How Meta-heuristic Algorithms Contribute to Deep Learning in the Hype of Big Data Analytics

Chapter · January 2018

DOI: 10.1007/978-981-10-3373-5_1

3 authors:

Simon Fong
University of Macau
498 PUBLICATIONS **1,714** CITATIONS
SEE PROFILE

Suash Deb
CVRCE, India
102 PUBLICATIONS **5,245** CITATIONS
SEE PROFILE

Xin-She Yang
Middlesex University, UK
418 PUBLICATIONS **19,864** CITATIONS
SEE PROFILE

How Meta-Heuristic Algorithms Contribute to Deep Learning in the Hype of Big Data Analytics

Simon Fong[1], Suash Deb[2], and Xin-she Yang[3]

[1] Department of Computer Information Science, University of Macau, Macau SAR
[2] Founding President, INNS-India Regional Chapter, Ashadeep, 7th Floor Jorar, Namkum, Ranchi 834010, JHARKHAND, INDIA
[3] School of Science and Technology, Middlesex University, London NW4 4BT, UK
ccfong@umac.mo, *suashdeb@gmail.com*, *x.yang@mdx.ac.uk*

Abstract. Deep learning (DL) is one of the most emerging type of contemporary machine learning techniques that mimic the cognitive patterns of animal visual cortex to learn the new abstract features automatically by deep and hierarchical layers. DL is believed to be a suitable tool so far for extracting insights from very huge volume of so-called big data. Nevertheless, one of the three "V" or big data is velocity that implies the learning has to be incremental as data are accumulating up rapidly. DL must be fast and accurate. By the technical design of DL, it is extended from feed-forward artificial neural network with many multi-hidden layers of neurons called deep neural network (DNN). In the training process of DNN, it has certain inefficiency due to very long training time required. Obtaining the most accurate DNN within a reasonable run-time is a challenge, given there are potentially many parameters in the DNN model configuration, and high-dimensionality of the feature space in the training dataset. Meta-heuristic has a history of optimizing machine learning models successfully. How well meta-heuristic could be used to optimize DL in the context of big data analytics is a thematic topic which we pondered on in this paper. As a position paper, we review the recent advances of applying meta-heuristics on DL, discuss about their pros and cons, and point out some feasible research directions for bridging the gaps between meta-heuristics and DL.

Keywords: Deep learning, meta-heuristic algorithm, neural network training, nature inspired computing algorithms, algorithm design.

1 Introduction

Deep learning (DL) is a new branch of machine learning mainly in the aspects of supervised learning. Given some suitable neural network architecture, logics on neuron weight updates and activation function, deep learning models and extracts high-level abstractions from voluminous data. It is usually done by using a series of inter-connected multiple processing layers setup in hierarchical structure. Since its inception in 2006 by Hinton [1], DL now is becoming one of the hottest research areas in the machine learning research community. DL has various versions which centered on collectively concept of a series of algorithms and models including but not limited Convolutional Neural Networks (CNN), Deep Boltzmann Machines (DBM), Deep

Belief Networks (DBN), Deep Representation, Recursive Auto encoders, and Restricted Boltzmann Machines (RBM), just to name a few. While their potential capabilities are to be exploited, some of the most popular applications are computer vision and image classification, by RBM and DBN.

Deep learning is regarded to be "deep" as the name coined in comparison to the well-known "shallow learning" algorithms such as Support Vector Machine (SVM), boosting and maximum entropy method and other discriminative learning methods. Those shallow learning recognizes data features mostly by artificial sampling or empirical sampling from the data, so the induced model or knowledge network learn the mappings between the features and prediction targets in a non-layer memory structure. In contrast, deep learning learns the relations between the raw data which are characterized by feature values and the targets, layer by layer, through transforming the data from raw feature space to transformed feature space. Additionally, deep structure can learn and approach non-linear function. All these advantages are beneficial to classification and feature visualization [2].

With the objective of useful deriving insights from the bottom of big data, deep learning has been formulated and tested in different research areas with respective to various performance indicators such as processing speed, accuracy and capabilities to adapt to continuous data collection environment. Case studies of DL applied on various industrial areas including image classification, pattern recognition, and natural language processing (NLP) etc, and more seem to come. For computer visions, there are proven successful examples as demonstrated in the large (CNN) scale visual recognition challenge (ILSVRC) by ImageNet [3]. Convolutional neural network is the first implemented DL tool in image classification and it showed effectiveness. The error rate drops from 25% to 15% when CNN was used over conventional neural network. Thereafter the success, the combination of techniques namely, deep learning for learning and prediction, big data or data warehousing, and GPU for parallel processing is integrated into large-scale image classification applications. Companies of search engine giants like Baidu and Google have upgraded their image searching capability using DL technology [4] in this big data analytics era.

Although DL outperformed most of the shallow learning methods and it has been tested in industrial applications, its design still carries some shortcomings. A large CNN typically is configured with millions of parameters and it is mostly trained by contrastive divergence (CD) learning algorithm which is iterative and known to be time consuming [5]. The most significant problem is that when facing very large scaled data the DNN will take several days or even months to learn, even though the greedy search strategy is in place. Regarding this, many companies who are seriously considering to deploy CNN would try to alleviate the speed limitation by investing heavily into hardware capabilities. Using high-power processing equipment such as multi-processors, large capacity of fast memories and parallel computing environment is a common approach. Some researchers alternatively try to use other training algorithms than CD to marginally speed up the training process. Apart from the hardware requirement and learning algorithm, the shortcomings lie in the fundamental structure of CNN where many parameters must be tuned properly. Some examples are training the weights for effective learning, controlling the 'attractors' which are related to stabilizing the system dynamics of the neural network states. The history of the inputs may need to be stored in a large set of attractors. All these could be possibly solved by

some kind of optimization algorithms. This belongs to hyperparameter search problem; just like any machine learning algorithm, CNN is able to induce a representative model that can capture some useful insights from a large data, given the model parameters are fine-tuned to its optimal state. A tuned machine learning model replies on balancing the learning at appropriate level of model complexity. Overfitting occurs if the model is trained with too much examples, making the model too complex. Consequently it overly fits the data into constructing the model on almost every instance was considered, but it lacks of generalization power to unseen data. On the other extreme, when the complexity of the model is too low, it will not be able to all the essential information in the data. This phenomenon is called underfitting. In the case of CNN, a set of hyperparameters should be determined before training big data commences. The choice of hyperparameters can remarkably affect the final model's performance in action. However, determining appropriate values of parameters for optimal performance is a complex process. Claesen and Moor in 2015 [6], has argued that it is an open challenge inherent to hyperparameter such as optimizing the architecture of neural networks [7], whereas the count of hidden layers of a neural network is one such hyperparameter, and the amount of neurons that associate with each layer gives rise to another set of additional hyperparameters. The search space gets increasingly complex when they depend conditionally upon the number of layers in the case of CNN. Tuning all these parameters is quite difficult in real-time computing environment. Instead of finding the perfect balance, as suggested by most hyperparameter optimization strategies, meta-heuristic could be used [8], allowing the best model in terms of optimization solution emerges by itself by stochastic and heuristic search over certain iterations in lieu of brute-force or Monte-Carlo that tries through all the alternatives.

