Research Article | Vol. 27, No. 26 / 23 December 2019 / Optics Express 37150
Optics EXPRESS

Efficient training and design of photonic neural network through neuroevolution

TIAN ZHANG,[1] JIA WANG,[1] YIHANG DAN,[1] YUXIANG LANQIU,[1] JIAN DAI,[1] XU HAN,[2] XIAOJUAN SUN,[3] AND KUN XU[1,*]

[1]State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]Huawei Technologies Co., Ltd, Shenzhen 518129, Guangdong, China
[3]School of Science, Beijing University of Posts and Telecommunications, Beijing 100876, China
[*]xukun@bupt.edu.cn

Abstract: Recently, optical neural networks (ONNs) integrated into photonic chips have received extensive attention because they are expected to implement the same pattern recognition tasks in electronic platforms with high efficiency and low power consumption. However, there are no efficient learning algorithms for the training of ONNs on an on-chip integration system. In this article, we propose a novel learning strategy based on neuroevolution to design and train ONNs. Two typical neuroevolution algorithms are used to determine the hyper-parameters of ONNs and to optimize the weights (phase shifters) in the connections. To demonstrate the effectiveness of the training algorithms, the trained ONNs are applied in classification tasks for an iris plants dataset, a wine recognition dataset and modulation formats recognition. The calculated results demonstrate that the accuracy and stability of the training algorithms based on neuroevolution are competitive with other traditional learning algorithms. In comparison to previous works, we introduce an efficient training method for ONNs and demonstrate their broad application prospects in pattern recognition, reinforcement learning and so on.

1. Introduction

Artificial neural networks (ANNs), in particular deep learning [1], have attracted a great amount of research attention for an impressively large number of applications, such as image processing [2], natural language processing [3], acoustical signal processing [4], time series processing [5], self-driving vehicles [6], games [7], and robots [8]. It should be noted that the training of the ANNs with deep hidden layers, especially for convolutional neural networks (CNNs) and recurrent neural networks (RNNs), for example AlexNet [9], VGGNet [10], GoogLeNet [11], ResNet [12] and long short-term memory [13], typically demands significant computational time and resources [1]. Thus, various electronic special-purpose platforms based on graphical processing units (GPUs) [14], field-programmable gate arrays (FPGAs) [15] and application-specific integrated circuits (ASICs) [16] were invented to accelerate the training and inference process of deep learning. On the other hand, in order to obtain general artificial intelligence, some brain-inspired chips including IBM TrueNorth [17], Intel Loihi [18], and SpiNNaker [19] were designed by imitating the structure of the human brain. However, even both energy efficiency and speed were improved, it was difficult for the performances of brain-inspired chips to compete with the state of the art of deep learning [20]. In the recent years, optical computing had been demonstrated to be an effective alternative to traditional electronic computing architectures, and it was expected to alleviate the bandwidth bottlenecks and power consumption disadvantages in electronics [21]. For example, new photonic approaches for spiking neurons and scalable network architecture based on excitable lasers, broadcast-and-weight protocol, and reservoir computing, had been reported [22–24]. Although an ultrafast spiking response was achieved, these neuromorphic photonic systems also faced the similar challenges in performance and integration issues [24].

#374781 https://doi.org/10.1364/OE.27.037150
Journal © 2019 Received 8 Aug 2019; revised 3 Nov 2019; accepted 5 Nov 2019; published 9 Dec 2019

Optics EXPRESS

Y. Shen *et al.* proposed a photonic implementation of ANNs and pointed out this integrated optical neural networks (ONNs) architecture could be used to improve the computational speed and consumption in comparison to conventional computers [25]. In addition, the photonic chips that could implement functions similar to those of CNNs [26] and RNNs [27] were outlined based on ONNs. X. Lin *et al.* introduced an all-optical diffractive deep neural network that could complete feature detection and image recognition [28]. Large scale and quantum ONNs were also proposed to enhance the overall capabilities of the on-chip optical computing [29,30]. The design of fault-tolerant ONNs was analyzed to offer guidelines for fabrication [31]. Besides, several different nonlinear activation functions based on electronic and photonic elements were proposed in ONNs to enhance their nonlinear capability [32,33]. It should be noted that the back-propagation (BP) and stochastic gradient descent (SGD) are currently being used as the training methods for ONNs [25]. However, the BP and SGD training strategies were found to be difficult to implement in integrated optical chips, thus determination of the weights in ONNs were generally mapped from the pre-trained results on a digital computer [25]. Obviously, this training method was inefficient due to the restricted accuracy of model representation and the loss of advantages in speed and energy [34].

In order to self-learn the weights in ONNs, an alternative approach was proposed that could directly obtain the gradient of the weights based on the forward propagation and finite difference method [25]. Similarly, a self-learning photonic signal processor trained by a modified SGD was demonstrated experimentally to implement a tunable filter, an optical switching and a descrambler [35]. T. W. Hughes *et al.* believed that this technique was a brute force method and had high computational complexity for large systems [36]. Thus, they proposed a novel training method to compute the gradients of weights by using the *in situ* intensity measurements and adjoint variable method (AVM) [36]. Here, the determinations of the gradients in ONNs were converted to an inverse design and sensitivity analysis process for photonic circuits [36]. In addition, some training strategies and tricks used in deep learning, such as adaptive moment estimation and initialization scheme were also applied in the training of ONNs [32,34]. The training methods proposed by T. W. Hughes inspire us to think that the learning process of ONNs can be converted to the inverse design problems that are solved by using gradient-based methods or gradient free methods [37,38]. Apart from gradient-based methods (such as AVM), gradient free methods, for example genetic algorithms (GA) and particle swarm optimization (PSO), can also be applied in the inverse design of photonic devices [39–42]. Additionally, as an alternative approach to training of ANNs, neuroevolution, which is derived from the evolution process that imitates the biological brain, is a typical gradient free method based on evolutionary algorithms [43]. In comparison to gradient-based methods, such as SGD and AVM, neuroevolution can determine the weights in the connections and optimize the network architectures of ANNs, the hyper-parameters of activation functions and the rules for the learning algorithm due to the advantages of diversity, parallelization and architecture search [43]. More interestingly, it has been proven that neuroevolution performs competitively with the learning algorithms in deep reinforcement learning (DRL), such as policy search and deep Q-network [44]. On a subset of Atari games, neuroevolution even outperforms the training speed of the best training algorithms in DRL because of its parallelizable [45]. Obviously, the possibility of training ONNs by neuroevolution, not only provides a novel learning method for ONNs, it is also expected to improve the training efficiency for traditional DRL.

