Swarm and Evolutionary Computation 63 (2021) 100863

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

Hybrid MPSO-CNN: Multi-level Particle Swarm optimized hyperparameters of Convolutional Neural Network

Pratibha Singh*, Santanu Chaudhury, Bijaya Ketan Panigrahi

Department of Electrical Engineering, Indian Institute of Technology Delhi, India

ARTICLE INFO

Keywords:
Convolution Neural Network
Evolutionary
Multiple swarms
Neural networksl
Particle Swarm Optimization

ABSTRACT

Recent advances in swarm inspired optimization algorithms have shown its extensive acceptance in solving a wide range of different real-world problems. Particle Swarm Optimization (PSO) is one of the most explored nature-inspired population-based stochastic optimization algorithm. In this paper, a Multi-level Particle Swarm Optimization (MPSO) algorithm is proposed to find the architecture and hyperparameters of the Convolutional Neural Network (CNN) simultaneously. This automated learning will reduce the overhead of human experts to find these parameters through manual analysis and experiments. The proposed solution uses multiple swarms at two levels. The initial swarm at level-1 optimizes architecture and multiple swarms at level-2 optimize hyperparameters. The proposed method has used sigmoid like inertia weight to adjust the exploration and exploitation property of particles and avoid the PSO algorithm to prematurely converge into a local optimum solution. In this paper, we have explored an approach to suggest the best well-conditioned CNN architecture and its hyperparameters using MPSO in a specified search space. The complexity and performance of MPSO-CNN will depend on the dimension of the search space. The experimental results on 5 benchmark datasets of MNIST, CIFAR-10, CIFAR-100, Convex Sets, and MDRBI have demonstrated one more effective application of PSO in learning a deep neural architecture.

1. Introduction

Neural network architectures with more than three layers are preferred to solve more complex real-world problems. It requires a large amount of data, fast computational resources, and a lot of computation time for proper training that raises challenges many times. Traditional machine learning methods depend on so-called handcrafted features. Deep Learning (DL) advances traditional methods of feature learning because of its auspicious performance. Some extensively used DL architectures are Convolutional Neural Network (CNN), Auto-encoder, Deep Belief Network (DBN), and restricted Boltzmann machine (RBN) [1]. In deep learning methods, a large number of parameters are required to be adjusted and its complexity increases with an increase in the number of hidden layers. Human expertise is required to decide these network parameters. CNN is extremely popular among all deep networks with its large set of applications in computer vision. Lots of research has been done in the past to optimize parameter selection in deep networks using swarm intelligence algorithms. Researchers have developed hybrid versions of CNNs for the optimization of parameter selection [2–8]. PSO proposed by J. Kennedy and R. Eberhart [9], is inspired by the social behavior of birds. PSO is similar to other evolutionary algorithms such as Genetic Algorithm (GA) [10], Differential Evolution (DE), Ant Colony Optimization (ACO) [11], Cuckoo Search Optimization [12], Bacterial Foraging Optimization (BFO) [13], Firefly Algorithm (FA) [14], Artificial Fish Swarm Optimization (AFSO) [15] etc. Mainstream swarm intelligence algorithms are ACO and PSO [16]. PSO is the most preferred choice of researchers in solving optimization problems because it has fewer parameters, a simple formulation, and easy computation. Recent advances in applications of PSO are in the emerging field of neural networks which is also our motivation for work [17]. PSO has improved a lot with its different variants to provide better solutions. PSO mainly suffers from its premature convergence which may occur due to the lack of diversity in the nature of particles. PSO variants can solve this problem in different ways by modifying the initialization of the swarm, improving the selection of parameters, adjusting its topological structure, and further proposing its hybrid versions with other algorithms. Modified-PSO is one of the recent developments in this case [18]. It uses chaos-based initialization to generate particles with uniform distribution and then sigmoid like inertia weight is used to balance between exploration and exploitation. They have also applied wavelet mutation to the particles with less fitness to maintain diversity in intermediate stages for better convergence [18].

The motivation of our work is to use PSO to automatically learn an efficient architecture and best set of hyperparameters of CNN without

* Corresponding author.
E-mail address: pratibhahbti2004@gmail.com (P. Singh).

https://doi.org/10.1016/j.swevo.2021.100863
Received 30 April 2018; Received in revised form 26 December 2020; Accepted 10 February 2021
Available online 2 March 2021

much human intervention. In general, it requires human expertise and intensive efforts for conducting experiments on many possible configurations of network parameters to finalize one with relatively better performance. It is difficult for human experts to select the parameters of CNN before applying it to solve any real-world problem. Manual search is not the same as guided by an algorithm that can continuously improve and converge to a steady state with consistent performance. The resulting solution may be error-prone and time-consuming due to many trivial experiments. PSO provides a way to automatically evolve these parameters. The search space is decided by the user and the complexity, computation cost, and performance of the proposed method depend on the dimension of search space. The proposed method evolves the architecture of CNN with its hyperparameters using multiple swarms. Although the proposed automated solution is computationally expensive it reduces human efforts extensively.

Contributions of the proposed method of optimum selection of hyperparameters in CNN using MPSO are:

- The proposed approach reduces the efforts of an individual to identify the effective configuration of a CNN with better performance, in comparison to default or random settings. The manual search of such configurations, specific to datasets needs a lot of human efforts and also expertise. So, this proposed approach will motivate non-experts also to use the neural network architectures efficiently.
- A modified structure of PSO is used here to search the optimal set of hyperparameters of a deep CNN architecture. We have extended the swarms at two levels to explore the solution in better way and used two dimensional structure of particles at second level which exhaustively searches for the hyperparameters for set of layers generated at first level.
- Automated search of the solution extends the search space that can be explored by human search. PSO is a guided search method that will explore and exploit the search space along with swarm at level-1 and swarm at level-2. The swarm at level-1 will define the search space with the maximum number of layers and swarm at level-2 will define the search space with the maximum number of hyperparameters associated with each layer. The swarm at level-1 explores the search space for finding the best set of layers and swarm level-2 exploits the sub-search-space to find the best set of parameters of these layers.
- MPSO used sigmoid-like inertia weight at both levels to explore and exploit the search space with the proper balance between them and as suggested in literature it avoids premature convergence of the solution.
- PSO algorithm initializes its population with a random set of configurations satisfying constraints of hyperparameters. The solution evolves with a better configuration with every iteration of MPSO. The fitness evaluation of each particle of the swarm depends on CNN. The fitness evaluations of these configurations can be executed in multiple machines in parallel as the evaluation of each particle of PSO is independent of the evaluation of another particle. Although, it requires large computation capability but can provide an optimal set of hyperparameters in the required time.
- Results demonstrate that automated search of configurations explores the configuration search space in a better way as a comparison to human search. Swarm optimized search is a guided search for the solution whereas human search depends on expertise. The dimension of the search space can be extended depending on the availability of computational resources. It reduces the human effort to a large extent and will require the least machine learning expertise.
- The proposed modified version of PSO optimizes hyperparameters of CNN for better performance. It is easier to reproduce this automated selection approach for problems with different domains, whereas it is a difficult task for a human expert or non-expert. This automated optimal approach will inspire the use of neural networks in diverse domains.

2. Literature survey

In this section, we will discuss a brief overview of PSO, its applications in the optimization of neural networks, effective architectures of CNN, hybrid versions of CNN with machine learning algorithms, and various contributions of PSO in the optimization of CNNs.

A survey on the contribution of Swarm Intelligence (SI) algorithms for optimization of problems has determined that Particle Swarm Optimization and Ant Colony Optimization algorithms have outperformed other SI algorithms [16]. There may not be any confirmed reason to decide a particular algorithm as a correct choice for solving a problem, but computation cost with fewer parameters to adjust is a major attraction of selecting PSO for optimization tasks. This paper compared 33 meta-heuristic methods proposed between 1960 and 2016. Its results demonstrate that PSO, Differential Evolution (DE), and Genetic Algorithm (GA) solve the largest number of problems efficiently and also acknowledged that PSO performs relatively better than other meta-heuristics with lower computation cost, DE and GA are better with higher computation cost [19]. This survey has supported our approach of using PSO for optimization of deep neural networks as these networks already have a large computation budget.

There are various methods proposed in the literature for optimizing different types of Neural Networks (NNs). A hybrid method that combines the Extreme Learning Machine (ELM) and Switching Delayed PSO algorithm (SDPSO) successfully predicted load in the power system [20,21]. ELM [11] with better generalization performance and fast learning speed had overcome the limitations of gradient-based learning in Single Hidden Layer Feed Forward Network (SLFN) [22]. But ELM algorithm with a more complex structure and excessive hidden neurons may pose ill-condition problems. An ill-conditioned system is sensitive for the selection of weights and data because the random selection of parameters does not confirm the robustness of the solution [23]. SDPSO-ELM outperforms over IPSO-ELM algorithm [23], E-ELM algorithm [11], and the basic ELM algorithm [11]. SDPSO improves the ability to reach the global optimum solution and avoids the limitation of the novel method of PSO of trapping into the local optimum solution and premature convergence [9]. PSO is also used to optimize the hidden nodes of well-trained SLFN for improvement in regression quality and attainment of better generalization performance in comparison to other constructive ELMs [24]. It consumes more time because of the use of PSO, but its outstanding performance over other ELMs will motivate researchers to apply it to solving more complex problems in classification. Manual selection of parameters and size of Radial Basis Function Neural Network (RBFNN) is very time-consuming in solving real-world problems and may also incline towards loss of generalization. Several learning algorithms were applied to optimize its model, and among those, evolutionary and gradient-based algorithms are most popular with certain limitations [25–27]. Evolutionary algorithms are more robust than gradient-based methods, and they provide global optimum results [25]. Although it has a few disadvantages as higher computational cost, overfitting problems, and sometimes convergence of the algorithm is more challenging [28]. PSO is useful in optimizing network size and parameters simultaneously of RBFNN and discover solutions faster than DE, GA, and other evolutionary methods [29].

