Mühendislik Bilimleri ve Tasarım Dergisi
10(4), 1251 – 1271, 2022
e-ISSN: 1308-6693

Journal of Engineering Sciences and Design
DOI: 10.21923/jesd.1104772

Araştırma Makalesi Research Article

A THEORETICAL INVESTIGATION ON TRAINING OF PIPE-LIKE NEURAL NETWORK BENCHMARK ARCHITECTURES AND PERFORMANCE COMPARISONS OF POPULAR TRAINING ALGORITHMS

Ozlem Imik SIMSEK*, Baris Baykant ALAGOZ

Inonu University, Engineering Faculty, Department of Computer Engineering, Malatya, Turkey

Keywords

Artificial Neural Networks, Network Architectures, Training Performance, Backpropagation Algorithms, Metaheuristic Training.

Abstract

Architectures of neural networks affect the training performance of artificial neural networks. For more consistent performance evaluation of training algorithms, hard-to-train benchmarking architectures should be used. This study introduces a benchmark neural network architecture, which is called pipe-like architecture, and presents training performance analyses for popular Neural Network Backpropagation Algorithms (NNBA) and well-known Metaheuristic Search Algorithms (MSA). The pipe-like neural architectures essentially resemble an elongated fraction of a deep neural network and form a narrowed long bottleneck for the learning process. Therefore, they can significantly complicate the training process by causing the gradient vanishing problems and large training delays in backward propagation of parameter updates throughout the elongated pipe-like network. The training difficulties of pipe-like architectures are theoretically demonstrated in this study by considering the upper bound of weight updates according to an aggregated one-neuron learning channels conjecture. These analyses also contribute to Baldi et al.'s learning channel theorem of neural networks in a practical aspect. The training experiments for popular NNBA and MSA algorithms were conducted on the pipe-like benchmark architecture by using a biological dataset. Moreover, a Normalized Overall Performance Scoring (NOPS) was performed for the criterion-based assessment of overall performance of training algorithms.

BORU-BENZERİ YAPAY SİNİR AĞI KARŞILAŞTIRMA MİMARİLERİNİN EĞİTİMİ HAKKINDA BİR TEORİK ARAŞTIRMA VE POPULAR EĞİTİM ALGORİTMALARIN PERFORMANS KARŞILAŞTIRILMALARI

Anahtar Kelimeler

Yapay Sinir Ağları, Ağ Mimarileri, Eğitim Performansı, Geriyeyayılım Algoritmaları, Metasezgisel Eğitim.

Öz

Sinir ağlarının mimarileri, yapay sinir ağlarının eğitim performansını etkiler. Eğitim algoritmalarının daha tutarlı performans değerlendirmesi için eğitimi zor kıyaslama mimarileri kullanılmalıdır. Bu çalışma, boru-benzeri mimari olarak adlandırılan bir referans sinir ağı mimarisini tanıtmakta ve popüler Sinir Ağı Geriyeyayılım Algoritmaları (SAGA) ve iyi bilinen Metasezgisel Arama Algoritmalarının (MAA) eğitim performansı analizlerini sunmaktadır. Boru-benzeri sinir mimarileri, temelde bir derin sinir ağının uzunlamasına bir kesitini temsil eder ve öğrenme süreci için bir daraltılmış uzun darboğaz oluşturur. Bu nedenle, uzun boru-benzeri ağ boyunca parametre güncellemelerinin geriye doğru yayılmasında gradyan kaybolma problemleri ve büyük eğitim gecikmelerine neden olarak eğitim sürecini önemli ölçüde zorlaştırır. Bu çalışmada boru-benzeri mimarilerin eğitim zorlukları birleştirilmiş tek-nöron öğrenme kanalları konjektörüne göre ağırlık güncellemelerinin üst sınırı dikkate alınarak teorik olarak gösterilmiştir. Bu analizler aynı zamanda Baldi ve arkadaşlarının sinir ağlarının öğrenme kanalı teoremine pratik açıdan da katkıda bulunmaktadır. Popüler NNBA ve MSA algoritmalarının eğitim deneyleri, bir biyolojik veri seti kullanılarak boru benzeri kıyaslama mimarisinde gerçekleştirmiştir. Ayrıca, eğitim

algoritmalarının genel performansının ölçüt tabanlı değerlendirmesi için Normalleştirilmiş Genel Performans Puanlaması (NGPP) uygulanmıştır.

Alıntı / Cite

Simsek, O.I., Alagoz, B.B., (2022). A Theoretical Investigation on Training of Pipe-Like Neural Network Benchmark Architectures and Performance Comparisons of Popular Training Algorithms, Journal of Engineering Sciences and Design, 10(4), 1251-1271.

Yazar Kimliği / Author ID (ORCID Number)	Makale Süreci / Article Process	
O. Imik Simsek, 0000-0002-4192-0255	**Başvuru Tarihi /Submission Date**	17.04.2022
B.B. Alagoz, 0000-0001-5238-6433	**Revizyon Tarihi / Revision Date**	15.07.2022
	Kabul Tarihi / Accepted Date	15.07.2022
	Yayım Tarihi / Published Date	30.12.2022

1. Introduction

Neural networks have been gaining growing popularity in many fields of engineering and applied science for almost two decades. Their architectures and computation schemes have been progressively developed since the first appearance of the fundamental neural network models (Kim,2017). Today, the learning power of deep neural networks is harnessed for processing much bigger data stacks, and theoretical works and research competitions for deeper neural networks are continuing to boost learning capability of the deep neural networks (Coleman et al., 2017; Schmidhuber, 2015; Shrestha and Mahmood, 2019; Deng and Yu, 2013; Winkler and Le, 2017; Mhaskar et al., 2016).

Architectures of neural networks and the training algorithms have been occasionally progressed. The gradient vanishing problem was a major problem when adding more hidden layers to neural networks to reach much deeper networks (Kim, 2017). As the hidden layer count has increased, gradient vanishing problems have emerged, slowed down the training, reduced efficiency of backpropagation algorithms throughout deep layers, and the practical benefits in use of deeper layers began to disappear. Since around 2000, researches have came up with several solutions for the training problems of deep hidden layers; for instance use of more relevant activation functions (RELU and variants)(Kim, 2017; Oostwal et al., 2019), pre-training approaches (Hinton and Salakhutdinov, 2006), better random initial scaling (Glorot and Bengio, 2010), employment of better optimization methods (Martens, 2010), selection of more suitable neural network architectures (Shrestha and Mahmood, 2019; Bahrami et al., 2019; Arifovic and Gençay, 2001) and improved initialization techniques such as the orthogonal initialization and the random walk initialization (Sussillo and Abbott, 2014).

Besides the implementation of gradient based optimization methods in neural network training, there have been attempts to use metaheuristic optimization algorithms in the training process of neural networks. The metaheuristic optimization can provide a gradient-free search option and this becomes advantageous when searching the optimal points in low-gradient parametric search spaces (Martens, 2010). Metaheuristic methods employ a set and trial search strategy to seek optimal values of parameters in the complicated optimization problems (Arifovic and Gençay, 2001; Sussillo and Abbott, 2014). Due to these advantageous, there are several research works that have addressed the training of feedforward neural networks by using popular metaheuristics(Sexton and Gupta, 2000; Che et al., 2011; Gudise and Venayagamoorthy, 2003; Ince et al., 2010; Mosavi et al., 2016) and results were compared with the backpropagation method in the shallow feedforward neural network training problems: Sexton et al. showed that genetic algorithm could be effectively used for training of shallow neural networks for chaotic time series data and reported a superior training performance of genetic algorithm over the backpropagation methods (Sexton and Gupta, 2000). In a similar study, Che et al. concluded that the backpropagation algorithm can be preferred since it provides faster training of neural networks than the genetic algorithm; however it can suffer from the gradient vanishing problem where the genetic algorithm does not suffer (Che et al., 2011). Gudise et al. compared the neural network training performance of particle swarm optimization with performance of a backpropagation algorithm and reported that particle swarm optimization algorithm can faster converge to optimal weights than the backpropagation algorithm (Gudise and Venayagamoorthy, 2003). These contradicting reports on training performances of backpropagation and metaheuristic methods indicate the need for well-designed, standardized test and evaluation procedures.

* İlgili yazar / Corresponding author: oimiksimsek@gmail.com, +90-530-416-8860

Later, performances of several contemporary metaheuristic algorithms were compared for the training of neural networks (Mosavi et al., 2016; Ghasemiyeh et al., 2017). Besides the training process of the neural network, metaheuristic optimization has also been performed for optimization of the network architectures to reach an improved training performance (Arifovic and Gençay, 2001). Although there are many efforts that compare performances of the learning methods (Sewak et al., 2018; Caruana and Niculescu-Mizil, 2006; Bala et al., 1992) and training algorithms (Bahrami et al., 2019; Zhao et al., 2010; Rusiecki, 2012; Karim et al., 2018; Can et al., 2019; Awolusi et al., 2019; Thakkar et al., 2020) in specific application domains, application-specific results obtained for arbitrary network architectures may not be relevant and consistent to have a common view on the training performances of the algorithms. For this reason, there is a demand for standard neural network training benchmark architectures, which are deliberately designed for hard-to-train tests to uncover advantages and/or shortcoming of the training algorithms (Zhu et al., 2018; Fong et al., 2018). Such application-independent benchmarking is particularly useful to pinpoint major drawbacks of existing popular training algorithms and helpful to indicate new research directions for the ongoing research efforts.

This study investigates training performances of 9 widely used backpropagation training algorithms (e.g. LM, BFG, CGB etc.), which are implications of the gradient based optimization approaches, and 3 popular metaheuristic search methods (GA, PSO GWO) in the training problem of pipe-like deep neural network benchmark architectures. The pipe-like architecture of deep neural networks complicates the training process due to forming a long and narrowed learning bottleneck via an elongated feedforward path of neurons. In the experimental work, the body fat percentage estimation dataset, which may express bio-complexity of human metabolism, is used in training of this architecture with different pipe lengths (hidden layer counts). For the overall performance assessment of the training methods in pipe-like neural network architecture, a NOPS scheme is employed and the overall training performance and performance criterion-based selection of the training algorithms are shown. The addressed problems, novelties and main contributions of this study can be summarized as:

(i) A pipe-like neural network benchmark architecture, which is called pipe-like neural network benchmark architecture, is introduced. The pipe-like neural network benchmark architecture forms an extended bottleneck for the learning process. The training difficulties of this hard-to-train benchmark architecture are theoretically analyzed in the aspect of Baldi et al.'s learning channel theorem of neural networks. This effort contributes to using practical implications of the learning channel theorem for investigating training problems in deep neural networks.