In this article, we propose a novel learning strategy to design and train ONNs based on neuroevolution. Two typically evolutionary algorithms, GA and PSO are used to determine the hyper-parameters of ONNs and to optimize the weights (phase shifters) in the connections. In order to demonstrate the effectiveness of neuroevolution, the trained ONNs are applied in classification tasks for different datasets. The calculated results demonstrate that these simple

Optics EXPRESS

training algorithms are competitive with traditional learning algorithms (such as SGD and AVM), and they have broad application prospects in pattern recognition and DRL.

2. Training methods based on neuroevolution

As shown in Fig. 1(a), the network architecture of ANNs imitates the structure of a biological neural network, which includes a large number of neurons and connections layer by layer [1]. It should be noted that, although ANNs are brain-inspired, there are significant differences in the network structure, learning method, information transmission and encoding rule between ANNs and the biological brain [24]. For example, the information propagated between the biological neurons is conveyed by synapses and the code scheme of it is spike timing [24]. Due to the temporal coding and event-driven manner, spiking neural networks (SNNs) are the closest approximation to a biological neural network that has computational advantages in energy efficiency and high speed [46]. Nevertheless, although SNNs have been applied in many areas (such as image and speech recognition [47,48]), in terms of actual performance and application range, it is difficult for them to compete with deep learning. In order to combine the advantages of ANNs and optical computing, as shown in Fig. 1(b), a photonic implementation of ANNs, which includes optical interference units (OIU) and optical nonlinearity units (ONU) has been outlined to complete the speech recognition task, experimentally [25]. As seen in Fig. 1(b), the network architecture of ONNs strictly imitates that of ANNs, namely, the OIU and ONU implement matrix multiplication and nonlinear transform functions, respectively [25]. As shown in Fig. 1(c), the physical implementation of OIUs in ONNs are composed of programmable Mach–Zehnder interferometers (MZIs) whose phase shifters are controlled by external voltage to construct any unitary matrix [25]. While the ONUs in ONNs can be realized by means of the strong nonlinear effects of two dimensional (2D) materials, such as graphene and sulfide [49]. However, the integrated fabrication of 2D materials into the silicon waveguide is complicated and the strength of the nonlinear effect is weaken when the 2D materials are integrated into the waveguide [34]. In order to alleviate the shortcomings of optical nonlinearity integrated in the waveguide, an electro-optic hardware platform, which implements various nonlinear activation functions with a low activation threshold is illustrated [34].

Training the neural network based on optimization algorithms is a critical requirement and procedure regardless of ANNs or ONNs [36]. The BP and SGD methods are representative gradient-based optimization methods that are used to compute the gradients of the model parameters based on the chain rule [36]. The derivative of the loss is calculated backwards from the output layer to the input layer of ANNs. The update rule of model parameters is given by:

$$\omega \leftarrow \omega - \eta(\alpha \frac{\partial R(\omega)}{\partial \omega} + \frac{\partial L}{\partial \omega}) \tag{1}$$

where η is the learning rate, L is a loss function that measures model performance, R is a regularization term, and α is the penalty, respectively [36]. Similar to ANNs, the ONNs in [25] were trained by using BP and SGD algorithms. As alternative approaches to gradient-based methods, evolutionary algorithms are representative gradient free methods used to optimize the weights of ANNs [44,45]. Through the selection, crossover, and mutation processes of a population, evolutionary algorithms provide a natural choice way to gradually optimize the model parameters to achieve superior fitness. There are many attractive reasons for using evolutionary algorithms to train ANNs: the most important one is that evolutionary algorithms optimize more hyper-parameters that are not considered in SGD. For example, the neuroevolution of augmenting topologies (NEAT) algorithm is a typical search method that solves the problem of crossing over variable network topologies through historical marking [43]. In this study, neuroevolution is proposed to determine the hyper-parameters of ONNs and to optimize the weights in the connections. The network architectures of ONNs are modelled by a chip-level simulation

Optics EXPRESS

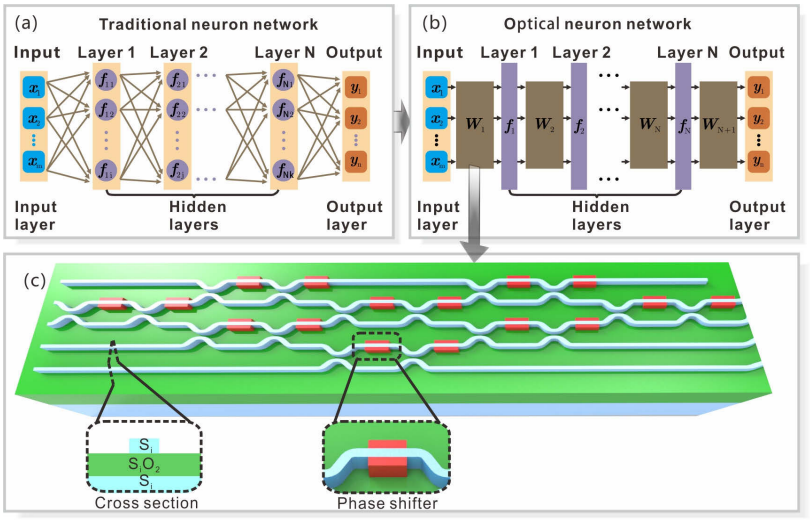

Fig. 1. (a) The network architecture of the ANNs, which includes input neurons, hidden layers, and output neurons. (b) Decomposition of ANNs into a series of layers which implement linear matrix multiplication and nonlinear transform functions. (c) Physical implementation of the OIU that are composed of programmable MZIs.