Other contributions of PSO are optimization of Self-Organizing RBF (SORBF) [30], Fuzzy PSO based RBFNN [31], Nonlinear-Time-Varying-Evolutionary PSO (NTVE-PSO) [32] which elevates the performance of RBFNN and Adaptive-PSO-Based SORBF (APSO-SORBF) [25] that has demonstrated its effectiveness in solving nonlinear problems with higher accuracy than other SORBF NNs. Results of APSO-SORBF are compared with other methods as SORBF [25], PSO-RBF [30], Adaptive Improved PSO-based RBF (AI-PSO-RBF) [33], and stability adaptive inertia weight PSO-based RBF (SAIW-PSO-RBF) [34] algorithms with same training data sets and test datasets. The results show that the final structure of the APSO-SORBF NN is the most compact, with the best generalization performance, requires less training time, and the least testing time amongst

all. The first novel framework of automatic tuning of the deep learning model is Marginalised Stacked Denoising Auto-encoder mSDA which is faster than a Stacked Autoencoder (SDA). It was optimized using PSO [35] which performed extremely well in tuning parameters of mSDA.

Recent developments in CNN have inspired us to automate learning of its structure and make it easy to use. CNN is one of the most extensively used deep learning techniques effective in computer vision [36]. It has major advantages of parameter sharing, the sparsity of connections, and translation invariance over other deep models as Logistic regression (LR), Multi-layer perceptron (MLP). [37]. CNNs have confirmed its significant contribution in numerous applications such as image classification [38], object detection [39], neural style transfer [40], speech recognition [41], recommender systems [42], natural language processing [43] and sentence classification [44]. There are several popular and very successful architectures of CNNs are proposed in the literature. As LeNet [45,46] proposed in 1990 used to read zip code/digits, AlexNet in 2012 which has outperformed state of the art methods in ImageNet ISLVRC challenge [38], GoogLeNet introduced inception model reducing parameters of AlexNet and won ISLVRC 2014 [47], VGGNet in 2014 has shown that depth of the network has key importance in the performance of CNN [48]. ResNet, a winner of ISLVRC in 2015, introduced the concept of residual learning, Clarifai presented insight into the function of in-between layers in 2013, and SPPNet proposed spatial pyramid pooling in 2014 [49–51].

A hybrid model of CNN with Long Short-Term Memory (LSTM) detects the unsafe behavior of workers on the construction site [3]. Many large-scale problems have received substantial attention from researchers working in machine learning and computer vision [52 53]. CNN with Naïve Bayes has demonstrated its successful application in object detection, analyzing each frame of video for detecting cracks on surfaces of nuclear reactors [4]. Experimental results show that CNN+NB performs better than LBP-SVM. Although it needs a large amount of annotated data and computation depends heavily on GPU [4]. Continuing with the literature of hybrid versions of CNN with other machine learning algorithms, researchers found that evolutionary algorithms can improve the performance of CNN by optimizing its parameters and network structure which is a complex problem. Though an automatic selection of training parameters of CNN is still a challenge [35], one of the few contributions of PSO in improving the accuracy of CNN based classification method is using Darwinian PSO for selecting informative bands of hyperspectral data of Indian Pines and Pavia University [2]. Traditional feature selection techniques require a large sample size to perfectly classify the test data but the issue of lack of sufficient training samples is addressed by optimal feature selection using evolutionary algorithms such as PSO and GA [54].

Fractional order Darwinian PSO (FODPSO) overcomes the limitation of premature convergence of traditional PSO algorithm that may cause due to lack of diversity in particles [55]. This self-improving CNN handled the unavailability of sufficient training samples using FODPSO based feature selection and solved the problem of overfitting using the dither method [56]. There is another approach of optimizing parameters of layers of CNN using PSO [5]. It has shown improvement in accuracy achieved from Alexnet-CNN architecture on CIFAR datasets. Genetic programming optimizes the architecture of CNN for image classification problems [7]. Evolving deep CNN (EvoCNN) determined layers in its architecture and weights for nine standard datasets using GA. This approach has outperformed over 22 states of the art algorithms [8]. An optimal architecture of CNN, proposed in another paper is a recent development is applying PSO for optimizing CNN parameters and structure for image classification problem [6]. It is IP-based variable-length particle encoding and a disabled layer for learning variable-length architecture with a fraction of datasets taken at a time to speed up the procedure.

Thus vast literature relevant to optimization of parameters of different types of neural networks using PSO is available but swarm inspired optimization of hyperparameters in CNN is still a thirst area and is required to explore more. This proposed approach does not require extensive knowledge to tune CNN's hyperparameters but a certain level of human efforts is essential for specifying a range of search space for PSO to find the optimal solution.

3. Proposed method

3.1. Particle Swarm Optimization

PSO is a metaheuristic, stochastic population based evolutionary optimization algorithm. It searches for the optimal solution in search space through swarm. Each particle in the swarm survives with a velocity and position in solution search space. Algorithm 1 shows the standard PSO algorithm developed by Kennedy, Eberhart [9]. The lower and upper bound of each dimension of the particle is denoted by lb and ub in the algorithm. It improves the best solution traversed so far by iteratively updating its velocity and position in the search space. Simple formulas of updating movement of an i^{th} particle in $(k+1)^{\text{th}}$ iteration are mentioned below:

$$\overrightarrow{v_i}(\mathrm{k}+1) = \omega v_i(\mathrm{k}) + c_1 \mathrm{r}_1(\overrightarrow{pbest}_{\mathrm{i}}(\mathrm{k}) - \overrightarrow{x}_i(\mathrm{k})) + c_2 r_2(\overrightarrow{gbest}(\mathrm{k}) - \overrightarrow{x}_{\mathrm{i}}(\mathrm{k})) \quad (1)$$

$$\vec{x}_{\mathrm{i}}(\mathrm{k}+1) = \vec{x}_{\mathrm{i}}(\mathrm{k}) + \overrightarrow{v_i}(\mathrm{k}+1) \quad (2)$$

The movement of each particle is influenced by the social and cognitive behavior of the particle. Social behavior is decided by swarm's best experience ($\overrightarrow{gbest}$) and cognitive behavior by particle's own best experience ($\overrightarrow{pbest}_{\mathrm{i}}$) travelled so far. Two random variables r_1 and r_2 take values between 0 and 1. Inertia weight (ω), social coefficient (c_1), and cognitive coefficient (c_2) are parameters to control the behavior of particles and maintain the balance between its exploration and exploitation properties. It has a large impact on performance and hence various optimization methods are proposed in the literature to optimize these parameters. Inertia weight (ω) which is the most important parameter [57] can be optimized by linear [58], non-linear [59], fuzzy based [60], and random strategies [61].

3.2. Convolution Neural Networks

Fig. 1. shows first successful deep architecture or multi-layer NN architecture of CNN emerged from the concept of sub-region of receptive fields proposed by Hubel and Wisel [62] in 1960. CNNs reduce the computational complexity of traditional neural networks as Multi-Layer Perceptron (MLP) by sharing filters for the complete image. The dimension of the filter/kernel matrix is much smaller than the size of an input image, and that is the reason it reduces the requirements of the number of parameters associated with each pair of connections between input and output. It supports the property of equivariance with sparse connections that further ensures that the output varies in the same way as the input [1]. The most important contribution of CNN is the requirement of fewer parameters as compared to other traditional NNs. It reduces memory requirements and computation complexity and improves performance. It mainly includes three different types of layers: 1. Convolutional layer 2. Pooling or Sub-sampling Layer 3. Fully connected layer.

