A Gradient Guided Evolutionary Approach to Training Deep Neural Networks

Article *in* IEEE Transactions on Neural Networks and Learning Systems · February 2021

DOI: 10.1109/TNNLS.2021.3061630

6 authors, including:

Shangshang Yang
Anhui University
28 PUBLICATIONS **195** CITATIONS
SEE PROFILE

Ye Tian
Anhui University
118 PUBLICATIONS **7,430** CITATIONS
SEE PROFILE

Cheng He
Huazhong University of Science and Technology
79 PUBLICATIONS **2,040** CITATIONS
SEE PROFILE

Xingyi Zhang
Anhui University
228 PUBLICATIONS **9,889** CITATIONS
SEE PROFILE

A Gradient Guided Evolutionary Approach to Training Deep Neural Networks

Shangshang Yang, Ye Tian, Cheng He, Xingyi Zhang, *Senior Member, IEEE*, Kay Chen Tan, *Fellow, IEEE*, and Yaochu Jin, *Fellow, IEEE*

***Abstract*—It has been widely recognized that the efficient training of neural networks (NNs) is crucial to the classification performance. While a series of gradient based approaches have been extensively developed, they are criticized for the ease of trapping into local optima and sensitivity to hyper-parameters. Owing to the high robustness and wide applicability, evolutionary algorithms (EAs) have been regarded as a promising alternative for training NNs in recent years. However, EAs suffer from the curse of dimensionality and are inefficient in training deep NNs. By inheriting the advantages of both the gradient based approaches and EAs, this paper proposes a gradient guided evolutionary approach to train deep NNs. The proposed approach suggests a novel genetic operator to optimize the weights in the search space, where the search direction is determined by the gradient of weights. Moreover, the network sparsity is considered in the proposed approach, which highly reduces the network complexity and alleviates overfitting. Experimental results on single-layer NNs, deep-layer NNs, recurrent NNs, and convolutional NNs demonstrate the effectiveness of the proposed approach. In short, this work not only introduces a novel approach for training deep NNs, but also enhances the performance of evolutionary algorithms in solving large-scale optimization problems.**

***Index Terms*—Deep neural networks, gradient descent, evolutionary algorithm, genetic operator, sparsity.**

Manuscript received –. This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0100100, in part by the National Natural Science Foundation of China under Grant 61672033, Grant 61822301, Grant 61876123, Grant 61906001, Grant 61903178, Grant U1804262, and Grant U20A20306, in part by the Hong Kong Scholars Program under Grant XJ2019035, in part by the Anhui Provincial Natural Science Foundation under Grant 1808085J06 and Grant 1908085QF271, in part by the State Key Laboratory of Synthetical Automation for Process Industries under Grant PAL-N201805, in part by the Research Grants Council of the Hong Kong Special Administrative Region, China under Grant PolyU11202418 and Grant PolyU11209219, and in part by a Royal Society International Exchanges Program under Grant IEC\NSFC\170279. *(Corresponding authors: Ye Tian and Xingyi Zhang.)*

S. Yang and X. Zhang are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education,School of Computer Science and Technology, Anhui University, Hefei 230039, China (email: yangshang0308@gmail.com; xyzhanghust@gmail.com).

Y. Tian is with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, Institutes of Physical Science and Information Technology, Anhui University, Hefei 230601, China (email: field910921@gmail.com).

C. He is with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (email: chenghehust@gmail.com).

Kay Chen Tan is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR (email: kctan@polyu.edu.hk).

Y. Jin is with the Department of Computer Science, University of Surrey, Guildford, Surrey, GU2 7XH, U.K. (email: yaochu.jin@surrey.ac.uk).

I. Introduction

DEEP neural networks (DNNs) have been widely applied to various domains, including image processing and computer vision [1], natural language processing [2], and object detection [3]. Since the weights in a DNN are decisive to the classification performance, the high complexity of DNNs poses stiff challenges for obtaining the optimal weights for a given dataset. During the last several decades, a number of optimization approaches have been developed for the training of neural networks (NNs), where most of them optimize the weights according to the gradient information. Typical approaches include the stochastic gradient descent (SGD) [4], RMSProp [5], and Adam [6], among many others.

Due to the rapid convergence speed provided by gradient information, the gradient based approaches have been shown to be promising in training DNNs. However, these approaches still suffer from several limitations as reported in [7]–[11]. On one hand, the gradient based approaches could be easily trapped into local optima and saddle points since they lack enough exploration ability due to their greedy nature [7], [8]. On the other hand, a regularization term usually needs to be introduced to the gradient based approaches for alleviating overfitting [12], which may not always lead to the desired result and bring extra regularization parameter. Furthermore, a number of hyper-parameters, such as the learning rate, momentum, and drop rate, should be carefully predefined [11].

As a group of meta-heuristics characterized by population evolution, evolutionary algorithms (EAs) have shown effectiveness in solving many complex optimization problems in various research domains, including a variety of non-linear [13], non-convex [14], and combinatorial optimization problems [15]. In comparison to the gradient based approaches, EAs have admiring exploration ability and insensitivity to local optima [11]. Therefore, many EAs have been suggested for training NNs since the 1980s [10], [11], [16]–[22]. For instance, Whitley *et al.* [18] adopted a genetic algorithm to optimize the weights of NNs, where experiments demonstrated the competitiveness of genetic algorithm on NNs. Montana and Davis [17] trained a feedforward NNs with a total of 126 weights via a tailored genetic algorithm, which was validated to outperform gradient based approaches.

In spite of their superiority in training NNs, EAs showed poor scalability in optimizing the weights of large-scale NNs [11], as large-scale optimization is always a challenging topic in evolutionary computation [23]. To address the curse of

dimensionality, some attempts have been made to enhance the scalability of EAs for training large-scale NNs, especially DNNs. Gong *et al.* [22] suggested a bi-objective EA to optimize the weights of sparse DNNs by only considering the biases instead of all weights, since the biases are the main factor for controlling the sparsity of hidden layers. Sun *et al.* [11] developed a single-objective EA to optimize the weights of DNNs, where the weight optimization was converted to the optimization of orthogonal bias vector in weight space. Instead of simplifying the optimization problem, Such *et al.* [24] used the parallel computing devices to enhance the search ability, aiming to improve the scalability of EAs in optimizing the weights of DNNs, by which millions of weights can be effectively optimized on a single modern desktop. Cui *et al.* [8] proposed an evolutionary approach to optimize the weights by SGD and EA alternately, where EA can enhance the stability of SGD and guarantee the steady reduction of training loss.

Among the evolutionary approaches for training large-scale NNs, most of them aimed to address the curse of dimensionality by reducing the search space (i.e., number of weights to be optimized), which may miss the optimal search region and increase the probability of getting trapped in local optima [25]. To remedy the shortcomings of existing evolutionary approaches, this paper proposes a gradient guided evolutionary approach to train DNNs, which aims to inherit the advantages of both gradient descent and EAs. Specifically, the contributions of this paper are summarized as follows:

1) A gradient based simulated binary crossover operator, termed gSBX, is proposed for optimizing the weights of DNNs. On one hand, the proposed gSBX generates offspring solutions by a similar strategy to the widely used simulated binary crossover (SBX) [26], which maintains a similar exploration ability to EAs. On the other hand, the search direction of gSBX is always set to the same as the gradient of the parent, aiming to enhance the exploitation ability by the gradient information. Therefore, the proposed gSBX can strike a balance between exploitation and exploration in optimizing the weights of DNNs, hoping to result in faster convergence speed than EAs and better optimization result than gradient descent.
2) A gradient guided evolutionary multi-objective approach to train DNNs is then proposed, called GEMONN. To be more specific, the evolutionary multi-objective optimization technique is adopted to optimize the training loss and control the network sparsity at the same time, where the proposed gSBX is adopted to generate promising offspring solutions.
3) To validate the performance of the proposed approach, it is used for optimizing the weights of four types of NNs, i.e., auto-encoder (AE), stacked AE (SAE), long short-term memory (LSTM), and convolutional NNs (CNNs). Experimental results indicate that the proposed approach can considerably enhance the scalability of EAs in optimizing the large-scale weights of DNNs, even arriving at millions. It is also verified that the proposed approach is superior over the state-of-the-art evolutionary approaches and competitive to gradient based approaches in training DNNs.

The rest of this paper is organized as follows. Section II reviews the existing evolutionary approaches for training NNs, then gives the motivation of this work. Section III elaborates on the details of the proposed approach. Section IV presents the experimental results to verify the performance of the proposed approach. Finally, conclusions and future work are given in Section V.

II. Related Work and Motivation

A. Multi-Objective Neural Network Training

Given a training set D and a network net, EAs usually search for the optimal values of the weights by minimizing the following bi-objective optimization problems [9]:

$$\min_{\mathbf{x}} \mathbf{F}(\mathbf{x}) = \begin{cases} f_1(\mathbf{x}) = f_{complexity}(net, \mathbf{x}) \\ f_2(\mathbf{x}) = f_{loss}(net, D, \mathbf{x}) \end{cases}, \tag{1}$$

where $\mathbf{x}$ is the set of all the weights to be optimized, $f_1(\mathbf{x})$ denotes the complexity of net with the weights being $\mathbf{x}$, and $f_2(\mathbf{x})$ denotes the training loss of net averaged over all the samples in D. In practice, the network complexity can be measured by the number of nonzero weights [27] or the sum of absolute weights [23], while the training loss can be classification error [9] or reconstruction error [10].

Since each solution contains two objective values, the quality of solutions is usually determined by the Pareto dominance relation. Specifically, a solution $\mathbf{x}_1$ is said to Pareto dominate solution $\mathbf{x}_2$, denoted as $\mathbf{x}_1 \prec \mathbf{x}_2$, if and only if the following conditions hold:

$$\begin{cases} \forall i = 1, 2, \ldots, m, \ f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2) \\ \exists j = 1, 2, \ldots, m, \ f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2) \end{cases}, \tag{2}$$

where m is the number of objectives and equals to 2 in this work. Besides, a solution is called Pareto optimal if it is not dominated by any other solutions. All the Pareto optimal solutions constitute the Pareto optimal set, and the projection of the Pareto optimal set in objective space is called the Pareto optimal front (PF). As a result, the goal of multi-objective neural network training is to find a set of solutions approximating the PF, which present trade-offs between the network complexity and training loss.

B. Evolutionary Approaches for Training Neural Networks

Since the 1990s, a series of work has been dedicated to training NNs by EAs [16]–[18]. Among these approaches, one of the major concerns is to enhance the scalability of EAs in tackling large-scale NNs, as EAs usually suffer from the curse of dimensionality in large-scale optimization [28]. Therefore, various strategies have been suggested to address this issue, where some representatives are reviewed in the following.

Stanley and Miikkulainen [20] suggested a hybrid encoding method to simultaneously optimize the structure and weights of an NN. Specifically, the network structure and the weights between nodes are represented by a variable-length vector, and a genetic algorithm is adopted to evolve a set of solutions using

two tailored genetic operators. However, the length of each solution will considerably increase due to the explicitly encoding of both structure and weights, which disables the algorithm from optimizing the weights for low training loss [11].

Wu *et al.* [29] optimized the weights as well as connection structure of NNs by a decomposition based multi-objective evolutionary algorithm (MOEA/D) [30] and two novel genetic operators. In this approach, the weights and connection structure are encoded in a hybrid decision vector, which is optimized by MOEA/D equipped with a tailored crossover operator and a tailored mutation operator. This approach exhibited better scalability than previous work for training large-scale NNs with approximately 6000 weights. Nevertheless, its efficiency in optimizing larger NNs needs further improvement.