platform, neuroptica [50]. Neuroptica provides a range of abstraction levels encapsulated by Keras-like application programming interfaces (APIs) [50]. In the neuroptica platform, the lowest abstraction levels and the highest abstraction levels manipulate the properties of the phase shifters and the whole network architectures of ONNs, respectively [50]. It has been demonstrated that the functions of a logical gate can be effectively implemented by using neuroptica based on BP and SGD training strategies [34]. Moreover, neuroptica provides two decomposition ways (Reck [51] and Clements [52] decompositions) to construct any unitary matrices by reasonably arranging the phase shifters and the MZIs in the optical meshes. The optical neurons in ONNs implement nonlinear activation functions based on the optical-to-optical nonlinearities, which have advantages in expressiveness in comparison to the all-optical nonlinearities [34]. The nonlinear activation function can be fabricated in an electro-optic hardware platform which is configured by three critical physical parameters: the amount of power tapped off to photodetector α, the phase gain g, and the biasing phase θ [34]. We include these three physical parameters as optimized variables in the training algorithms except for the weights between the optical neurons. Moreover, two kinds of traditional optimizers in the neuroptica platform can be used to train the ONNs based on the AVM and SGD (or adaptive moment estimation) [34].

Two kinds of typical gradient free algorithms, GA and PSO, are used to validate the effectiveness of the training algorithms based on neuroevolution. GA is a neuroevolution algorithm that is widely used in inverse design and performance optimization of photonic devices [40,41]. In this study, we use GA to train ONNs by optimizing for the phase shifters of the optical mesh and hyper-parameters of the nonlinear activation functions. The algorithmic details of GA are outlined as follows: In Step 1, a reasonable network architecture for ONNs is selected for the targeted dataset. Then a population of N individuals (ONNs), that have the same network architectures, but different weights and hyper-parameters, is randomly generated. For example, we construct a simple network architecture consisting of $L = 3$ layers for the famous iris plants dataset. In that dataset, the features of each iris plant and the categories of all iris plants are 4 and 3, respectively. Consequently, the input layers of ONNs include 4 input ports, while the output layers include 3 output ports. Each hidden layer in the ONNs includes a 4×4 unitary

Optics EXPRESS

matrix implemented by an optical mesh and a layer of 4 parallel electro-optic activation functions, as proposed in [34]. In the final layer, the electro-optic activation functions are replaced by a detection layer, which corresponds to the power measurement by using a photodetector [34]. After the final layer, a dropmask layer is added to the network to reduce the dimension from 4 to 3 by abandoning a output port. Here, the dropmask layer can be added in the hidden layer to make the dimension of the intermediate layer more flexible (larger or smaller than 4). To implement an arbitrary unitary matrix, the unitary optical mesh includes a layer of phase shifters at the beginning of the optical mesh [50]. The optical responses of the electro-optic activation functions are configured by three hyper-parameters, α, g, and θ, which are also considered in GA. For all the generated ONNs, all the optimization variables are initialized in the different ranges, which are specified by minimum and maximum values $0<\varphi<2\pi$, $0<\alpha<1$, $0<g<\pi$ and $-2\pi<\theta<0$, where φ is the phase of the phase shifter. In Step 2, for the generated ONNs, training instances are inputted into the ONNs and transferred from the input layer to output layer. This process is called the forward-propagation step, where the training instances are performed by linear operations (unitary optical mesh) and nonlinear operations (electro-optic activation function) layer by layer. In the output layer, the predicted results for current iteration are collected, and the prediction losses between the prediction results and target results are calculated based on the categorical cross entropy or mean squared error (MSE). The prediction losses for all the generated N ONNs are calculated and sorted in ascending order. In Step 3, a new population of ONNs is generated by using the standard selection, crossover, and mutation procedures. In the selection process, the prediction loss of ONNs is regarded as fitness. Two parent individuals are selected from the previous generation based on the roulette-wheel selection method or tournament strategy [39]. The ONNs with a smaller prediction loss are selected with the higher probability in the selection process. In order to maintain the diversity of the population or to keep some superior individuals, some percentage of the superior ONNs or inferior ONNs are retained in the next population. In the crossover process, the optimization variables, consisting of weights and hyper-parameters, are extracted from the ONNs and converted into the binary values. It should be noted that the conversion of a decimal number to a binary number is likely to result in the loss of digital precision. The optimization variables of the parent individuals (ONNs) cross over to generate a new individual based on the uniform crossover algorithm [39]. In the uniform crossover algorithm, the probabilities of gene exchange and crossover are 50% and 80%, respectively. In the mutation process, each element in the binary number has a 5% probability to flip from 0 (1) to 1 (0). After converting the weights and the hyper-parameters from binary numbers to decimal numbers, new individuals (ONNs) are generated. The new generated ONNs and the remaining ONNs selected from the previous generation form a new population. In Step 4, the performance metrics of the generated population are evaluated to determine if the training process should be stopped. In our training algorithm, if the generation of ONNs evolves for 1000 times or the prediction losses remain unchanged for more than 5 generations, the training process stops; otherwise, it proceeds to Step 2. The flowchart of the learning process for the ONNs based on GA is shown in Fig. 2(a).