Convolutional layer extract features and learns filters using the back-propagation method. Each filter connects the output to certain inputs which may be overlapping receptive fields of the previous layer. The filter has weight and bias parameters to learn and these parameters are shared by multiple regions. The spatial relationship among neighboring pixels helps in the learning process. Convolution operation enables layers to recognize the same object in multiple images with different locations. The pooling layer is used to reduce the dimension of input to avoid overfitting, leads to better robustness, and also to maintain translation invariance towards feature transformation. Max pooling method summarizes the statistics of the output of previous layers but contains at most information [63]. Max pooling layer is included in between convolution layers to preserve features detected anywhere and is performed

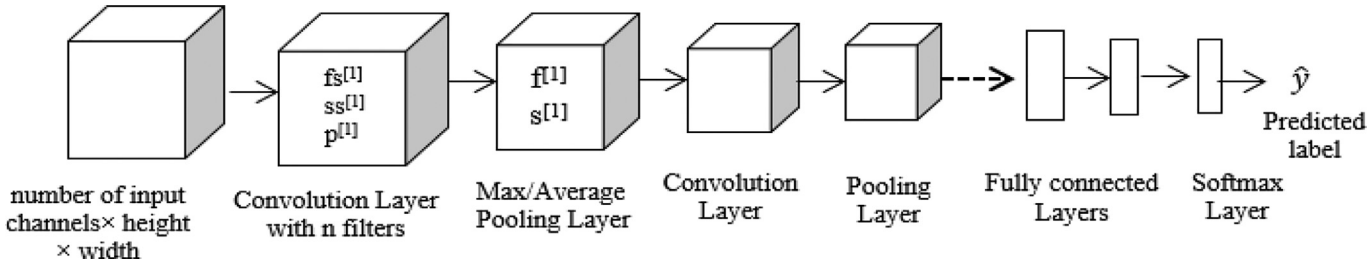

Fig. 1. Convolutional neural network architecture.

independently for each channel. It leads to better robustness and invariance. Another method is average pooling that can also be used, but max pooling is used widely due to its faster convergence property. Some other relevant approaches are stochastic pooling for better generalization, spatial pyramid pooling for improving the recognition accuracy of images of random sizes, and def-pooling for learning from the deformation of images [64]. Fully connected layers are added after the last pooling layer. The first fully connected layer reduces the dimension of the feature vector obtained from the previous layer to a single-dimensional vector. It is densely connected with other layers and hence requires many parameters with the increase in efforts in computation. The final fully connected layer is followed by classification layer or a regression layer.

3.3. The proposed method: MPSO-CNN

The training of CNN involves a backpropagation method, which has mainly two steps. The first step is the feed-forward step in that significant features are obtained by applying multiple filters at each layer. These features pass through different layers in a forward direction, and the final layer computes actual output. In the second step, the error that is a difference between the actual output and the expected output is computed and then back propagated to previous layers for fine-tuning of parameters using a gradient descent approach. Various regularization methods are encouraged to use to increase generalization ability and reduce overfitting.

This section describes the basic foundation of hybrid approach, flow diagram, algorithms and detailed architecture of the proposed MPSO-CNN. As shown in Fig. 2, MPSO uses multiple swarms to evolve the architecture of CNN and also its hyperparameters. Each particle represents a possible configuration of CNN. The final layer of CNN is a softmax classification layer to compute the probability of occurrence of samples of each class. The obtained accuracy represents the fitness value of each particle. PSO iteratively optimizes hyperparameters of a CNN, and it finally converges to a configuration with the best fitness value. The evolved CNN architecture with an optimal set of hyperparameters will be trained with a larger number of training samples in a go. The optimized CNN with trained parameter values is used further for the classification of unknown samples.

The workflow diagram of MPSO-CNN is shown in Fig. 3. A swarm $[P_1, P_2 \ldots P_m]$ at level-1 is initialized to random values of convolution, pooling, and fully connected layers. Multiple swarms ($[P_{11}, P_{12\ldots} P_{1n}]$, $[P_{21}, P_{22} \ldots P_{2n}] \ldots [P_{m1}, P_{m2} \ldots P_{mn}]$) are initialized at swarm level-2 where each swarm has n particles. For each particle P_i of swarm level-1, a swarm $[P_{i1}, P_{i2} \ldots P_{in}]$ is initialized at level-2 as presented in Algorithm 2. The dimension of P_{ik} depends on the number of parameters listed for hidden, pooling, and fully connected layers in P_i. Each particle of swarm level-2 is randomly initialized corresponding to the number of filters, filter size, stride size, padding for convolution layer, filter size, stride size and padding requirements for pooling layer, and the number of output neurons for fully connected layers. Now CNN extract features and softmax layer calculates accuracy (fitness value) for each particle of swarm level-2. The velocity and position of particles of each swarm are computed using Eqs. (1) and (2). Personal and global best experiences are updated accordingly. This procedure repeats until the termination criterion is met as shown in Algorithm 3. Now, the fitness value of each P_i is set to the global best value obtained from its associated swarm $[P_{i1}, P_{i2\ldots.}P_{in}]$. The personal and global best of $[P_1, P_2 \ldots P_m]$ are updated along with its velocity and position as presented in Algorithm 2. This procedure is repeated for swarm level-1 followed by swarm level-2 and global best value obtained after the termination of the algorithm gives maximum accuracy for the CNN obtained in the given search space. Particle traversed so far with minimum error value is represented as $[P_i, P_{ij}]$ where P_i represents a particle of swarm level-1 with the number of layers of each type and P_{ij} represents a particle of swarm level-2 providing all hyperparameters required at each layer of an evolved CNN.

We used sigmoid like inertia weight as mentioned in Eq. (3) for computing velocity. It is preferred to maintain a balance between exploration and exploitation in search space to avoid premature convergence. In its formulation symbol "t" denotes current iteration, t_{max} is the maximum number of iterations, the value of α is set to 0.2 which has shown significant improvement in results as mentioned in paper [18].

$$\omega(t) = \{\{ \begin{matrix} 0.9 \ when \ t < \alpha t_{max} \\ \frac{1}{1+ e^{(10t-2t_{max})/t_{max}}} \ otherwise \end{matrix} \quad (3)$$

Detailed design of proposed system of Hybrid MPSO-CNN is shown in Fig. 4 followed by description of each step of the process.

Step 1. Swarms initialization in hyperparameters search space

Table 1 describes the minimum and maximum values of hyperparameters that controls the dimensions of the particles in the search space. The specified range of hyperparameters is used for all benchmark datasets in experimental analysis. The multi-level PSO optimizes 11 hyperparameters of a Convolutional Neural Network. The first level of the swarm has three hyperparameters in a particle: number of convolution layers (nC), number of pooling layers (nP), number of fully connected layers (nF), and the second level of a swarm contains eight hyperparameters: number of filters in the convolutional layer (c_nf), size of filter/kernel in the convolutional layer (c_fs), padding (valid or same) requirement in the convolutional layer (c_pp), size of stride in the convolutional layer (c_ss), size of a filter in the max-pooling layer (p_fs), size of stride in the max-pooling layer (p_ss), padding pixels in pooling layer (p_pp) and the number of output neurons in the fully connected layer (op). The particles are randomly initialized in the specified range to determine the optimal set of hyperparameters of a CNN. We have limited the search space constrained to the resources available to us while ensuring that the suggested approach explores the search space exhaustively and obtained results confirm its effectiveness for hyperparameters optimization.

Step 2. PSO algorithm parameter values

Table. 1 defines the range of hyperparameters of a CNN architecture and is used for the initialization of a particle. Table 2 shows the parameters of the swarm for controlling the movements of the particles in the provided search space as mentioned in Table 1. The dimension

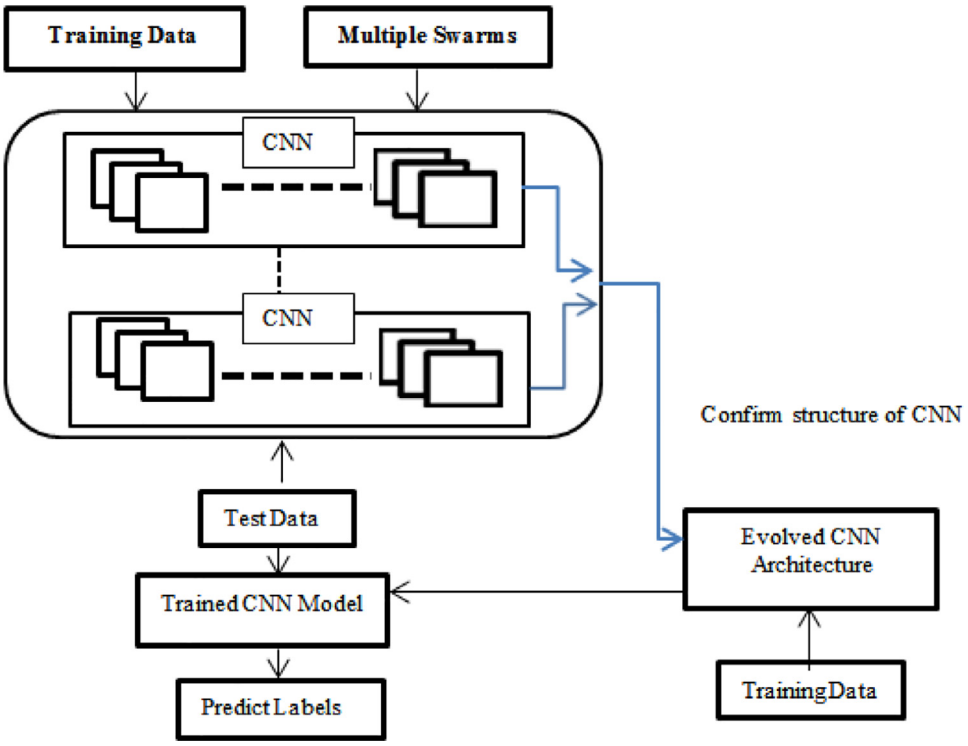

Fig. 2. Hybrid MPSO-CNN basic building block diagram.

Table 1
Range of hyperparameters.