(ii) A theorem to consider upper bounds of the training performance for gradient-based training algorithms is suggested. This theorem implements sensitivity derivative analysis on the pipe-like deep neural learning channel and it conjectures essential mechanisms that lead to gradient vanishing problems in deep neural networks. Suggestions of this theorem were observed in the experimental study and used to explain essential reasons for training performance degradations of gradient-based training in our experimental test.

(iii) To the best of our knowledge, this is the first experimental study that tests training performances of 9 popular NNBA algorithms and 3 fundamental MSA algorithms on the elongated pipe-like benchmark architectures from shallow one to deeper networks. The training performances of these algorithms are reported without any application-specific bias. The experimental results revealed performance drawbacks of these training algorithms and results indicated a requirement of designing new deep learning dedicated optimization algorithms, which are particularly specialized for training of deep neural network algorithms.

(iv) To enable performance criterion-based selection of the training algorithms, a NOPS scheme is illustrated. By allowing importance weighting of the performance criterion, NOPS can contribute to the solution of automated training algorithm evaluation and selection problems.

2. Preliminaries And Theoretical Background

This section provides preliminary knowledge on NNBA and MSA training algorithms that are tested in experimental study. Then, training difficulties of the pipe-like neural network benchmark architecture are theoretically analyzed by suggesting a theorem for upper bounds of training performance of this benchmark architecture. This theoretical background establishes a theoretical foundation to demonstrate suitability of the pipe-like neural network benchmark architecture for testing of training algorithm performances.

2.1. Backpropagation Training Algorithms

Training of an artificial neural network mathematically refers to finding optimal values of weight coefficients in order to minimize a predefined loss function. The loss functions are mathematical expressions that are used to evaluate the quality of the learning activity on the net. This optimization problem complicates as the number of layers in a neural network increases because of the increased function compositions to represent the network function (functional complexity of the network function) and growing weight and bias coefficient numbers to be optimized (high dimensionality of the optimization problem). To deal with complications associated with the training of multilayer feedforward neural network architectures, a "backpropagation" algorithm was proposed to cope with functional complexity of multilayer networks (Kim, 2017; Goodfellow et al., 2016). Then, the backpropagation has become the most fundamental training algorithm, and it has found a wide application area for the training of multilayer neural networks. It involves two essential stages (Kim, 2017): firstly, the input data propagates towards the output layer of neural networks; this stage is forward propagation of information, afterward, the error information, which is calculated at the output of the neural network, propagates back into the input layer of the neural network. This is the process of the backward propagation of the error information, where the term "backpropagation" comes from. Several training algorithms have been proposed to improve performance of the backpropagation training approach (Hagan et al., 1996; Cömert and Kocamaz, 2017; Hagan and Menhaj, 1994; Dennis and Schnabel, 1996; Mosavi et al., 2016; Pan et al., 2013). Table 1 shows a list of up-to-date backpropagation training algorithms that have been preferred in the training of multilayer feedforward neural networks in the current study. Training performance of backpropagation algorithms decrease when the hidden layer number of the feedforward neural networks increases because of the gradient vanishing problems: The decrease in gradient magnitudes can severely deteriorate the backward error propagation toward input layers, the weight updates slows down, and the training in deep layers begins to cease, practically. Therefore, consistent training performance analysis of algorithms should be carried out for deep and complicated neural network configurations.

Table 1. The NNBA Types, Which Are Tested in This Study, and Their Abbreviations

Training Algorithms	Abbreviations	Related Works
Levenberg-Marquardt backpropagation	LM	(Hagan et al., 1996; Hagan and Menhaj, 1994; Powell, 1977)
Quasi-Newton backpropagation with Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update	BFG	(Dennis and Schnabel, 1996)
Conjugate gradient backpropagation with Powell-Beale restarts	CGB	(Powell, 1977; Beale, 1972)
Conjugate gradient backpropagation with Polak-Ribiére updates	CGP	(Hagan et al., 1996; Scales, 1985 ; Fletcher, 1964)
Conjugate gradient backpropagation with Fletcher-Powell updates	CGF	(Hagan et al., 1996; Scales, 1985; Fletcher, 1964
Variable learning rate gradient descent	GDX	(Vogl et al., 1988)
One-step secant backpropagation	OSS	(Battiti, 1992)
Resilient backpropagation	RP	(Riedmiller and Braun, 1993)
Scaled conjugate gradient backpropagation	SCG	(Moller, 1993)

2.2. Metaheuristic Training for Artificial Neural Networks

Metaheuristic optimization methods are easy-to-use, optimal solution seeking tools that have been widely implemented in engineering problems. They have been particularly used when the optimization problem is too complex to be solved numerically or analytically (Wong and Ming, 2019). These methods employ a set and trial search strategy that can provide a straightforward solution for empirical, complicated, even not-well structured optimization problems. This introduces an important advantage that makes them preferable in real-world engineering application works. However, they have some disadvantages such as concerns about the dependability of solutions, inefficiency of these methods while searching in high dimensional search spaces etc.

The candidate solution selection strategies of metaheuristic algorithms are very substantial for convergence performance of metaheuristic optimization, and they establish major discrimination points between metaheuristic optimization techniques. Some major problems, which were observed in use of the metaheuristic methods, can be summarized as (Gogna and Tayal, 2013; Chopard and Tomassini, 2018; Hinton, Salakhutdinov, 2006; Igel, 2014):

(i) Contemporary metaheuristic optimization methods cannot convey definite information whether or not the solution is globally optimal (Gogna and Tayal, 2013). However, the quality of solutions can be evaluated with the value of objective functions in applications (Parejo et al., 2012; Gunantara et al., 2019).

(ii) Metaheuristic optimization is commonly effective in the low dimensional search spaces. An increase in the dimension of search spaces, that is, more parameters to optimize, severely reduces the converge performance in the metaheuristic searching because of the exponential growth in the exploration fields for the set and trial searching of search agents. When the number of parameters, namely the dimension of the problem, increases, the computation time of the algorithms severely arises (Chopard and Tomassini, 2018; Parejo et al., 2012).

(iii) Many metaheuristic methods need the finite search spaces that are confined by the predefined search ranges of the optimized parameters. This may become an important shortcoming that can reduce practical efficiency of metaheuristic methods because it is not always an easy problem to determine predefined search ranges of parameters, which can include the global minimum or maximum. Therefore, improper configuration of parameter search regions can limit performance of metaheuristic search (Chopard and Tomassini, 2018; Birattari and Kacprzyk, 2009).

(iv) Metaheuristics methods commonly utilize random number generation in the search processes to differentiate search paths. However, this brings a problem of unrepeatability of the solutions even though the computers can generate pseudo-random numbers. The reliable results require a statistical evaluation of the repeated optimization tasks. However, the rerunning of the algorithms many times can considerably increase the computational burden (Gogna and Tayal, 2013; Parejo et al., 2012).

When the depth of a multi-layer feedforward neural network increases, training of the neural network introduces difficulties associated with gradient magnitudes for gradient-based optimization techniques. Therefore, there exist several efforts to implement gradient-free metaheuristic optimization as a substitute for backpropagation methods (Sexton and Gupta, 2000; Che et al., 2011; Gudise and Venayagamoorthy, 2003; Ince et al., 2010; Mosavi et al., 2016). Since a majority of metaheuristic methods perform gradient-free optimization algorithms, they are expected to present advantages over the gradient-based optimization algorithms when gradient calculations are problematic or ineffective. For the training of the multilayer feedforward networks, three widely preferred popular metaheuristic optimization methods in the literature are listed in Table 2 (Che et al., 2011; Ince et al., 2010; Mosavi et al., 2016). These methods are commonly used to minimize the sum of square error loss function of neural networks, which is written by

$$\min E(w) = \frac{1}{2}\sum_{i=1}^{k} e_i^2 = \frac{1}{2}\sum_{i=1}^{k} (f_i(w, x_j) - y_i)^2 \qquad (1)$$

where, $f_i(w, x_j)$ represents i^{th} output function of the neural network for the input vector x_j. Training set is formed by the input-output data pairs $(x_j, y_1, y_2, .., y_k)$. The weight vector w represents a collection of the weight coefficients of neural networks, which are optimized during the training of the network via a metaheuristic method. (Bias coefficients are computationally assumed as an input with a weight value of 1). Number of parameters to optimize, n layer neural network can be written by $D = \sum_{h=1}^{n} (k_{(h-1)} k_h + k_h)$, where k_h is the number of neurons in the hidden layer h and k_0 is the number of inputs in a neural network. Parameter number D determines the dimension of the search space. The search space expands a D dimensional hypercube by the term $[w_l, w_u]^D$, where $[w_l, w_u]$ is the upper and lower boundaries of weight coefficient search ranges. Due to the exponential growth of the search space volume, the metaheuristic algorithms in Table 2 have been utilized in the training of shallow neural networks (Che et al., 2011; Ince et al., 2010; Mosavi et al., 2016). It will be useful to test these methods in training of the pipe-like architecture of deep neural networks to better observe the inherent shortcomings of the metaheuristic training approach.

Table 2. The MSA Types, Which Are Tested in This Study, and Their Abbreviations

Metaheuristic Search Algorithms	Abbreviations	Related Works
Genetic Algorithm	GA	(Sexton and Gupta, 2000; Che et al., 2011; Melanie, 1996 ; Michalewicz, 1992
Particle Swarm Optimization	PSO	(Gudise and Venayagamoorthy, 2003 ; Ince et al., 2010 ; Zeugmann et al., 2011)
Gray Wolf Optimization	GWO	(Mosavi et al., 2016 ;Mirjalili and Mirjalili S., 2014 ;Faris et al., 2018)

2.3. A Theoretical Background on Pipe-like Architecture Deep Neural Network

Architecture types of neural networks affect the learning skills, and the proper selection of network architecture is a very important stage for the training performance (Ince et al., 2010; Hornik, 1991). Fundamentally, feedforward neural network architectures are divided into two main categories: These are the single layer neural networks in Figure 1 and the multi-layer neural networks in Figure 1. Later, multi-layer feedforward neural networks were grouped into two major subcategories that are the shallow neural networks and the deep neural networks. Theoretical discussions on the architecture or the configuration of artificial neural networks are continuing on the bases of universal approximation theorems (Schmidhuber, 2015 ; Winkler and Le, 2017; Hornik, 1991; Csaji, 2001; Kratsios and Bilkoptov, 2020).