In [22], an EA based induced learning approach was designed by Gong *et al.* for optimizing the large-sale weights of AE. In addition to minimizing the reconstruction error, the sparsity of hidden units is also considered for better generalization. More importantly, only the biases vector is optimized by EA to directly control the sparsity, with the assistance of a gradient based local search strategy. Thus, the weight optimization is simplified as a small-scale optimization problem and can be easily solved. Since EA can guarantee that the best fitness in population never degrades, the population can be stably evolved for better convergence.

Liu *et al.* [10] suggested an EA based layer-wise structure learning approach for optimizing the connection of weights. To reduce the search space, the value of each weight is first limited to either 0 or 1. Then, a multistage initialization strategy is designed for reducing the search space to different degrees in different stages, where multiple weights share the same value according to the adjacency relations to the input layer. By doing so, this approach holds good scalability in training DNNs.

Zhu and Jin [31] devised an EA based hyper-parameter optimization approach for searching the optimal structure of NNs. To enhance the scalability in evolving large-scale NNs, a modified sparse evolutionary training (SET) [32] strategy is suggested to remove a fraction of weights with the smallest update at the last training step in NNs. By encoding the modified SET as only two hyper-parameters, the entire learning process can be realized by optimizing all hyper-parameters based on a tailored multi-objective algorithm. In spite of its promising performance on DNNs, the training process of NNs mainly relies on gradient based approach.

In [33], Koutník proposed a Fourier transform based EA to optimize the weights of large-scale NNs for vision-based reinforcement learning. This approach represents the weights by a sequence of Fourier-type coefficients, which is variable-length and much shorter than the vector of weights. In short, this approach also converts the training of large-scale NNs into small-scale optimization problems.

Real *et al.* [34] employed an EA to evolve the whole structure of NNs from the trivial initial structure. To discover more various models, a fairly unrestricted search space with no fixed depth and arbitrary skip connections is designed for their approach. Equipped with tailored mutation operators, more promising offspring models can be generated based on the parent models. Although this approach merely optimizes the structure of NNs, it provides a bright prospect of combining EA with NNs.

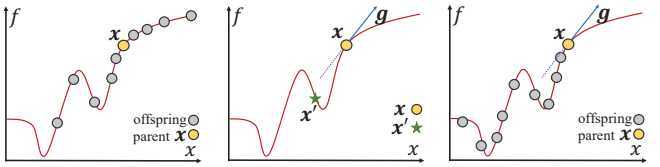
The parallel computing technique has also been adopted for optimizing the weights of DNNs [24]. In order to store a large number of weights compactly, an efficient encoding method is designed for the distributed deep genetic algorithm, which stores a list of random seeds that produce the series of mutations for offspring generation. Moreover, a novel search strategy is adopted to avoid local optima by ignoring the reward function during evolution.

More recently, an innovative evolutionary approach was proposed by Sun *et al.* [11] for training unsupervised DNNs, called EUDNN. This approach reduces the search space by representing the large number of weights by the weighted sum of a set of orthogonal vectors, where the number of these transformed variables is just the square root of the number of weights. In addition, other hyper-parameters are also optimized together with the weights. Experimental results demonstrated the superiority of EUDNN over state-of-the-art approaches in training DNNs with millions of weights.

C. Motivation of This Work

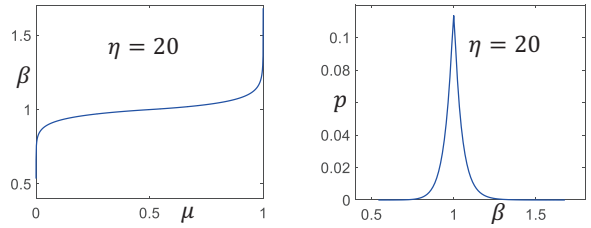
In spite of the promising performance of the above evolutionary approaches in training NNs, most of them encounter difficulties in optimizing a large number of weights in DNNs [11]. The limitation of these approaches is mainly attributed to the employment of dimensionality reduction strategy, which addresses the curse of dimensionality to some extent but is more vulnerable to getting trapped in local optima [25]. However, these approaches have to reduce the number of optimized weights due to the limitation of the search ability of EAs for large-scale optimization [23]. Besides, there are some approaches [8], [22] directly combining EA and gradient descent to search in the original search space, where the weights are successively updated by a traditional genetic operator and gradient descent. However, the obstacle of EA having poor performance on large-scale optimization problems has not been overcome. In other words, the EA in these approaches merely guarantees that the best fitness in the population does not degrade, while the gradient descent is mainly responsible for finding better solutions in the original search space.

To better illustrate the limitation of traditional genetic operators, Fig. 1 illustrates the difference between the search behaviors of genetic operators and gradient descent in optimizing a multimodal function $f(x)$. More specifically, the popular genetic operator SBX [26] is taken as an example in Fig. 1(a), where the generated offspring solutions distribute on both sides of the parent $\mathbf{x}$, since SBX determines the search direction randomly and enables more unknown areas to be exploited. In other words, SBX makes many unnecessary search efforts, and it becomes even more serious for optimizing the weights of DNNs in a high-dimensional search space. While for the gradient descent shown in Fig. 1(b), it has better exploitation ability since the search direction is always in contrast to the gradient of $\mathbf{x}$; however, an unsuitable step size (determined by the learning rate) is likely to drive $\mathbf{x}$ to a local optimum.



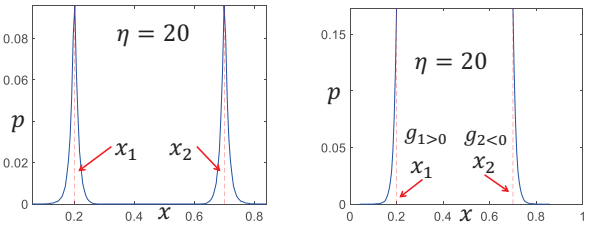
(a) The offspring solutions generated by genetic operator. (b) The offspring solution generated by gradient descent. (c) The offspring solutions generated by combining genetic operator and gradient descent.

Fig. 1. Search behaviors of genetic operator and gradient descent on a minimization problem $f(\mathbf{x})$.



(a) Relationship between β and μ. (b) Probability distribution of β.

Fig. 2. Relationship between β and μ in SBX.



(a) Offspring solutions generated by SBX. (b) Offspring solutions generated by modified SBX.

Fig. 3. Probability distribution of the offspring solutions generated based on two parents $x_1 = 0.2$ and $x_2 = 0.7$ by SBX and modified SBX.

As illustrated in Fig. 1(c), if the genetic operator generates offspring solutions only in the expected direction, which is often hidden in the gradient information, the shortcomings of genetic operator and gradient descent can be remedied by each other. To be more specific, the introduction of gradient information in the genetic operator can limit the search behavior in promising directions, thus highly reducing the search effort in unnecessary directions and improving the exploitation ability. Meanwhile, the generation of multiple offspring solutions can highly reduce the probability of trapping into local optima and improve the exploration ability of gradient descent; besides, it eliminates the use of learning rate, which is difficult to be predefined or adapted.

Following this idea, this paper serves as the first attempt to design a gradient based genetic operator for training DNNs, which enables EAs to optimize the weights of DNNs without any reduction of the search space. More importantly, existing approaches are usually tailored for a specific type of DNN; while this paper aims to propose a generic approach for training different types of DNNs. In the next section, the proposed genetic operator as well as the gradient guided evolutionary approach is elaborated.

III. The Proposed Approach

A. Gradient Based Simulated Binary Crossover

In recent years, there are many crossover operators proposed for EAs, such as SBX, UNDX [35], and DE [36]. Among these crossovers, SBX exhibits better performance on most test problems as demonstrated in [37]. Therefore, we adopt SBX as the basic genetic operator, which is widely used for continuous optimization problems. Formally, given two parents $\mathbf{x}_1$ and $\mathbf{x}_2$ with n decision variables, SBX generates two offspring solutions $\mathbf{c}_1$ and $\mathbf{c}_2$ by

$$\begin{cases} \mathbf{c}_1^i = [(1+\beta)\mathbf{x}_1^i + (1-\beta)\mathbf{x}_2^i]/2 \\ \mathbf{c}_2^i = [(1-\beta)\mathbf{x}_1^i + (1+\beta)\mathbf{x}_2^i]/2 \end{cases}, \ 1 \le i \le n\ , \quad (3)$$

where $\mathbf{x}_1^i$ denotes the i-th variable of decision vector $\mathbf{x}_1$ and β is a random value that needs to be regenerated for each dimension by

$$\beta = \begin{cases} (2\mu)^{\frac{1}{\eta+1}} & , \ \mu \le 0.5 \\ (2(1-\mu))^{-\frac{1}{\eta+1}} & , \ \mu > 0.5 \end{cases}, \quad (4)$$

where μ is a uniformly distributed random value sampled in $[0, 1]$ and η is a parameter controlling the distribution of the offspring solution. According to [26], the value of η is suggested to be 20. Fig. 2(a) plots the relationship between β and μ, and Fig. 2(b) plots the probability distribution of β. Obviously, β is very likely to be around 1, where $\beta = 1$ will make $\mathbf{c}_1 = \mathbf{x}_1$ and $\mathbf{c}_2 = \mathbf{x}_2$ according to (3).

For a visual observation of the search behavior of SBX, Fig. 3(a) depicts the probability distribution of the offspring solutions generated by SBX, based on two one-dimensional parents $x_1 = 0.2$ and $x_2 = 0.7$. It can be found that the offspring solutions locate on the left and right sides of each parent with the same probability. While if the gradient information is taken into consideration, the offspring solutions should only locate on the left side of the first parent x_1 and the right side of the second parent x_2, where the gradient g_1 of the first parent is assumed to be positive and the gradient g_2 of the second parent is assumed to be negative. Next, the details of incorporating gradient information into SBX are elaborated.