Convolution Neural Network (CNN) Architecture			
Layers	Hyperparameter	Range	
		Minimum value	Maximum value
Convolution	1. Number of convolutional layers (nC)	1	5
Pooling	2. Number of pooling layers (nP)	1	5
Fully Connected	3. Number of fully connected layers (nF)	1	5
Convolution	1. Number of filters (c_nf)	1	64
	2. Filter Size (c_fs) (odd)	1	13
	3. Padding pixels (c_pp)	0 (valid)	1 (same) $p = \frac{c\_fs-1}{2}$
	4. Stride Size (c_ss)(< c_fs)	1	5
Pooling	5. Filter Size (p_fs)(odd)	1	13
	6. Stride Size (p_ss)	1	5
	7.Padding pixels (p_pp) (< p_fs)	0 (valid)	1 (same) $p = \frac{p\_fs-1}{2}$
Fully Connected	8. Number of neurons (op)	1	1024

Table 2
Values of parameters.

Particle Swarm Optimization (PSO)	
Parameter	Value
Swarm size at Swarm Level-1 (nP≤ nC, nF ≤ nC)	5 × 3
Swarm size at Swarm Level-2	5 X nC X 8
Social coefficient (c_1)	2
Cognitive coefficient (c_2)	2
Inertia weight (ω)	as in Eq. (3)
Maximum iterations at Swarm Level-1	between 5 and 8
Maximum iterations at Swarm Level-2	5

of the updated position vector of the particle is truncated to the integer value closest to the lower value in the range. We can extend this search space for exploring deeper CNN architectures that will depend on the availability of the computational resources and time.

Table. 2 shows the values of parameters fixed for implementation of multi-level swarms of PSO. The swarm at level one (SL1) has 5 particles and each particle is a vector of size 3. Thus, the dimension of the swarm at this level is 5 × 3. The first element represents the number of convolutional layers (nC), the second element represents the number of pooling layers (nP) and the third element represents the number of fully connected layers (nF). The generated number of pooling layers and fully connected layers will always be less than the number of convolutional layers. It helped us to frame the structure of two-dimensional particles of the swarm at level two (SL2). The social and cognitive coefficients c_1 and c_2 are set to integer value 2, as stated in the basic version of PSO [9]. Although the choice of parameters plays a major role in optimization performance, we have used the simplest version of PSO with a modification in computing inertia coefficient. Inertia weight (ω) is calculated using Eq. (3) which generates the values starting from 0.9 and then it decreases gradually towards zero.

Step 3. Structure of swarms

Figs. 5 and 6 explain the structure of multi-level, multi-dimensional swarms modeled at swarm level-1 and swarm level-2 for better understanding of its working.

There is a swarm at level-1 which has five particles, each of them represents a configuration of the number of layers of each type as shown

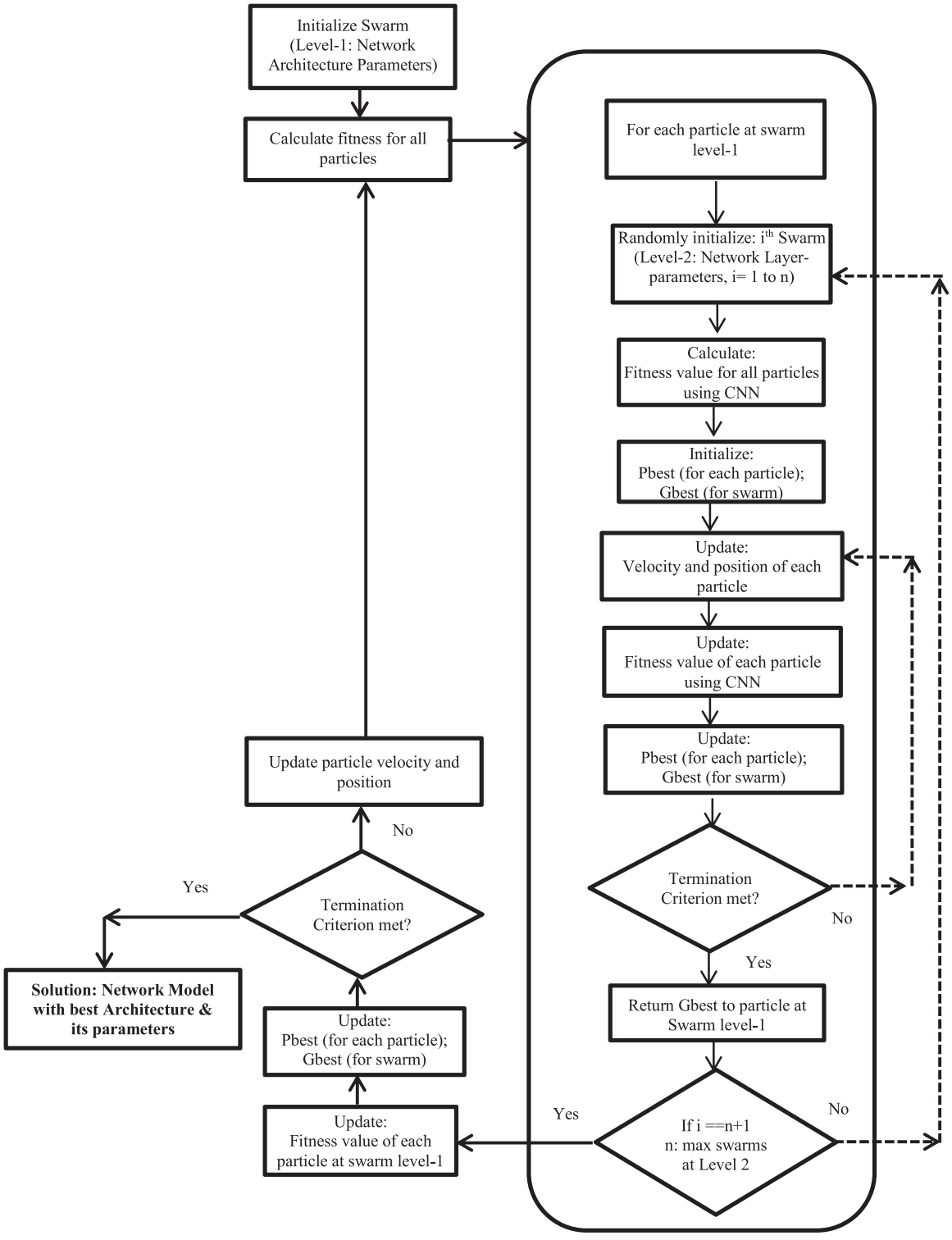

Fig. 3. Hybrid MPSO-CNN flow diagram.

in Fig. 5. Each particle is extended at level-2 with a population of a possible set of hyperparameters that a CNN may converge to at the end of evolution as shown in Fig. 6. Five swarms are explored at level-2, where each swarm is modeled according to the number of layers specified in a particle at level-1. Overall five swarms will be generated at level-2. Each swarm at level-2 has five particles and each particle is of dimension $nC \times 8$ as shown in Fig. 7. Thus, the dimension of the swarm at level-2 is $5 \times nC \times 8$. These eight parameters that need to be optimized are the number of filters, filter size, padding bits, stride size in a convolutional layer, size of the filter, stride size, and padding bit in a pooling layer, and the number of output neurons in a fully connected layer.

MPSO iterates to determine the best CNN configuration in a search space as specified in Table 1. Swarm with five particles at level-1 iterates five times. Each particle at level-1 decides the dimension of particles of swarm generated at level-2 for finding the hyperparameters used in