Conceptually, increasing the tunable parameter numbers (layers, weights, bias, generic activation function) can enhance fitting capability to data because it increases complexity in the compositional function representation of neural network and accordingly allows representation of more composite function models by using the deep neural network function $f_i(w, x_j)$. Thus, deep neural networks present potential of better approximating to higher complexity compositional function models that cannot be well approximated by a shallow neural network (Mhaskar et al., 2016). On the other hand, in addition to enhancing approximation capability, increasing the depth of feedforward network was observed to enable more abstraction of the learned knowledge in the deeper layers similar to the biological neural network can do, for instance the convolution neural network can detect more complex features (irregular shape) in deep layers by using more primitive features (e.g. edges, curves) that are detected in the shallow layers (Deng and Yu, 2013). However, the overfitting problem, which causes a reduction in the generalization property in learning, emerges in deep neural networks because too many tunable parameters highly increase the approximation capability of neural network function to each data point in the training set. Overfitted models cause serious performance degradations in data applications in the cases of noisy data or insufficient data in training sets. Therefore, a proper generalization is preferable in data applications to obtain satisfactory practical performance.

In order to compare training performances of the training methods, hard-to-train neural network architectures are more suitable to reveal the performance shortcomings and superiorities. In this manner, authors used a pipe-like architecture deep neural network in the performance tests. The width of layers was set to 5 neurons and the length of the neural network was increased up to 20 layers by adding a hidden layer at each test. Figure 2 shows a depiction of the trained 5 neurons wide and 20 layers long pipe-like benchmark architecture. This generic pipe-like structure enables testing several pipe architecture versions from the shallow one to the deep networks by adjusting the hidden layer counts.

The communication and signaling phenomenon between neurons indicate the communication nature of the neural networks (Tagluk and Isik, 2019). A neural network resembles an information-adaptive narrowband communication channel with a learning property between the input layer and the output layer of the neural network. Previously, Baldi et al. have suggested the existence of a physical learning channel to convey information via the weights of the network (Isik and Sadowski, 2016) and they investigated the capacity of learning algorithms by considering the error gradients per weight. Thus, Baldi et al. conjectured a foundation for the learning channel theorem of neural networks. The current study contributes to the learning channel theorem of neural networks in a practical aspect by considering distribution of weight update magnitudes throughout a pipe-like deep neural network. Figure 2 depicts a one-neuron wide, pipe-like network fraction (at middle schematic) that is composed of the hidden neuron model (at bottom schematic). Let's investigate the weight update magnitudes for this network to show hard-to-train nature of pipe-like benchmark networks:

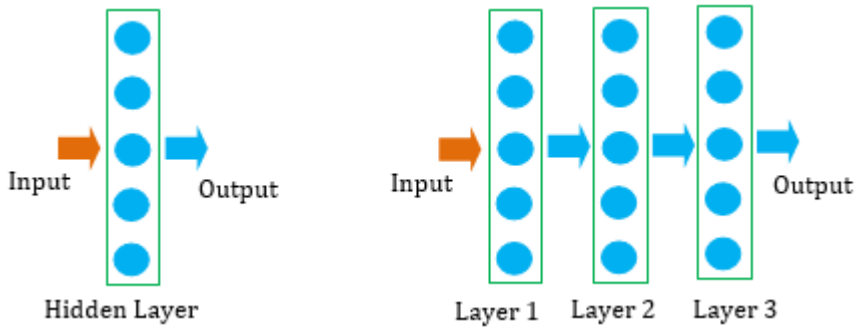

Figure 1. A Representation Of A Single Layer Neural Network With 5 Neurons (On The Left-Hand Side) And A Multi Layer Neural Network With Three 3 Hidden Layer Of The 5 Neurons (On The Right-Hand Side)

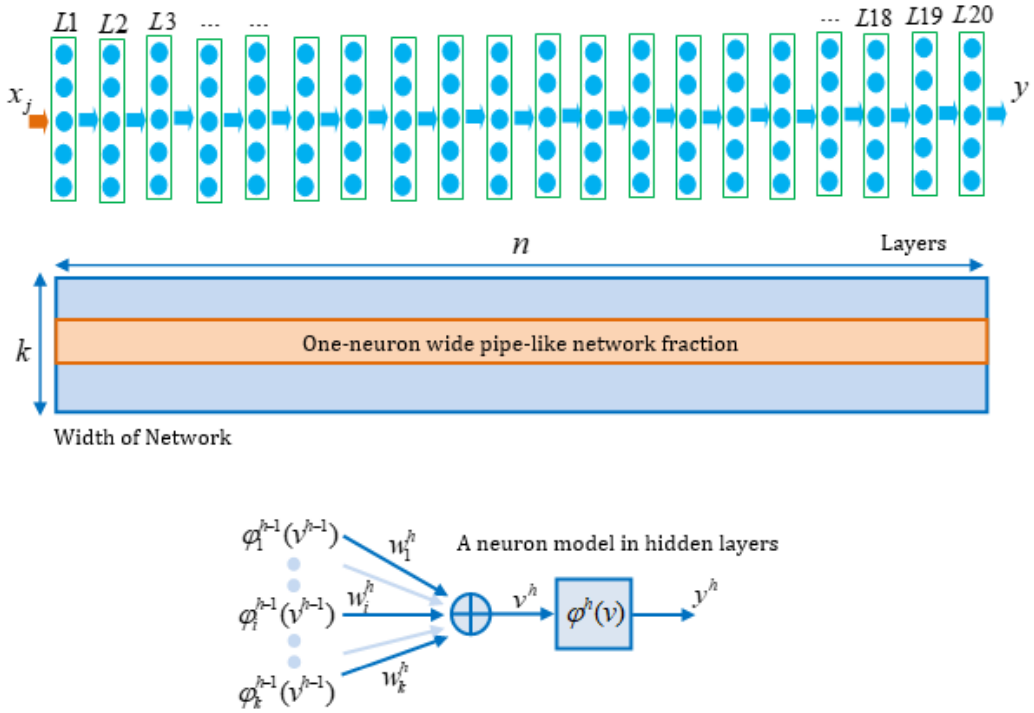

Figure 2. A Block Diagram Of A 5 Neuron Wide And 20 Layer Long Pipe Architecture Neural Network (Top Schema), A Depiction Of One-Neuron Wide Pipe-Like Network Segment (Middle Schema), One Neuron In The Hidden Layer (Bottom Schema)

Essentially, the pipe-like architecture results in a long bottleneck for the training process because a pipe-like architecture forms a narrowed and elongated network for forward and backward propagation of neural information and it exhibits two major training drawbacks for gradient based training algorithms that are the propagation delay problems related to narrowing of network and gradient vanishing problems related to elongation of network. To analyze these properties associated with the network architecture, let's denote the output function of the layer h in the neural network by the function $f^h(w^h, x_j^h)$, where the superscript h is the layer index. When the layer number increases by adding more layers, the resulting output function of a deep neural network with n layers can be commonly expressed by using function compositions as (Mhaskar et al., 2016; Isik and Sadowski, 2016; Strang, 2018)

$$y = f^n(w^n, f^{n-1}(w^{n-1}, f^{n-2}(w^{n-2}, \ldots f^2(w^2, f^1(w^1, x_j^1))\ldots))) \quad (2)$$

where the superscript n is the depth of this function composition.

By considering activation function and the weighted sum of neuron inputs in the form of $y^h = f^h(w^h, x^h) = \varphi^h(\sum w_i^h . \varphi_i^{h-1}) = \varphi^h(w^h \varphi^{h-1})$ (Isik and Sadowski, 2016; Mhaskar et al., 2016), the output of whole neural network is considered in the form of

$$y = \varphi^n(\sum w^n \varphi^{n-1}(\sum w^{n-1}\varphi^{n-2}(\sum w^{n-2}....\varphi^2(\sum w^2\varphi^1(\sum w^1 x_j))...))) \quad (3)$$

This equation suggests a composite learning function family of the weight coefficient vector w^h. The $\varphi^h(.)$ stands for the activation in the layer h. The weighted sum of neuron inputs in the layer h can be represented by the weighted sum of previous layer outputs $v^h = \sum w_i^h.\varphi_i^{h-1}$. For every optimal determination of the weight coefficient set of $w^* = \{w^1, w^2, w^3, .., w^h\}$ at a minima of the loss function, the results yield a learned function $y_j = f(w^*, x_j)$ for an input vector x_j from the training set. The pipe-like architecture mainly narrows the width of the network, and this decreases the number of the weighted sum terms in the output function of the network. Consequently, this decreases the complexity of the composite learning function family that is represented by the Equation (3). Accordingly, a reduction in the complexity of the composite learning function tree decreases the representative nature of the output function. This is a factor that complicates the training tasks of a pipe-like architecture for all training algorithms.

The gradient-based algorithms use delta rule for the weight updates,

$$w^h \leftarrow w^h + \Delta w^h \quad (4)$$

where Δw^h is the weight update (Kim, 2017; Isik and Sadowski, 2016). The weight updates Δw^h are commonly performed in directions where the loss function decreases. Therefore, sensitivity function $\frac{\partial E}{\partial w^h}$ is widely used for the weight update $\Delta w^h = -\eta \frac{\partial E}{\partial w^h}$ in order to detect the descent directions of the loss function, where $\eta > 0$ is the learning rate that is commonly used to regulate converge rate of gradient-based optimization techniques. The magnitude of the weight updates $\left|\Delta w^h\right| = \eta \left|\frac{\partial E}{\partial w^h}\right|$ is an indicator to evaluate penetration of learning through the network. To consider training difficulties of pipe-like neural network architecture for gradient-based backpropagation training algorithms, it is useful to investigate an upper boundary of sensitivity function magnitude in the network.