Meta-heuristic algorithms are designed find global or near optimal solutions within acceptable search time, at reasonable computational cost. In case of CNN, the whole model of neural network could technically be represented by a solution vector which could be optimized to produce the best fitness in terms of prediction accuracy. This can be easily done by encoding a vector of weights from the neural network, with the value in each vector cell representing the weight of a linkage between a pair of neuron in the neural network. Figure 1 illustrates this simple encoding concept that represents a neural network to be optimized as a solution vector.

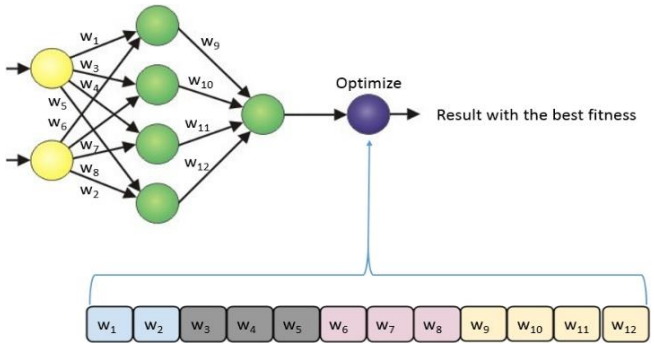

Fig. 1. Encoding the weights of neural network into a solution vector

Once the encoding is in place, we can train a neural network using a meta-heuristic search algorithm, for finding a solution vector that represents a combination of weights that gives the highest fitness. In this position paper, the relevant works of applying meta-heuristic algorithms for artificial neural network's training are reviewed. In addition, we survey the possibility of implementing meta-heuristic algorithms on restricted Boltzmann machine's (RBM) parameter training process.

The reminder of this paper is organized as follow: meta-heuristic algorithms and their fundamental design constructs are introduced in Section 2. Section 3 describes meta-heuristic algorithms are implemented on optimizing neural network training. Section 4 shows a case of deep learning and restricted Boltzmann machine which could be empowered by meta-heuristics. Discussion about the prospects of apply metaheuristics on DL and conclusion are drawn in Section 5.

2 Meta-heuristic Algorithm

Meta-heuristic is another emerging trend of research, mostly found its application in optimization including combinatorial optimization, constrained based optimization, fixed-integer, continuous numeric and mixed-type search space optimization [9]. Meta- stands for some high-level control logics, that controls some rules underneath or embraced in an iterative operation which try to improve the current solution generation after generation. It is a collectively concept of a series of algorithms including evolutionary algorithm, the most famous one is Genetic algorithm (GA) [10]. Recently a branch of population-based meta-heuristics has gained popularity and showed effectiveness in stochastic optimization. Optimal or near optimal answers are almost always guaranteed, by going through the search repeatedly through certain number of iteration. The movements of the search agents are mostly naturally inspired or bio-logical algorithm, foraging the food hunting patterns and/or social behavior of insects/animals towards to global best situations which sets as the objective of the search operation [11]. These search-oriented algorithms found success recently in many optimization applications, from scientific engineering to enhancing data mining methods. Lately, with the hype of big data analytics, and the rise of neural network in the form of CNN being shown useful in DL within big data, meta-heuristics may again shows its edge in probably complementing the shortcomings of CNN, improving its ultimate performance like fitting a hand into glove.

Some of the most prevalent population-based meta-heuristics is Particle Swarm Optimization (PSO) [12]; trajectory algorithm, such as Tabu search [13], and so on. In this review paper, we focus mainly on the underlying logics of GA and PSO, partly because these two are the most popular meta-heuristics that have demonstrated their merits. Another reason is that GA and PSO represent two very fundamental concepts in terms of the movement logics, how they converge and how the solutions emerge through trying out heuristically alternative solutions in the search process. GA on one hand, represents a tightly coupled evolutionary mechanism, having a population of chromosomes that represent the solutions, being mutate, cross-over and the fittest ones pass onto the future generations till only the few fittest solutions stay.

PSO on the other hand, work in similar approach but with an explicit concept of separating global velocity and local velocity among the swarm of moving particles (search agents). While the global velocity is controlling the search momentum towards the best possible solution (to be found), the local velocity associate with each particle enables some randomness and alternatives in the hope of finding better solutions just than the current ones. In a way, PSO unifies two subtle objectives in the logic design, namely local intensification (LI) and global exploration (GE) [14]. These two very underlying forces which are usually embedded in the search mechanism empower the overall search, often yielding good results. Since PSO was launched, and the two subtle searching forces that complement each other towards the final goal were discovered, a number of variants of metaheuristics was created in the meta-heuristic research community. Many of the new variants are either extensions of PSO embracing the two concepts of LI and GE or hybrids of the existing meta-heuristic algorithms, interchanging some implementations of LI and GE.

The latter type of new variants which are hybrid, is founded on the shortcomings of the original prototype where the algorithm is inspired by a certain nature phenomenon. The original prototype normal would faithfully follow the salient features of an animal or natural manifestation, thereby limiting its algorithmic efficacy for mimicking the animal as closely as possible. As a result, standalone and original prototype may work as efficiently as it is wished to be, leaving some rooms for improvement by modifications.

Under this observation that mods are better than the original (they have to be better in order to get the papers published), researchers have been trying to combine multiple meta-heuristics in the hope of yielding some better results since it is already known that no meta-heuristic alone is able to offer the best. In the process of thinking of a new hybrid, the original meta-heuristic algorithm is dissected into different parts, checking of its unique function(s) and how they were built suitable for some particular types of complex problem. Blum and Roli [15] explained that the power of the meta-heuristics search is somehow due to the dual efforts and cooperation between the local exploitation and global exploration strategies. Figure 2 shows the dual steps are integral part of the original metaheuristic algorithm, in general sense.

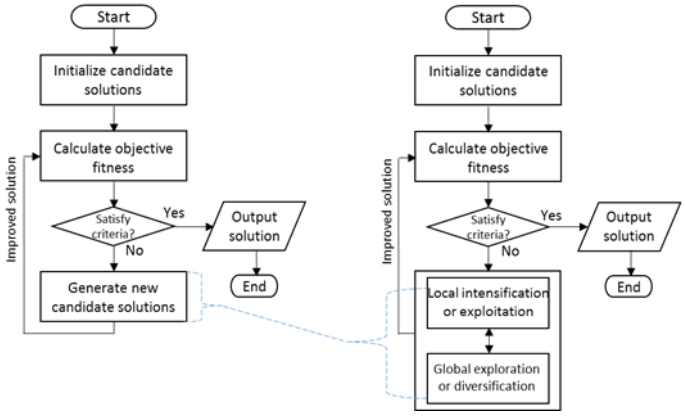

Fig. 2. LI and GE in a general meta-heuristic algorithmic logic.

By understanding these two underlying forces in the meta-heuristic design, it helps finding suitable optimization constructs for improving any machine learning algorithms, including CNN of course. GI is designed to continuously find a global optimum from some afar positions of the search space. Therefore in some metaheuristics the GI components enable the search agents to wide-spread the search agents from their current positions through some random mechanism. So it is in the hope that by venturing far away, the swarm is able to escape from being stuck at local optimum as well as finding a better terrain undiscovered previously. On the other hand, LI is designed to guide the search agents to scout intensively at the current proximity for refining the locally best solution they have found so far.