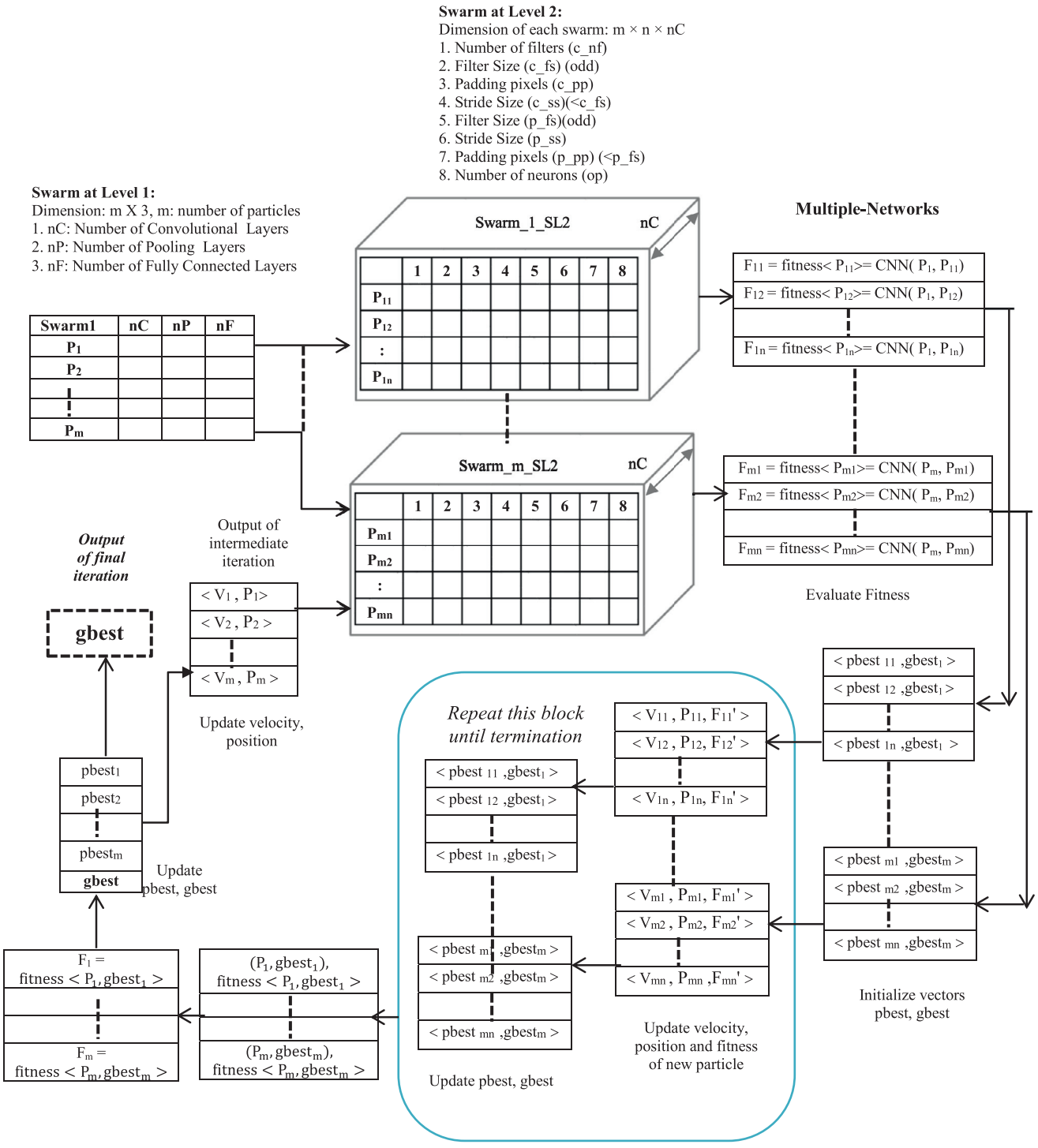

Fig. 4. Hybrid MPSO-CNN system architecture.

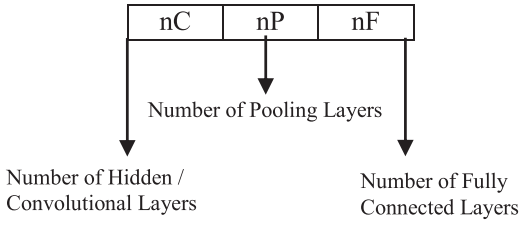

Fig. 5. Structure of a particle at swarm level-1.

each layer. 5 particles at level-2 are generated to exploit the possible search space provided by each particle at SL-1. Thus, a total of 625 CNNs were evaluated to find the best configuration for each dataset with better performance than default settings and comparable performance with state-of-the-art methodologies. We have extended the number of maximum iterations for CIFAR-10, CIFAR-100, and MDRBI datasets. It helped us to find better configurations for these datasets. The evaluation of any CNN is independent of another one and therefore its steps can run in parallel. We have executed the algorithm for 20 times independently

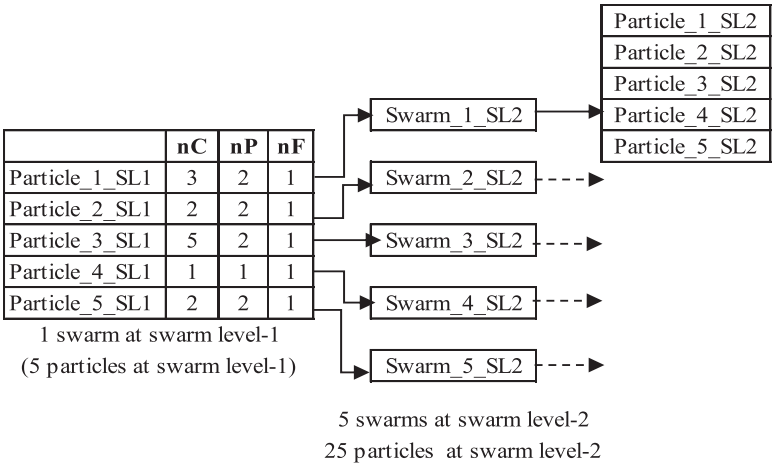

Fig. 6. Hierarchical structure of swarms at level-1 and level-2.

Convolutional layer				Pooling layer			Fully Connected Layer
Number of filters	Filter Size (Odd)	Padding pixels	Stride Size	Filter Size	Stride Size	Padding pixels	Number of neurons
c_nf1	c_fs1	c_pp1	c_ss1	p_fs1	p_ss1	p_pp1	op1
c_nf2	c_fs2	c_pp2	c_ss2	p_fs2	p_ss2	p_pp2	op2
c_nf3	c_fs3	c_pp3	c_ss3	p_fs3	p_ss3	p_pp3	op3
c_nf4	c_fs4	c_pp4	c_ss4	p_fs4	p_ss4	p_pp4	op4
c_nf5	c_fs5	c_pp5	c_ss5	p_fs5	p_ss5	p_pp5	op5

Particle_1_SL2 : Two dimensional particle of swarm level-2

Convolutional layer				Pooling layer			Fully Connected Layer
Number of filters	Filter Size (Odd)	Padding pixels	Stride Size	Filter Size	Stride Size	Padding pixels	Number of neurons
57	1	'same'	1	5	1	'valid'	944
3	1	'valid'	2	5	5	'same'	0
17	5	'same'	4	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Particle_1_SL2 : Values of two dimensional particle of swarm level-2

Fig. 7. Structure of a 2-dimensional particle at swarm level-2.

on each dataset to study the nature of the convergence of the multi-level, multi-dimensional PSO algorithm.

Sep 4. Fitness evaluation

In the proposed MPSO, CNN followed by softmax layer evaluates the fitness of each particle i.e. accuracy. The set of hyperparameters of CNN that provide better accuracy than another is fittest solution in comparison to another particle with configuration that provides lower accuracy. Fitness of the particle is computed at level-2 for a set of parameters represented by (P_i, P_{ij}) as in Eq. (4). P_i is a particle at swarm level-1 and P_{ij} is particle at swarm level-2. Both particles define all the hyperparameters required to model a CNN architecture.

$$\text{Fitness } (\mathrm{P_{ij}}) = \text{CNN } (\mathrm{P_i}, \mathrm{P_{ij}}) \tag{4}$$

where $P_i = (nC, nP, nF)$, $P_{ij} = (\text{c\_nf}, \text{c\_fs}, \text{c\_pp}, \text{c\_ss}, \text{p\_fs}, \text{p\_ss}, \text{p\_pp}, \text{op})$

$$V_{ij} = \omega V_{ij} + c_1 r_1 (pbest_{ij} - P_{ij}) + c_2 r_2 (gbest_i - P_{ij}) \tag{5}$$

$$V_{ij} = \begin{cases} V_{max}, & if\ V_{ij} > V_{max} \\ V_{min}, & if\ V_{ij} < V_{min} \end{cases} \tag{6}$$

where $V_{max} = P_{ij}^{max} - P_{ij}$, $V_{min} = P_{ij}^{min} - P_{ij}$

$$P_{ij} = P_{ij} + V_{ij} \tag{7}$$

$$V_i = \omega V_i + c_1 r_1 (pbest_i - P_i) + c_2 r_2 (gbest - P_i) \tag{8}$$

$$V_i = \begin{cases} V_{max}, & if\ V_i > V_{max} \\ V_{min}, & if\ V_i < V_{min} \end{cases} \tag{9}$$

where $V_{max} = P_i^{max} - P_i$, $V_{min} = P_i^{min} - P_i$

$$P_i = P_i + V_i \tag{10}$$

$$\text{Fitness } (\mathrm{P_i}) = \text{CNN } (\mathrm{P_i}, gbest_i) \tag{11}$$

Velocity (V_{ij}) of each particle($\mathrm{P_{ij}}$) is computed using Eq. (5) with $pbest_{ij}$ as the best experience of j^{th} particle of i^{th} swarm at level-2 and $gbest_i$ is the best experience of i^{th} swarm at level-2. The maximum and minimum velocity of the particle of swarm level-2 is referred from Table 1. It controls the velocity of the particle $\mathrm{P_{ij}}$ as mentioned in Eq. (6) and the new position of the particle is given in Eq. (7) which updates the hyperparameters used in each layer of CNN. Further, the velocity of i^{th} particle of swarm level-1 is computed as in Eq. (8) and controlled according to Eq. (9). The new position of the particle P_i is calculated in Eq. (10) which updates the number of layers of each type. Thus, the fitness of P_i depends on the layers that it consists of itself and the global best hyperparameters that it obtained from its extended level of the particle in i^{th} swarm, also shown in Eq. (11). The solution finally converges to the optimal best solution gbest which will be the maximum of $gbest_1, gbest_2, \ldots, gbest_m$.

3. Implementation results

The selection of proper architecture and hyperparameters of a CNN model is a very time-consuming process when it is to be decided on a trial-and-error basis. Thus, it is required to develop an automated approach which can reach to the best CNN model with minimal human expertise and efforts. The proposed work uses the random but guided nature of PSO to find the best structure of CNN. It optimizes its hyperparameters on given datasets in predefined search-space. The optimization process requires human intervention at the beginning to decide this search-space. In this section, we demonstrate the effectiveness

Table 3
Benchmark Datasets.