Theorem 1 (An upper bound for sensitivity derivatives in pipe-like neural network models):
In cases of k numbers of the neuron in the each layer and gradient-based training algorithms, an upper bound for sensitivity derivatives at the first layer neurons can be expressed as

$$\left|\frac{\partial E}{\partial w^1}\right| \le \sum_i^k \left|\frac{\partial E}{\partial \varphi_i^n}\right| \left|\frac{\partial \varphi_i^n}{\partial v^n}\right| \left|w^n\right| \left|\frac{\partial \varphi_i^{n-1}}{\partial v^{n-1}}\right| \left|w^{n-1}\right| ..... \left|w^3\right| \left|\frac{\partial \varphi_i^2}{\partial v^2}\right| \left|w^2\right| \left|\frac{\partial \varphi_i^1}{\partial v^1}\right| \left|x_j\right| \quad (5)$$

Proof: To calculate sensitivity derivative $\frac{\partial E}{\partial w^h}$ through the layers, a chain rule of derivative operators is implemented to cope with the composite function form of the neural network output function. When the chain rule of derivative operator is used for the one-neuron wide and n layer pipe-like architecture, the sensitivity derivative can be written by

$$\Delta w^1 = -\eta \frac{\partial E}{\partial w^1} = -\eta \frac{\partial E}{\partial \varphi^n} \frac{\partial \varphi^n}{\partial v^n} \frac{\partial v^n}{\partial \varphi^{n-1}} \frac{\partial \varphi^{n-1}}{\partial v^{n-1}} \frac{\partial v^{n-1}}{\partial \varphi^{n-2}} ..... \frac{\partial v^3}{\partial \varphi^2} \frac{\partial \varphi^2}{\partial v^2} \frac{\partial v^2}{\partial \varphi^1} \frac{\partial \varphi^1}{\partial v^1} \frac{\partial v^1}{\partial w^1} \quad (6)$$

for an update of weights in the first hidden layer (Roodschild et al., 2020). Here, by considering at the hidden layers $\frac{\partial v^h}{\partial \varphi^{h-1}} = w^h$ and the first hidden layer $\frac{\partial v^1}{\partial w^1} = x_j$, the magnitude of sensitivity at the first hidden layer is expressed by the weight update magnitude as

$$\left|\frac{\partial E}{\partial w^1}\right| = \left|\frac{\partial E}{\partial \varphi^n}\right|\left|\frac{\partial \varphi^n}{\partial v^n}\right| w^n \left|\frac{\partial \varphi^{n-1}}{\partial v^{n-1}}\right| w^{n-1} \ldots\ldots w^3 \left|\frac{\partial \varphi^2}{\partial v^2}\right| w^2 \left|\frac{\partial \varphi^1}{\partial v^1}\right| x_j \tag{7}$$

For k numbers of the neuron in the each layer (i.e., the learning channel width of k-neurons), one can aggregate contributions of all one-neuron path segments (See middle schema in Figure 2) in order to state an upper boundary for the sensitivity function magnitudes:

$$\left|\frac{\partial E}{\partial w^1}\right| \le \sum_i^k \left|\frac{\partial E}{\partial \varphi_i^n}\right|\left|\frac{\partial \varphi_i^n}{\partial v^n}\right| \left\|w^n\right\| \left|\frac{\partial \varphi_i^{n-1}}{\partial v^{n-1}}\right| \left|w^{n-1}\right| \ldots\ldots \left|w^3\right| \left|\frac{\partial \varphi_i^2}{\partial v^2}\right| \left|w^2\right| \left|\frac{\partial \varphi_i^1}{\partial v^1}\right| \left\|x_j\right|$$

Some Remarks and A Numerical Example:

By using Equation (5), an upper boundary for the weight updates at the first layer neurons can be obtained

$$\left|\Delta w^1\right| = \eta \left|\frac{\partial E}{\partial w^1}\right| \le \eta \sum_i^k \left|\frac{\partial E}{\partial \varphi_i^n}\right| \left|\frac{\partial \varphi_i^n}{\partial v^n}\right| \left\|w^n\right\| \left|\frac{\partial \varphi_i^{n-1}}{\partial v^{n-1}}\right| \left|w^{n-1}\right| \ldots\ldots \left|w^3\right| \left|\frac{\partial \varphi_i^2}{\partial v^2}\right| \left|w^2\right| \left|\frac{\partial \varphi_i^1}{\partial v^1}\right| \left|x_j\right| \tag{8}$$

This boundary reveals a restrictive factors for the learning activity of the pipe-like neural networks: The learning tendency of the network at each training epoch depends on the magnitude of activation function derivatives $\left|\frac{\partial \varphi^h}{\partial v^h}\right|$, the magnitude of weight coefficients $\left|w^h\right|$, the learning rate η and the depth of the network (n). For the worst-case analysis, each partial derivative term of activation functions can be assumed $\left|\frac{\partial \varphi^h}{\partial v^h}\right| < 1$ and the weight magnitudes are considered $\left|\frac{\partial v^h}{\partial \varphi^{h-1}}\right| = \left|w^h\right| < 1$, these conditions cause the whole chain rule approximating to zero, $\left|\frac{\partial E}{\partial w^1}\right| \to 0$, as the number of hidden layer (n) increases because of the subsequent multiplication of $\left|\frac{\partial \varphi^h}{\partial v^h}\right| < 1$ and $\left|w^h\right| < 1$ terms. This effect results in the weight updates also converging to zero, $\left|\Delta w^1\right| \to 0$, and it can severely slow down the training process.

According to Theorem 1, the deep pipe-like network encounters two major training complications for NNBA methods: (i) gradient vanishing problem ($\frac{\partial E}{\partial w^1} \to 0$) because of the multiplication of terms with $\left|\frac{\partial \varphi^h}{\partial v^h}\right| < 1$ and $\left|w^h\right| < 1$ as the hidden layer number increases, and (ii) slowing down the training process though the pipe-like networks as neuron counts k (the channel width) decreases in the narrowed layers.
For a numerical illustration of the gradient vanishing problem in the pipe-like neural networks, it is useful to consider upper bounds of sensitivity derivatives at the first hidden layer for a fundamental activation function. For illustration purpose, a pipe-like neural network with the sigmoid activation function ($\varphi(v) = \frac{1}{1+e^{-v}}$) is investigated based on the inequality (5). The unit weight coefficients $\left|w^i\right| = 1$ and the

unit input $|x_j| = 1$ are assumed in order to compensate for effects of weight and input parameters. Thus, one can consider only effects of the activation function selection on the training process. Let us consider sigmoid activation functions, it is obvious that $\left|\frac{\partial \varphi^h}{\partial v^h}\right| \leq 0.25$. For an upper boundary analysis, the maximum value $\left|\frac{\partial \varphi^h}{\partial v^h}\right| = 0.25$ can be used in inequality (5). Figure 3 shows change of logarithmic scaled sensitivity function amplitude ($\left|\frac{\partial E}{\partial w^1}\right|$) at the first layer neurons while increasing hidden layer numbers up to 20 layers. The upper bound of the sensitivity derivative up to 20 hidden layer network is about $\left|\frac{\partial E}{\partial w^1}\right| \leq 9.09 \cdot 10^{-13}$, which also implies that the weight updating $|\Delta w^1| \leq \eta 9.09 \cdot 10^{-13}$ is at a negligible level at the first layer. This problem suppresses backward propagation of error signals through deep neural networks. This result indicates a severe gradient vanishing problem with almost cessation of the training process under the presumed conditions. However, according to this conjecture, one can suggest that the learning rate η, can be adaptively used to amplify the sensitivity function amplitudes to deal with this issue.

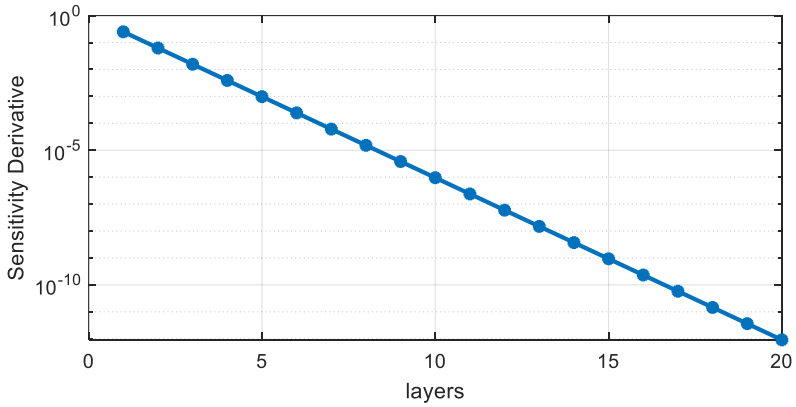

Figure 3. The Level Of Sensitivity Function In Logarithmic Scale At The First Hidden Layer ($\left|\frac{\partial E}{\partial w^1}\right|$) For Sigmoid Activation Functions

3. An Experimental Study

This section reports experimental results for training performance of 9 popular NNBA algorithms and 3 fundamental MSA algorithms on the elongated pipe-like benchmark architectures. The advantages and disadvantages of the tested training algorithms are revealed in training of this benchmark architecture.

3.1. A Comparison of Training Algorithms from Shallow to Deep Training

Artificial neural networks have been widely used for the black-box modeling of the physical and biological systems and they have been implemented in the model-based prediction problems (Chen et al., 2020; Zhao et al., 2020). Due to higher level of complexity and chaotic dynamics, biological system modeling benefits from highly nonlinear function approximation skills of the artificial neural networks. To conduct the training experiments, the body fat rate estimation problem is solved by using body fat data from Matlab. This dataset is composed of 13 anatomical measurements of the human body as input data, and the neural network is trained to predict body fat percentages based on these anatomical features (Zamri et al., 2018). This dataset involves highly nonlinear relations and noisy data, which make it preferable for training algorithm tests.