Similar to GA, PSO is an evolutionary algorithm that is suitable for the decimal number rather than the binary number [42]. The steps used to generate the initial population for the GA are also used for that purpose for the PSO. However, the generation of new population for PSO is not through selection, crossover and mutation procedures. It indicates that there is no need to convert the decimal number to the binary number in PSO, which can effectively avoid the loss of digital precision. The flowchart of the learning process for the ONNs based on PSO is shown in Fig. 2(b). For PSO, the individuals in the population depend on the currently optimal individual and the historically optimal solution to evolutionally optimize for fitness [42]. Similarly, when the PSO is used to train ONNs, each ONN in the population searches for the optimal variables (weights and hyper-parameters) by synthetically considering the globally optimal variables and

Optics EXPRESS

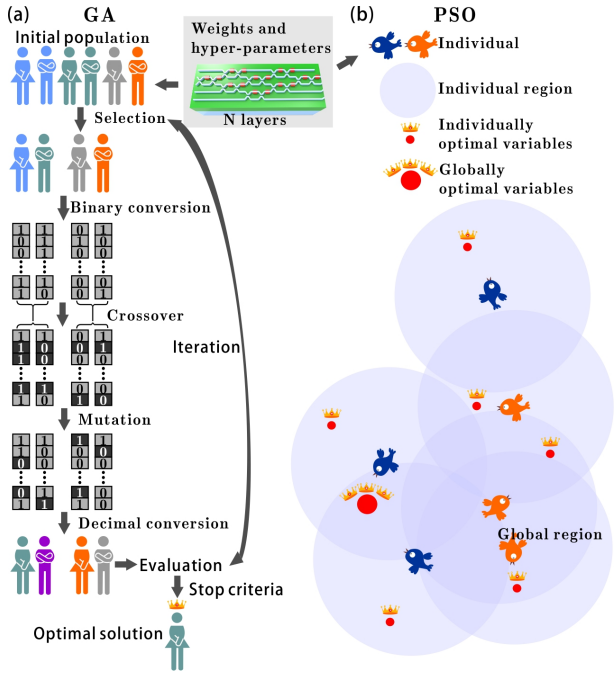

Fig. 2. The flowcharts of the learning algorithms for the ONNs based on GA (a) and PSO (b).

the individually optimal variables. It indicates that the evolution of the optimization variables for all ONNs are controlled by a specified velocity [53]:

$$V_i^{k+1} = WV_i^k + c_1 r_1 (pb_i^k - X_i^k) + c_2 r_2 (gb_k^d - X_i^k) \tag{2}$$

where i represents the ith ONN in the population, k is the iteration number, W is the inertia weight, c_1=c_2=1.49445 are the acceleration constants, r_1 (r_2) are the random values between 0 and 1, $gb_k{}^d$ relates to the globally optimal weights and hyper-parameters for all ONNs, and X_{ik} and $pb_i{}^k$ are the current variables and the individually optimal variables for the ith ONN in the kth iteration, respectively. The weights and hyper-parameters of the ith ONN are updated according to following equation [53]:

$$X_i^{k+1} = X_i^k + V_i^{k+1} \tag{3}$$

In order to avoid the premature problem, the velocities of evolution are limited to a certain range (−2~2). Finally, in each iteration, the prediction losses of the newly generated population are evaluated to determine if the training process should be stopped. If the generation evolves for 1000 times, or if the prediction losses remain unchanged for more than 5 generations, the PSO stops.

We define the effectiveness for the training algorithms based on three criteria. First of all, all the prediction losses of our proposed training algorithm have downward trends with the epoch, and, finally, they are smaller than 0.2. Secondly, the prediction accuracies of our proposed training algorithm have upward trends with the epoch, and, finally, they are larger than 0.85. Thirdly, the prediction losses and accuracies of our proposed training algorithm are equal to or better than the AVM training algorithms.

Optics EXPRESS

3. Calculated results and discussions

As typical datasets in the classification tasks, the iris plants dataset [54] and wine recognition dataset [55] are selected as the test datasets to demonstrate the effectiveness of the training algorithm. The iris plants dataset is a simple dataset that includes 150 instances (four attributes for each instance). In comparison to the iris plants dataset, the wine dataset is more complex, for example, it has 13 features, leading to the need for a more complex network architecture for ONNs. Thus, an ONN that includes 13 input ports and 3 output ports must be constructed to recognize the wine dataset. In addition, to verify the practical applications of the training algorithm based on neuroevolution, we use ONNs to recognize the modulation formats in a communication system. We did not utilize the complete digital signals to train the ONNs because of their high dimensions. Alternatively, four effective attributes γ_{max}, σ_{aa}, σ_{dp} and σ_{af} are extracted from 800 digital signals randomly modulated by the 4ASK, 4FSK, BPSK and QPSK modulation formats [56]. It has been demonstrated that these statistical features can exhibit obvious separation in the distributions for those modulation formats [56]. Here, the four effective features are inputted into the ONNs, which are trained by supervised learning, to identify the maximum possible category of the modulation formats. Besides, we also use two kinds of randomly generated datasets provided by the neuroptica simulation platform to compare the training effects between the evolutionary algorithms (GA and PSO) and AVM. As this article shows later, the data points in the example datasets are randomly generated to segment specific square areas into a triangle (ring) part and some other part [50]. This indicates that the recognition of the example datasets is a binary classification problem whose categories are coded by one-hot labelling. We use 80% (320) and 20% (80) of all the instances (400) in the example datasets as the training set and the test set, respectively.

Figure 3 presents the calculated results of the ONNs trained by GA. First of all, we use the APIs provided by the neuroptica simulation platform to generate a simple dataset that segments the square space into two triangle areas. As shown in Fig. 3(a), the red cross marks and blue circles correspond to two triangle areas respectively. The network architectures of ONNs includes $L = 5$ layers decomposed by the Clements method [52] and each layer includes five neurons. Similar to the demo provided by the neuroptica simulation platform, we reshape the input data to fit into the specified mesh size, and we normalize them to have the same total power. The ONNs are trained by GA and AVM, and the calculated results are presented in Fig. 3(b). Here, the population size of GA is set as $N = 500$, and the MSEs between the prediction results and the target results are chosen as the fitness for GA. As seen in Fig. 3(b), the prediction losses of AVM (red circles) decrease from 0.35 to 0.005, while the prediction losses of GA (blue circles) decrease from 0.38 to 0.09. This suggests that the learning algorithms are convergent for the training datasets. Although both the GA and AVM are effective for simple pattern recognition, the performance is better for the AVM than the GA. This conclusion can be verified by the variations in the accuracies (solid line) for the GA and AVM, as seen in Fig. 3(b). It can be observed that the accuracies of the AVM and GA can reach 1 and 0.95, respectively. Moreover, the AVM can achieve excellent accuracy (>0.95) faster than the GA. Obviously, although the GA training method cannot compete with the AVM learning method in terms of accuracy and speed, it is still effective for the training of ONNs. The contours shown in Fig. 3(a) are the classification boundary for GA. Except for the simple example dataset, we also use the ONNs trained by GA to recognize three practical datasets, namely, the iris plants dataset, the wine dataset, and the modulation format recognition dataset. For particular network architectures consisting of $L = 3$ layers for the iris plants dataset (modulation format recognition dataset) and $L = 2$ layers for the wine recognition dataset, the variations of the prediction loss and classification accuracy are illustrated in Fig. 3(c) and Fig. 3(d), respectively. For the ONNs with L = 3 layer and the ONNs with L = 2 layer, the totals of the optimization variables, including the weights and hyper-parameters, are 54 and 35, respectively. As shown in Fig. 3(c), the MSEs of the three