Referring to the other trend of metaheuristics than hybrids, variants of PSO in the names of some animals have subsequently been arisen. They can be considered as enhanced versions of PSO, with unique features of keeping LI and GE explicitly defined. For typical examples, are Wolf Search Algorithm (WSA) [16] and Elephant Search Algorithm (ESA) [17] that have been recently proposed and they are considered as "semi-swarm" meta-heuristics. The search agents by the semi-swarm design have certain autonomous capacity focusing in LI, yet they cooperatively follow the guidelines of GE to outreach for global exploration. Their pseudo codes are shown in Figures 3, 4 and 5 respectively. In particular the LI and GE parts are highlighted in the pseudo-codes showing their similarities, yet loosely coupled movements by the search agents enforcing these two crucial search forces.

In Figure 3, it can be seen that PSO the global velocity and the individual particles' velocities are tightly coupled in the rule. By the design of WSA as shown in Figure 4, WSA relaxes this coupling by allowing every wolf search agent to roam over the search space individually by taking their local paths in various dimensions. They sometimes merge when they are bound within certain visual ranges. At a random chance, the wolf search agents jump out of their proximity for the sake of GE. At the end, the search agents would unite into their nuclear family packs which eventually would have migrated to the highest achievable solution (that has the maximum fitness found so far by both GE and LI). In Figure 5, ESA is comprised of search agents of two genders. Each gender group of elephant search agents will explicitly do GE and LI. The two search forces are separately enforced by the two genders in ESA. The leader of the female herd which has the local best fitness guides her peers in doing local search intensively around the proximity. The male elephants venture far away to search for terrains that may yield better fitness. When positions of higher fitness is found by the male elephants, the female elephant herds will migrate over there. Unlike PSO and WSA, the two underlying forces are being fulfilled diligently by two genders of elephant groups, totally separately and autonomously. In addition to separation of GE and LI, the unique design of ESA is all elephants follow a limited life-span. Aged elephants will expire, be relinquished from the search process, and new elephants will be born in locations which are inferred from a mix of best positions from their parents, female group leaders and some randomness. This extra force, evolution, is shown in Figure 4. To certain perspective, ESA carries the goodness of semi-swarm metaheuristics with respect to GE and LI, and combine this virtue into some evolutionary mechanism, ensuring the future generations progress into better solutions than the existing ones. Such designs will hopefully shed some light into applying meta-heuristic into optimizing DL tools such as CNN.

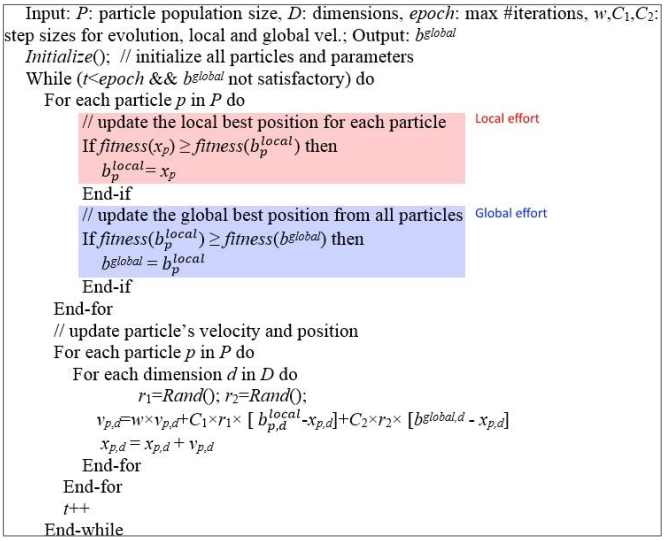

Fig. 3. Simplified version of PSO with local and global search efforts highlighted.

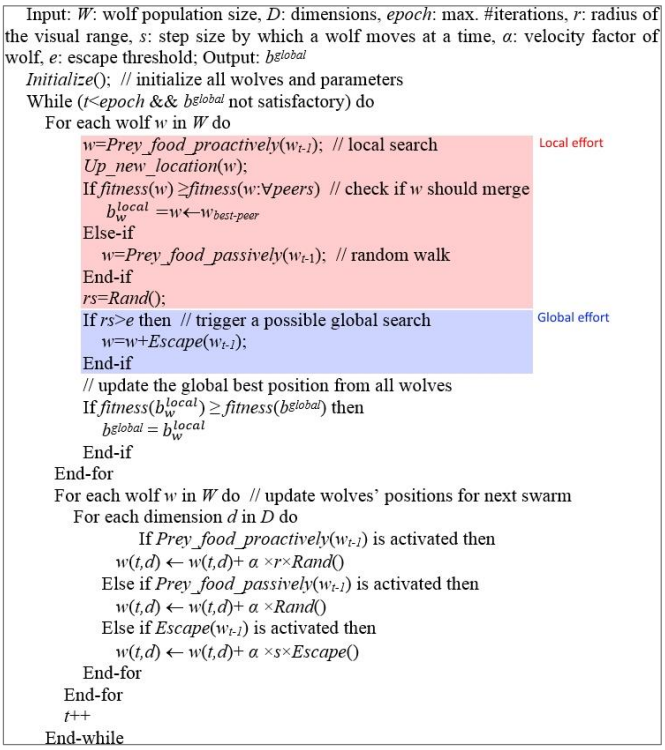

Fig. 4. Simplified version of WSA with local and global search efforts highlighted.

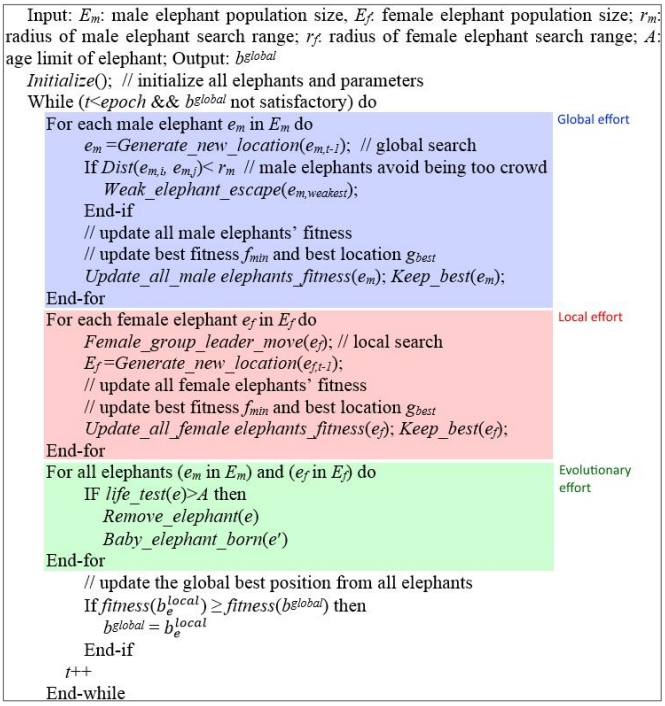

Fig. 5. Simplified version of ESA with local and global search efforts highlighted.

Depending on the complexity of the hyperparameter challenges in CNN optimization and the presences of multi-modals (local optima), meta-heuristics ranging from fully swarm, to semi-swarm and loosely coupled with evolutionary abilities are available.