1) Core Idea of Gradient Based SBX (gSBX): As a matter of fact, (3) can be reformulated as

$$\begin{cases} \mathbf{c}_1^i = \mathbf{x}_1^i + \alpha_1(\mathbf{x}_1^i - \mathbf{x}_2^i)/2 \\ \mathbf{c}_2^i = \mathbf{x}_2^i - \alpha_2(\mathbf{x}_1^i - \mathbf{x}_2^i)/2 \end{cases}, \ 1 \le i \le n \quad (5)$$

with

$$\begin{aligned} \alpha_1 &= \beta_1 - 1 \\ \alpha_2 &= \beta_2 - 1\ . \\ \beta_1 &= \beta_2 = \beta \end{aligned} \quad (6)$$

To incorporate the gradient $\mathbf{g}_1$ and $\mathbf{g}_2$ into the genetic operator, the following conclusion can be easily obtained:

$$\begin{cases} \mathbf{c}_{1,2}^i < \mathbf{x}_{1,2}^i,\ \mathbf{g}_{1,2}^i > 0 \\ \mathbf{c}_{1,2}^i \geq \mathbf{x}_{1,2}^i,\ \mathbf{g}_{1,2}^i \leq 0 \end{cases},\ 1 \leq i \leq n, \tag{7}$$

where $\mathbf{c}_{1,2}$ stands for $\mathbf{c}_1$ or $\mathbf{c}_2$, $\mathbf{x}_{1,2}$ stands for $\mathbf{x}_1$ or $\mathbf{x}_2$, and $\mathbf{g}_{1,2}$ stands for $\mathbf{g}_1$ or $\mathbf{g}_2$.To this end, according to (5), $\alpha_1(\mathbf{x}_1^i - \mathbf{x}_2^i)$ should have a different sign with $\mathbf{g}_1^i$ while $\alpha_2(\mathbf{x}_1^i - \mathbf{x}_2^i)$ should have the same sign with $\mathbf{g}_2^i$. Since the value of $(\mathbf{x}_1^i - \mathbf{x}_2^i)$ is determined, the sign of α_1^i and α_2^i can be varied as expected, which is controlled by μ according to (4) and (6). As a consequence, the proposed gSBX incorporates the gradient information into the genetic operator by setting μ in the following way:

$$\mu_1 = \begin{cases} \text{sample in } [0, 0.5], & \mathbf{g}_1^i(\mathbf{x}_1^i - \mathbf{x}_2^i) > 0 \\ \text{sample in } (0.5, 1], & \mathbf{g}_1^i(\mathbf{x}_1^i - \mathbf{x}_2^i) < 0 \\ \text{sample in } [0, 1], & \text{otherwise} \end{cases} \tag{8}$$

and

$$\mu_2 = \begin{cases} \text{sample in } [0, 0.5], & \mathbf{g}_2^i(\mathbf{x}_1^i - \mathbf{x}_2^i) < 0 \\ \text{sample in } (0.5, 1], & \mathbf{g}_2^i(\mathbf{x}_1^i - \mathbf{x}_2^i) > 0 \\ \text{sample in } [0, 1], & \text{otherwise} \end{cases}, \tag{9}$$

where μ_1 and μ_2 are used for generating α_1 and α_2 by (4) and (6), respectively, and they need to be regenerated for each dimension. Besides, μ_1 and μ_2 are sampled in the original interval $[0, 1]$ if the gradient is equal to zero or unavailable, which is the same to the original SBX.

2) Uniform Crossover in gSBX: The idea of uniform crossover is also adopted in gSBX, which has been used in many existing genetic operators for improving the diversity [28]. Specifically, the uniform crossover exchanges the variables in each dimension of the two offspring solutions $\mathbf{c}_1$ and $\mathbf{c}_2$ with a probability of 0.5. According to (5), we have

$$\begin{aligned} \mathbf{c}_1^i &= \mathbf{x}_i^1 + (\beta_1 - 1)(\mathbf{x}_1^i - \mathbf{x}_2^i)/2 \\ &= (1 - \beta_1)\mathbf{x}_2^i/2 + (1 + \beta_1)\mathbf{x}_1^i/2, \\ &= \mathbf{x}_2^i - (-\beta_1 - 1)(\mathbf{x}_1^i - \mathbf{x}_2^i)/2 \end{aligned} \tag{10}$$

hence $\mathbf{c}_2^i$ can be changed to $\mathbf{c}_1^i$ if we set $\alpha_2 = -\beta_1 - 1$. Similarly, $\mathbf{c}_1^i$ can be changed to $\mathbf{c}_2^i$ if we set $\alpha_1 = -\beta_2 - 1$.

There is another thing needing to be considered when exchanging $\mathbf{c}_1^i$ and $\mathbf{c}_2^i$. Assuming that the gradient $\mathbf{g}_1^i > 0$, it is unreasonable to change $\mathbf{c}_1^i$ to $\mathbf{c}_2^i$ when $\mathbf{x}_1^i < \mathbf{x}_2^i$, since $\mathbf{x}_1^i$ should be moved along the negative direction of $\mathbf{g}_1^i$. Therefore, some conditions should be imposed to limit the uniform crossover, where the uniform crossover can be performed only when $\mathbf{g}_1^i(\mathbf{x}_1^i - \mathbf{x}_2^i) > 0$ and $\mathbf{g}_2^i(\mathbf{x}_1^i - \mathbf{x}_2^i) < 0$. To summarize, the proposed gSBX sets α_1 and α_2 as

$$\alpha_1 = \begin{cases} -\beta_2 - 1, & \lambda \leq 0.5 \text{ and } \mathbf{g}_1^i(\mathbf{x}_1^i - \mathbf{x}_2^i) > 0 \\ & \text{and } \mathbf{g}_2^i(\mathbf{x}_1^i - \mathbf{x}_2^i) < 0 \\ \beta_1 - 1, & \text{otherwise} \end{cases} \tag{11}$$

and

$$\alpha_2 = \begin{cases} -\beta_1 - 1, & \lambda \leq 0.5 \text{ and } \mathbf{g}_1^i(\mathbf{x}_1^i - \mathbf{x}_2^i) > 0 \\ & \text{and } \mathbf{g}_2^i(\mathbf{x}_1^i - \mathbf{x}_2^i) < 0 \\ \beta_2 - 1, & \text{otherwise} \end{cases}, \tag{12}$$

Algorithm 1: $gSBX(P)$

Input: P: Parents;
Output: O: Offspring population;
1: $O \leftarrow \emptyset$;
2: **while** $P \neq \emptyset$ **do**
3: Randomly select two solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ from P;
4: $P \leftarrow P \setminus \{\mathbf{x}_1, \mathbf{x}_2\}$;
5: **for** $i = 1$ to n **do**
6: Sample μ_1 and μ_2 by (8) and (9);
7: Calculate β_1 and β_2 by (4);
8: Calculate α_1 and α_2 by (11) and (12);
9: Repair α_1 and α_2 by (13) and (14);
10: Calculate $\mathbf{c}_1^i$ and $\mathbf{c}_2^i$ by (5);
11: **end for**
12: $O \leftarrow O \cup \{\mathbf{c}_1, \mathbf{c}_2\}$;
13: **end while**
14: **return** O;

where λ is sampled from a uniform distribution $U[0, 1]$ for each dimension.

3) Considering Sparsity in gSBX: In order to better control the network sparsity, the sparsity of the offspring solutions is maintained in gSBX, where an intuitive idea is to make the offspring solutions have the same or higher sparsity than their parents. To this end, the proposed gSBX sets each variable of an offspring solution to zero if the variable of the corresponding parent is zero. Thus, the number of zero weights in the offspring solution is always equal to or larger than the number of zero weights in the parent. According to (5), $\mathbf{c}_1^i = \mathbf{x}_1^i = 0$ can be held if we set $\alpha_1 = 0$. As a result, the proposed gSBX further repairs α_1 and α_2 by

$$\alpha_1 = \begin{cases} 0, & \mathbf{x}_1^i = 0 \\ \alpha_1, & \text{otherwise} \end{cases} \tag{13}$$

and

$$\alpha_2 = \begin{cases} 0, & \mathbf{x}_2^i = 0 \\ \alpha_2, & \text{otherwise} \end{cases}. \tag{14}$$

4) Procedure of gSBX: The detailed procedure of gSBX is presented in Algorithm 1. As can be seen, two parents $\mathbf{x}_1$ and $\mathbf{x}_2$ are selected from the population P randomly, and they are used to generate two offspring solutions $\mathbf{c}_1$ and $\mathbf{c}_2$. For each dimension, the values of μ_1, μ_2, β_1, β_2, α_1, and α_2 are determined in sequence, and the decision variables $\mathbf{c}_1^i$ and $\mathbf{c}_2^i$ are calculated accordingly. This procedure repeats until all the solutions in P are visited, and all the newly generated solutions are returned as the offspring population.

A comparative experiment is conducted to briefly illustrate the superiority of the proposed gSBX over original SBX. Here, both gSBX and SBX are embedded in a simple genetic algorithm to minimize a function $f(\mathbf{x})$ having a mixed landscape:

$$\min_{\mathbf{x}} \mathbf{f}(\mathbf{x}) = \sum_{i=1}^{40} x_i^2 - \frac{1}{2}\sum_{i=41}^{60} x_i \sin(\sqrt{|x_i|}), \tag{15}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_{60}) \in [-500, 500]^{60}$. The population size is set to 100 and the number of generations is set to 100. Besides, the gradient descent with a learning rate of 0.1 is also adopted to minimize the function for 10000 steps. Fig. 4 shows the decline in minimum objective value found by gradient descent, genetic algorithm with SBX, and genetic algorithm with the proposed gSBX. It can be found that the

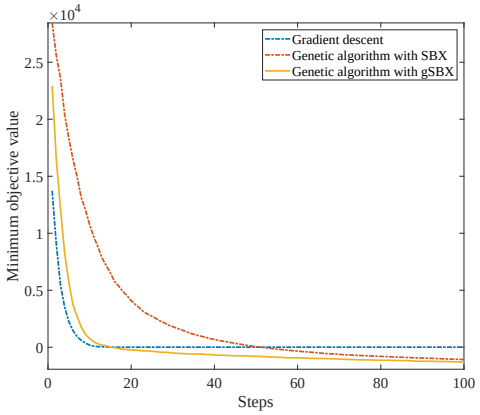


Fig. 4. Decline in minimum objective value found by gradient descent, genetic algorithm with SBX, and genetic algorithm with the proposed gSBX on an objective function with mixed landscape.

Algorithm 2: Procedure of the Proposed GEMONN

Input: Gen: Maximum number of generations; Pop: Population size; Net: A DNN to be trained;
Output: $\mathbf{x}$: Weights of Net;
- - Weights Optimization - -
$P \leftarrow$ Randomly initialize Pop solutions;
Calculate the objective values and gradient of each solution in P;
for $g = 1$ to Gen **do**
  $Parent \leftarrow$ Select Pop solutions from P by the mating selection of NSGA-II;
  $O \leftarrow gSBX(Parent)$;
  Calculate the objective values and gradient of each solution in O;
  $P \leftarrow$ Select Pop solutions from $P \cup O$ by the environmental selection of NSGA-II;
end for
- - Pareto Optimal Solutions Update - -
$P \leftarrow$ Delete the dominated solutions from P;
for each $\mathbf{x} \in P$ **do**
  $\mathbf{x} \leftarrow$ Fine-tune $\mathbf{x}$ by sparse stochastic gradient descent;
end for
- - Final Solution Selection - -
$\mathbf{p} \leftarrow$ Select one solution from P in the knee area;
return $\mathbf{p}$;

gradient descent quickly gets trapped into a local optimum; by contrast, SBX converges slower than gradient descent but it can find a smaller objective value at last. As for the proposed gSBX, it has a similar convergence speed to gradient descent and finds a similar objective value to SBX. As a result, gSBX can strike a balance between exploitation and exploration, and it is expected to be efficient in training DNNs with a large number of weights.

Based on the proposed gSBX, a gradient guided evolutionary approach is proposed to train DNNs, the details of which are presented in the next subsection.

B. Gradient Guided Evolutionary Approach to Train DNNs

Algorithm 2 presents the procedure of the proposed GEMONN. To begin with, a population with predefined size is created, where the decision variables (i.e., weights) of each initial solution is set to a random value within $[-1, 1]$ or 0 with the same probability. Then, the objective values of each solution are evaluated and the gradient in terms of the training loss is calculated. In each generation of GEMONN, a number of solutions are selected from the population by the mating selection strategy of NSGA-II [38] and used to generate the same number of offspring solutions by the proposed gSBX. Afterwards, the offspring population is combined with the original population, and half the solutions in the combined population are retained for next generation by the environmental selection strategy of NSGA-II. After the evolution terminates, a single solution should be picked up from the final population as the output. For this aim, a Pareto optimal solution update strategy and a final solution selection strategy are developed.