Figure 4 shows the average MSE performance of NNBA training tests. Figure 5 and Figure 6 illustrate the maximum MSE and the minimum MSE performances of NNBA training processes for 10 repeated tests. A large increase in layer counts cannot consistently improve MSE performance because of emergence of the

gradient vanishing and saturation problems as suggested in remarks of theorem 1. In overall, one can observe that the BFG, LM, CGB and RP backpropagation algorithms provide an improved average MSE performance when training the neural network architectures from 1 layer to 10 layers. However, for 10 repeated tests, the minimum MSE performances are produced by the LM algorithm for all configurations in Figure 6. These results indicate the potential of the LM algorithm to reach the lowest MSE when training of the neural networks is repeated adequately.

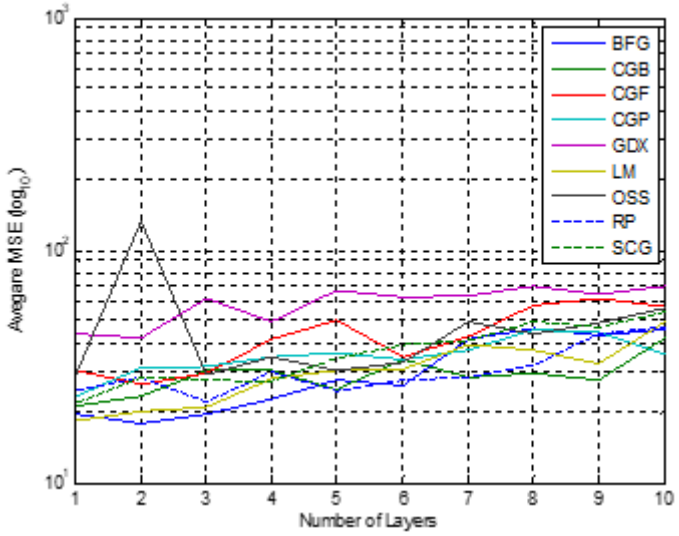

Figure 4. Average MSE Performances of The NNBA Trainings

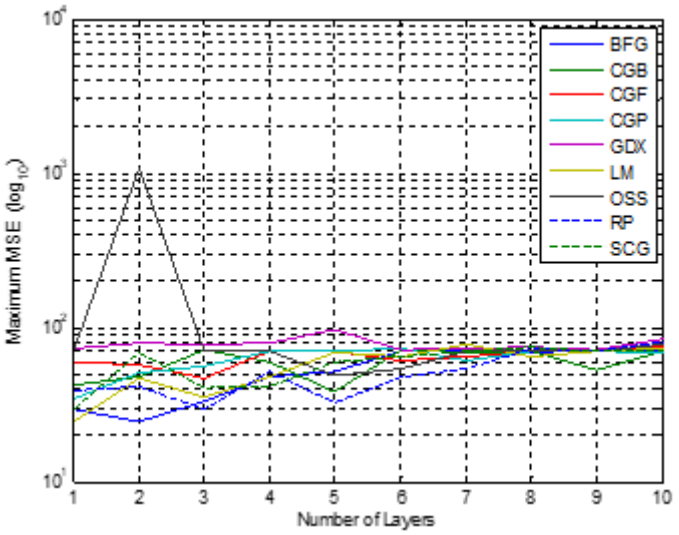

Figure 5. Maximum MSE Performances of The NNBA Trainings

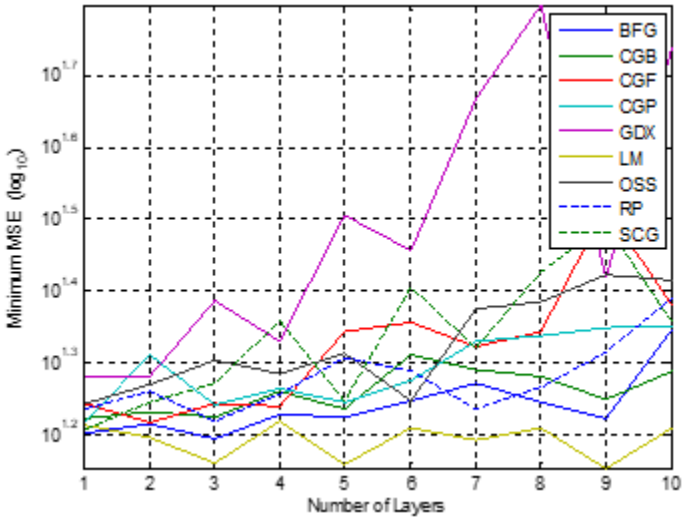

Figure 6. Minimum of MSE performances of the NNBA trainings

Figure 7 shows the average MSE performance of the training tests by using MSAs. Figure 8 and 9 illustrate the maximum MSE and the minimum MSE performances for 10 repeated training tests of each neural network configuration. Increasing the hidden layer number to 2 layers or more severely deteriorated the tested MSE performances because MSAs need much more computation time in searching optimal points as the dimension of search spaces increases. The dimension of search space ($D = \sum_{h=1}^{n} (k_{(h-1)} k_h + k_h)$) grows fast with the layer number n in neural networks. Another factor that affects the performance of MSAs is the geometry of search spaces. Loss functions of shallow networks introduce rather multimodal search spaces and a higher exploration skill becomes an advantage to find better solutions, and this effect can increase average performance of search agents in multimodal search spaces. As the number of hidden layers increases in neural networks, the convexity of their search space increases because more hidden layers increase depth and optimization parameters, and accordingly approximation performance of the composite neural network function.

Figure 7 and 8 reveal that the GA can provide better average MSE and maximum MSE performances up to 8 hidden layer networks. A main reason for this result is that the GA is more explorative than swarm-based search algorithms (GWO, PSO) because of randomly applied genetic processes such as random mutation and crossover. These processes occasionally lead to random spreading of population into the search space at each generation, and such dispersion of individuals makes it more probable to find better solutions in multi-modal search space of shallow networks. On the other hand, GWO and PSO algorithms perform more exploitative search because search agents (individuals) of the swarm tend to move towards the best individual. Therefore, minimum MSE performance of GWO and PSO begins to improve after 8 hidden layers in Figure 9 as a result of the increase in convexity of the search space. Due to enhanced exploitation capability of GWO, the GWO algorithm can provide the lowest minimum MSE values when the training is repeated adequately. These results reveal that the GWO algorithm can be advantageous to obtain a minimum MSE in the case that the training of the deep neural networks is repeated.

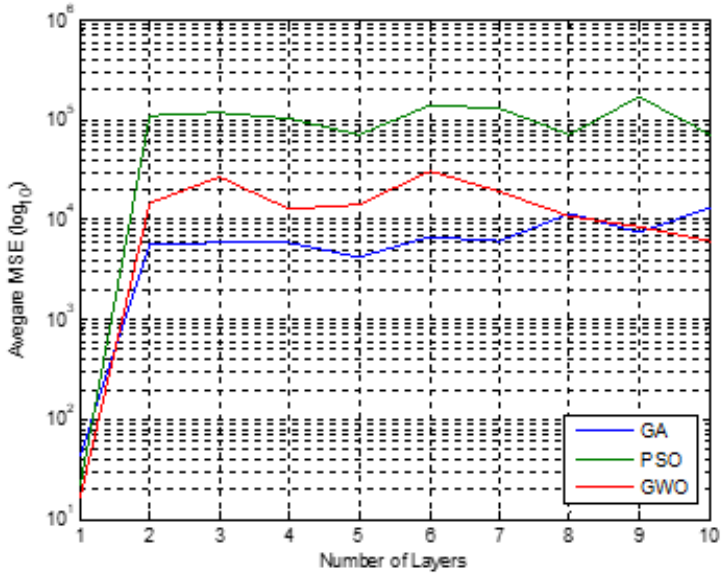

Figure 7. Average MSE Performances of The MSAs

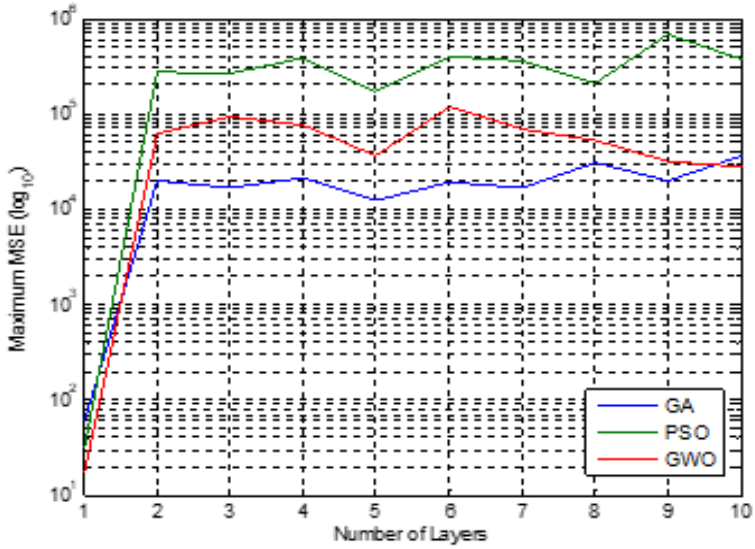

Figure 8. Maximum MSE Performances of The MSAs

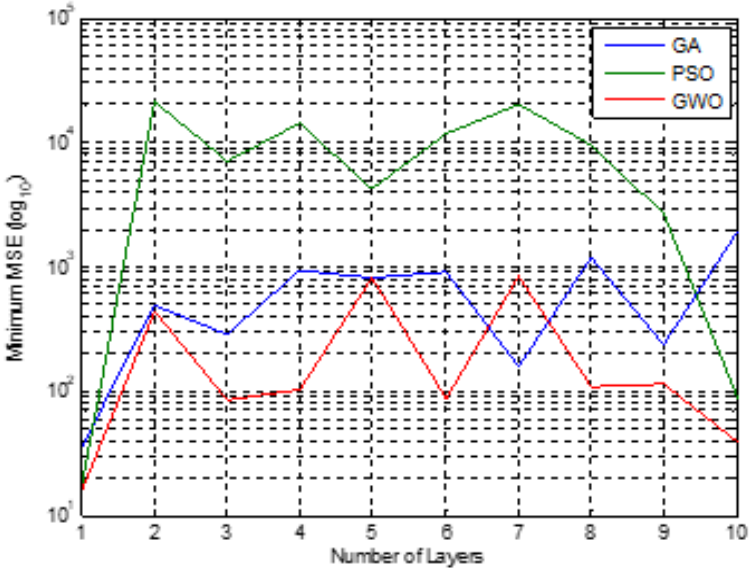

Figure 9. Minimum MSE Performances of The MSAs

The computation time per weight parameter is useful to consider the computation load of an optimization method in the training process. It expresses the average of training time that a training algorithm is used to perform a weight update. It eliminates effects of network depth on training time measurements of algorithms. It is computed by dividing the total computation time (T_c) by the number of weight parameters (N_w) as T_c / N_w. Figure 10 shows the average computation times per weight in the training processes of NNBAs. The figure reveals that the BFG training method consumes considerably more time during the training process. Figure 11 shows the average computation times per weight in MSAs during the training process. The figure reveals a lower computation time of the GA algorithm compared to GWO and PSO algorithms in this regression problem because automatic stopping criteria of Matlab ga() can decrease the computation time. GWO and PSO algorithms do not stop before performing the maximum iteration number (Maximum iteration number is 1000 iterations for GWO and PSO). Figure 12 compares NNBAs and MSAs in terms of computation time per the weight coefficient. For these network configurations (up to 10 hidden layers), it is apparent that the computation times of MSAs is much higher than those of the NNBAs and such a high computation time is another important disadvantage of MSAs for the deep neural network training tasks. A reason for high computation time of MSAs is that MSAs are multi-agent (population-based) search algorithms, and they perform a loss function calculation for each agent of the population. This is an important factor that increases the training time of MSAs depending on the population size. Since population size and iteration numbers of MSAs are the same while the number of layers increases, the computation time per weight is relatively steady in Figure 11.