3 Applying Meta-heuristic Algorithm on Neural Network Training

There have been some debates among researchers in the computer science research community, on the choice of optimization methods for optimizing shallow and deep neural networks like CNN for instance. Some argued that meta-heuristic algorithms should be used in lieu of classical optimization methods such as Gradient Descent, Nesterov, Newton-Raphson etc., because meta-heuristics were designed to avoid falling stuck at local minima.

Researchers who are skeptical about the use of meta-heuristics usually share the concern that local minima are not of a serious problem that needs to be heavily optimized at the neural networks. The presences of local minima which are believed to come in mild intensity and quantity are caused by some permutation of the neurons at the hidden layers, depending on the symmetry of the neural network. It was supposed that finding a good local minimum by minimizing the errors straightforwardly is good enough. It is an over-kill using extensive efforts in searching for the global minima to

the very end. Moreover some are wary that overly optimizing a neural network limits its flexibility, hence leading to overfitting the training data if metaheuristic is used or excessively used. The overfitted neural network may become lack of generalization power when compared to a neural network that was trained by gradient descent that achieved a local minima which is good enough. When this happens some regularization function would be required to keep the complexity of the model under check. To the end of this, some researchers suggested using an appropriate kernel or radial basis function provides simple and effective solution.

Nevertheless researchers from the other school of thoughts believed that applying meta-heuristic on neural network training has its edge on providing the weight training to its optimum state. Like the epitome of a doctrine, the name meta-heuristic consists of the terms “meta” and ‘‘heuristic” are Greek where, “meta” is “higher level” or “beyond” and heuristics implies ‘‘to find”, ‘‘to know”, ‘‘to guide an investigation” or ‘‘to discover”. Heuristics are simple logics to find best (or near-best) optimal solutions which are on par with the absolute best (which is extremely difficult to find) at a reasonable computational cost. In a nutshell, meta-heuristics are a collection of simple but intelligent strategies which could fit into a wide range of application scenarios for enhancing the efficiency of some heuristic procedures [18]. Optimizing the configuration of neural network is one of such heuristic procedures.

By tapping on the searching ability of global optimum by meta-heuristic algorithms, researchers aim to train a neural network to execute faster than traditional gradient descent algorithm. In the following part, I reviewed four researchers’ work on implementing meta-heuristic on neural network training including GA on NN, PSO on NN and hybrid GA&PSO on NN.

Though in the above-mentioned section, the basic constructs of most of the meta-heuristics algorithms are GE and LI (in addition to initialization, stopping criteria checking and other supporting functions), meta-heuristic operate by implementing different forms of agents such as chromosome (GA), particles (PSO), fireflies (firefly algorithm). These agents collectively keep moving close to the global optimum or near global optimum through iterative search. Many strategies such as evolutionary strategy, social behaviour and information exchange are implemented, thereby many versions were made possible. Readers who want to probe into details of the variety of meta-heuristics are referred to a latest review [19].

Artificial neural network (ANN) are traditionally constructed in layout of multi-layered feed-forward neural network; some used back-propagation (BP) as error feedback to modify the weights for training up the cognitive power of the network. The weight training strategy has been traditionally gradient descent (GD). However, in the literature, there have been numerous cases of applying meta-heuristic on optimizing such traditional neural network for speeding up the training process. This is primarily achieved by replacing the GD strategy with iterative evolutionary strategy or swarm intelligence strategy by meta-heuristics [20, 21, 22, 23, 24].

Gudise, V. G., et al [20] compared the performance of feed-forward neural network optimized by PSO and feed-forward neural network with BP, the experiment result shows that feed-forward network with PSO is better than that with BP in terms of non-linear function.

Leung, F. H., et al [24] showed their work on the efficacy of tuning up the structure and parameters of a neural network using an improved genetic algorithm

(GA). The results indicate that the improved GA performs better than the standard GA when the neural networks are being tested under some benchmarking functions.

Juang, C. F. [22] proposed a new evolutionary learning algorithm based on a hybrid of genetic algorithm (GA) and particle swarm optimization (PSO), called HGAPSO. It takes the best of the both types of meta-heuristics: swarming capability and evolutionary capability. Defining the upper-half of the GA population as elites and enhancing them by PSO, while the rest of the population are processed by GA, the hybrid method outperforms PSO and GA individually in training a recurrent or fuzzy neural network.

Meissner, M., et al [23] used Optimized Particle Swarm Optimization (OPSO) to accelerate the training process of neural network. The main idea of OPSO is to optimize the free parameters of the PSO by generating swarms within a swarm. Applying the OPSO to optimize neural network training it aims to build a quantitative model. OPSO approach produces a suitable parameter combination which is able to improve the overall optimization performance.

Zhang, J. R., Zhang [21] proposed a hybrid algorithm of PSO coupled with BP for neural network training. By leveraging the advantage of PSO's global searching ability as well as BP's deep search capability, the hybrid algorithm showed very good performance respective to convergent speed and convergent accuracy.

Optimizing shallow learning by traditional neural network approaches has been shown successfully possible using the meta-heuristic methods as above-mentioned. In contrast, DL by CNN is a relatively unexplored field. Very few papers have been published except for one by [25] which used Simulated Annealing (SA) algorithm to optimize the performance of CNN and showed improved results. Nonetheless, there is a still good prospects in trying out different meta-heuristics for optimizing CNN for DL, because the fundamental problems and solutions are about the same: you have a number of unknown variables on hand, and meta-heuristics attempt to offer the best possible solution.

Structures of deep learning model is similar to the traditional artificial neural network, except for some modifications are implemented for better learning ability. For instance, the CNN is a traditional ANN modified with pooling procession and the structure of RBM is an undirected graph or a bidirectional neural network. DL model shares similar models with neural network; more importantly, different training algorithm may be called upon instead of gradient descent strategy. This warrants further exploration into this research arena. Several important contributions which have been mentioned about are elaborated as follow.

3.1 Genetic Algorithm on Neural Network

Leung et al [24] first tried implementing genetic algorithm (GA) on neural network training in 2003. Though Leung et al may not be the pioneer in apply GA on neural network, an improved version of GA that is made suitable for ANN is put forward. Crossover operations, mutation operations and fitness function of GA are all re-defined, custom-made. Firstly, when it comes to encoding the chromosome and perform the crossover operation, four possible offspring candidates will be generated and the one with the largest fitness value will be chosen as offspring. The four possible crossover offspring are generated as regulations listed below:

$$os_c^1 = [os_1^1, os_2^1, \dots os_n^1] = \frac{p_1 + p_2}{2}$$
$$os_c^2 = [os_1^2, os_2^2, \dots os_n^2] = p_{max}(1-w) + max(p_1, p_2)w$$
$$os_c^3 = [os_1^3, os_2^3, \dots os_n^3] = p_{min}(1-w) + min(p_1, p_2)w$$
$$os_c^4 = [os_1^4, os_2^4, \dots os_n^4] = \frac{(p_{min} + p_{max})(1-w) + (p_1 + p_2)w}{2}$$

where the p_{max} and p_{min} are calculated respectively according to the parameters of the neural network by maximizing and minimizing them such as $p_{max} = [para_{max}^1, para_{max}^2, \dots, para_{max}^n]$, $p_{min} = [para_{min}^1, para_{min}^2, \dots, para_{min}^n]$. For example, Max([1,-1,4],[-3,3,2])=[1,3,4] and Min([1,-1,4],[-3,3,2])=[-3,-1,2]. Secondly, the mutation operations are re-defined. The regulations are given below:

$$os = os + [b_1 \Delta nos_1, b_2 \Delta nos_2, \dots, b_n \Delta nos_n]$$

os is the chromosome with biggest fitness value in all four possible offspring. b_1 random equals to 0 or 1 and Δnos_i is random number making sure $para_{min}^i \leq os_i + b_i \Delta nos_i \leq para_{max}^i$. os' is the final generation after crossover operation and mutation operation.