Fig. 5 plots the flowchart of the proposed GEMONN. In the following, the details of the main components of GEMONN, including solution evaluation, Pareto optimal solution update, and final solution selection, are described separately.

1) Quality Evaluation: Given a training set $D = \{(\mathbf{s}_i, \mathbf{y}_i)\}_{i=1}^N$ and a DNN net, the proposed GEMONN calculates the following two objectives for each solution $\mathbf{x}$:

$$\min_{\mathbf{x}} \mathbf{F}(\mathbf{x}) = \begin{cases} f_1(\mathbf{x}) = ||\mathbf{x}||_0 \\ f_2(\mathbf{x}) = \dfrac{1}{N}\sum_{i=1}^{N} L(\hat{\mathbf{y}}_i) \end{cases}, \tag{16}$$

where the decision vector $\mathbf{x}$ denotes the set of all the weights in net, and $\hat{\mathbf{y}}$ is the output of net with the weights being $\mathbf{x}$. The first objective f_1 indicates the complexity of the DNN, where a smaller f_1 means to a sparser network. Here the l_0-norm is adopted to measure the complexity, which is validated to be effective since it directly counts the number of nonzero weights [9], [28]. The second objective f_2 indicates the training loss averaged over all the training samples, where the cross entropy loss (CE) [39] is adopted for supervised learning tasks and the mean squared error (MSE) [9] is adopted for unsupervised learning tasks. Specifically, f_2 is defined as

$$f_2(\mathbf{x}) = \begin{cases} L_{CE}(\hat{\mathbf{y}}_i) = -\sum_{j}^{l} \mathbf{y}_i^j log(\hat{\mathbf{y}}_i^j) \\ L_{MSE}(\hat{\mathbf{y}}_i) = \dfrac{1}{m}\sum_{j}^{m} (\mathbf{s}_i^j - \hat{\mathbf{y}}_i^j)^2 \end{cases}, \tag{17}$$

where $\mathbf{y}_i^j$ denotes the j-th label of the i-th sample, $\mathbf{s}_i^j$ denotes the j-th feature of the i-th sample, l is the total number of labels, and m is the total number of features.

Since the network may overfit the training set by the mere minimization of training loss, we consider the network complexity to alleviate overfitting. Intuitively, the two objectives f_1 and f_2 are conflicting with each other since a network with lower training loss usually corresponds to a higher complexity. But, owing to the delicate selection strategy provided by evolutionary multi-objective optimization, a set of solutions making good trade-offs between the two objectives can be found, and a network with promising training loss but does not overfit the training set is expected to be obtained.

2) Pareto Optimal Solution Update: After the evolution terminates, a Pareto optimal solution update strategy is tailed for the final exploitation. To be specific, the dominated solutions in the population are removed, and then each non-dominated solution is fine-tuned by SGD [40] with a learning rate of 0.1 for one step. It is worth noting that only the nonzero weights are updated here, while the zero weights are kept unchanged.

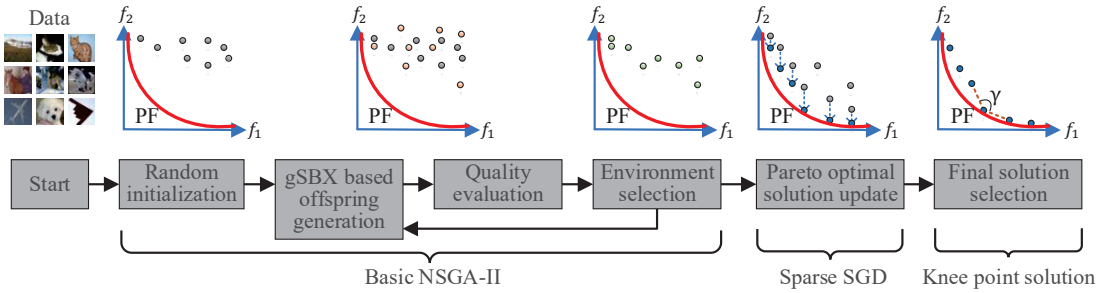


Fig. 5. Whole flowchart of the proposed GEMONN.

Such a trick can decrease the training loss but does not lose the network sparsity, and we call it sparse SGD (SSGD) hereafter.

3) Final Solution Selection: Last but not the least, a single solution should be selected from the fine-tuned population as the output. Intuitively, it is unreasonable to select the solution with the lowest training loss or complexity, as the former tends to overfit the training set and the later is usually not well trained. Since the solutions in the knee area of the population have a better trade-off between the two objectives [41], they are usually preferred in many scenarios [10], [22]. Based on this idea, the proposed GEMONN selects a solution $\mathbf{p}$ according to the following criteria:

$$\begin{aligned} X_1 &= \{\mathbf{x} | \text{int}(\log_{10} f_2(\mathbf{x})) - \text{int}(\log_{10} f_2(\mathbf{x}^+)) < 1\} \\ X_2 &= \{\mathbf{x} | f_2(\mathbf{x}) < 2 * f_2(\mathbf{x}^+),\ \mathbf{x} \in X_1\} \\ \mathbf{p} &= \arg\min_{\mathbf{x}} \gamma(\mathbf{x}),\ \mathbf{x} \in X_2 \end{aligned} \quad . \quad (18)$$

The first criterion aims to find the solutions whose training losses are with the same order of magnitude as $\mathbf{x}^+$, where $\mathbf{x}^+$ is the solution having the lowest training loss in the population, i.e., $\mathbf{x}^+ = \arg\min_{\mathbf{x} \in P} f_2(\mathbf{x})$. The second criterion is also to guarantee the training performance of the selected solution. The third criterion is used to find the final solution according to $\gamma(\mathbf{x})$, which calculates the angle between the left and right neighbors of the solution [42].

C. Time Complexity Analysis

The time complexity of proposed GEMONN is mainly determined by two components, i.e., the calculation of gradients and the optimization process of NSGA-II. According to [43], the time complexity for calculating gradients is $O(N \times m)$, where N and m are the size of a mini-batch and feature dimensionality, respectively. The time complexity of one generation of NSGA-II is $O(Pop^2)$ [38], where Pop is the population size. Therefore, the overall time complexity of GEMONN is $O(N \times m \times Pop \times Gen) + O(Pop^2 \times Gen)$. Since $N \times m \gg Pop$ in general, the time complexity of GEMONN can be regarded as $O(N \times m \times Pop \times Gen)$, which is same as that of SGD [43].

IV. Empirical Studies

A. Neural Networks to be Trained

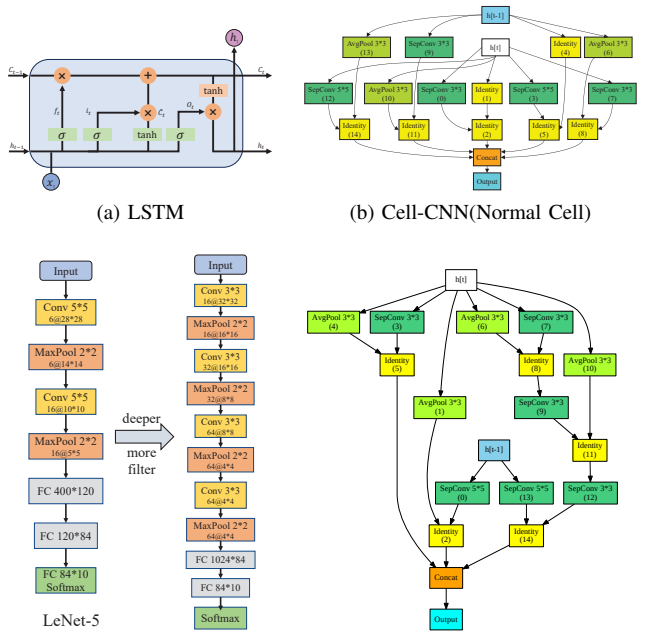
The performance of the proposed GEMONN is verified on four types of NNs, including AE, SAE [44], LSTM [45], and CNNs [46]. A brief introduction to the training of the four types of NNs is given in the following.

1) Auto-Encoder (AE): As a fully connected (FC) feedforward NN with a single hidden layer, AE is often used for dimensionality reduction, where the output is expected to be the same to the input and the hidden layer can be regarded as a meaningful and reduced representation of the training sample. As a result, AE can be trained in an unsupervised manner, and many existing evolutionary approaches have been adopted to train it [10], [11], [22]. To train AE by the proposed GEMONN, the MSE is adopted as the second objective. Besides, the sigmoid activation function is applied to the hidden layer.

2) Stacked Auto-Encoder (SAE): By using AE as the building blocks, SAE can reduce the training sample layer by layer and do classification at last. With the greedy layer-wise training process, the dependency of weights in different layers can be well handled, and the weights in each layer can be optimized separately. The training of SAE is composed of two phases [11], [47], i.e., pre-training and fine-tuning. The pre-training phase trains a set of AEs with different numbers of hidden nodes by minimizing the reconstruction error. Then, these AEs are stacked together and tailed by a FC layer, and the fine-tuning phase optimizes all the weights for minimizing the classification error. To train SAE by the proposed GEMONN, the MSE is adopted as the second objective in the pre-training phase and the CE is adopted in the fine-tuning phase.

3) Long Short-Term Memory (LSTM): LSTM is currently one of the most popular recurrent NNs for many deep learning tasks, in which the connections between layers are much more complex than those in feedforward NNs. Since all the weights in LSTM should be optimized together, it is much more difficult than training SAE whose weights are optimized layer by layer, and none of the existing evolutionary approaches have been applied to it. In order to demonstrate the effectiveness of the proposed genetic operator in training DNNs without dimensionality reduction, we also use the proposed GEMONN to train LSTM with its structure shown in Fig. 6(a), where the CE is adopted as the second objective.

4) Convolutional Neural Network (CNN): CNNs are also powerful NNs for many deep learning tasks, especially preferred by researchers in computer vision. To investigate the ability of GEMONN in training CNNs, we consider the LeNet-5 [48] and its variant (termed LeNet-deeper) having



(a) LSTM (b) Cell-CNN(Normal Cell)

(c) LeNet-5 and its variant (d) Cell-CNN(Reduction Cell)

Fig. 6. The structure of used NNs, including LSTM, LeNet-5 as well as its variant(deeper and more filters), and Cell-based CNN.

more convolutional operations and more filters as shown in Fig. 6(c), where a ReLu [49] activation function follows each convolutional operation. To investigate the results on deeper CNNs, a Vgg16 [50] variant with channel setting [8, 16, 32, 64, 64] is utilized, whose last two FC layers are set to 512 and 256. Besides, an effective Cell based CNN [51] is also considered, termed Cell-CNN, whose normal cell and reduction cell are set to the best architecture founded in [46] as plotted in Figs. 6(b) and (d). To facilitate the training of the Cell based CNN, as suggested in [52], we make the neural network contain 6 cells, 16 filters in all the nodes, and an auxiliary head classifier with a loss weight of 0.4.

B. Benchmark Datasets

As listed in Table I, 17 widely used datasets are involved in the experiments. The first ten datasets including MNIST [48] as well as its variants [53] and Cifar10-bw [54] are used in the experiments of training AE and SAE. The next six datasets are collected from the UCR time series classification archive [55] and used in the experiments of training LSTMs. Besides, the last dataset Cifar10 [54] is used in the experiments of CNNs.