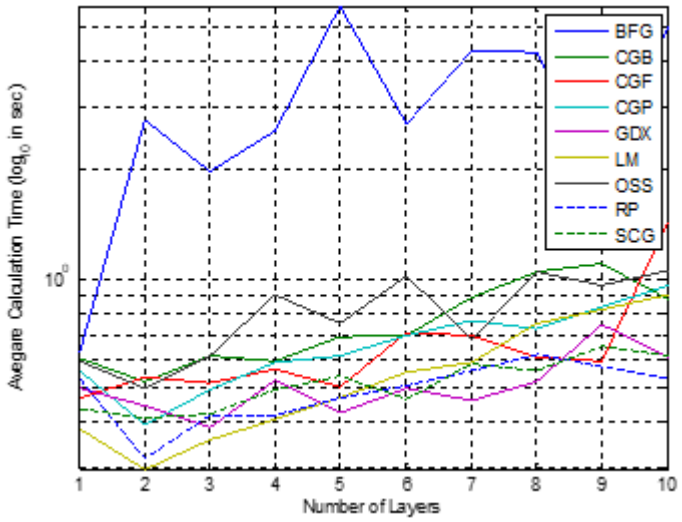

Figure 10. Computation Time (In Sec) During Training of The Neural Network Configurations By using NNBAs

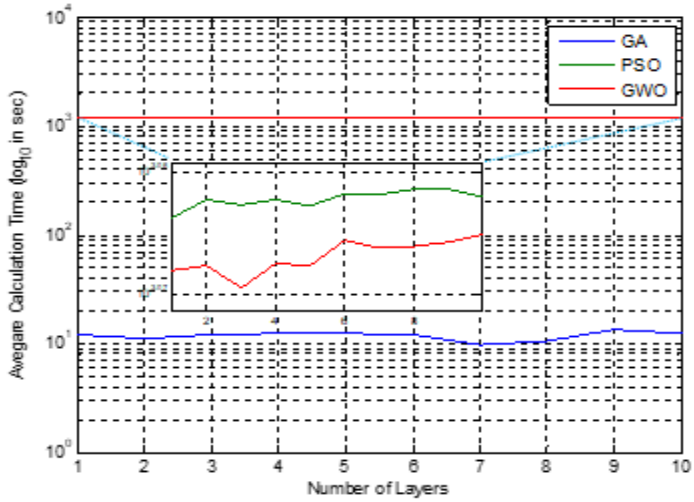

Figure 11. Computation Times (In Sec) During Training of The Neural Network Configurations By Using MSAs

Table 3 lists the training test results of algorithms on a pipe-like deep neural network architecture with 20 hidden layers. In the case of 20 hidden layers, the lowest average MSE is provided by the GWO algorithm

and the lowest maximum MSE is provided by the PSO algorithm as a result of the increased convexity in the search space. On the other hand, due to the gradient vanishing problem of the pipe-like deep neural network, which was analyzed by using Theorem 1, training performances of NNBAs can reduce to levels that are comparable with training performances of MSAs. The performance of MSAs deteriorated because of the high-dimensional search space with 646 optimization parameters. However, the lowest minimum MSE is still provided by the LM algorithm (The lower minimum MSE implies accessibility to the best MSE performance in the case of multiple training.) and the lowest standard deviation of the MSE is provided by the CGF algorithm. (The lower standard deviation implies consistency of the MSE performance in multiple tests.) The lowest average computation time per weight is provided by the RP algorithm because The RP uses the sign of the partial derivative of the activation functions and this improves convergence speed however reduces convergence accuracy (Riedmiller an Braun, 1993). Although the lowest average MSE is possible by using the GWO, it's computation time is much higher than all NNBAs. Table 3 showed that NNBA methods consume quite less computation time. Deep neural network training applications require faster algorithms in order to process large amounts of data. The speed of training algorithms is an important asset for deep neural network training in big data applications (Zhu et al.,2018).

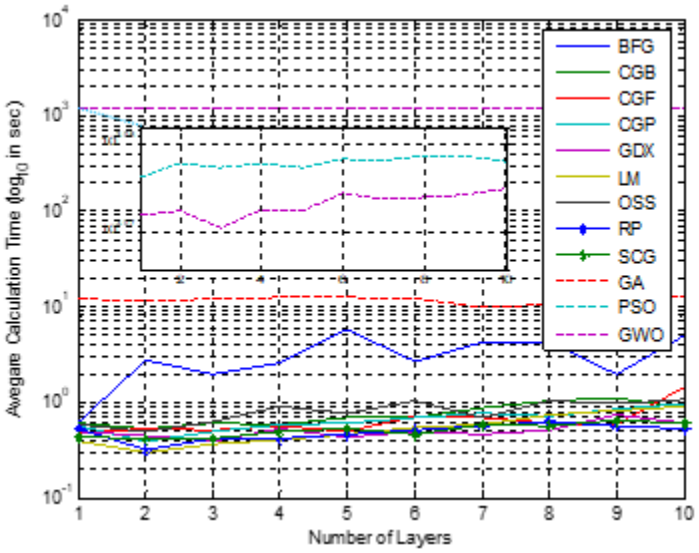

Figure 12. The Computation Time (In Sec) Comparison Between NNBAs And MSAs

When an optimization algorithm is developed to be successful in hard-to-training benchmark networks, it will be more specialized and more dedicated for deep neural network training tasks. This unveils a research motivation that the optimization algorithms should be designed dedicated to the neural networks "deep learning dedicated optimization algorithms" in order to surpass the training performance standards of general purpose optimization algorithms.

Table 3. Test Results of The Pipe Architecture Deep Neural Network With 20 Hidden Layers For Analyses Of MSE Performances And Computation Times For 10 Repeated Training (The Number Of Weights To Be Updated At Each Iteration Is 646)

Training Algorithm Types	Algorithm	Average MSE	Maximum MSE	Minimum MSE	Standard Deviation	Average Computation Time (Sec.) Per Weight Update
NNBA	LM	49.26	69.84	16.25	26.56	2.12
NNBA	BFG	54.00	74.24	30.08	15.61	11.28
NNBA	CGB	63.15	70.84	28.08	13.67	1.22
NNBA	CGP	62.60	82.50	28.86	15.09	1.23
NNBA	CGF	69.73	70.14	68.86	0.34	1.07
NNBA	GDX	89.43	259.83	69.78	59.88	1.00
NNBA	OSS	67.84	72.50	45.75	7.82	1.36
NNBA	RP	71.37	74.60	70.18	1.62	0.89
NNBA	SCG	62.60	69.92	43.00	11.52	1.25
MSA	GA	73.30	106.95	46.65	18.32	129.11
MSA	PSO	52.30	60.16	44.17	5.14	20964.90
MSA	GWO	43.26	66.89	31.30	10.21	20698.54

3.2. Overall Performance Analysis and Criterion-based Selection of Training Algorithms

For criterion-based assessment on the practical effectiveness of training algorithms, a normalized overall performance scoring (NOPS) is adopted to statistical properties in Table 3 as follows:

$$NOPS = \sum_{i=1}^{p} W_i \frac{P_i}{\sum_{j=1}^{m} P_i} \tag{9}$$

where the parameter P_i, $i = 1,2,..,p$ stands for the value of the properties (performance indices) in the analyses of m different algorithm options. Authors used 5 properties to evaluate training algorithm performance ($p = 5$). Accordingly, the property P_1 is the average MSE value, the property P_2 is the maximum MSE value, the property P_3 is the minimum MSE value, the property P_4 is the standard deviation and the property P_5 is the average computation time per weight. Previously, the weighted sum formula has been considered for decision-making problems (Stanujkic and Zavadskas, 2015 ; Goh et al., 1996). We modified it for the criterion-based selection of the training algorithms by using importance weighting. It should be noticed that all properties should be minimized for a desirable performance in this training algorithm selection problem. Therefore, an algorithm with a lower NOPS is better in criterion-based overall performance. The parameter W_i stands for the importance weight of the property i and the importance weight should satisfy the normalization condition $\sum_{i=1}^{p} W_i = 1$ to perform a weighted average. For the equal importance of the properties, the importance weight is set to a constant value of $W_i = \frac{1}{p}$. Table 4 and Table 5 show NOPS lists of NNBAs and MSAs for three type importance weighting: the equal importance weights with $W_i = [0.2\ 0.2\ 0.2\ 0.2\ 0.2]$, the NOPS_1 importance weights with $W_i = [0.4\ 0.1\ 0.2\ 0.0\ 0.3]$ (it attributes more importance for average and minimum MSE performances to express accuracy and for computation time to express speed of the training algorithms) and NOPS_2 importance weights with $W_i = [0.4\ 0.0\ 0.0\ 0.3\ 0.3]$ (it attributes more importance for average MSE and standard deviation to express accuracy and consistency and for computation time to express speed of the training algorithms). Due to their higher speed and accuracy requirements, NOPS_1 can be preferable for training algorithm selection tasks of deep learning applications for big data analytics. Since MSAs have very high computation time, NOPS calculations were separately performed for NNBAs and MSAs categories. According to the NOPS analyses, the LM, CGB and CGP training algorithms can be advantageous for speed and accuracy requirements due to their low scores in Table 4. Improvements on nonlinear optimization (Marquardt, 1963) can contribute to NNBAs. Among the tested MSAs in Table 5, GA algorithms can be useful in terms of speed and accuracy weighting. However, one should consider that these MSAs have extremely high computation time as shown in Table 3, and they are not effective for deep neural network training.