Datasets	Training samples	Test samples	Number of classes	Input image size (height × width × input channels)	Images
MNIST	60,000	10,000	10	28 × 28 × 1	Grayscale
CIFAR-10	50,000	10,000	10	32 × 32 × 3	Colored
CIFAR-100	50,000	10,000	100	32 × 32 × 3	Colored
MDRBI	12,000	50,000	10	28 × 28 × 1	Grayscale
Convex Set	8000	50,000	2	28 × 28 × 1	Grayscale

Table 4
Parameter values for a single independent run of MPSO-CNN with median value of accuracy obtained out of 20 independent runs of MPSO-CNN.

Datasets	Parameter values			
	Average number of trainable parameters in one CNN model	Total CNNs evaluations for this independent run	Average execution time of one CNN Model (in s)	Attained validation accuracy (%)
MNIST Dataset	283,258	625	457.696	99.54
CIFAR-10 Dataset	416,356	750	212.168	93.33
CIFAR-100 Dataset	340,523	750	1848.816	74.56
Convexset Dataset	454,318	625	92.868	96.55
MDRBI Dataset	327,602	1000	3127.53	69.77

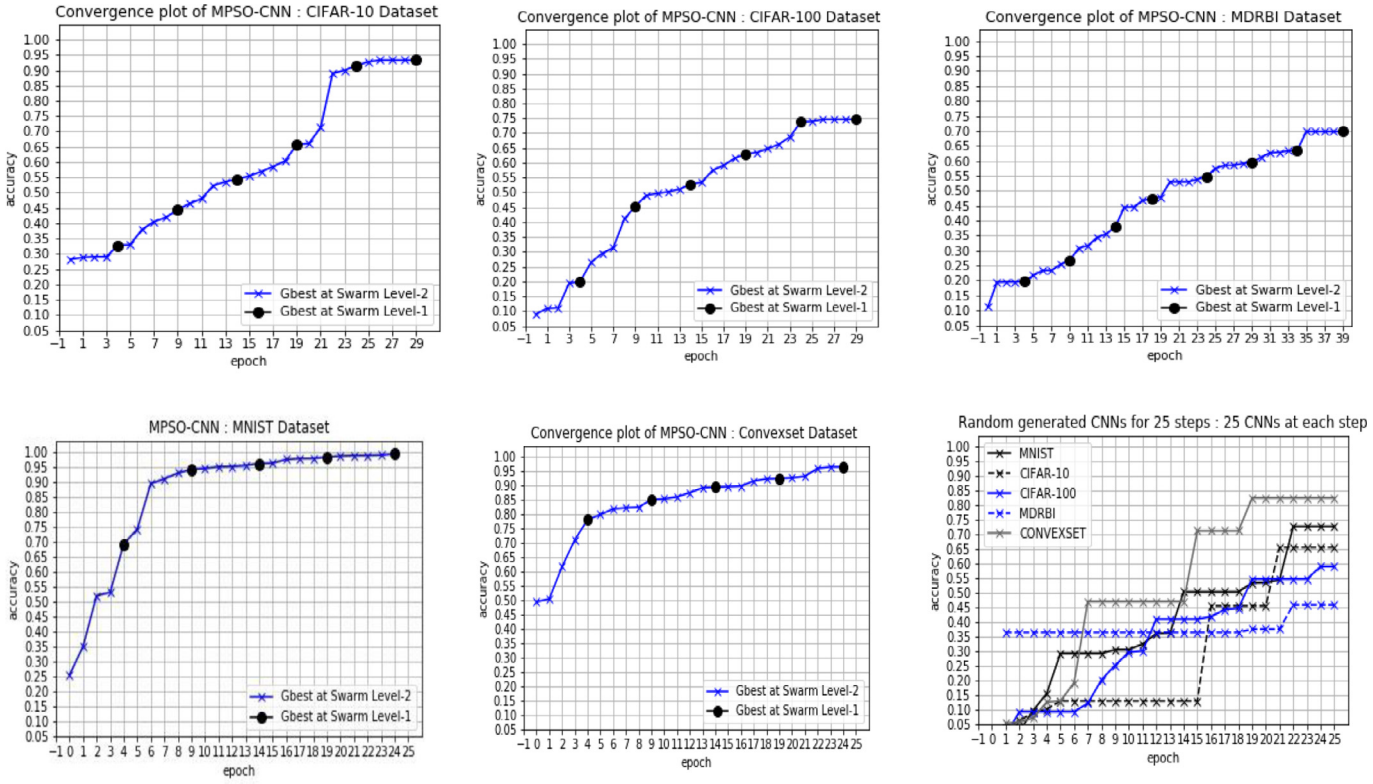

Fig. 8. Evolution of solution obtained from MPSO-CNN and random generated CNNs for 5 datasets: MNIST, CIFAR-10, CIFAR-100, Convexset, and MDRBI (Median accuracy plot).

of MPSO-CNN on five benchmark datasets. The proposed hybrid MPSO-CNN model is implemented, trained and tested with Keras framework on Google's Colaboratory.

3.1. Benchmark datasets

We trained and tested the MPSO-CNN model on five benchmark datasets. These are the MNIST dataset of handwritten digits, CIFAR-10 and CIFAR-100 datasets of colored images, MDRBI dataset of rotated MNIST images with black and white background, and CONVEX dataset of convex and non-convex images. A description of the datasets is given in Table 3.

3.2. Result analysis

The range of the hyperparameters to evolve the configurations is set manually as mentioned in Table 1 and movement of the particles is guided by PSO parameters as given in Table 2. Firstly, we observe how the PSO evolves the population of possible configurations of CNN and iteratively improves the performance. It reaches the better configuration at the end. The algorithm converges to the optimum solution in a given search-space and a specified limited number of iterations. The configuration search space and the maximum number of iterations can be increased depending on the computation time and resources available to us. Next, we present the average accuracy obtained on five bench-

Table 5
Comparison of results obtained from MPSO optimized CNN models and randomly generated CNN models (20 runs).

	MNIST dataset		CIFAR-10 dataset		CIFAR-100 dataset		Convexset dataset		MDRBI dataset	
	Average accuracy (%)	Best accuracy (%)	Average accuracy (%)	Best accuracy (%)	Average accuracy (%)	Best accuracy (%)	Average accuracy (%)	Best accuracy (%)	Average accuracy (%)	Best accuracy (%)
Randomly generated CNN	51.61	74.34	42.81	58.49	39.54	62.9	73.47	82.5	31.62	52.98
MPSO optimized CNN	99.13	99.89	87.34	89.56	66.97	71.55	90.33	92.73	65.77	68.1

Table 6
Experimental results on average accuracy.

MNIST dataset	Avg. accuracy (%)	CIFAR100 dataset	Avg. accuracy (%)
PSO-mSDA[35]	98.70	Alexnet [5]	49.87
1-layer DAE [35]	98.63	PSO-CNN [5]	53.28
2-layer DAE [35]	98.71	MPSO-CNN	66.97
mlDAE [35][65]	92.83	Deep Net [68]	67.76
MPSO-CNN	99.13	HD-CNN [69]	67.38
CIFAR10 Dataset	Avg. Accuracy (%)	Convex Sets (CS) Dataset	Avg. Accuracy (%)
Alexnet [5]	77.75	IPPSO [6]	87.94
PSO-CNN [5]	80.15	EvoCNN [8]	94.61
MPSO-CNN	87.34	MPSO-CNN	90.33
ReLU-CNN [66]	88.8	MDRBI Dataset	Avg. Accuracy (%)
ReNet [67]	87.65	IPPSO [6]	67
CGP-CNN (ConvSet) [7]	93.25	EvoCNN [8]	62.62
CGP-CNN (ResSet) [7]	94.02	MPSO-CNN	65.77

mark datasets for 20 independent runs, and the results obtained from other state-of-the-art methods. It recommends that optimization methods can be used to automate the selection of the best CNN model and it is possible to achieve even better results if we increase the number of particles at both levels, search space of hyperparameters, and the number of iterations.

A *Convergence plots of median value of 20 independent runs for each dataset.*

In this section, we analyze the results of 20 independent runs. Table 4 shows the average number of trainable parameters in one CNN model for each dataset, the total CNNs evaluated in an independent run that is decided by the number of particles and the number of iterations at each level of swarm. It also shows the average execution time for a CNN model and attained validation accuracy at the end of applying all steps of this stochastic method of MPSO–CNN. All runs take 128 mini-batch size, RELU activation function and dropout rate of 20%. CNN generated features are used in softmax layer for classification of images.

Fig. 8 shows the convergence plot of an independent run out of 20 runs of MPSO-CNN which has obtained median accuracy. Further, we have also shown the randomly generated CNNs for 25 iterations, 25 CNNs are mutated at each step and the best of the accuracies is indicated after each iteration. As shown in Fig. 8, the solution converges or starts converging after a few iterations to the potential solution. The nature of the convergence of the stated algorithm is examined and the execution is interrupted after the solution gets stable. The required number of iterations for convergence depends on datasets.