Thirdly, the fitness value is defined. By adding parameters in the neural network mathematical expression, the actual output of GA-optimized neural network y_k equals to:

$$y_k = \sum_{j=1}^{n_h} \delta(s_{jk}^2) w_{jk} logsig\left[\sum_{i=1}^{n_i} \delta(s_{ij}^1) w_{ij} x_i - \delta(s_j^1 b_j^1)\right] - \delta(s_k^2) logsig(b_k^2)$$

in which k=1,2, .., n_{out}, s_{ij} denotes link from i^{th} neuron in input layer to j^{th} neuron in hidden layer, s_{jk} denotes link from j^{th} neuron in hidden layer to k^{th} neuron in output layer, w_{jk} denotes weight between each neuron, b_k^1 and b_k^2 denote bias in input layer and hidden layer respectively, n_{in}, n_h and n_{out} denote the number of neurons of input layer, hidden layer and output layer, respectively. The error of the whole network is defined as mean of all chromosomes:

$$error = \sum_{k=1}^{n_{out}} \frac{|y_k - y_k^d|}{n_d}$$

in which n_d denotes the number of chromosome used in the experiment, y_k^d denotes the desired output of output neuron k. Given the error of the network, GA is implemented to optimize the network thus minimize the error. The fitness function is defined as

$$fitness = \frac{1}{1 + error}$$

where the smaller the error and the bigger the fitness value. GA is implemented to find the global optimum of the fitness function thus the parameter combinations of weight w are the trained weight for the network.

3.2 Particle Swarm Optimization on Neural Network

Gudise and Venayagamoorthy [5] implemented PSO on neural network training in 2003. The fitness value of each particle (member) of the swarm is the value of the error function evaluated at the current position of the particle and position vector of the article corresponds to the weight matrix of the network.

Zhang et al [21] developed a hybrid algorithm of BP & PSO that could balance training speed and accuracy. The particle swarm optimization algorithm was showed to converge rapidly during the initial stages of a global search, but around global optimum, the search process will become very slow. On the contrary, the gradient descending method can achieve faster convergent speed around global optimum, and at the same time, the convergent accuracy can be higher.

When the iteration process is approaching end and current best solution is near global optimum, if the change of the weight in PSO is big, the result will vibrate severely. Under this condition, Zhang supposed with the increase of iteration time, the weight in PSO should decline with the iteration time's increasing to narrow the search range thus pay more attention to local search for global best. He suggest the weight decline linearly first then decline nonlinearly as shown in Figure 6.

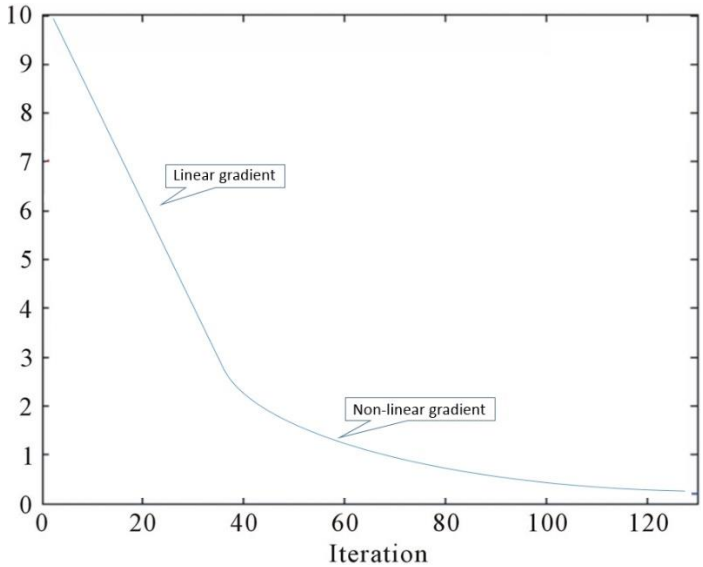

Fig. 6. Change of inertia weight through linear and non-linear curve

The concrete working process is summarized as follow: For all the particles p_i, they have a global best location p^{global_best}. If the p^{global_best} keeps unchanged for over 10 generations, that may infer the PSO pays too much time on global search thus BP is implemented for p^{global_best} to deep search for a better solution.

Similar to GA's implementation in neural network, the fitness function defined is also based on whole network's error and to minimize the error as the optimization of PSO. The learning rate μ of neural network is also controlled in the algorithm, as

$$\mu = k \times e^{-\mu_0 \cdot epoch}$$

where μ is learning rate, k and μ_0 are constants, *epoch* is a variable that represents iterative times, through adjusting k and μ_0, the acceleration and deceleration of learning rate can be controlled.

By implementing the strategy that BP focusing on deep searching and PSO focusing on global searching, the hybrid algorithm has a very good performance.

3.3 Hybrid GA & PSO on Neural Network

Juang et al [22] proposed in 2004, hybrid genetic algorithm and particle swarm optimization for recurrent network's training.

The hybrid algorithm called HGAPSO is put forward because the learning performance of GA may be unsatisfactory for complex problems. In addition, for the learning of recurrent network weights, many possible solutions exist. Two individuals with high fitness values are likely to have dissimilar set of weights, and the recombination may result in offspring with poor performance.

Juang put forward a conception of "elite" of the first half to enhance the next generation's performance. In each generation, after the fitness values of all the individuals in the same population are calculated, the top-half best-performing ones are marked. These individuals are regarded as elites.

In every epoch, the worse half of the chromosome is discarded. The better half is chosen for reproduction through PSO's enhancement. All elite chromosomes are regarded as particles in PSO. By performing PSO on the elites, we may avoid the premature convergence in elite GAs and increase the search ability. Half of the population in the next generation are occupied by the enhanced individuals, the others by crossover operation. The working flow of algorithm has mechanism of evolving the offsprings as well as enhanced elites through crossover and mutation.

The crossover operation of HGAPSO is similar to normal GA, random selecting site on chromosome and exchange the sited piece of chromosome to finish the crossover operation. In HGAPSO, uniform mutation is adopted, that is, the mutated gene is drawn randomly, uniformly from the corresponding search interval.

3.4 Simulated Annealing on CNN

Rere et al [25] claimed to be the first which applied a meta-heuristic called Simulate Annealing (SA) on optimizing CNN for improved performance. Their results showed that SA is effective on optimizing CNN with better results than the original CNN although it comes with an increase of computation time cost. The SA works by progressively improving the solution; the logics are outlined as follow: Step 1. Encode the initial solution vector at random, compute the objective function; Step 2. Initialize the temperature which is a crucial parameter relating to the convergence speed and hitting onto the global optimum; Step 3. Pick a new solution in the neighborhood of the current position. New solutions are generated depending on T – this is similar to the local intensification; Step 4. Evaluate the goodness of the new solution candidates.