C. Compared Algorithms

Since the experiments are conducted on AE, SAE, LSTM, and CNNs, different comparison approaches have to be selected for different NNs.

1) Approaches for Training AE: The following comparison approaches are used for training AE:

- SGD [4] : Stochastic gradient descent;
- Adam [6] : A method for stochastic optimization;
- NSGA-II [38] : NSGA-II with SBX operator;

TABLE I
CHARACTERISTICS OF SEVENTEEN CHOSEN BENCHMARK DATASETS.

Benchmark	Dimension	Classes	Size of		Network
			training	test	
MNIST [48][1]	28∗28	10	60000	10000	
MNIST-basic [53][2]	28∗28	10	12000	50000	
MNIST-rot [53][2]	28∗28	10	12000	50000	
MNIST-back-image [53][2]	28∗28	10	12000	50000	**AE**
MNIST-back-rand [53][2]	28∗28	10	12000	50000	**/**
MNIST-rot-back-image [53][2]	28∗28	10	12000	50000	**SAE**
Rectangles [53][3]	28∗28	2	1200	50000	
Rectangles-image [53][3]	28∗28	2	12000	50000	
Convex [53][3]	28∗28	2	8000	50000	
Cifar10-bw [54][4]	32∗32	10	50000	10000	
FordA [55][5]	500	2	3601	1320	
TwoPatterns [55][5]	128	4	1000	4000	
ECG5000 [55][5]	140	5	500	4500	**RNN**
ElectricDevices [55][5]	96	7	8926	7711	
UWaveGestureLibraryX [55][5]	315	8	896	3582	
InsectWingbeatSound [55][5]	256	11	220	1980	
Cifar10 [54][4]	3∗32∗32	10	50000	10000	**CNN**

1. http://yann.lecun.com/exdb/mnist/.
2. https://sites.google.com/a/lisa.iro.umontreal.ca/public_static_twiki/variations-on-the-mnist-digits.
3. http://www-labs.iro.umontreal.ca/ lisa/icml2007data/.
4. http://www.cs.toronto.edu/ kriz/cifar.html.
5. https://www.cs.ucr.edu/ eamonn/time_series_data_2018/.

- NSGA-II (+gradient): Based on NSGA-II, update all offspring solutions by SGD for one step at each generation;
- NSGA-II (+gradient_0): Based on NSGA-II, update all offspring solutions by SSGD for one step at each generation;
- GEMONN (-SSGD): GEMONN without updating the Pareto optimal solutions by SSGD;
- GEMONN(single): GEMONN without optimizing sparsity by (16);
- GEMONN (-S): GEMONN without considering sparsity by (13)(14);
- GEMONN (-G): GEMONN without using gradient information by (8)(9)(11)(12).

SGD and Adam are the most popular gradient based approaches for training DNNs with millions of weights, which can be used as baselines to validate whether the performance of GEMONN is promising or not. NSGA-II, NSGA-II (+gradient), and NSGA-II (+gradient_0) are used to verify the effectiveness of the proposed genetic operator gSBX, since the other components of NSGA-II and GEMONN (i.e., solution evaluation, mating selection, and environmental selection) are totally the same. The remaining GEMONN (-SSGD), GEMONN(single), GEMONN (-S), and GEMONN (-G) are designed for ablation study, validating the effectiveness of the novel strategies proposed in GEMONN.

2) Approaches for Training SAE: The following state-of-the-art approaches are used for training SAE:

- SDAE [56]: Stacked denoising auto-encoders;
- SCAE [57]: Stacked contractive auto-encoders;
- SSAE [58]: Stacked sparse auto-encoders;
- EUDNN [11]: An evolutionary approach for training unsupervised DNNs.

As introduced in [11], SDAE, SCAE, and SSAE are three state-of-the-art gradient based approaches for training SAE, which incorporate various problem independent regularization terms to decrease the network complexity for alleviating overfitting. Besides, the innovative evolutionary approach EUDNN is considered as the state-of-the-art, which can effectively train

large-scale DNNs with the help of dimensionality reduction. As a result, the comparison between GEMONN and them is meaningful and convincing.

3) Approaches for Training LSTM and CNNs: Only SGD and Adam are considered as comparison approaches for training LSTM and CNNs, since these two approaches are very popular and effective for optimizing the large number of weights together. Besides, existing evolutionary approaches including EUDNN are not suitable for training them due to their poor performance caused by the curse of dimensionality.

D. Parameter Settings

The parameter settings for all the compared approaches are given below.

1) Parameters in Training AE: The number of hidden units of the AE is set to 500, which means that the total number of weights to be optimized is $(28 * 28 + 1) * 500$. Besides, the following settings are used in the compared approaches:

- SGD and Adam: Learning rate varies in $\{0.1, 0.01, 0.001\}$; batch size is set to 64; maximal number of training steps is set to 50*500 to be consistent with the evolutionary approaches;
- NSGA-II and its variants: Population size is set to 50; maximal generation is set to 500;
- GEMONN and its variants: Population size is set to 50; maximal generation is set to 500.

As a result, all the compared approaches have the same number of function evaluations (i.e., calculating the output and gradient over all the training samples) for fairness.

2) Parameters in Training SAE: The number of the units in each layer of SAE is empirically set to [$input$, 500, 400, 300, 200, 100, $output$], where $input$ is the dimension of features and $output$ is the number of classes of the dataset. The following parameter settings are used in the compared approaches, which also ensures that all the compared approaches consume the same number of function evaluations:

- SDAE, SCAE, and SSAE: Following the settings in [11], the SGD is chosen as the training approach, where the learning rate varies in $\{0.1, 0.01, 0.001\}$,and the batch size is set to 64; the maximal number of training steps is set to 50*500; the binary corrupted level of SDAE, the sparsity of SSAE, the coefficient of contractive term in SCAE are set to $\{0.1, 0.3, 0.5, 0.7\}$, respectively;
- EUDNN: Population size is set to 50; maximal iteration is set to 500; other parameters follow the original settings in the literature;
- GEMONN: Population size is set to 50; maximal generation is set to 500.

3) Parameters in Training LSTM and CNNs: The sizes of the input layer, hidden layer, and output layer of a 2-layer LSTM are set to $input$, 100, and $output$, respectively, and the parameters in CNNs are presented in Section IV-A. The following settings are used in the compared approaches:

- SGD and Adam: Learning rate varies in $\{0.1, 0.01, 0.001\}$; batch size is set to 64 for LSTM and CNNs; maximal number of training steps is set to 50*100 for LSTM and 50*500 for CNNs;
- GEMONN: Population size is set to 50; maximal generation is set to 100 for LSTM and 500 for CNNs.

All models and gradient based approaches are implemented in Pytorch [59], and GEMONN as well as variants of NSGA-II are implemented in Numpy [60]. These experiments were performed on a Windows 10 computer with an Intel i7-8700k CPU and an NVIDIA 1080 GPU. The source code of the proposed approach is provided by the first author and available from https://github.com/smartdonckey/GEMONN.

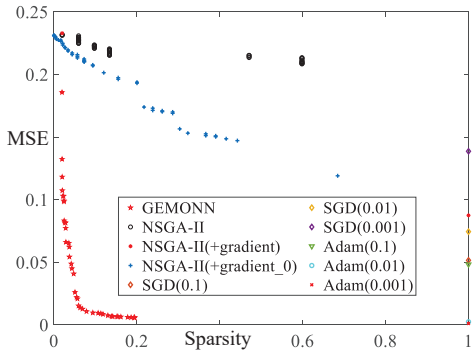


Fig. 7. Final solutions obtained by the compared approaches on the MNIST dataset, where "SGD (0.1)" denotes the SGD with a learning rate of 0.1, and so on.

E. Experimental Results on AE

Fig. 7 presents the solutions obtained by all the compared approaches in objective space. It can also be found that the performance of SGD and Adam highly relies on the learning rate, as the solutions obtained by SGD (0.1) outperform SGD (0.01) and SGD (0.001), while Adam (0.1) is also much worse than Adam (0.01) and Adam (0.001). Moreover, the solutions obtained by SGD and Adam are not sparse at all, which may affect the generalization of the model and result in overfitting.

On the other hand, the solutions obtained by NSGA-II have poor convergence in terms of MSE and we can not find any valuable solution due to the poor search ability of existing genetic operators. With the assistant of gradient information, the solutions obtained by NSGA-II (+gradient_0) are better than those obtained by NSGA-II, which verifies the importance of introducing gradient information into EAs. Nevertheless, the performance of NSGA-II and its variants is much worse than the gradient based approaches and the proposed GEMONN.

As for the proposed GEMONN, it exhibits quite competitive performance to SGD, Adam, and NSGA-II. On one hand, it can be seen that the solutions acquired by GEMONN are very promising, whose MSE is just worse than those obtained by Adam (0.001) and Adam (0.01) but much better than those obtained by the other approaches. On the other hand, the sparsity of the solutions acquired by GEMONN is similar to those obtained by NSGA-II but much better than those obtained by SGD and Adam. In short, the AE trained by the proposed GEMONN can have a relatively low MSE with satisfactory sparsity.

To investigate the convergence speed of the compared approaches, Fig. 8 depicts the decline in MSE of all the compared approaches, with the increase of the number of function evaluations, where the observation is consistent with Fig. 7. More specifically, GEMONN outperforms NSGA-II and most gradient based approaches, while it converges slower than Adam (0.01) and Adam (0.001).

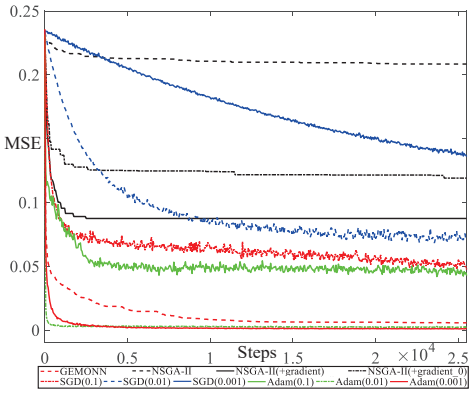


Fig. 8. Decline in MSE of the compared approaches on the MNIST dataset.

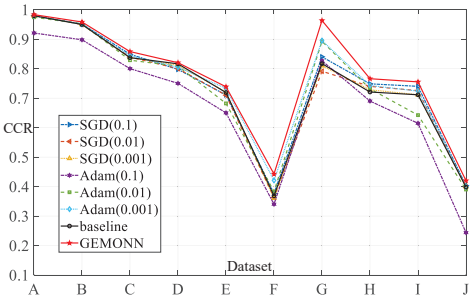
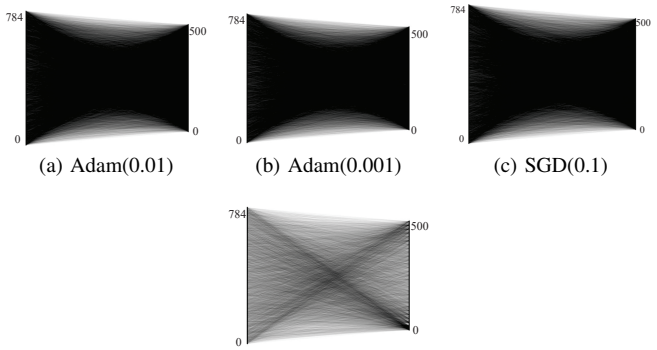


Fig. 9. The mean CCR values of the proposed GEMONN against SGD, Adam, and the baseline for training the AE tailed by a classification layer. The tick labels "A-J" denote the datasets of MNIST, MNIST-basic, MNIST-rot, MNIST-back-rand, MNIST-back-image, MNIST-rot-back-image, Rectangles, Rectangles-image, Convex, and Cifar10-bw, respectively.