The results demonstrate here that proposed structure of particles of swarm search the hyperparameters of CNN efficiently and improves the performance for each dataset. The PSO guided search can further explore more search space to obtain much better solutions than what has been shown. The results shown here are of an independent run which shows the median of fitness values obtained from 20 independent runs of MPSO-CNN. Each run starts with a new random set of hyperparameters. The random generated CNNs are initialized with the same random set of hyperparameters as in the MPSO-CNN model presented here for a fair comparison of performances. This search is not guided by PSO and 25 steps are executed with random generations of CNNs. Each step generates 25 CNNs (as in swarms at level-2) and the best configuration among 25 CNNs is noted. Total 625 CNNs are generated at random and results have shown that MPSO guided CNN search is a systematic approach that leads to better performance in comparison to a random search of a CNN model. The complexity of the MPSO-CNN is on the higher side, it requires more computational resources in comparison to random selection but it has its own benefits of performing better than default setting of configuration. It systematically removes the least performing configurations, and also reduces human efforts. Experimental setup is same for MPSO-CNN and random generated CNNs.

The comparison of results obtained from PSO generated CNN models and randomly generated CNN models is shown in Table 5. The average accuracy and best accuracy of MPSO optimized CNN is better than the randomly generated CNNs for MNIST, CIFAR-10, CIFAR-100, CONVEX set and MDRBI datasets. The guided method outperforms the completely random method. It automatically search for the best CNN configuration which is a difficult task for a non-expert user.

A *Contribution of the proposed method of particle swarm optimized CNN architecture*

This section demonstrates the contribution of proposed MPSO-CNN, an automated selection and optimization method and represents its existence along with other state-of-the-art methods. The solution space for the parameters of CNNs is mentioned in Table 1. We may extend the maximum value and can use Graphics Processing Unit (GPU) for fast processing of the algorithm. We have taken a limited range to show the effectiveness of MPSO to converge to the solution in a given search space. Experiments are ex-

Algorithm 1
Standard PSO Algorithm.

```
for each particle i = 1 to m d
    a. Randomly initialize the particle's position with uniform distribution in given range: x_i ← U(lb,ub)
    b. Calculate fitness value of x_i: fitness(x_i)
    c. Initialize particle's best position to its initial position: pbest_i ← x_i
    d. Initialize swarm's best position to the best of all particles' position: gbest ← pbest_i
end
while (termination criterion or maximum number of iterations are not encountered)
    for each particle i = 1 to m do
        for each dimension of particle d = 1 to n
            Update the particle's velocity (v_id) using Eq. (1)
            Update the particle's (x_id) position using Eq. (2)
        end
        if fitness(x_i) < fitness(pbest_i) then
            Update particle's best position: pbest_i ← x_i
                if fitness(pbest_i) < fitness(gbest) then
                    Update swarm's best position: gbest ← pbest_i
    end
end
```

Algorithm 2
Hybrid MPSO-CNN algorithm at swarm level-1.

Input: maximum number of iterations ($tmax^1$) and search space for hyperparameters
Output: (CNN hyperparameters: *[nC, nP, nF, c_nf, c_fs, c_pp, c_ss, p_fs, p_ss, p_fs, op]*, fitness value*)*

Algorithm:

```
initialize particle's position vector in specified range: [nC, nP, nF]
while (maximum number of iterations is not reached: tmax^1)
Calculate ω using Eq. (3)
    for each particle i = 1 to m of swarm at level-1 do
        find hyperparameters and its fitness value (as in Algorithm 3)
        update fitness value: F_i =fitness<P_i > = fitness (P_i, gbest_i)
        update personal best: pbest_i
        update global best: gbest
        update particle's velocity and position: (V_i, P_i)
    end
end
return(gbest, CNN (gbest))
```

Algorithm 3
Hybrid MPSO-CNN algorithm at swarm level-2.

Input: particle (P_i) of swarm level-1, maximum number of iterations ($tmax^2$) and search space for hyperparameters
Output: (CNN hyperparameters, fitness value))

Algorithm:

```
for each particle j = 1 to n of swarm at level-2 do
    initialize particle's position in specified range: [c_nf, c_fs, c_pp, c_ss, p_fs, p_ss, p_fs, op]
    setup a CNN model: CNN(P_i, P_ij)
    compute fitness value using CNN: F_ij
    initialize personal best: pbest_ij
    initialize global best: gbest_i
end
while (maximum number of iterations is not reached: tmax^2)
    for each particle j = 1 to n of swarm at level-2 do
        update particle's velocity and position: (V_ij, P_ij)
        setup a CNN model: CNN(P_i, P_ij)
        compute fitness value using CNN: F'_ij
        update personal best: pbest_ij
        update global best: gbest_i
    end
end
return((P_i, gbest_i), CNN(P_i, gbest_i))
```

ecuted 20 times and the average accuracy obtained is presented in Table 6. Results for MNIST dataset is compared with PSO-mSDA[35], 1-layer DAE [35], 2-layer DAE [35] and mlDAE [65]. MPSO-CNN performed with 99.13% of average accuracy which is better than others. The average accuracy for the CIFAR-10 dataset is 87.34 which is better than Alexnet [5], PSO-CNN [5], and competitive with ReLU-CNN [66] and ReNet [67]. The average accuracy of CIFAR-100 is 66.97 which is better than Alexnet [5] and PSO-CNN [5] but reasonably lesser than Deep Net [68]. Results for Convex Sets are comparable with IPPSO [6] and lesser than EvoCNN [8], whereas average accuracy for MDRBI dataset is better than results quoted for IPPSO [6] and EvoCNN [8]. It is the overview of results showing that MPSO-CNN performs significantly better but the results can clearly be compared when implemented in the same environment with identical computational resources.

The structure of MPSO supports parallel implementation on multiple machines, it can be extended to more levels optimizing more hyperpa-

rameters. We can increase the range of hyperparameters search-space, the population size, and the number of iterations also to explore and exploit the solution in a better way. It will increase the movements of particles in the larger space and may converge to a better CNN structure than before. Most importantly the stated method automatically decides the configuration of CNN and requires the least efforts of human experts.

4. Conclusion and future work

In this work, we have exhaustively surveyed the literature on evolutionary algorithms and explored their contributions in optimizing neural networks. The relevant state-of-the-art methods determine that there is a good scope of applying evolutionary based optimization methods for neural architectural search and hyperparameters optimization. It is substantially important to reduce the overhead of human experts to determine the architecture and its hyperparameters through trial and error basis which is computationally expensive and may lead to inefficient solutions.

The proposed MPSO approach has given satisfactory results in limited search space which can be extended for attaining a more effective structure of CNN. The MPSO has a simple hierarchical representation of swarms. It has fewer parameters to compute in comparison to other evolutionary based methods. Swarm at first level optimizes network layers and swarms at level two optimize hyperparameters used in each layer. It also reasonably explore the number of layers and then exploit the space with a possible set of hyperparameters for the stated layers. We have used sigmoid like-inertia to generate population and guide its movements to avoid premature convergence. The results on five benchmark datasets confirmed the successful application of MPSO-CNN to automate the selection of structure of CNN without much human intervention.

The process is computationally expensive but efficient and the learned optimized CNN structure may be used to solve other similar problems. The populations generated can be implemented in parallel. It will be easier for a non-expert user to identify the CNN model suitable for an application. This approach can further be enhanced by expanding the search space, increasing the maximum number of iterations, incorporating more hyperparameters to optimize, and using other PSO variants.

Declaration of Competing Interest

None.

Author statement

Conceptualization, Methodology, Software, Writing- Original draft preparation, Results: **Pratibha Singh**.

Supervision**: Prof.** Santanu Chaudhury, Prof. Bijaya Ketan Panigrahi.

References

[1] W Liu, et al., A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.

[2] P Ghamisi, Y Chen, X X Zhu, A self-improving convolution neural network for the classification of hyperspectral data, IEEE Geosci. Remote Sens. Lett. 13 (10) (2016) 1537–1541.

[3] L Ding, et al., A deep hybrid learning model to detect unsafe behavior: integrating convolution neural networks and long short-term memory, Autom. Constr. 86 (2018) 118–124.

[4] Fu-C Chen, M.R. Jahanshahi, NB-CNN: deep learning-based crack detection using convolutional neural network and Naïve Bayes data fusion, IEEE Trans. Ind. Electron. 65 (5) (2018) 4392–4400.

[5] T. Yamasaki, T. Honma, K. Aizawa, Efficient optimization of convolutional neural networks using particle swarm optimization, in: Proceedings of the 2017 IEEE Third International Conference on InMultimedia Big Data (BigMM) 2017 Apr 19, IEEE, 2017, pp. 70–73.

[6] Bin Wang, Yanan Sun, Bing Xue, Zhang Mengjie, in: Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification, IEEE, 2018, pp. 1–8.

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

[8] Yanan Sun, Bing Xue, Mengjie Zhang, G. Yen Gary, Evolving deep convolutional neural networks for image classification, IEEE Transactions on Evolutionary Computation 24 (2) (2019) 394–407.

[9] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Network, 1995, pp. 1942–1948.

[10] J. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

[11] G.B. Huang, Q. Zhu, C. Siew, Extreme learning machine: a new learning scheme of feed forward neural networks, in: Proceedings of the 2004 International Joint Conference on Neural Networks, 2, 2004, pp. 985–990.