Update the global best if a better candidate is found; Step 5. Periodically lower the temperature during the search process. So that the chance of receiving deteriorating moves drops, so the search is converging; Step 6. Repeat the above steps until the stopping criterion is satisfied.

In the meta-heuristic search design, SA is trying to minimize the standard error on fitness function of the vector solution and the training set. The fitness function is formulated as follow:

$$fitness = 0.5 \times \sqrt{\frac{\sum_{i=1}^{N}(|y - y'|)^2}{N}}$$

where y' is the expected output, y is the actual output, N is the amount of training samples. SA should converge hence stop when the minimum neural network complexity have attained the most optimal (lowest) state and the approximate error accuracy indicates a very low value. In the optimization process, the CNN computes the weights and the bias of the neural network, the results are past from the last layers to evaluate the lost function. This is being proceeded as SA tries to find the best solution vector, like a wrapper process. SA attempts at scouting for a better solution in each iteration by randomly picking a new solution candidate from the proximity of the current solution by adding some random Δx. This optimization is relatively simple, because only the last layer of the CNN is used to instill the solution vector, where the convolution part is almost left untouched. The SA seems doing only local search, and it does not require a large number of search agents, nor doing much of global exploration. The results show that the larger the neighborhood sizes (10, 20 and 50) the better the accuracy is achieved. However, the gain in accuracy was most obvious in small numbers of epoch, from 1 to 9. From epoch 10 onwards, the marginal difference in performance between the original CNN and SA applied on CNN diminish. The time cost however becomes more than double at epoch 10 between the CNN and SA+CNN with 50 neighbor size. This is doubtful where SA is a suitable meta-heuristic to be applied on CNN especially if the epoch reaches a large number in magnitudes greater than 10 for some complex big data analytics. This work nevertheless promotes the use of meta-heuristics, researchers would be encouraged to try other meta-heuristics.

4 Deep Learning and Restricted Boltzmann Machine

When dealing with image classification or other problems, traditional method is using pre-processing transforming data as input values for neural network learning while using deep learning method for classification, raw data (pixel values) are used as input values. This will keep to the maximum extent protecting all information regardless of useful or not from being destroyed by extraction methods. The most advantage lies that all the extraction methods are based on expert knowledge and expert choice thus are not extensible to other problems, while deep learning algorithm can overcome these limitations by using all data with its powerful processing ability. A convolutional neural network (CNN) is shown in Figure 7.

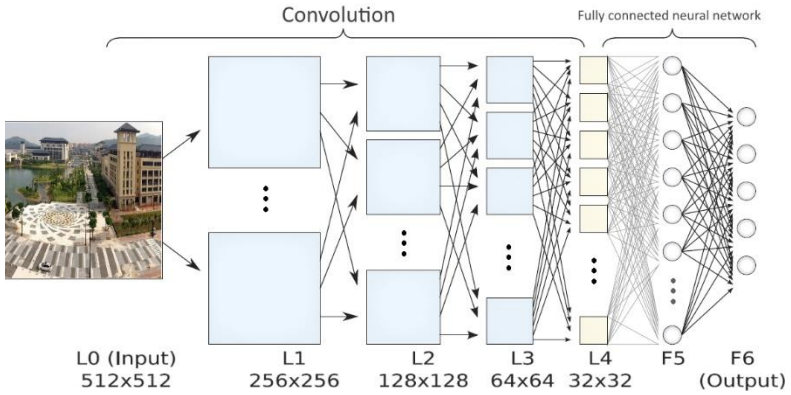

Fig. 7 Typical architecture of a convolutional neural network

The Boltzmann machine only has two layers of neurons, namely input and output layers respectively. The first layer is the input layer and the second layer is the output layer, although the structure is very simple only contains two layers, but it mathematical function in it is not simple. In here we need to introduce the following probability equation to know the RBM.

From the equation we are knowing there are three item in it, they are the energy of the visible layer neural, the energy of the hidden layer neural, and the energy is consisted of the two layer. From the energy function of the RBM, we can see there are three kinds of parameters lie in the RBM, different from the neural network, each Neuron has a parameter too, but for the neural network only the connection of two layers has the parameters. Also, the neural network does not carry any energy function. They use the exponential function to express the potential function. There are also the probabilities existing in the RBM. They work exactly in the same way as the two kind of the probabilities such as $p(v|h)$ and $p(h|v)$. It has a similarity with probability graph model in details, examples are Bayesian network and Markov network. It looks like a Bayesian network because of the conditional probability. On the other hand it does not look like a Bayesian because of the two directions probabilities. These probabilities are lying on the two variables that have only one direction.

Compared with the Markov network, the RBM seems to be having a little bit relation with it, because the RBM has the energy function just as the Markov network. But it is not so much alike because the RBM's variable has parameter, and the Markov network does not have any parameter. Furthermore, the Markov network does not have conditional probability because it has no direction but just the interaction. From the graph's perspective, variables in the Markov network use cliques or clusters to represent the relations of close and communicated variables. It uses the production of the potentials of the clique to express the joint probability instead of conditional probability just like the RBM. Its input data are the kind of the Boolean data, within the range between 0 and 1.

The training way of the RBM is to maximize the probability of the visible layer, and to generate the distribution of the input data. RBM is a kind of special

Markov random function and a special kind of Boltzmann machine. Its graphical model is corresponding to the factor product analysis. Different from the probability graphical model, the RBM's joint distribution directly uses the energy function of both visible layer *v* and hidden layer *h* to define instead of potential of it given as

$$E(v,h) = -a^T v - b^T h - v^T W h$$
$$P(v,h) = \frac{1}{Z} e^{-E(v,h)}$$

Later we will know that Z is the partition function defined as the sum of $e^{-E(v,h)}$ over all the possible configurations. In other words, it is just a constant normalizing the sum over all the possible hidden layer configurations.

$$P(v) = \frac{1}{Z} \sum_{h} e^{-E(v,h)}$$

The hidden unit activations are mutually independent given the activations. That is, for m visible and n hidden units, the conditional probability of a configuration of the visible unit *v*, given a configuration

$$P(v|h) = \prod_{i=1}^{m} P(v_i|h)$$

Conversely, the conditional probability of *h* given *v* is *P*(*h*|*v*). Our goal is to infer the weights that maximize the marginal of the visible, in details we can step through the following equation to infer and learn the RBM.

$$\arg \underset{w}{Max}\, E\left[\sum_{v \in V} logP(v)\right]$$

As for the training algorithm, the main idea is also applied gradient descent idea into RBM. Hinton put forward Contrastive Divergence [26] as a faster learning algorithm. Firstly, the derivative of the log probability of a training vector with respect to a weight is computed as

$$\frac{\partial logP(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

where the angle brackets are used to denote expectations under the distribution specified by the subscript that follows. This leads to a very simple learning rule for performing stochastic steepest ascent in the log probability of the training data:

$$\Delta w_{ij} = \varepsilon\left(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}\right)$$

where ε is a learning rate.

