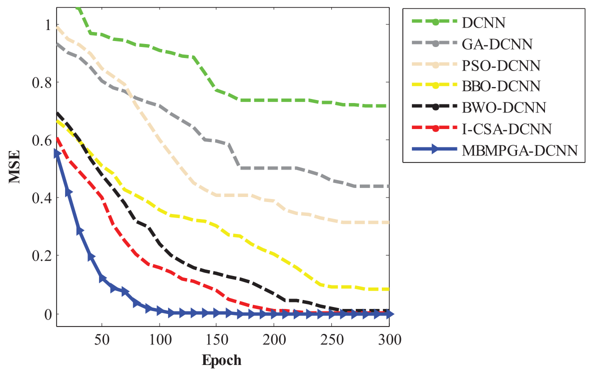


Figure 14: The convergence of architectures in the accuracy criterion

Table 6 demonstrates the trend of accuracy and execution time of all architectures in different epochs. Based on the results, the MBMPGA-DCNN has obtained the best accuracy (99.53%) in the shortest execution time. The accuracy of the I-CSA-DCNN, BWO-DCNN, BBO-DCNN, PSO-DCNN, GA-DCNN, DCNN, MLPNN, and SVM is 98.62%, 98.19%, 97.75%, 96.60%, 96.73%, 94.06%, 93.74%, and 92.04%, respectively. Fig. 15 compares the total "Runtime (s)" of the different models. As can be observed, the execution time of MBMPGA-DCNN is shorter than other models.

Table 6: The trend of accuracy and runtime of the architectures in different epochs

DL models	Metric	Epoch									
		30	60	90	120	150	180	210	240	270	300
MBMPGA-DCNN	Accuracy (%)	89.18	92.15	92.96	93.86	96.33	97.88	98.89	99.53	99.53	99.53
	Runtime (s)	70	149	791	279	331	374	460	523	600	589
I-CSA-DCNN	Accuracy (%)	88.89	90.54	92.09	93.88	94.49	95.91	97.89	98.20	98.62	98.62
	Runtime (s)	81	159	244	300	347	430	490	553	600	710
BWO-DCNN	Accuracy (%)	88.29	90.18	92.49	93.27	94.51	95.20	96.88	97.83	98.01	98.19
	Runtime (s)	96	179	240	302	370	439	504	586	630	735
BBO-DCNN	Accuracy (%)	86.12	88.47	91.05	91.96	92.70	93.43	95.76	96.15	97.29	97.75
	Runtime (s)	120	195	240	300	390	422	500	589	669	740
PSO-DCNN	Accuracy (%)	87.08	88.84	89.15	90.45	91.86	92.19	93.47	95.16	96.50	96.60
	Runtime (s)	141	215	259	320	399	462	566	651	700	790
GA-DCNN	Accuracy (%)	85.26	86.76	88.73	90.83	92.73	93.40	94.10	95.19	95.89	96.73
	Runtime (s)	119	200	259	320	410	460	541	600	683	753
Standard DCNN	Accuracy (%)	80.12	81.86	84.12	88.40	89.43	90.56	91.30	92.19	93.76	94.06
	Runtime (s)	170	263	360	414	490	542	630	730	800	879
MLPNN	Accuracy (%)	78.63	79.29	82.15	84.25	88.86	89.63	90.14	91.79	92.19	93.74
	Runtime (s)	163	251	369	425	501	552	627	719	796	865
SVM	Accuracy (%)	75.96	77.85	81.25	82.59	85.17	88.19	89.49	90.86	91.19	92.04
	Runtime (s)	112	205	308	395	474	521	603	689	762	829