Table 4. NOPS Analyses of NNBAs For The Pipe Architecture Deep Neural Network With 20 Hidden Layers

Training Algorithm Types	Algorithm	NOPS for Equal Importance	NOPS_1 for Speed and Accuracy Importance	NOPS_2 for Speed and Consistency Importance
NNBA	LM	0.0608	0.0795	0.1155
NNBA	CGB	0.0552	0.0823	0.0869
NNBA	CGP	0.0591	0.0838	0.0894
NNBA	SCG	0.0583	0.0897	0.0827
NNBA	OSS	0.0572	0.0965	0.0805
NNBA	RP	0.0612	0.1047	0.0640
NNBA	CGF	0.0581	0.1049	0.0629
NNBA	GDX	0.1629	0.1402	0.1927
NNBA	BFG	0.0563	0.2184	0.2254

Table 5. NOPS Analyses of MSAs For The Pipe Architecture Deep Neural Network With 20 Hidden Layers

Training Algorithm Types	Algorithm	NOPS for Equal Importance	NOPS_1 for Speed and Accuracy Importance	NOPS_2 for Speed and Consistency Importance
MSA	GA	0.0773	0.2967	0.3378
MSA	GWO	0.1598	0.3309	0.3420
MSA	PSO	0.1476	0.3724	0.3202

3.3. A Discussion on Experimental Results:

Some significant observations from experimental results can be summarized as

(i) For training of the shallow networks with a hidden layer of 5 neurons, MSE performances of the tested NNBAs and MSAs are comparable. However, when two or more hidden layers were added to the network, MSE values of the MSAs sharply increased and the training performances severely deteriorated. While increasing the hidden layer numbers, MSE performances of NNBAs rather slowly deteriorate because the gradient vanishing problem gradually becomes effective as the layer number increases as suggested by Theorem 1. Therefore, authors concluded that the tested MSAs are suitable for the training of shallow neural networks. This shortcoming is mainly caused by the fact that the tested MSA methods are practicable for low dimensional optimization problems (Fong et al., 2018). An increase in hidden layers largely increases the number of weight and bias coefficients, namely the dimension of the search space of the loss function. For this reason, in order to train deep neural networks via MSAs, it is necessary to design algorithms that can be particularly effective for high dimensional optimization problems. Consequently, research efforts on high dimensional optimization problems will be very strategic for the deep learning research community. Authors anticipated that development of deep learning dedicated optimization methods can be more effective than the adaptation of general purpose optimization methods for neural network training. In a recent work, Manoharan et al. discussed performance improvement of neural network dedicated metaheuristics for several datasets (Manoharan and Sathesh, 2020).MSAs were preferably utilized in network architecture optimization problems because architecture optimization introduces lower-dimensional optimization problems compared to training tasks. MSAs can more effectively optimize hyper-parameters of neural networks such as layer numbers, neuron numbers and configuration of neural elements. Several works have discussed benefits of such neuroevolutionary approaches in deep learning (Stanley et al., 2019; Floreano et al., 2008; Suganuma, 2017; Ding et al., 2013; Galván and Mooney, 2021).

(ii) The computation load of the MSAs is much larger than those of the tested NNBAs. This is another substantial shortcoming of MSAs in the training of deep neural networks. The main reason is that metaheuristic optimization commonly uses multi-agents (population) global search techniques. NNBAs use the single search agent and perform the local search according to the gradient direction. This property of NNBAs becomes an advantage for reduction of algorithmic computation complexity and it can significantly reduce the computation load of the training process. Nonetheless, low average MSE results of GWO in Table 3 can support the idea, suggesting that the gradient free metaheuristic search can be a solution to deal with the complications associated with the gradient-based deep neural network training, such as the gradient vanishing and gradient exploding problems. However, the computation time of GWO is severely high, which becomes an important disadvantage for deep learning applications.

For consistent evaluation of training algorithms, hyperparameters of each training algorithm type have been configured to similar values. Some significant hyperparameters of the training algorithms are summarized in Table 6. Upper and lower bounds of weight coefficients are important parameters because they can affect the training performance for MSA methods. When these bounds are set symmetrical in positive and negative ranges, MSAs can obtain both negative and positive weight coefficients in the optimization stage. Besides, we configured the upper and lower bounds of weight coefficients in the narrow range of [-10, 10] as in Table 6. This can lead to a regularization effect (data generalization) in the neural learning process by preventing large differences between weight coefficients. Since the number of optimized weight coefficients in the training process exponentially grows the search space of MSAs, population size of MSAs can be increased to maintain the search performance for large search spaces.

Table 6. Hyperparameter Setting of Training Algorithms

Training Algorithm Types	Algorithm	HyperParameter Setting
NNBA	LM, BFG, CGB, CGP, CGF, GDX, OSS, RP, SCG	Number of epoch: 200, Minimum gradient magnitude: 1e-20, Error goal: 0, Numbers of neurons in hidden layers are 5, Activation function in hidden layers is hyperbolic tangent sigmoid transfer function, Activation function in output layer is linear transfer function, Number of weight coefficient: 646.
MSA	GA	Iteration number: Default stopping criteria of Matlab ga() function is used, Population size: 200, Number of parameters (dimension of optimization): 646, Lower bound of parameters: -10, Upper bound of parameters:10.
MSA	PSO	Maximum iteration number: 1000, Population size: 200, Number of parameters (dimension of optimization): 646, Lower bound of parameters: -10, Upper bound of parameters:10, Damping Ratio: 0.99, Personal Acceleration Coefficient: 2, Social Acceleration Coefficient: 2.
MSA	GWO	Maximum iteration number: 1000, Population size: 200, Number of parameters (dimension of optimization): 646, Lower bound of parameters: -10, Upper bound of parameters:10.

4. Conclusions

This study introduced the pipe-like neural network benchmark architecture and performance analysis results of popular training algorithms were reported for different depths of the generic pipe-like neural network architecture. Training difficulties of this network were theoretically demonstrated by extending the learning channel theory to an aggregated one-neuron learning channel conjecture in Theorem 1. The weight update bounds of aggregated one-neuron learning channels and gradient vanishing problem of pipe-like deep networks were analyzed for a fundamental neuron model. In the experimental studies, training processes of a pipe-like deep neural network took several weeks run-time for a high performance computer (Intel I7 processors and 16 GB RAM) without interruptions.

Some remarks and suggestions can be summarized as follows:

* Upper bounds theorem of training performance demonstrated training complications of the pipe-like neural network benchmark architecture on the basis of learning channel theorem, and the experimental study validates these performance complications for 9 popular NNBA algorithms and 3 fundamental MSA algorithms.

* A major weaknesses of the tested popular NNBA algorithms originates from decrease of sensitivity function magnitudes ($\left|\frac{\partial E}{\partial w^h}\right|$) through hidden layers according to remarks of Theorem 1. This effect causes severe attenuation of backward error signal propagation in the deep neural networks for backpropagation algorithms. Possible solutions to this problem may be reducing of gradient dependence of optimization methods (e.g., use of gradient-free optimization methods (Sexton and Gupta, 2000; Che et al., 2011; Gudise and Venayagamoorthy, 2003; Ince et al., 2010; Mosavi et al., 2016)), implementation of gradient magnitude balancing mechanisms to regulate backward error propagation (e.g., adjustment of additional gain coefficients (Roodschild et al., 2020) to balance gradient magnitudes through hidden layers), improving activation function (Kim, 2017; Oostwal et al., 2019) to enhance gradient magnitude of the activation function ($\left|\frac{\partial \varphi^h}{\partial v^h}\right|$).

*An essential shortcoming of the tested popular MSA algorithms is related to a high increase of computation burden as the dimension of optimization problems (the number of parameters to optimize) increases. Dimension of optimization problem for the training of neural networks grows very fast as the layer number increases. Population sizes and iteration numbers of MSAs should be increased to deal with the high dimension issue, however this can severely increase the computation time and cause the tested MSAs to be impractical for the deep neural network training task. Possible solutions of this problem may be enhancement of search strategies by specializing them to perform more effective searching on neural network function types. Such metaheuristics are referred to as deep learning dedicated MSAs.

The current study introduced a hard-to-train benchmark network that can be utilized as a test bench for development of training algorithms. Future works can address designing and testing of the deep learning dedicated optimization algorithms by using a pipe-like neural network test benches.

Conflict of Interest

No conflict of interest was declared by the authors.

References

Aliev, R.A., Fazlollahi, B., Guirimov, B.G., Aliev, R.R., 2008. Recurrent Fuzzy Neural Networks and Their Performance Analysis. in: Recurr. Neural Networks, InTech. https://doi.org/10.5772/5540.

Arifovic, J., Gençay, R., 2001. Using genetic algorithms to select architecture of a feedforward artificial neural network. Phys. A Stat. Mech. Its Appl., 289:574–594. https://doi.org/10.1016/S0378-4371(00)00479-9.

Awolusi, T.F., Oke, O.L., Akinkurolere, O.O., Sojobi, A.O., Aluko, O.G., 2019. Performance comparison of neural network training algorithms in the modeling properties of steel fiber reinforced concrete. Heliyon 5:e01115. https://doi.org/10.1016/j.heliyon.2018.e01115.

Bahrami, M., Akbari, M., Bagherzadeh, S.A., Karimipour, A., Afrand, M., Goodarzi, M., 2019. Develop 24 dissimilar ANNs by suitable architectures & training algorithms via sensitivity analysis to better statistical presentation: Measure MSEs between targets & ANN for Fe–CuO/Eg–Water nanofluid. Phys. A Stat. Mech. Its Appl. 519:159–168. https://doi.org/10.1016/j.physa.2018.12.031.