Optics EXPRESS

datasets decrease to 0.12 (iris plants dataset), 0.22 (wine dataset), and 0.26 (modulation format recognition dataset) after 2000 iterations, which indicates the convergence of GA. As seen in Fig. 3(d), the classification accuracies for the test datasets are 0.97 (iris plants dataset), 0.89 (wine dataset), and 0.92 (modulation format recognition dataset). In fact, the trained algorithm has converged in the vicinity of 100 iterations (the classification accuracies for all the datasets are greater than 0.80), which means that the GA is effective for the training of ONNs in practical classification applications. It should be noted that the individuals in the new population may not be better than those in the previous generation. And better ONNs may be reproduced through efforts of several generations. Therefore, the prediction accuracies and losses in Fig. 3 change in a stepwise way.

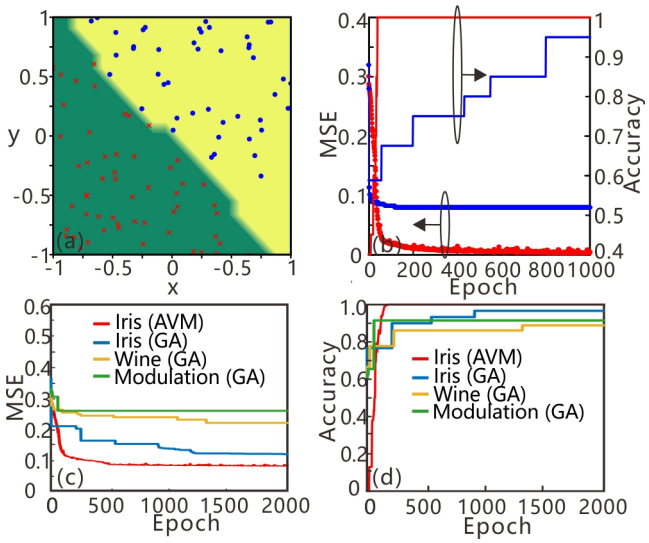

Fig. 3. The calculated results of the ONNs trained by GA. (a) A simple dataset provided by the neuroptica simulation platform segments the square space into two triangle areas. (b) The MSEs and classification accuracies of the ONNs trained by AVM and GA for simple dataset. (c) The MSEs of the ONNs trained by GA for three test datasets. (d) The classification accuracies of the ONNs trained by GA for three test datasets.

We also compare the training effects under different GA parameters, such as population sizes, mutation probabilities, crossover probabilities, and selection operators. For simplicity, we will use the iris plants dataset as an example. The information presented in Fig. 4(a) shows the influences of different population sizes ($N = 50$, 200, and 500) on the MSE and classification accuracy. Obviously, as the population sizes increase, the final values of MSE decrease (0.11 for $N = 500$, 0.12 for $N = 200$, and 0.20 for $N = 50$), and classification accuracies increase (0.97 for $N = 500$, 0.93 for $N = 200$, and 0.9 for $N = 50$). This occurs because the large population size enhances the global searching ability of GA. In addition, we also consider the influences of the mutation probability and crossover probability of GA on the performance metrics. In Fig. 4(b), it can be observed that an increase in mutation probability may lead to a decrease in accuracy. We should note that a large mutation can cause instability and reduce the convergence speed of GA. Thus, the mutation probability is set to 10% in the GA. On the other hand, as seen in Fig. 4(c), there is no linear correlation between the crossover probability and performance metrics. Moreover, we compare the calculated results for different selection operators, such as tournament strategy (TS), roulette-wheel selection (RWS), linear ranking selection (LRS), and exponential ranking selection (ERS). As shown in Fig. 4(d), the classification accuracies of the TS, LRS,

Optics EXPRESS

RWS, and ERS are 0.97, 0.97, 0.87 and 0.90, respectively. This indicates that TS and LRS are more effective than the other two selection operators.

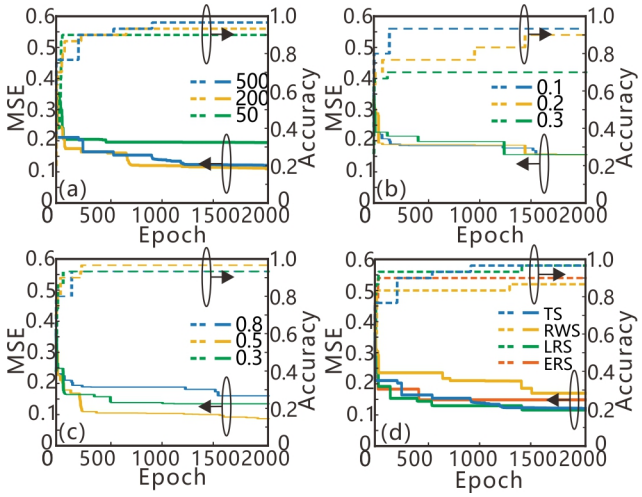

Fig. 4. The MSEs and classification accuracies of the trained ONNs with different population sizes (a), mutation probabilities (b), crossover probabilities (c) and selection operators (d) of GA.