As a result, solutions obtained by the proposed GEMONN have better sparsity but worse training loss than those obtained by Adam (0.01) and Adam (0.001). To further investigate which solutions are better for classification, a FC layer is tailed to the AE and trained in a supervised manner. Fig. 9 presents the mean Correct Classification Rate (CCR) [11] values on test set obtained by SGD, Adam, and the proposed GEMONN, averaged over 30 runs. Besides, a three-layer fully connected network with the same size is directly trained for the classification task and used as the baseline. By contrast, the results of NSGA-II are ignored since the obtained CCR values are quite poor. It can be observed from Fig. 9 that the mean CCR values obtained by GEMONN are better than all the other compared approaches including Adam (0.01) and Adam (0.001), though the solutions found by Adam (0.01) and Adam (0.001) are with lower MSE than GEMONN as shown in Fig. 8. Therefore, it can be inferred that the AEs trained by Adam overfit the training set to some extent, whereas the sparse AEs obtained by GEMONN can alleviate overfitting.

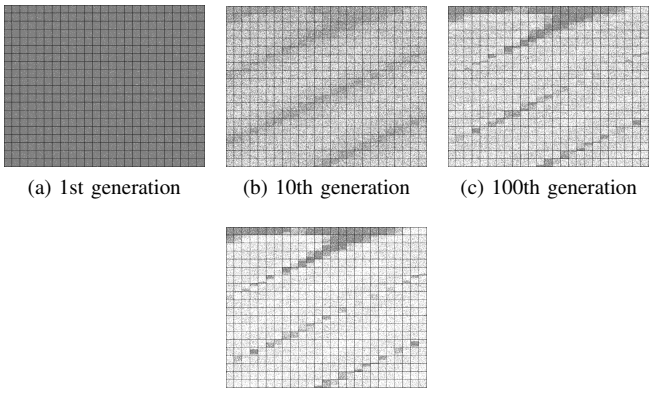
For visual observation of the difference between the sparsity of the AEs obtained by the compared approaches, Fig. 10 draws the connections between the input layer and hidden layer of the AEs obtained by Adam (0.01), Adam (0.001), SGD (0.1), and GEMONN on the MNIST dataset, where a darker color indicates a denser connection between weights. It is obvious that the AE obtained by GEMONN is much



(a) Adam(0.01) (b) Adam(0.001) (c) SGD(0.1)

(d) GEMONN

Fig. 10. Connections between the input layer and hidden layer of the AEs obtained by Adam (0.01), Adam (0.001), SGD (0.1), and GEMONN on the MNIST dataset. A darker color indicates a denser connection between weights.



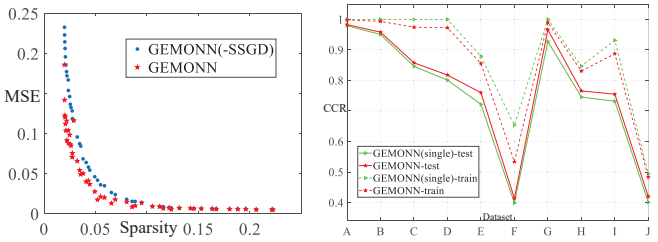
(a) 1st generation (b) 10th generation (c) 100th generation

(d) 500th generation

Fig. 11. Connections between the input layer and hidden layer of the AE obtained by GEMONN at different generations on the MNIST dataset. A gray pixel indicates an input unit is connected to a hidden unit, while a white pixel indicates there is no connection.

sparser than those obtained by the gradient based approaches. Moreover, Fig. 11 plots the connections between the input layer and hidden layer of the AE obtained by GEMONN at different generations on the MNIST dataset. For better observation, each figure in Fig. 11 consists of 500 binary images, where each image has 28×28 pixels and each pixel indicates whether the corresponding input unit is connected to a hidden unit or not. Besides, a gray pixel indicates a connection between the input unit and a hidden unit, while a white dot indicates there is no connection. As can be seen from Fig. 11, the AE is not sparse at the beginning, while it gradually becomes sparse as the population evolves. As a consequence, the proposed GEMONN can effectively control the sparsity of the obtained NN, and such a sparse nature is crucial to preventing the model from overfitting.

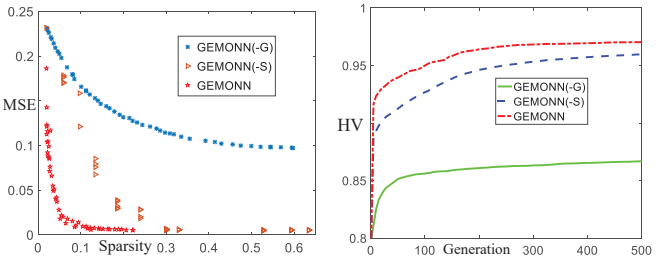
Afterwards, the effectiveness of the novel strategies proposed in GEMONN is verified by ablation study. Firstly, GEMONN is compared to GEMONN (-SSGD) on the MNIST dataset to verify the effectiveness of the SSGD based Pareto optimal solution update strategy. According to the solutions obtained by GEMONN and GEMONN (-SSGD) shown in



(a) Solutions obtained by GEMONN and GEMONN (-SSGD) on the MNIST dataset.

(b) Comparison between GEMONN and GEMONN(single) in terms of training and test CCR values.

Fig. 12. The validation of proposed strategies by comparing the performance of GEMONN and its variants.



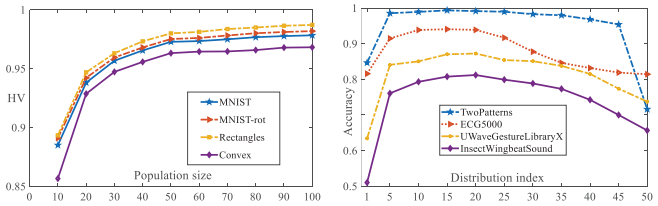
(a) Solutions obtained by GEMONN and its variants.

(b) Convergence profiles of HV of GEMONN and its variants.

Fig. 13. Comparison between the performance of GEMONN and its variants on the MNIST dataset.

Fig. 12(a), it is obvious that the solutions obtained by GEMONN dominate most solutions obtained by GEMONN (-SSGD), hence the effectiveness of SSGD is confirmed. Secondly, Fig. 12(b) depicts the effectiveness of considering the optimization of sparsity within GEMONN. As can be seen that the final solutions of GEMONN (single) have higher training CCR values than that of GEMONN on all ten datasets, but GEMONN can obtain better test CCR values than GEMONN (single). The crucial reason is that GEMONN (single) encounters the overfitting problem to some extent due to the optimization not considering the sparsity restriction. According to the above comparisons in Figs. 8, 9, and 12(b), it is indeed an effective way to mitigate the overfitting by considering sparsity into optimization. Thirdly, GEMONN is compared to GEMONN (-G) and GEMONN (-S) on the MNIST dataset to verify the effectiveness of the gradient information in the proposed genetic operator. As can be seen from Fig. 13(a), the solutions obtained by GEMONN are superior over those obtained by GEMONN (-G) and GEMONN (-S) in terms of both the sparsity and MSE; besides, GEMONN converges obviously faster than GEMONN (-G) and GEMONN (-S) in terms of the hypervolume indicator (HV) [61] as shown in Fig. 13(b). Therefore, the effectiveness of the novel strategies proposed in GEMONN can be verified.

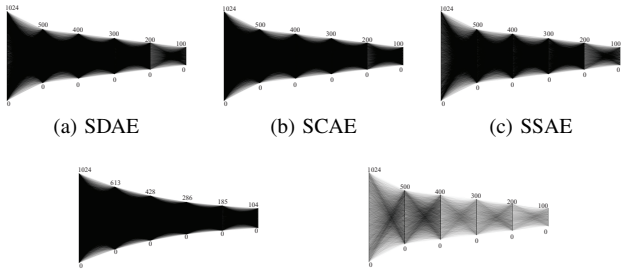
Lastly, a necessary parameter sensitivity analysis is made for population size Pop and distribution index η in GEMONN. Fig. 14(a) shows the HV values obtained by GEMONN on four datasets MNIST, MNIST-rot, Rectangles, and Convex with different population sizes, where it can be seen that GEMONN



(a) Sensitivity to population size

(b) Sensitivity to distribution index

Fig. 14. Parameter sensitivity analysis to population size and distribution index.



(a) SDAE (b) SCAE (c) SSAE

(d) EUDNN (e) GEMONN

Fig. 15. Connections between the layers of the SAEs obtained by SDAE, SCAE, SSAE, EUDNN, and GEMONN on the Cifar10-bw dataset. A darker color indicates a denser connection between weights.

can obtain a relatively good performance when the population size is larger than 50. It is not necessary to set Pop to 100 for a slightly higher HV value at the expense of a doubled number of evaluations. The results of sensitivity to distribution index η in GEMONN on four datasets TwoPatterns, ECG5000, UWaveGestureLibraryX, and InsectWingbeatSound are plotted in Fig. 14(b). From this figure, it can be seen that the proposed GEMONN can obtain the best performance with the distribution index being equal to 15 or 20. Thus, the recommended η =20 in [26] is also suitable and effective for GEMONN.

F. Experimental Results on SAE

Table II lists the CCR values of the SAEs obtained by GEMONN, EUDNN, SSAE, SCAE, SDAE on the nine Datasets, where each cell in the table records the mean and standard deviation of the CCR values obtained in 30 independent runs. In addition, the Wilcoxon rank sum test [62] at a significance level of 0.05 is employed to perform statistical analysis on the experimental results, where the symbols "+", "−", and "≈" indicate that the result obtained by GEMONN is significantly better than, significantly worse than, and statistically similar to that of the corresponding peer competitors, respectively.

It is clearly presented in Table II that the proposed GEMONN gains the best results on 7 out of 9 datasets, which is followed by EUDNN and SSAE obtaining 1 best result. In terms of the Wilcoxon rank sum test, the numbers of datasets where GEMONN performs statistically better than EUDNN, SSAE, SCAE, and SDAE are 6, 7, 8, and 9, respectively. Furthermore, Fig. 15 plots the connections between the layers

TABLE II
CORRECT CLASSIFICATION RATE (CCR) VALUES OF THE SAES OBTAINED BY GEMONN, EUDNN, SSAE, SCAE, SDAE ON TEN DATASETS. BEST MEAN VALUES ARE HIGHLIGHTED IN BOLDFACE. THE SYMBOLS "+", "−", AND "≈" DENOTE GEMONN IS STATISTICALLY BETTER THAN, WORSE THAN, AND SIMILAR TO THAT OF THE COMPARISON APPROACH, RESPECTIVELY.