Because there are no direct connections between hidden units in an RBM, it is very easy to get an unbiased sample of $\langle v_i h_j \rangle_{data}$. Given a randomly selected training image, v, the binary state, h_j, of each hidden unit, j, is set to 1 with probability

$$p(h_j = 1|v) = \sigma\left(b_j + \sum_i v_i w_{ij}\right)$$

where b_j is the current state of hidden neuron j, $\sigma(x)$ is the logistic sigmoid function $\sigma(x) = \frac{1}{1+exp(-x)}$. $v_j h_j$ is then an unbiased sample. The contrastive divergence (CD) is used to calculate the latter part $\langle v_i h_j \rangle_{model}$. Details can be found in their respective publications.

Considering the complicated computation of implementing CD, the training process of RBM is not easy. Under this condition, implementing meta-heuristic on RBM training to substitute CD is of high possibility.

3 Discussion and Conclusion

Meta-heuristic has successfully implemented in neural network training in the past years. The algorithm used includes GA, PSO, their hybrids as well as many other meta-heuristic algorithms. Moreover, feed-forward back-propagation neural network and spiking neural network [27] are all trained with tests on famous classification problems, where benchmarking their performance is easy. Given the rises of these two emerging trends of meta-heuristics and deep learning which are gaining momentums both in academic research communities and industries, these two trends do cross-road. At this junction, this paper contributes as a timely review on how meta-heuristics contribute to optimizing deep learning tools such as convolutional neural network (CNN).

The basic structure of deep learning network is similar to traditional neural network. CNN is a special neural network with different weight computation regulations and Restricted Boltzmann Machine (RBM) is a weighted bio-dimensional neural network or bio-dimensional graph. Their training processes are also mainly executed through iterative formula on error which is similar to traditional neural network's training. Given their iterative natures in execution and the fact that a neural network (at least a part of it) could be coded as a solution vector, it is natural to tap onto the power of meta-heuristics to optimize their layouts, configurations, weights computations and so forth.

In the near future, it is anticipated that it is of high possibility of applying meta-heuristic in deep learning to speed up training without declining performance. However, relevant publications along this direction are still rare. Nevertheless the interests which are expressed in terms of Google Trends show some take-off. In Figure 8, it is seen that the four trends are indeed converging to about the same level of popularity in 2016 and beyond. The four trends are by the keywords of deep learning, machine learning, mathematical optimization and big data. Interestedly, optimization was very much hyped up in the early millennium and now it is approach a steady state.

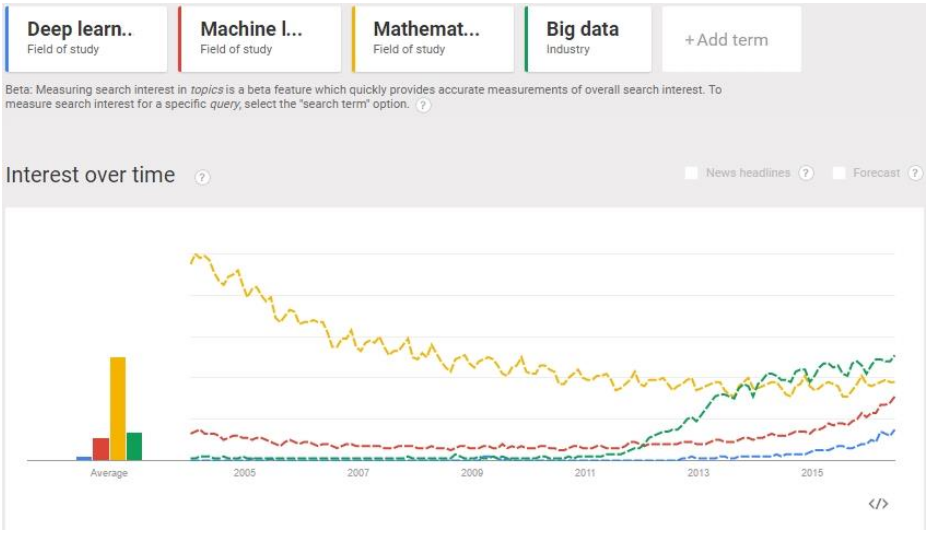

Fig. 8 Google Trend screenshot that shows the popularities of the four fields of study from 2004 till present

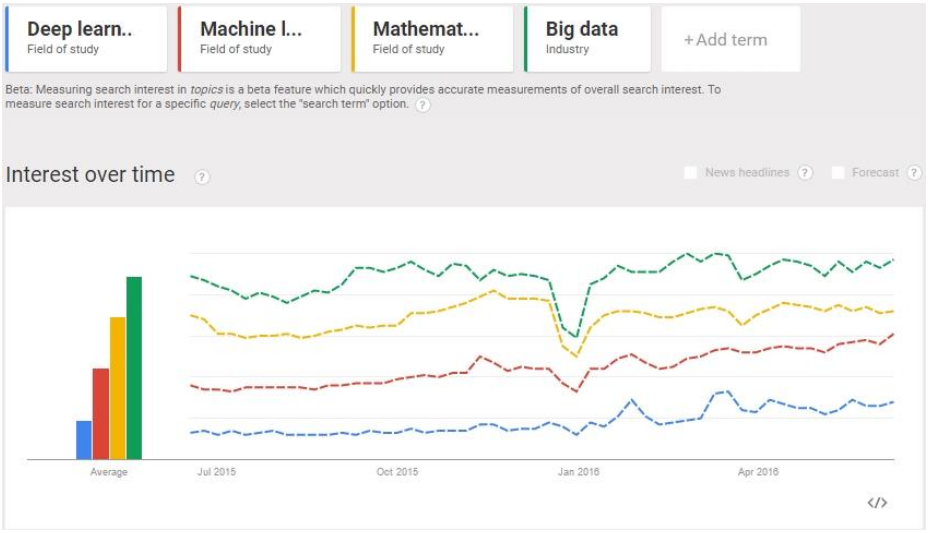

Fig. 9 Google Trend screenshot that shows the popularities of the four fields of study from the past 12 months

The other three trends, especially big data rose up from nowhere since 2011, exceeding the rest by now. The trends of machine learning and deep learning share the same curve patterns. These trends imply sufficient interest levels for these techniques to fuse and cross-apply over one another. In Figure 7, which is the Google Trend result of these four fields of study, the timeline is of past 12 months. It is apparent to see that they do share similar patterns and fluctuation over time, though they are setting at different interest levels.