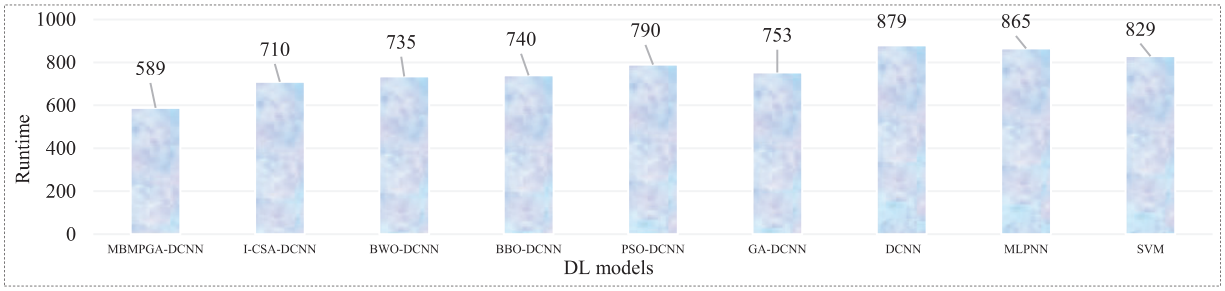


Figure 15: Total runtime of DL models

Table 7 exhibits how effective the proposed MBMPGA-DCNN model is in attacking XOR-APUF in comparison to other methods. The "Number of CRPs" column indicates the size of the training datasets required for successful machine learning training of the models, and the "Attacking Model" column mentions the machine learning methods used. As demonstrated in Table 7, our approach is more efficient than other related works in terms of the number of CRPs utilized, the needed time, and the accuracy of the results.

Table 7: Comparison of our PUF modeling with the related works

K-XOR-APUF	Method	Attacking model	Number of CRPs	Train time	Test accuracy
5-XOR-APUF	[28]	LR	80×10^4	2:8:0	99%
	[29]	MLP	14.5×10^4	0:10:12	98%
	[31]	MLP	80×10^4	0:0:58	99%
	[33]	DL	4.5×10^4	0:2:46	98%
	[34]	ECP-TRN	8×10^4	0:18:00	98%
	[36]	FC-LSTM	9×10^4	0:1:20	99%
	Ours	MBMPGA-DCNN	6.5×10^4	0:1:10	>99%
6-XOR-APUF	[28]	LR	20×10^4	31:01:0	99%
	[29]	MLP	68×10^4	0:20:52	97%
	[31]	MLP	200×10^4	0:7:42	99%
	[33]	LD	21×10^4	0:30:24	98%
	[34]	ECP-TRN	32×10^4	0:40:20	97%
	[36]	FC-LSTM	21×10^4	0:5:41	99%
	Ours	MBMPGA-DCNN	16×10^4	0:5:11	>99%
7-XOR-APUF	[29]	MLP	120×10^4	-	-
	[31]	MLP	500×10^4	0:11:8	99%
	[33]	DL	300×10^4	2:43:0	-
	[34]	ECP-TRN	56×10^4	6:20:0	97%
	[36]	FC-LSTM	99×10^4	0:9:54	99%
	Ours	MBMPGA-DCNN	67×10^4	0:8:49	>99%

(Continued)

Table 7 (continued)

K-XOR-APUF	Method	Attacking model	Number of CRPs	Train time	Test accuracy
8-XOR-APUF	[31]	MLP	30×10^6	0:23:3	99%
	[33]	DL	40×10^6	6:31:0	-
	[34]	ECP-TRN	2.7×10^6	15:50:0	97%
	[36]	FC-LSTM	2×10^6	0:16:18	99%
	Ours	MBMPGA-DCNN	8×10^5	0:19:12	>99%

6 Conclusion and Future Research Direction

In this paper, we propose a novel DL-based modeling attack to break the resistance of complex XAPUFs. More specifically, we introduce a new algorithm named MBMPGA to train the deep architecture for obtaining a higher rate of accuracy and convergence speed while decreasing the run-time of the attack. First, we evaluated the performance of MBMPGA on some well-known benchmark datasets. After successfully passing the benchmark tests, we used the generated CRPs from [33] to model complex XAPUFs using the proposed MBMPGA-based deep architecture. The simulation results on the XAPUF dataset show that the suggested modeling attack obtains more than 99% modeling accuracy and a high rate of specificity and sensitivity. The accuracy of MBMPGA-DCNN on the training and validation datasets was 99.53% and 98.21%, respectively, which were the highest accuracy rates.

The precise configuration of the MBMPGA's initial parameters can pose a constraint, and it may be necessary to utilize specific techniques, such as the Taguchi method. Another issue in practical applications can the computational overhead of MBMPGA. For instance, in a single generation of the MBMPGA, many chromosomes are produced. Typically, the computation of the loss function for all possible solutions and selecting the best one is a complex process that requires significant computing resources, especially for solving real-world problems, e.g., modeling more complex PUF structures.

There are several avenues for future research that can be recommended. Firstly, exploring the adaptation of variants of MBMPGA to address a wide range of real-world problems, including multi-objective and discrete problems, would be valuable. This would expand the applicability and versatility of the methodology. Moreover, further work can focus on optimizing specific thresholds within the equations of MBMPGA to enhance its efficiency. Fine-tuning these aspects can lead to more effective optimization outcomes. The utilization of hybrid algorithms is another promising direction for future research. Incorporating elements from different algorithms can enhance the operators within MBMPGA and potentially improve its overall effectiveness. Considering the challenges posed by limited training data in big data problems, future research should address this issue by proposing DL-based approaches that utilize multiple training datasets. By leveraging diverse datasets, DLs can overcome the limitations imposed by data scarcity and enhance their performance. Furthermore, the next generation of DL methods is anticipated to be semi-supervised and unsupervised. Future research should explore incorporating clustering concepts and techniques into DLs to improve their performance and efficiency.