Benchmark	GEMONN	EUDNN	SSAE	SCAE	SDAE
MNIST	**0.9879(0.0017)**	$0.9812(0.0018)^{+}$	$0.9828(0.0079)^{+}$	$0.9827(0.0009)^{+}$	$0.9822(0.0049)^{+}$
MNIST-basic	**0.9775(0.0012)**	$0.9634(0.0017)^{+}$	$0.9725(0.0046)^{+}$	$0.9613(0.0019)^{+}$	$0.9565(0.0032)^{+}$
MNIST-rot	**0.9173(0.0013)**	$0.8871(0025)^{+}$	$0.8857(0.0041)^{+}$	$0.8975(0.0043)^{+}$	$0.8765(0.0081)^{+}$
MNIST-back-image	0.8771(0.0064)	**0.8932(0.0074)**$^{-}$	$0.8721(0.0073)^{\approx}$	$0.7984(0.0065)^{+}$	$0.8637(0.0033)^{+}$
MNIST-back-rand	0.8609(0.0024)	$0.8857(0.0012)^{-}$	**0.8862(0.0087)**$^{-}$	$0.8612(0.0073)^{\approx}$	$0.7974(0.0076)^{+}$
MNIST-rot-back-image	**0.4975(0.0033)**	$0.4858(0.0035)^{+}$	$0.4712(0.0013)^{+}$	$0.4683(0.0015)^{+}$	$0.4513(0.0025)^{+}$
Rectangles	**0.9757(0.0011)**	$0.9692(0.0021)^{+}$	$0.9457(0.0023)^{+}$	$0.9662(0.0021)^{+}$	$0.9453(0.0024)^{+}$
Rectangles-image	**0.7886(0.0027)**	$0.7678(0.0034)^{+}$	$0.7764(0.0017)^{+}$	$0.7677(0.0016)^{+}$	$0.7613(0.0026)^{+}$
Convex	**0.8186(0.0049)**	$0.8135(0.0051)^{\approx}$	$0.8087(0.0057)^{+}$	$0.7891(0.0016)^{+}$	$0.8017(0.0032)^{+}$
Cifar10-bw	0.4817(0.0052)	$0.4832(0.0013)^{\approx}$	$0.4753(0.0039)^{+}$	**0.4911(0.0107)**$^{-}$	$0.4694(0.0152)^{+}$
$+/-/\approx$		6/2/2	8/1/1	8/1/1	10/0/0

TABLE III
AVERAGE RUNTIME OF THE PROPOSED GEMONN, EUDNN, AND THREE APPROACHES TRAINED BY SGD (INCLUDING SSAE, SCAE, AND SDAE) ON TEN DATASETS. NOTE THAT '$>$ 5h' MEANS THAT THE RUNTIME IS SIGNIFICANTLY LARGER THAN 5 HOURS FOR ONE RUN.

Dataset	GEMONN	EUDNN	SGD		
			SSAE	SCAE	SDAE
MNIST	2158.4s	$>$5h	1226.8s	1235.6s	1270.8s
MNIST-basic	2163.4s	$>$5h	1232.4s	1233.7s	1266.8s
MNIST-rot	2179.3s	$>$5h	1229.9s	1239.8s	1264.3s
MNIST-back-image	2166.3s	$>$5h	1210.9s	1229.4s	1241.6s
MNIST-back-rand	2211.7s	$>$5h	1219.1s	1240.4s	1262.8s
MNIST-rot-back-image	2138.3s	$>$5h	1226.2s	1231.9s	1235.9s
Rectangles	2158.8s	$>$5h	1212.4s	1227.6s	1243.9s
Rectangles-image	2162.4s	$>$5h	1231.8s	1235.2s	1262.8s
Convex	2201.6s	$>$5h	1227.7s	1229.2s	1258.7s
Cifar10-bw	2681.1s	$>$5h	1338.5s	1356.4s	1384.4s

of the SAEs obtained by SDAE, SCAE, SSAE, EUDNN, and GEMONN on the Cifar10-bw dataset. Obviously, the SAE obtained by GEMONN is much sparser than those obtained by the other approaches. Therefore, it can be concluded that the SAE obtained by GEMONN has not only better classification performance but also much lower complexity than those obtained by the other compared approaches. In short, the proposed GEMONN shows significant superiority over the state-of-the-art approaches in training SAE.

Next, Table III lists the runtimes of the proposed GEMONN, EUDNN, and three approaches trained by SGD on the ten datasets. It can be found that the efficiency of EUDNN is the worst due to its time expensive evaluation, and the efficiency of GEMONN is better than EUDNN but worse than SGD, though it has the same time complexity to SGD as analyzed in Section III-C. This is mainly because SGD is implemented by Pytorch on GPU, while the proposed GEMONN is implemented on CPU since some complex steps in EA cannot be easily performed on GPU, which is a common difficulty of most EA based approaches including EUDNN.

G. Experimental Results on LSTM and CNNs

The comparison between the proposed GEMONN and gradient based approaches in training LSTM is presented in Table IV. The results are calculated based on 30 independent runs, and the Wilcoxon rank sum test is also employed. Note

TABLE IV
CORRECT CLASSIFICATION RATE (CCR) VALUES OF THE LSTMS OBTAINED BY GEMONN, ADAM, AND SGD ON SIX DATASETS. BEST VALUES ARE HIGHLIGHTED IN BOLDFACE. THE SYMBOLS "+", "−", AND "≈" DENOTE GEMONN IS STATISTICALLY BETTER THAN, WORSE THAN, AND SIMILAR TO THAT OF THE COMPARISON APPROACH, RESPECTIVELY.

Benchmark	CCR	GEMONN	Adam (0.001)	Adam (0.01)	SGD (0.01)	SGD (0.1)
FordA	$best$	**1.0000**	0.9848	0.7977	0.7364	**1.0000**
	avg	**0.9994**	0.9702^{+}	0.7737^{+}	0.7192^{+} s	0.9973^{+}
	Std	0.0004	0.0065	0.0120	0.0108	0.0011
TwoPatterns	$best$	**0.9975**	0.9748	0.9765	0.7480	0.9828
	avg	**0.9914**	0.9701^{+}	0.9744^{+}	0.7152^{+}	0.9799^{+}
	Std	0.0112	0.0028	0.0011	0.0265	0.0021
ECG5000	$best$	0.9409	0.9409	**0.9607**	0.9153	0.9389
	avg	0.9387	$0.9372^{\approx}$	**0.9531**$^{-}$	0.9108^{+}	0.9334^{+}
	Std	0.0017	0.0089	0.0150	0.0032	0.0042
Electric--Devices	$best$	**0.9662**	0.8771	0.9598	0.4970	0.8605
	avg	**0.9547**	0.8701^{+}	0.9501^{+}	0.4672^{+}	0.8505^{+}
	Std	0.0075	0.0094	0.0091	0.0195	0.0086
UWaveGesture--LibraryX	$best$	0.8816	0.8451	0.8068	0.7672	**0.8833**
	avg	0.8724	0.8414^{+}	0.7939^{+}	0.7495^{+}	**0.8772**$^{-}$
	Std	0.0083	0.0029	0.0105	0.0084	0.0043
InsectWing--beatSound	$best$	**0.8197**	0.7990	0.7884	0.8020	0.8172
	avg	**0.8121**	0.7899^{+}	0.7732^{+}	0.7881^{+}	$0.8087^{\approx}$
	Std	0.0103	0.0142	0.0081	0.0082	0.0055
$+/-/\approx$			5/0/1	5/1/0	6/0/0	4/1/1

that due to the poor performance of Adam (0.1) and SGD (0.001), only the results of Adam (0.001), Adam (0.01), SGD (0.01), and SGD (0.1) are listed in the table. As can be seen, the overall performance of GEMONN is better than the gradient based approaches, having achieved the largest average CCR values on 4 out of 6 datasets. Besides, Adam (0.01) and SGD(0.1) obtain the largest average CCR values on the remaining 2 datasets. In short, the performance of GEMONN is significantly better than Adam (0.001) as well as SGD (0.01) and competitive to Adam (0.01) and SGD (0.1). In addition, it is worth to note that the performance of Adam is sensitive to the learning rate since Adam (0.01) outperforms Adam (0.001) on ECG5000 and ElectricDevicess, but Adam (0.001) outperforms Adam (0.01) on FordA and UWaveGestureLibraryX. By contrast, the GEMONN does not contain the learning rate or any other sensitive parameter, which is also a superiority of evolutionary approaches over gradient based approaches.

The comparison between GEMONN and gradient based

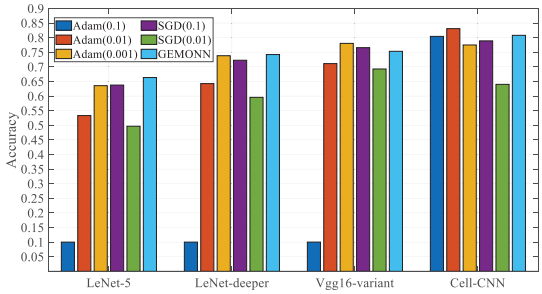


Fig. 16. The mean Cifar10 test accuracy of the compared approaches on four different CNNs over 30 independent runs.

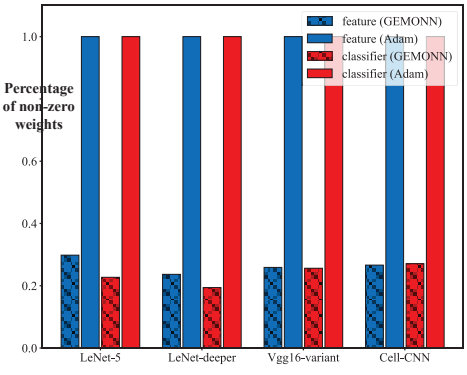


Fig. 17. The percentage of non-zero weights in feature (extraction) part and classifier part included in four CNNs trained by GEMONN and Adam.

approaches in training CNNs on Cifar10 dataset is shown in Fig. 16, where the mean test accuracy over 30 independent runs is plotted. Note that the result of SGD (0.001) is not given due to its poor performance. From this figure, it can be observed that GEMONN outperforms gradient based approaches on LeNet-5, and the performance of GEMONN is also slightly better than the opponents on LeNet-deeper. However, Adam (0.001) and Adam (0.01) get the best performance on Vgg16-variant as well as Cell-CNN, while GEMONN obtains a competitive performance. It is worth noting that the performance of GEMONN degenerates with the architecture of CNN becoming complex since the proportion of convolutional operation parameters increases from 5% to 98%. Essentially, the weights of a whole CNN can be divided into two parts, including the feature extraction (convolutional) part and the classifier (FC) part. Compared to weights of the FC layers, convolutional operations have much fewer weights due to the intrinsic merit of local connection and weight-sharing, where most weights of convolutional layers should not be optimized to zero during the training process [63]. For a deep insight into the weights of each part obtained by GEMONN, Fig. 17 depicts the percentage of non-zero weights in four CNNs obtained by GEMONN and Adam. It is obvious that the weights of convolutional layers obtained by the proposed approach are much sparser than that of Adam, hence the final performance of GEMONN may be slightly worse than the best gradient based approaches. Nevertheless, EMONN still holds a competitive performance compared to SGD and Adam.

V. Conclusions and Future Work

In this paper, we have proposed a gradient guided evolutionary approach to train DNNs. The proposed GEMONN focuses on designing a powerful genetic operator gSBX to directly optimize the weights of DNNs, by means of taking advantage of the gradient information. Moreover, the network sparsity is also optimized together with the training loss, which can reduce the network complexity and alleviate overfitting.