The calculated results of the ONNs trained by PSO are shown in Fig. 5. Firstly, we use the function provided by the neuroptica simulation platform to generate a simple dataset that segments the square space into a ring and another part. As shown in Fig. 5(a), the red cross marks correspond to the ring area, while the blue circles correspond to the other area. The ONNs, which consist of $L = 5$ layers decomposed by the Clements method [52] (each layer includes five neurons) are trained by PSO and AVM, and the calculated results are shown in Fig. 5(b). The prediction losses that are chosen as the fitness for PSO are calculated based on MSE in each iteration. As seen in Fig. 5(b), the prediction losses (red and blue circles) of AVM and PSO, decrease from 0.35 (0.37) to 0.05 (0.07), suggesting that the learning algorithms are convergent for the training datasets. It is noteworthy that the final prediction loss of the PSO is approximately equal to that of the AVM. This phenomenon can be confirmed by the classification accuracies (red and blue solid line) illustrated in Fig. 5(b). We observe that the classification accuracies of PSO achieve an excellent accuracy (>0.85) faster than AVM, although, ultimately, both AVM and PSO achieve an accuracy of 0.9. And the contours shown in Fig. 5(a) are the classification boundary for PSO. Obviously, the accuracy and stability of the PSO training method are competitive with the AVM learning algorithms. Similar to GA, we also use the trained ONNs to recognize the iris plants dataset, wine dataset and modulation format dataset. In this case, the network architecture of the ONNs contains three layers, and each layer includes an optical mesh followed by an electro-optic activation function with intensity modulation [34]. As seen in Fig. 5(c), when the population of PSO evolves 1000 generations, the MSEs between the prediction results and the target results show tendencies toward low values, indicating the convergence of the training algorithm. Correspondingly, we observe in Fig. 5(d) that the classification accuracies for the iris plants dataset and the wine dataset reach to 100%. For the relatively complex dataset (modulation format dataset), the classification accuracy also increases to 93%. Obviously, the accuracies of the ONNs trained by PSO are superior to those of the ONNs trained by GA. The finding is attributed to the fact that there is no need to convert the optimization variables from decimal numbers to binary numbers for PSO, which efficiently avoids the loss of digital precision.

Optics EXPRESS

Moreover, the optimization strategy based on the globally optimal variables and individually optimal variables accelerates the convergence of the training algorithm.

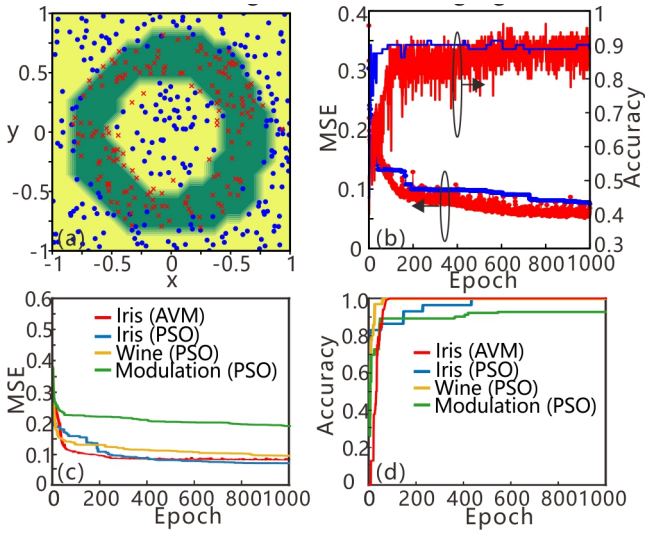

Fig. 5. The calculated results of the ONNs trained by PSO. (a) A simple dataset provided by the neuroptica simulation platform segments the square space into ring part and other part. (b) The MSEs and classification accuracies of the ONNs trained by PSO and AVM for simple dataset. (c) The MSEs of the ONNs trained by PSO for three test datasets. (d) The classification accuracies of the ONNs by PSO for three test datasets.

We also compare the training results under different optimization parameters, such as population sizes, inertia weights, velocity ranges of PSO, and the number of the layers in ONNs. For simplicity, we provide an example of the iris plants dataset. In Fig. 6(a), it can be observed that the classification accuracies can reach 1.0, 0.9 and 0.87 when the population sizes of PSO are $N = 500$, 200, and 50, respectively. The prediction losses for the large populations ($N = 500$ and 200) are superior to that of the small population ($N = 50$). This is because the large populations enhance the global searching ability of evolution algorithms [36]. Although we can increase the population size of PSO to achieve a lower MSE value and a higher classification accuracy, it's at the expense of the training time. In addition, we also consider the influences of inertia weight and velocity range on the performance metrics. As seen in Fig. 6(b) and Fig. 6(c), there is no linear correlation between the inertia weight (velocity range) and MSE (classification accuracy). In this study, we set the inertia weight and velocity range as 1 and -2~2, respectively. In order to achieve rapid convergence, the variable inertia weight for different iterations can be employed in PSO. The velocity range in PSO should be reasonably configured to control the relationship between the convergence speed and the search scope. Besides, we also consider the influence of different network architectures of ONNs on the performance metrics. As shown in Fig. 6(d), the number of layers in the ONNs does not have a linear relationship with MSE and classification accuracy. We observe that the simplest ONN with 3 layers has a low MSE (0.08) and a high accuracy (100%) in the last iteration. The complex network architecture of ONNs may lead to the over-fitting problem, which reduces the accuracy on the test dataset. Consequently, the 3-layer ONN performs better than the 5-layer ONN, as seen in Fig. 6(d).