As APUFs become more prevalent in security-critical applications, the development of effective defense mechanisms against adversarial attacks becomes crucial. Future research can focus on designing and evaluating robust defense mechanisms that can withstand the proposed attack methodology and mitigate its impact. While the proposed methodology primarily focuses on complex APUFs, future research could explore the applicability of the developed MBMPGA and DCNN-based attack techniques to other security-sensitive domains. This could involve investigating their effectiveness in attacking other hardware security primitives or cryptographic systems.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] X. Yang, L. Shu, Y. Liu, G. P. Hancke, M. A. Ferrag *et al.,* "Physical security and safety of IoT equipment: A survey of recent advances and opportunities," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 7, pp. 4319–4330, 2022.

[2] E. Lee, Y. D. Seo, S. R. Oh and Y. G. Kim, "A survey on standards for interoperability and security in the internet of things," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1020–1047, 2021.

[3] M. Stoyanova, Y. Nikoloudakis, S. Panagiotakis, E. Pallis and E. K. Markakis, "A survey on the internet of things (IoT) forensics: Challenges, approaches, and open issues," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1191–1221, 2020.

[4] J. Zhang, C. Shen, H. Su, M. T. Arafin and G. Qu, "Voltage over-scaling-based lightweight authentication for IoT security," *IEEE Transactions on Computers*, vol. 71, no. 2, pp. 323–336, 2021.

[5] M. Kaveh, S. Aghapour, D. Martin and M. R. Mosavi, "A secure lightweight signcryption scheme for smart grid communications using reliable physically unclonable function," in *IEEE Int. Conf. on Environment and Electrical Engineering and IEEE Industrial and Commercial Power Systems*, Europe, pp. 1–6, 2020.

[6] X. Jiang, X. Liu, J. Fan, X. Ye, C. Dai *et al.,* "Enhancing IoT security via cancelable HD-sEMG-based biometric authentication password, encoded by gesture," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16535–16547, 2021.

[7] S. Aghakhani, A. Larijani, F. Sadeghi, D. Martín and A. A. Shahrakht, "A novel hybrid artificial bee colony-based deep convolutional neural network to improve the detection performance of backscatter communication systems," *Electronics*, vol. 12, no. 10, pp. 2263, 2023.

[8] V. Vijay, K. Chaitanya, C. S. Pittala, S. S. Susmitha, J. Tanusha *et al.,* "Physically unclonable functions using two-level finite state machine," *Journal of VLSI Circuits and Systems*, vol. 4, no. 1, pp. 33–41, 2022.

[9] P. Mall, R. Amin, A. K. Das, M. T. Leung and K. K. R. Choo, "PUF-based authentication and key agreement protocols for IoT, WSNs and smart grids: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8205–8228, 2022.

[10] W. Lalouani, M. Younis, M. Ebrahimabadi and N. Karimi, "Countering modeling attacks in puf-based iot security solutions," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 18, no. 3, pp. 1–28, 2022.

[11] B. Gao, B. Lin, X. Li, J. Tang, H. Qian *et al.,* "A unified PUF and TRNG design based on 40-nm RRAM with high entropy and robustness for IoT security," *IEEE Transactions on Electron Devices*, vol. 69, no. 2, pp. 536–542, 2022.

[12] A. Shamsoshoara, A. Korenda, F. Afghah and S. Zeadally, "A survey on physical unclonable function (PUF)-based security solutions for Internet of Things," *Computer Networks*, vol. 183, no. 1, pp. 107593, 2020.

[13] Y. Zheng, W. Liu, C. Gu and C. H. Chang, “PUF-based mutual authentication and key exchange protocol for peer-to-peer IoT applications,” *IEEE Transactions on Dependable and Secure Computing*, vol. 2022, pp. 1–18, 2022.

[14] M. El-Hajj, A. Fadlallah, M. Chamoun and A. Serhrouchni, “A taxonomy of PUF schemes with a novel Arbiter-based PUF resisting machine learning attacks,” *Computer Networks*, vol. 194, no. 1, pp. 108133, 2021.

[15] P. Gope and B. Sikdar, “A comparative study of design paradigms for PUF-based security protocols for IoT devices: Current progress, challenges, and future expectation,” *Computer*, vol. 54, no. 11, pp. 36–46, 2021.