In the experiments, four widely used neural networks AE, SAE, LSTM, and CNNs have been utilized to validate the effectiveness of the proposed approach. For the experiments on AEs, SAE, and LSTM, GEMONN exhibits better performance than the counterparts, including traditional EAs, SGD, Adam, and other state-of-the-art approaches. Besides, the NNs trained by GEMONN have much better sparsity than that of gradient based approaches, which can not only alleviate the overfitting but also benefit many deep learning tasks. For example, the sparser NNs become especially crucial for image classification [64], speech recognition [65], and object detection [66] to reduce the high memory and runtime cost when deploying in the application on computationally constrained architectures like mobile devices. Although GEMONN holds a slightly worse performance than SGD and Adam in training CNNs due to the difficulty brought by sparsity during the training process, it can be believed that the proposed GEMONN is a promising and effective approach for training DNNs.

This work has shown the promising prospect of combing gradient descent and EAs in deep learning, and we would like to further explore this issue from the following aspects: Firstly, it is desirable to apply gradient information to the evolutionary process in a more comprehensive way, which is not limited to the genetic operators. Secondly, evolutionary approaches are often criticized for the high complexity, therefore it is advisable to decrease the runtime by efficient strategies (e.g., using surrogates [67]). Lastly, the robustness is also crucial to neural networks [68], hence designing better strategies to incorporate robustness into EAs is also very important.

References

[1] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of the 2016 International Conference on Machine Learning*, 2016, pp. 2014–2023.

[2] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proceedings of the 2016 European Conference on Computer Vision*. Springer, 2016, pp. 21–37.

[4] H. Robbins and S. Monro, "A stochastic approximation method," *The Annals of Mathematical Statistics*, pp. 400–407, 1951.

[5] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, pp. 26–31, 2012.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

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

[8] X. Cui, W. Zhang, Z. Tüske, and M. Picheny, "Evolutionary stochastic gradient descent for optimization of deep neural networks," in *Proceedings of the 2018 Advances in Neural Information Processing Systems*, 2018, pp. 6048–6058.

[9] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, 2008.

[10] J. Liu, M. Gong, Q. Miao, X. Wang, and H. Li, "Structure learning for deep neural networks based on multiobjective optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2450–2463, 2017.

[11] Y. Sun, G. G. Yen, and Z. Yi, "Evolving unsupervised deep neural networks for learning meaningful representations," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 89–103, 2018.

[12] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, no. 2, pp. 219–269, 1995.

[13] B. Babu and R. Angira, "Optimization of non-linear functions using evolutionary computation," in *Proceedings of the 12th ISME International Conference on Mechanical Engineering, India*. Citeseer, 2001, pp. 153–157.

[14] S.-K. Wang, J.-P. Chiou, and C.-W. Liu, "Non-smoothn-convex economic dispatch by a novel hybrid differential evolution algorithm," *IET Generation, Transmission & Distribution*, vol. 1, no. 5, pp. 793–803, 2007.

[15] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 580–593, 2002.

[16] L. D. Whitley *et al.*, "The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best." in *Proceedings of the 1989 ICGA*, vol. 89. Fairfax, VA, 1989, pp. 116–123.

[17] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms." in *Proceedings of the 1989 IJCAI*, vol. 89, 1989, pp. 762–767.

[18] D. Whitley, T. Starkweather, and C. Bogart, "Genetic algorithms and neural networks : optimizing connections and connectivity," *Parallel computing*, vol. 14, no. 3, pp. 347–361, 1990.

[19] E. Ronald and M. Schoenauer, "Genetic lander: An experiment in accurate neuro-genetic control," in *Proceedings of the 1994 International Conference on Parallel Problem Solving from Nature*. Springer, 1994, pp. 452–461.

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

[21] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131–162, 2007.

[22] M. Gong, J. Liu, H. Li, Q. Cai, and L. Su, "A multiobjective sparse feature learning model for deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 12, pp. 3263–3277, 2015.

[23] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multiobjective optimization based on a competitive swarm optimizer," *IEEE Transactions on Cybernetics*, 2019, in press.

[24] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," *arXiv preprint arXiv:1712.06567*, 2017.

[25] C. He, L. Li, Y. Tian, X. Zhang, R. Cheng, Y. Jin, and X. Yao, "Accelerating large-scale multiobjective optimization via problem reformulation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 949–961, 2019.

[26] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 2, pp. 115–148, 1995.

[27] Y. Jin, T. Okabe, and B. Sendhoff, "Neural network regularization and ensembling using multi-objective evolutionary algorithms," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, 2004.

[28] Y. Tian, X. Zhang, C. Wang, and Y. Jin, "An evolutionary algorithm for large-scale sparse multi-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, 2019.

[29] Y. Wu, Y. Zhang, X. Liu, Z. Cai, and C. Yaoming, "A multiobjective optimization-based sparse extreme learning machine algorithm," *Neurocomputing*, vol. 317, pp. 88–100, 2018.

[30] Q. Zhang and H. Li, "MOEA/D: A multi-objective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[31] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[32] D. C. Mocanu, E. Mocanu, P. Stone, P. H. Nguyen, M. Gibescu, and A. Liotta, "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science," *Nature Communications*, vol. 9, no. 1, pp. 1–12, 2018.

[33] J. Koutník, G. Cuccu, J. Schmidhuber, and F. Gomez, "Evolving large-scale neural networks for vision-based reinforcement learning," in *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2013, pp. 1061–1068.

[34] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," *arXiv preprint arXiv:1703.01041*, 2017.

[35] H. Kita, I. Ono, and S. Kobayashi, "Multi-parental extension of the unimodal normal distribution crossover for real-coded genetic algorithms," *Transactions of the society of Instrument and Control Engineers*, vol. 36, no. 10, pp. 875–883, 2000.

[36] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[37] N. Hitomi and D. Selva, "A classification and comparison of credit assignment strategies in multiobjective adaptive operator selection," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 294–314, 2016.

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[39] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*. ACM, 2008, pp. 1096–1103.

[40] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[41] L. Rachmawati and D. Srinivasan, "Multiobjective evolutionary algorithm with controllable focus on the knees of the Pareto front," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 810–824, 2009.

[42] J. Branke, K. Deb, H. Dierolf, and M. Osswald, "Finding knees in multi-objective optimization," in *Proceedings of the 2004 International Conference on Parallel Problem Solving from Nature*, 2004, pp. 722–731.

[43] Y. Mu, W. Liu, X. Liu, and W. Fan, "Stochastic gradient made stable: A manifold propagation approach for large-scale optimization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 2, pp. 458–471, 2016.

[44] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," in *Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics*, 2018, pp. 415–419.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[47] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

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

[49] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8609–8613.

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

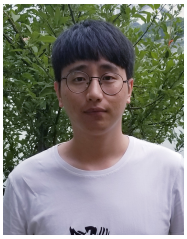
[51] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8697–8710.

[52] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "Nsga-net: neural architecture search using multi-objective genetic algorithm," in *Proceedings of the 2019 Genetic and Evolutionary Computation Conference*, 2019, pp. 419–427.

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

[54] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2012.

[55] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML, "The ucr time series classification archive," October 2018.
[56] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. Dec, pp. 3371–3408, 2010.
[57] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Omnipress, 2011, pp. 833–840.
[58] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.
[59] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.
[60] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.
[61] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, 2006.
[62] R. G. Steel, J. H. Torrie *et al.*, *Principles and procedures of statistics, a biometrical approach*. McGraw-Hill Kogakusha, Ltd., 1980, no. Ed. 2.
[63] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations*, 2018.
[64] M. D. Collins and P. Kohli, "Memory bounded deep convolutional networks," *arXiv preprint arXiv:1412.1442*, 2014.
[65] D. Yu, F. Seide, G. Li, and L. Deng, "Exploiting sparseness in deep neural networks for large vocabulary speech recognition," in *Proceeding of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4409–4412.
[66] J. Park, S. Li, W. Wen, P. T. P. Tang, H. Li, Y. Chen, and P. Dubey, "Faster cnns with direct sparse convolutions and guided pruning," *arXiv preprint arXiv:1608.01409*, 2016.
[67] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.
[68] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy*, 2017, pp. 39–57.



Shangshang Yang received the B.Sc. degree from Anhui University, Hefei, China, in 2017, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Anhui University, Hefei, China.

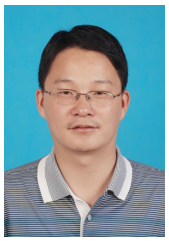
His current research interests include evolutionary multi-objective optimization and neural architecture search.

Ye Tian received the B.Sc., M.Sc., and Ph.D. degrees from Anhui University, Hefei, China, in 2012, 2015, and 2018, respectively.

He is currently an Associate Professor with the Institutes of Physical Science and Information Technology, Anhui University, Hefei, China. His current research interests include evolutionary computation and its applications. He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award and the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award.

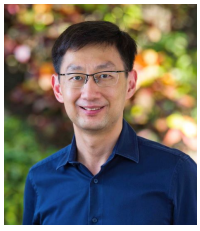
Cheng He received the B.Eng. degree from Wuhan University of Science and Technology, Wuhan, China, in 2012, and the Ph.D. degree from Huazhong University of Science and Technology, Wuhan, China, in 2018.

He is currently a Research Assistant Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. His current research interests include model-based evolutionary algorithms, multiobjective optimization, large-scale optimization, deep learning, and their applications. He is a recipient of the SUSTech Presidential Outstanding Postdoctoral Award from Southern University of Science and Technology, and the leading Guest Editor for Special Issue: Emerging Topics in Evolutionary Multiobjective Optimization of the Complex and Intelligent Systems.



Xingyi Zhang (SM'18) received the B.Sc. degree from Fuyang Normal College, Fuyang, China, in 2003, and the M.Sc. and Ph.D. degrees from Huazhong University of Science and Technology, Wuwan, China, in 2006 and 2009, respectively.

He is currently a Professor with the School of Computer Science and Technology, Anhui University, Hefei, China. His current research interests include unconventional models and algorithms of computation, multi-objective optimization, and membrane computing. He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award and the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award.



Kay Chen Tan (SM'08-F'14) received the B.Eng. (First Class Hons.) degree in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively. He is currently a Chair Professor of the Department of Computing at the Hong Kong Polytechnic University, Hong Kong. He has published over 200 refereed articles and six books, and holds one U.S. patent on surface defect detection.

Prof. Tan is currently the Vice-President (Publications) of IEEE Computational Intelligence Society, USA. He has served as the Editor-in-Chief of IEEE Transactions on Evolutionary Computation from 2015-2020 and IEEE Computational Intelligence Magazine from 2010-2013, and currently serves as the Editorial Board Member of over 10 journals. He is currently an IEEE Distinguished Lecturer Program (DLP) speaker and Chief Co-Editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.

Yaochu Jin (SM'02-F'16) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Bochum, Germany, in 2001.

He is currently a Distinguished Chair Professor in Computational Intelligence, Department of Computer Science, University of Surrey, Guildford, U.K., where he heads the Nature Inspired Computing and Engineering Group. He was a Finland Distinguished Professor and a Changjiang Distinguished Visiting Professor. He has (co)authored over 300 peer-reviewed journal and conference papers and been granted eight patents on evolutionary optimization.

He is the Editor-in-Chief of the IEEE Transactions on Cognitive and Developmental Systems and Complex & Intelligent Systems. He is also an Associate Editor or Editorial Board Member of the IEEE Transactions on Evolutionary Computation, IEEE Transactions on Cybernetics, Evolutionary Computation, and Soft Computing. He was an IEEE Distinguished Lecturer (2013-15, 2017-19). He is the recipient of the 2014, 2016, and 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award, the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, and the Best Paper Award of the 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. He has been named a Highly Cited Researcher for 2019 and 2020 by the Web of Science group. He is a Fellow of IEEE.