It should be noted that any elevated performance over one class of problems is offset by its performance over another class for optimization algorithms [57]. Therefore, it is important to discuss the advantages and disadvantages of the neuroevolution that is applied in the training of ONNs. First of all, until now, with the exception of the 'brute force' method [25] and the AVM

Optics EXPRESS

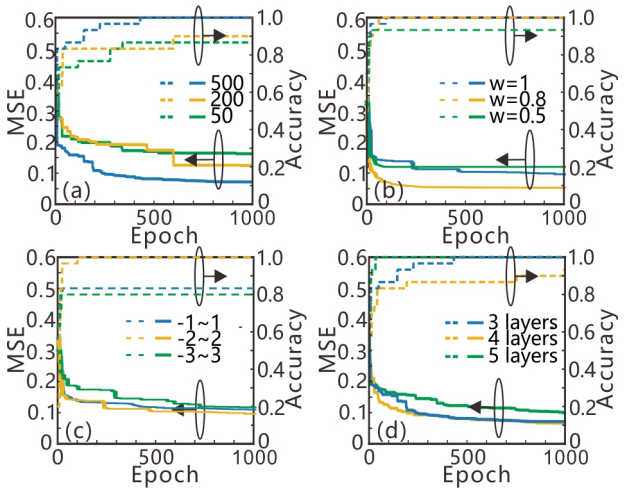

Fig. 6. The MSEs and classification accuracies of the trained ONNs for different population sizes (a), inertia weights (b), velocity ranges (c) of the PSO and the number of the layers in the ONNs (f).

method [36], no other effective learning methods have been identified for training ONNs. Here, we propose another on-chip training approach that can take no account of the fabrication errors and mapping errors. In comparison to gradient-based methods (such as AVM), evolutionary algorithms are relatively simple, but effective, because there is no need to have a deep physical background to derive the gradients of the objective functions. Secondly, neuroevolution can determine the weights in the connections and optimize more hyper-parameters of the network, such as activation functions. Thirdly, if ONNs can be trained by neuroevolution, it is expected to improve the training efficiency for the traditional DRL. We have used neuroevolution to train the ONNs that are applied in cart pole games. The calculated results demonstrate that the ONNs that are trained by using neuroevolution can amass 325 points (full score is 500). Fourth, it is expected that an effective data cluster can be achieved by using the ONNs trained by neuroevolution. In comparison to the SGD training algorithm, neuroevolution is more appropriate for training a self-organizing map which has one layer in its network structure. Certainly, the neuroevolution has many disadvantages. The slow training speed is the main disadvantage for the evolutionary algorithms that are used in the training of ANNs. We compare the training time between the AVM and evolutionary algorithms on a 2.9-GHz Intel Core i5 processor. The calculated results show that the total training time of the AVM is 36 s when the batchsize is set to 20, while the total training time for the GA and PSO is 167 s and 112 s, respectively, when the population size is set to 100. Obviously, the training consumption is larger for the neuroevolution than the AVM. However, it should be noted that many evolutionary algorithms can use parallel computing to accelerate the optimization process. We use the message passing interface to reduce the training time of GA. The calculated results show that the parallel GA simulated on a cluster of two nodes can double the speed of the model training. Secondly, GA and PSO are inclined to fall into local optima. For both the GA and PSO, we set the population size to 500 to alleviate this problem, to some extent. In order to further alleviate this problem, we can adopt a hybrid evolutionary algorithm that includes GA and simulated annealing to accept a worse solution with a variable probability [58].

4. Conclusions

In conclusion, we propose a novel learning strategy to design and train ONNs based on neuroevolution. Two typical gradient free algorithms, GA and PSO are used to determine the

Optics EXPRESS

hyper-parameters of ONNs and optimize the weights (phase shifters) in the connections. To demonstrate the effectiveness of the training algorithms, the trained ONNs are applied in the classification tasks for different datasets. The calculated results demonstrate that these simple training algorithms are competitive with other traditional learning algorithms. Compared with previous works, we introduce an efficient training method for the ONNs and demonstrate their broad application prospects in pattern recognition and DRL.

Funding

National Natural Science Foundation of China (61705015,61625104,61431003); Beijing Municipal Science and Technology Commission (Z181100008918011); Fundamental Research Funds for the Central Universities (2019RC15, 2018XKJC02); National Key Research and Development program (2016YFA0301300).

References

1. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**(7553), 436–444 (2015).
2. J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks **61**, 85–117 (2015).
3. T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," IEEE Comput. Intell. Mag. **13**(3), 55–75 (2018).
4. G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and T. N. Sainath, "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups," IEEE Signal Process. Mag. **29**(6), 82–97 (2012).
5. M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," Pattern Recognition Letters **42**, 11–24 (2014).
6. M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, and J. Zhang, "End to end learning for self-driving cars," arXiv preprint arXiv:1604.07316 (2016).
7. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602 (2013).
8. S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*, (IEEE, 2017), 3389–3396.
9. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, (NIPS, 2012), 1097–1105.
10. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556 (2014).
11. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (CVPR, 2015), 1–9.
12. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, (IEEE, 2016), 770–778.
13. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation **9**(8), 1735–1780 (1997).
14. J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. Warde-Farley, I. Goodfellow, and A. Bergeron, "Theano: Deep learning on gpus with python," in *NIPS 2011, BigLearning Workshop, Granada, Spain*, (Citeseer, 2011), 1–48.
15. C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, (ACM, 2015), 161–170.
16. T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *ACM Sigplan Notices*, (ACM, 2014), 269–284.
17. F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, and G.-J. Nam, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **34**(10), 1537–1557 (2015).
18. M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, and S. Jain, "Loihi: A neuromorphic manycore processor with on-chip learning," IEEE Micro **38**(1), 82–99 (2018).
19. S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," Proc. IEEE **102**(5), 652–665 (2014).
20. X. Wu, V. Saxena, K. Zhu, and S. Balagopal, "A CMOS Spiking Neuron for Brain-Inspired Neural Networks With Resistive Synapses andIn SituLearning," IEEE Trans. Circuits Syst. II **62**(11), 1088–1092 (2015).
21. J. Touch, A.-H. Badawy, and V. J. Sorger, "Optical computing," Nanophotonics **6**(3), 503–505 (2017).
22. M. A. Nahmias, B. J. Shastri, A. N. Tait, and P. R. Prucnal, "A leaky integrate-and-fire laser neuron for ultrafast cognitive computing," IEEE J. Sel. Top. Quantum Electron. **19**(5), 1–12 (2013).