[16] M. Kaveh and A. Falahati, “An improved Merkle hash tree based secure scheme for bionic underwater acoustic communication,” *Frontiers of Information Technology & Electronic Engineering*, vol. 22, no. 7, pp. 1010–1019, 2021.

[17] X. Zhao, T. Xu, C. Xie, X. Pan, W. He *et al.,* “A CRP-space-extended RRAM PUF with in-cell zero-overhead salicide-blocked contact,” *IEEE Transactions on Electron Devices*, vol. 68, no. 7, pp. 3702–3705, 2021.

[18] H. Rabiei, M. Kaveh, M. R. Mosavi and D. Martín, “MCRO-PUF: A novel modified crossover RO-PUF with an ultra-expanded CRP space,” *Computers, Materials and Continua*, vol. 74, no. 3, pp. 4831–4845, 2023.

[19] N. Wisiol, B. Thapaliya, K. T. Mursi, J. P. Seifert and Y. Zhuang, “Neural network modeling attacks on Arbiter-PUF-based designs,” *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2719–2731, 2022.

[20] A. Lotfy, M. Kaveh, D. Martín and M. R. Mosavi, “An efficient design of Anderson PUF by utilization of the Xilinx primitives in the SLICEM,” *IEEE Access*, vol. 9, pp. 23025–23034, 2021.

[21] S. Aghapour, M. Kaveh, M. R. Mosavi and D. Martín, “An ultra-lightweight mutual authentication scheme for smart grid two-way communications,” *IEEE Access*, vol. 9, pp. 74562–74573, 2021.

[22] P. Afrasyabi, M. S. Mesgari, M. Razban and M. Kaveh, “Multi-modal routing using NSGA-II algorithm considering COVID-19 protocols: A case study in Tehran,” *Earth Observation and Geomatics Engineering*, vol. 6, no. 1, pp. 1–14, 2022.

[23] M. Kaveh, M. S. Mesgari and A. Khosravi, “Solving the local positioning problem using a four-layer artificial neural network,” *Engineering Journal of Geospatial Information Technology*, vol. 7, no. 4, pp. 21–40, 2020.

[24] O. Rostami and M. Kaveh, “Optimal feature selection for SAR image classification using biogeography-based optimization (BBO), artificial bee colony (ABC) and support vector machine (SVM): A combined approach of optimization and machine learning,” *Computational Geosciences*, vol. 25, no. 3, pp. 911–930, 2021.

[25] S. Baniasadi, O. Rostami, D. Martín and M. Kaveh, “A novel deep supervised learning-based approach for intrusion detection in IoT systems,” *Sensors*, vol. 22, no. 12, pp. 4459, 2022.

[26] P. Gope, Y. Wang, Z. Li and B. Sikdar, “QR-PUF: Design and implementation of A RFID-based secure inpatient management system using XOR-Arbiter-PUF and QR-Code,” *IEEE Transactions on Network Science and Engineering*, vol. 2022, pp. 1–9, 2022.

[27] U. Ruhrmair, F. Sehnke, J. Solter, G. Dror, S. Devadas *et al.,* “Modeling attacks on physical unclonable functions,” in *Proc. of the 17th ACM Int. Conf. on Computer and Communications Security*, Chicago, USA, pp. 237–249, 2010.

[28] P. Santikellur, A. Bhattacharyay and R. S. Chakraborty, “Deep learning-based model building attacks on Arbiter PUF compositions,” *Cryptology ePrint Archive*, vol. 2019, pp. 566, 2019.

[29] A. O. Aseeri, Y. Zhuang and M. S. Alkatheiri, “A Machine learning-based security vulnerability study on XOR PUFs for Resource-constraint Internet of Things,” in *Proc. of the 2018 IEEE Int. Conf. on Internet of Things (ICIOT)*, San Francisco, CA, USA, pp. 49–56, 2018.

[30] J. Tobisch and G. T. Becker, "On the scaling of machine learning attacks on PUFs with application to noise bifurcation," in *Proc. of the 11th RFIDsec Int. Conf. on Radio Frequency Identification: Security and Privacy Issues*, New York, NY, USA, pp. 17–31, 2015.

[31] P. Santikellur and R. S. Chakraborty, "A computationally efficient tensor regression network-based modeling attack on XOR Arbiter PUF and its variants," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1197–1206, 2020.

[32] G. Hospodar, R. Maes and I. Verbauwhede, "Machine learning attacks on 65 nm Arbiter PUFs: Accurate modeling poses strict bounds on usability," in *Proc. of the 2012 IEEE Int. Conf. on Information forensics and security (WIFS)*, Costa Adeje, Spain, pp. 37–42, 2012.