[12] Z. Cui, B. Sun, G. Wang, et al., A novel oriented cuckoo search algorithm to improve DV-Hop performance for cyber-physical systems, J. Parallel Distr. Comput. 103 (2017) 42–52.

[13] K. Passino, Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Syst. Mag. 22 (3) (2002) 52–67.

[14] H. Wang, Z. Cui, H. Sun, et al., Randomly attracted firefly algorithm with neighborhood search and dynamic parameter adjustment mechanism, Soft Comput. 21 (2017) 5325–5339.

[15] L. Li, Z. Shao, J. Qian, An optimizing method based on autonomous animals: fish-swarm algorithm, Syst. Eng. Theory Pract. 22 (11) (2002) 32–38.

[16] M Mavrovouniotis, C Li, S Yang, A survey of swarm intelligence for dynamic optimization: algorithms and applications, Swarm Evol. Comput. 33 (2017) 1–17.

[17] S. Leung, Y. Tang, W. Wong, A hybrid particle swarm optimization and its application in neural networks, Expert Syst. Appl. 39 (1) (2012) 395–405.

[18] D. Tian, Z. Shi, MPSO: modified particle swarm optimization and its applications, Swarm. Evol. Comput. 41 (2018) 49–68.

[19] A.P. Piotrowski, et al., Swarm intelligence and evolutionary algorithms: performance versus speed, Inf. Sci. (Ny) 384 (2017) 34–85.

[20] N. Zeng, H. Zhang, W. Liu, J. Liang, F.E. Alsaadi, A switching delayed PSO optimized extreme learning machine for short-term load forecasting, Neurocomputing 240 (2017) 175–182.

[21] N. Zeng, Z. Wang, H. Zhang, F.E. Alsaadi, A novel switching delayed PSO algorithm for estimating unknown parameters of lateral flow immunoassay, Cogn. Comput. 8 (2) (2016) 143–152.

[22] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, A. Lendasse, OP-ELM: optimally pruned extreme learning machine, IEEE Trans. Neural Netw. 21 (1) (2010) 158–162.

[23] F. Han, H.-.F. Yao, Q.-.H. Ling, An improved evolutionary extreme learning machine based on particle swarm optimization, Neurocomputing 116 (2013) 87–93.

[24] F Han, et al., An improved incremental constructive single-hidden-layer feed-forward networks for extreme learning machine based on particle swarm optimization, Neurocomputing 228 (2017) 133–142.

[25] H-G Han, W Lu, Y Hou, J-F Qiao, An adaptive-PSO-based self-organizing RBF neural network, IEEE Trans. Neural Netw. Learn. Syst. 29 (1) (2016) 104–117.

[26] P. Singla, K. Subbarao, J.L. Junkins, Direction-dependent learning approach for radial basis function networks, IEEE Trans. Neural Netw. 18 (1) (2007) 203–222.

[27] Z.B. Xu, R. Zhang, W.F. Jing, When does online bp training converge? IEEE Trans. Neural Netw. 20 (10) (2009) 1529–1539.

[28] T.H. Oong, N.A.M. Isa, Adaptive evolutionary artificial neural networks for pattern classification, IEEE Trans. Neural Netw. 22 (11) (2011) 1823–1836.

[29] J. Tian, M. Li, F. Chen, N. Feng, Learning subspace-based RBFNN using coevolutionary algorithm for complex classification tasks, IEEE Trans. Neural Netw. Learn. Syst. 27 (1) (2016) 47–61.

[30] H.M. Feng, Self-generation RBFNs using evolutional PSO learning, Neurocomputing 70 (1–3) (2006) 241–251.

[31] A. Alexandridis, E. Chondrodima, H. Sarimveis, Radial basis function network training using a nonsymmetric partition of the input space and particle swarm optimization, IEEE Trans. Neural Netw.Learn. Syst. 24 (2) (2013) 219–230.

[32] C.M. Lee, C.N. Ko, Time series prediction using RBF neural networks with a nonlinear time-varying evolution PSO algorithm, Neurocomputing 73 (1–3) (2009) 449–460.

[33] A. Nickabadi, M.M. Ebadzadeh, R. Safabakhsh, A novel particle swarm optimization algorithm with adaptive inertia weight, Appl. Soft Comput. 11 (4) (2011) 3658–3670.

[34] M. Taherkhani, R. Safabakhsh, A novel stability-based adaptive inertia weight for particle swarm optimization, Appl. Soft Comput. 38 (2016) 281–295.

[35] C Sui, M Bennamoun, R Togneri, Deep feature learning for dummies: a simple auto-encoder training method using Particle Swarm Optimisation, Pattern Recognit. Lett. 94 (2017) 75–80.

[36] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, Pattern Anal. Mach. Intell. IEEE Trans. 35 (8) (2013) 1798–1828.

[37] I.G.Y. Bengio, A. Courville, Deep Learning, MIT Press [Online], 2016 Book in Preparation For.

[38] A Krizhevsky, I Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Adv. Neural Inf. Process Syst. 25 (2012) (NIPS 2012).

[39] W. Zhiqiang, L. Jun, A review of object detection based on convolutional neural network, in: Proceedings of the Control Conference (CCC), 2017 36th Chinese, IEEE, 2017, pp. 11104–11109.

[40] L.A. Gatys, A.S. Ecker, M. Bethge, Image style transfer using convolutional neural networks, in: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),, IEEE, 2016, pp. 2414–2423.

[41] O Abdel-Hamid, Li Deng, D Yu, Exploring Convolutional Neural Network Structures and Optimization Techniques For Speech Recognition, Interspeech, Lyon, France, 2013 2013, 5 - 29 August.

[42] A. Van den Oord, S. Dieleman, B. Schrauwen, Deep content-based music recommendation, in: Proceedings of the Advances in Neural Information Processing Systems, 2013, pp. 2643–2651.

[43] R. Collobert, J. Weston, A unified architecture for natural language processing: deep

neural networks with multitask learning, in: Proceedings of the 25th international Conference on Machine learning, ACM, 2008, pp. 160–167. Jul 5.
[44] Y. Kim, in: Convolutional Neural Networks for Sentence Classification, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2014, pp. 1746–1751.
[45] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551.
[46] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.
[47] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, p. 19.
[48] K. Simonyan and A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.
[49] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[50] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional neural networks, in: Proceedings of the ECCV, 2014.
[51] K. He, X. Zhang, S. Ren, et al., Spatial pyramid pooling in deep convolutional networks for visual recognition, in: Proceedings of the ECCV, 2014.
[52] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117, doi:10.1016/j.neunet.2014.09.003.
[53] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: International Conference on Neural Information Processing Systems, 2012, pp. 1097–1105, doi:10.1145/3065386.
[54] P. Ghamisi, J.A. Benediktsson, Feature selection based on hybridization of genetic algorithm and particle swarm optimization, IEEE Geosci. Remote Sens. Lett. 12 (2) (2015) 309–313.
[55] P. Ghamisi, M.S. Couceiro, J.A. Benediktsson, A novel feature selection approach based on FODPSO and SVM, IEEE Trans. Geosci. Remote Sens. 53 (5) (2015) 2935–2947.
[56] A. J. R. Simpson, “Dither is better than dropout for regularising deep neural networks,” unpublished paper. [Online]. Available: http://arxiv.org/abs/1508.04826
[57] Y. Peng, X. Peng, Z. Liu, Statistic analysis on parameter efficiency of particle swarm optimization, Acta Electron. Sin. 32 (2) (2004) 209–213.
[58] R. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC’00, 2000, pp. 84–88.
[59] G. Chen, J. Jia, Q. Han, Study on the strategy of decreasing inertia weight in particle swarm optimization algorithm, J. Xi’an Jiaotong Univ. 40 (1) (2006) 53–56.
[60] G. Chen, J. Jia, Q. Han, Study on the strategy of decreasing inertia weight in particle swarm optimization algorithm, J. Xi’an Jiaotong Univ. 40 (1) (2006) 53–56.
[61] R. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC’01), 2001, pp. 94–100.
[62] D.H. Hubel, T.N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex, J. Physiol. 160 (1) (1962) 106–154.
[63] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[64] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, M.S. Lew, Deep learning for visual understanding: a review, Neurocomputing 187 (2016) 27–48.
[65] J. Park, I.W. Sandberg, Universal approximation using radialbasis-function networks, Neural Comput. 3 (2) (1991) 246–257.
[66] Xu, B, et al. "Empirical evaluation of rectified activations in convolutional network". arXiv preprint arXiv:1505.00853 (2015).
[67] Visin, Francesco, Kyle Kastner, Kyunghyun Cho, Matteo Matteucci, Aaron Courville, and Yoshua Bengio. "Renet: A recurrent neural network based alternative to convolutional networks." *arXiv preprint arXiv:1505.00393* (2015).
[68] R.K. Srivastava, K Greff, J Schmidhuber, Training very deep networks, Proceedings of the Advances in Neural Information Processing Systems (NIPS) (2015) 2377–2385.
[69] Z Yan, H Zhang, R Piramuthu, V Jagadeesh, D DeCoste, W Di, Y Yu, HD-CNN: hierarchical deep convolutional neural networks for large scale visual recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2740–2748.