Optics EXPRESS

23. A. N. Tait, T. F. De Lima, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Multi-channel control for microring weight banks," Opt. Express **24**(8), 8895–8906 (2016).
24. P. R. Prucnal, B. J. Shastri, T. F. de Lima, M. A. Nahmias, and A. N. Tait, "Recent progress in semiconductor excitable lasers for photonic spike processing," Adv. Opt. Photonics **8**(2), 228–299 (2016).
25. Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund, and M. Soljačić, "Deep learning with coherent nanophotonic circuits," Nat. Photonics **11**(7), 441–446 (2017).
26. H. Bagherian, S. Skirlo, Y. Shen, H. Meng, V. Ceperic, and M. Soljacic, "On-Chip Optical Convolutional Neural Networks," arXiv preprint arXiv:1808.03303 (2018).
27. J. Bueno, S. Maktoobi, L. Froehly, I. Fischer, M. Jacquot, L. Larger, and D. Brunner, "Reinforcement learning in a large-scale photonic recurrent neural network," Optica **5**(6), 756–760 (2018).
28. X. Lin, Y. Rivenson, N. T. Yardimci, M. Veli, Y. Luo, M. Jarrahi, and A. Ozcan, "All-optical machine learning using diffractive deep neural networks," Science **361**(6406), 1004–1008 (2018).
29. M. Y.-S. Fang, S. Manipatruni, C. Wierzynski, A. Khosrowshahi, and M. R. DeWeese, "Design of optical neural networks with component imprecisions," Opt. Express **27**(10), 14009–14029 (2019).
30. R. Hamerly, L. Bernstein, A. Sludds, M. Soljačić, and D. Englund, "Large-scale optical neural networks based on photoelectric multiplication," Phys. Rev. X **9**(2), 021032 (2019).
31. G. R. Steinbrecher, J. P. Olson, D. Englund, and J. Carolan, "Quantum optical neural networks," npj Quantum Inf **5**(1), 60 (2019).
32. N. Passalis, G. Mourgias-Alexandris, A. Tsakyridis, N. Pleros, and A. Tefas, "Variance preserving initialization for training deep neuromorphic photonic networks with sinusoidal activations," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (IEEE, 2019), 1483–1487.
33. G. Mourgias-Alexandris, A. Tsakyridis, N. Passalis, A. Tefas, K. Vyrsokinos, and N. Pleros, "An all-optical neuron with sigmoid activation function," Opt. Express **27**(7), 9620–9630 (2019).
34. I. A. Williamson, T. W. Hughes, M. Minkov, B. Bartlett, S. Pai, and S. Fan, "Reprogrammable Electro-Optic Nonlinear Activation Functions for Optical Neural Networks," arXiv preprint arXiv:1903.04579 (2019).
35. H. Zhou, Y. Zhao, X. Wang, D. Gao, J. Dong, and X. Zhang, "Self-learning photonic signal processor with an optical neural network chip," arXiv preprint arXiv:1902.07318 (2019).
36. T. W. Hughes, M. Minkov, Y. Shi, and S. Fan, "Training of photonic neural networks through in situ backpropagation and gradient measurement," Optica **5**(7), 864–871 (2018).
37. S. Molesky, Z. Lin, A. Y. Piggott, W. Jin, J. Vucković, and A. W. Rodriguez, "Inverse design in nanophotonics," Nat. Photonics **12**(11), 659–670 (2018).
38. W. Bogaerts and L. Chrostowski, "Silicon photonics circuit design: methods, tools and challenges," Laser Photonics Rev. **12**(4), 1700237 (2018).
39. T. Zhang, J. Wang, Q. Liu, J. Zhou, J. Dai, X. Han, Y. Zhou, and K. Xu, "Efficient spectrum prediction and inverse design for plasmonic waveguide systems based on artificial neural networks," Photonics Res. **7**(3), 368–380 (2019).
40. Z. Yu, H. Cui, and X. Sun, "Genetically optimized on-chip wideband ultracompact reflectors and Fabry–Perot cavities," Photonics Res. **5**(6), B15–B19 (2017).
41. P.-H. Fu, S.-C. Lo, P.-C. Tsai, K.-L. Lee, and P.-K. Wei, "Optimization for Gold Nanostructure-Based Surface Plasmon Biosensors Using a Microgenetic Algorithm," ACS Photonics **5**(6), 2320–2327 (2018).
42. J. C. Mak, C. Sideris, J. Jeong, A. Hajimiri, and J. K. Poon, "Binary particle swarm optimized 2× 2 power splitters in a standard foundry silicon photonic platform," Opt. Lett. **41**(16), 3868–3871 (2016).
43. K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, "Designing neural networks through neuroevolution," Nat. Mach. Intell. **1**(1), 24–35 (2019).
44. 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).
45. M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, (AAAI, 2018).
46. A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," Neural Networks **111**, 47–63 (2019).
47. S. R. Kulkarni and B. Rajendran, "Spiking neural networks for handwritten digit recognition—Supervised learning and network optimization," Neural Networks **103**, 118–127 (2018).
48. M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," Front. Neurosci. **12**(774), 1–18 (2018).
49. A. Autere, H. Jussila, Y. Dai, Y. Wang, H. Lipsanen, and Z. Sun, "Nonlinear optics with 2D layered materials," Adv. Mater. **30**(24), 1705963 (2018).
50. https://github.com/fancompute/neuroptica.
51. M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator," Phys. Rev. Lett. **73**(1), 58–61 (1994).
52. W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, "Optimal design for universal multiport interferometers," Optica **3**(12), 1460–1465 (2016).

Optics EXPRESS

53. P. Ghamisi and J. A. Benediktsson, "Feature selection based on hybridization of genetic algorithm and particle swarm optimization," IEEE Geosci. Remote Sens. Lett. **12**(2), 309–313 (2015).
54. R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, (Montreal, Canada, 1995), 1137–1145.
55. S. J. Roberts, R. Everson, and I. Rezek, "Maximum certainty data partitioning," Pattern Recognition **33**(5), 833–839 (2000).
56. E. E. Azzouz and A. K. Nandi, "Automatic identification of digital modulation types," Signal Processing **47**(1), 55–69 (1995).
57. D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," IEEE Trans. Evol. Computat. **1**(1), 67–82 (1997).
58. W. Li, S. Ong, and A. Nee, "Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts," Int. J. Prod. Res. **40**(8), 1899–1922 (2002).