[33] K. T. Mursi, B. Thapaliya, Y. Zhuang, A. O. Aseeri and M. S. Alkatheiri, "A fast deep learning method for security vulnerability study of XOR PUFs," *Electronics*, vol. 9, no. 10, pp. 1715, 2020.

[34] J. Shi, Y. Lu and J. Zhang, "Approximation attacks on strong PUFs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2138–2151, 2019.

[35] Y. Cui, C. Gu, Q. Ma, Y. Fang, C. Wang *et al.,* "Lightweight modeling attack-resistant multiplexer-based multi-PUF (MMPUF) design on FPGA," *Electronics*, vol. 9, no. 5, pp. 815, 2020.

[36] S. S. Fard, M. Kaveh, M. R. Mosavi and S. B. Ko, "An efficient modeling attack for breaking the security of XOR-Arbiter PUFs by Using the fully connected and long-short term memory," *Microprocessors and Microsystems*, vol. 94, no. 6, pp. 104667, 2022.

[37] H. Mirhajianmoghadam, M. R. Akbarzadeh-T and E. Lotfi, "A harmonic emotional neural network for non-linear system identification," in *Proc. of the 2016 IEEE 24th ICEE Conf. on Electrical Engineering (ICEE)*, Shiraz, Iran, pp. 1260–1265, 2016.

[38] F. Sadeghi, A. Larijani, O. Rostami, D. Martín and P. Hajirahimi, "A novel multi-objective binary chimp optimization algorithm for optimal feature selection: Application of deep-learning-based approaches for SAR image classification," *Sensors*, vol. 23, no. 3, pp. 1180, 2023.

[39] M. Kaveh and M. S. Mesgari, "Hospital site selection using hybrid PSO algorithm-case study: District 2 of Tehran," *Scientific-Research Quarterly of Geographical Data (SEPEHR)*, vol. 28, no. 111, pp. 7–22, 2019.

[40] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk *et al.,* "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large-Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, 2005.

[41] W. H. Bangyal, R. Qasim, N. U. Rehman, Z. Ahmad, H. Dar *et al.,* "Detection of fake news text classification on COVID-19 using deep learning approaches," *Computational and Mathematical Methods in Medicine*, vol. 2021, no. 12, pp. 1–14, 2021.

[42] R. Qasim, W. H. Bangyal, M. A. Alqarni and A. Ali Almazroi, "A fine-tuned BERT-based transfer learning approach for text classification," *Journal of Healthcare Engineering*, vol. 2022, no. 2, pp. 1–17, 2022.

[43] L. Rukhsar, W. H. Bangyal, M. S. Ali Khan, A. A. Ag Ibrahim, K. Nisar *et al.,* "Analyzing RNA-seq gene expression data using deep learning approaches for cancer classification," *Applied Sciences*, vol. 12, no. 4, pp. 1850, 2022.

[44] M. Kaveh and M. S. Mesgari, "Application of meta-heuristic algorithms for training neural networks and deep learning architectures: A comprehensive review," *Neural Processing Letters*, vol. 2022, no. 3, pp. 1–104, 2022.

[45] M. Kaveh, M. S. Mesgari, D. Martín and M. Kaveh, "TDMBBO: A novel three-dimensional migration model of biogeography-based optimization (case study: Facility planning and benchmark problems)," *The Journal of Supercomputing*, vol. 2023, no. 9, pp. 1–56, 2023.

[46] M. Kaveh, M. S. Mesgari and B. Saeidian, "Orchard algorithm (OA): A new meta-heuristic algorithm for solving discrete and continuous optimization problems," *Mathematics and Computers in Simulation*, vol. 208, no. 3, pp. 95–135, 2023.

[47] W. Wang, Y. Yang, J. Li, Y. Hu, Y. Luo *et al.,* "Woodland labeling in Chenzhou, China, via deep learning approach," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 1393–1403, 2020.

[48] H. Fan, L. Zheng, C. Yan and Y. Yang, "Unsupervised person re-identification: Clustering and fine-tuning," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 14, no. 4, pp. 1–18, 2018.

[49] Y. Sun, B. Xue, M. Zhang, G. G. Yen and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3840–3854, 2020.

[50] F. Sadeghi, O. Rostami, M. K. Yi and S. Hwang, "A deep learning approach for detecting COVID-19 using the chest X-ray images," *Computers, Materials & Continua*, vol. 74, no. 1, pp. 751–768, 2023.