Furthermore, there still exists a question that goes under today's computation ability, especially the GPU parallel computing whose computation ability is many times stronger than CPU that has been widely used in industrial area [28]. This would further propel the use of meta-heuristic in solving DL problems. The original design of meta-heuristics especially those population-based search algorithms, are founded on parallel processing. The search agents are supposed to operate in parallel, thereby maximizing their true power in GPU environment. When the time cost overhead is removed from the design consideration in GPU environment, searching for near optimal results by meta-heuristic is still attractive as they can offer an optimal configuration of CNN. It is also anticipated that the parameters and structures of CNN will only become increasingly complex when they are applied to solve very complex real-life problems in the big data analytic era. It is foreseen that classical meta-heuristics such as PSO and GA will be first applied to solve DL problems, followed by variants and hybrids of those. While most of the papers in the first wave will concentrate on local intensification in scouting for new solution candidates by adding some randomness and Levy flight distribution, more are expected to follow which embrace both local intensification and global exploration. GE will be useful only when the search space is sufficiently huge, e.g. self-adaptive versions of meta-heuristics such as Bat algorithm [29] and Wolf search algorithm [30], the search space is the summation of the dimensions of the existing possible solutions and the allowable ranges of the parameters values. This advanced meta-heuristic technique translate to parameter-free optimization algorithms which are suitable to deal with only very complex neural network configuration and performance tuning problems. Then again, considering the big challenges of big data analytics in solving big problems (e.g. analyzing in real-time of road traffic optimization problems using Internet-of-Things), corresponding powerful DL tools should be equipped with effective optimization strategies.

Acknowledgement

The authors are thankful for the financial support from the Research Grant called "A Scalable Data Stream Mining Methodology: Stream-based Holistic Analytics and Reasoning in Parallel", Grant no. FDCT/126/2014/A3, offered by the University of Macau, FST, RDAO and the FDCT of Macau SAR government.

References

1. Hinton, G. E., Osindero, S., and Teh, Y. W. "A fast learning algorithm for deep belief nets", Neural Computation. 2006;18(7):1527-1554
2. Yu Kai, Jia Lei, Chen Yuqiang, and Xu Wei. "Deep learning: yesterday, today, and tomorrow", Journal of Computer Research and Development. 2013;50(9):1799-1804
3. ILSVRC2012. Large Scale Visual Recognition Challenge 2012 [Internet].[Updated 2013-08-01]. Available from: http:/www.imagenet. Org/challenges/LSVRC/2012/
4. Izadinia, Hamid, et al. "Deep classifiers from image tags in the wild". In: Proceedings of the 2015 Workshop on Community-Organized Multimodal Mining: Opportunities for Novel Solutions; ACM; 2015

5. Gudise, V. G. and Venayagamoorthy, G. K. "Comparison of particle swarm optimization and back propagation as training algorithms for neural networks". In: Proceedings of In Swarm Intelligence Symposium SIS'03; 2006. p. 110-117
6. Marc Claesen, Bart De Moor, "Hyperparameter Search in Machine Learning", MIC 2015: The XI Metaheuristics International Conference, Agadir, June 7-10, 2015, pp.14-1 to 14-5
7. Steven R. Young, Derek C. Rose, Thomas P. Karnowski, Seung-Hwan Lim, Robert M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm", Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, ACM, 2015
8. Papa, Joao P.; Rosa, Gustavo H.; Marana, Aparecido N.; Scheirer, Walter; Cox, David D. "Model selection for Discriminative Restricted Boltzmann Machines through meta-heuristic techniques". Journal of Computational Science, v.9, SI, p.14-18, July 2015.
9. Xin-She Yang, "Engineering Optimization: An Introduction with Metaheuristic Applications", Wiley, ISBN: 978-0-470-58246-6, 347 pages, June 2010
10. Goldberg, D. E. and Holland, J. H. "Genetic algorithms and machine learning". Machine Learning. 1988;3(2):95-99.
11. Iztok Fister Jr., Xin-She Yang, Iztok Fister, Janez Brest, Dusan Fister, "A Brief Review of Nature-Inspired Algorithms for Optimization
", ELEKTROTEHNISKI VESTNIK, 80(3): 1–7, 2013
12. Kennedy, J. "Particle Swarm Optimization"; Springer, USA; 2010. p.760-766
13. Glover, F. "Tabu search-part I". ORSA Journal on Computing. 1989;1(3):190-206
14. Xin-She Yang, Suash Deb, Simon Fong, "Metaheuristic Algorithms: Optimal Balance of Intensification and Diversification", Applied Mathematics & Information Sciences, 8(3), May 2014, pp.1-7
15. C. Blum, and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison", ACM Computing Surveys, Volume 35, Issue 3, 2003, pp.268–308
16. Simon Fong, Suash Deb, Xin-She Yang, "A heuristic optimization method inspired by wolf preying behavior", Neural Computing and Applications 26 (7), Springer, pp.1725-1738
17. Suash Deb, Simon Fong, Zhonghuan Tian, "Elephant Search Algorithm for optimization problems", 2015 Tenth International Conference on Digital Information Management (ICDIM), IEEE, Jeju, 21-23 Oct. 2015, pp.249-255
18. Beheshti, Z. and Shamsuddin, S. M. H. "A review of population-based meta-heuristic algorithms". International Journal of Advances in Soft Computing & Its Applications, 2013;5(1):1-35.
19. Simon Fong, Xi Wang, Qiwen Xu, Raymond Wong, Jinan Fiaidhi, Sabah Mohammed, "Recent advances in metaheuristic algorithms: Does the Makara dragon exist?", The Journal of Supercomputing, Springer, 24 December 2015, pp.1-23
20. Gudise, V. G. and Venayagamoorthy, G. K. "Comparison of particle swarm optimization and back propagation as training algorithms for neural networks". In: Proceedings of In Swarm Intelligence Symposium SIS'03; 2006. p. 110-117
21. Zhang, J. R., Zhang, J., Lok, T. M., and Lyu, M. R. "A hybrid particle swarm optimization–back-propagation algorithm for feed forward neural network training". Applied Mathematics and Computation. 2007;185(2):1026-1037
22. Juang, C. F. "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design". Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions 2004;34(2):997-1006
23. Meissner, M., Schmuker, M., and Schneider, G. "Optimized particle swarm optimization (OPSO) and its application to artificial neural network training". BMC Bioinformatics. 2006;7(1):125
24. Leung, F. H., Lam, H. K., Ling, S. H., and Tam, P. K. "Tuning of the structure and parameters of a neural network using an improved genetic algorithm". IEEE Transactions on Neural Networks. 2003;14(1):79-88

25. L.M. Rasdi Rere, Mohamad Ivan Fanany, Aniati Murni Arymurthy, "Simulated Annealing Algorithm for Deep Learning", The Third Information Systems International Conference, Procedia Computer Science 72 (2015), pp.137–144
26. Hinton, G. E., "Training products of experts by minimizing contrastive divergence.", Neural Computing, 2002 Aug;14(8):1771-800
27. Maass, W. "Networks of spiking neurons: The third generation of neural network models". Neural Networks. 1997;10(9):1659-1671.
28. Simon Fong, Ricardo Brito, Kyungeun Cho, Wei Song, Raymond Wong, Jinan Fiaidhi, Sabah Mohammed, "GPU-enabled back-propagation artificial neural network for digit recognition in parallel", The Journal of Supercomputing, Springer, 10 February 2016, pp.1-19
29. Iztok Fister Jr., Simon Fong, Janez Brest, and Iztok Fister, "A Novel Hybrid Self-Adaptive Bat Algorithm," The Scientific World Journal, vol. 2014, Article ID 709738, 12 pages, 2014. doi:10.1155/2014/709738
30. Qun Song, Simon Fong, Rui Tang, "Self-Adaptive Wolf Search Algorithm", 5th International Congress on Advanced Applied Informatics, July 10-14, 2016, Kumamoto City International Center, Kumamoto, Japan