Bala, J.W., Analytics, D., Bloedorn, E., Bratko, I., 1992. The MONK's Problems A Performance Comparison of Different Learning Algorithms. http://robots.stanford.edu/papers/thrun.MONK.html Accessed 05 August 2021.

Battiti, R., 1992. First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method. Neural Comput., 4:141–166. https://doi.org/10.1162/neco.1992.4.2.141.

Beale, E.M.L., 1972. A derivation of conjugate gradients. in F.A. Lootsma, Ed., Numerical methods for nonlinear optimization, Academic Press, London, 39-43.

Birattari, M., Kacprzyk, J., 2009. Tuning metaheuristics: a machine learning perspective, Springer, Berlin.

Can, A., Dagdelenler, G., Ercanoglu, M., Sonmez, H., 2019. Landslide susceptibility mapping at Ovacık-Karabük (Turkey) using different artificial neural network models: comparison of training algorithms. Bull. Eng. Geol. Environ., 78:89–102. https://doi.org/10.1007/s10064-017-1034-3.

Caruana, R., Niculescu-Mizil, A., 2006. An empirical comparison of supervised learning algorithms. ACM Int. Conf. Proceeding Ser., 148:161–168. https://doi.org/10.1145/1143844.1143865.

Che, Z.G., Chiang, T.A., Che, Z.H., 2011. Feed-forward neural networks training: a comparison between genetic algorithm and back-propagation learning algorithm. International Journal of Innovative Computing Information and Control , 7(10), 5839-5850.

Chen, Z., Ashkezari, A.Z., Tlili, I., 2020. Applying artificial neural network and curve fitting method to predict the viscosity of SAE50/MWCNTs-TiO2 hybrid nanolubricant. Phys. A Stat. Mech. Its Appl., 549:123946. https://doi.org/10.1016/j.physa.2019.123946.

Chopard, B., Tomassini, M., 2018. Performance and limitations of metaheuristics. in: Nat. Comput. Ser., Springer Verlag, 191–203. https://doi.org/10.1007/978-3-319-93073-2_11.

Coleman, C., Narayanan, D., Kang, D., Zhao, T., Zhang, J., Nardi, L., Bailis, P., Olukotun, K., Zaharia, C.M., 2017. DawnBench: An end-to-end deep learning benchmark and competition, Training. In NIPS ML Systems Workshop.

Cömert, Z., Kocamaz, A., 2017. A Study of Artificial Neural Network Training Algorithms for Classification of Cardiotocography Signals. Bitlis Eren Univ. J. Sci. Technol., 7 , 93–103. https://doi.org/10.17678/beuscitech.338085.

Csaji, B.C., 2001. Approximation with Artificial Neural networks, Faculty of Science; Eötvö Lorand University, Hungary.

Dennis, J.E., Schnabel, R.B., 1996. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Society for Industrial and Applied Mathematics, SIAM. https://doi.org/10.1137/1.9781611971200.

Deng, L., Yu, D., 2013. Deep learning: Methods and applications, Found. Trends Signal Process. 7:197–387. https://doi.org/10.1561/2000000039.

Ding, S., Li, H., Su, C., et al., 2013. Evolutionary artificial neural networks: a review. Artificial Intelligence Review, 39:251–260. https://doi.org/10.1007/s10462-011-9270-6.

Faris, H., Aljarah, I., Al-Betar, M.A., Mirjalili, S., 2018. Grey wolf optimizer: a review of recent variants and applications. Neural Comput. Appl., 30:413–435. https://doi.org/10.1007/s00521-017-3272-5.

Fletcher, R., 1964. Function minimization by conjugate gradients. Comput. J., 7:149–154. https://doi.org/10.1093/comjnl/7.2.149.

Floreano, D., Dürr, P., Mattiussi, C., 2008. Neuroevolution: from architectures to learning. Evolutionary intelligence, 1(1), 47-62.

Fong, S., Deb, S., Yang, X.S., 2018. How meta-heuristic algorithms contribute to deep learning in the hype of big data analytics. Adv. Intell. Syst. Comput., 518:3–25. https://doi.org/10.1007/978-981-10-3373-5_1.

Galván, E., Mooney, P., 2021. Neuroevolution in deep neural networks: Current trends and future challenges. IEEE Transactions on Artificial Intelligence, 2: 476-493. https://doi.org/10.1109/TAI.2021.3067574.

Ghasemiyeh, R., Moghdani, R.., Sana, S.S., 2017. A Hybrid Artificial Neural Network with Metaheuristic Algorithms for Predicting Stock Price. Cybern. Syst., 48:365–392. https://doi.org/10.1080/01969722.2017.1285162.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, PMLR 9:249-256.
Gogna, A.., Tayal, A., 2013. Metaheuristics: Review and application. J. Exp. Theor. Artif. Intell., 25:503–526. https://doi.org/10.1080/0952813X.2013.782347.
Goh, C.H., Tung, Y.C.A., Cheng, C.H., 1996. A revised weighted sum decision model for robot selection. Comput. Ind. Eng., 30:193–199. https://doi.org/10.1016/0360-8352(95)00167-0.
Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning, MIT Press.
Gudise, V.G., Venayagamoorthy, G.K., 2003. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. in: 2003 IEEE Swarm Intell. Symp. SIS 2003 - Proc., Institute of Electrical and Electronics Engineers Inc., 2003:110–117. https://doi.org/10.1109/SIS.2003.1202255.
Gunantara, N., Nurweda, Putra I.D.N., 2019. The Characteristics of Metaheuristic Method in Selection of Path Pairs on Multicriteria Ad Hoc Networks. J. Comput. Networks Commun., 2019:7983583. https://doi.org/10.1155/2019/7983583.
Hagan, M.T., Demuth, H.B., Beale, M.H., 1996. Neural Network Design, Boston, MA: PWS Publishing.
Hagan, M.T., Menhaj, M.B., 1994. Training Feedforward Networks with the Marquardt Algorithm. IEEE Trans. Neural Networks., 5:989–993. https://doi.org/10.1109/72.329697.
Hinton, G.E., Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. Science 313:504–507. https://doi.org/10.1126/science.1127647.
Hornik, K., 1991. Approximation capabilities of multilayer feedforward networks. Neural Networks, 4:251–257. https://doi.org/10.1016/0893-6080(91)90009-T.
Igel, C., 2014. No free lunch theorems: Limitations and perspectives of metaheuristics. In Theory and principled methods for the design of metaheuristics. Springer, Berlin.
Ince, T., Kiranyaz, S., Pulkkinen, J., Gabbouj, M., 2010. Evaluation of global and local training techniques over feed-forward neural network architecture spaces for computer-aided medical diagnosis. Expert Syst. Appl., 37:8450–8461. https://doi.org/10.1016/j.eswa.2010.05.033.
Isik, P.X., Sadowski, P., 2016. A theory of local learning, the learning channel. and the optimality of backpropagation. Neural Netw., 83:51-74. https://doi.org/10.1016/j.neunet.2016.07.006
Karim, H., Niakan, S.R., Safdari, R., 2018. Comparison of neural network training algorithms for classification of heart diseases. IAES Int. J. Artif. Intell. , 7:185–189. https://doi.org/10.11591/ijai.v7.i4.pp185-189.
Kim, P., 2017. Matlab Deep Learning With Machine Learning. Neural Networks and Artificial Intelligence, Apress.
Kratsios, A., Bilkoptov, E., 2020. Non-Euclidean Universal Approximation. arXiv preprint arXiv:2006.02341.
Manoharan, S., Sathesh, A., 2020. Population Based Meta Heuristics Algorithm for Performance Improvement of Feed Forward Neural Network. Journal of Soft Computing Paradigm, 2(1), 36-46. https://doi.org/10.36548/jscp.2020.1.004.
Marquardt, D.W., 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. J. Soc. Ind. Appl. Math. 11:431–441. https://doi.org/10.1137/0111030.
Martens, J. , 2010. Deep learning via hessian-free optimization. in ICML, 27:735-742.
Mhaskar, H., Liao, Q., Poggio, T., 2016. Learning Functions: When Is Deep Better Than Shallow. arXiv preprint arXiv:1603.00988.
Mirjalili, S., Mirjalili, S.M., 2014. A. Lewis, Grey Wolf Optimizer. Adv. Eng. Softw., 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007.
Michalewicz, Z., 1992. Genetic algorithm + data structures = evolutionary programs. Springer-Verlag, New York. Melanie, M., 1996. An Introduction to Genetic Algorithms. MIT Press, Cambridge.
Moller, M.F., 1993. A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks , 1993:6525–533. https://doi.org/10.1016/S0893-6080(05)80056-5.
Mosavi, M.R., Khishe, M., Ghamgosar, A., 2016. Classification Of Sonar Data Set Using Neural Network Trained By Gray Wolf Optimization. Neural Netw. World. 26:393-415 https://doi.org/10.14311/nnw.2016.26.023.
Oostwal, E., Straat, M., Biehl, M., 2019. Hidden Unit Specialization in Layered Neural Networks: ReLU vs. Sigmoidal Activation. Phys. A Stat. Mech. Its Appl, 564:125517. https://doi.org/10.1016/j.physa.2020.125517.
Pan, X., Lee, B., Zhang, C., 2013. A comparison of neural network backpropagation algorithms for electricity load forecasting. In 2013 IEEE International Workshop on Inteligent Energy Systems (IWIES), 22-27.
Parejo, J.A., Ruiz-Cortés, A., Lozano, S., Fernandez, P., 2012. Metaheuristic optimization frameworks: A survey and benchmarking. Soft Comput., 16:527–561. https://doi.org/10.1007/s00500-011-0754-8
Powell, M.J.D., 1977. Restart procedures for the conjugate gradient method. Math. Program, 12:241–254. https://doi.org/10.1007/BF01593790.
Riedmiller, M., Braun, H., 1993. Direct adaptive method for faster backpropagation learning: The RPROP algorithm. in: 1993 IEEE Int. Conf. Neural Networks, Publ by IEEE, 586–591. https://doi.org/10.1109/icnn.1993.298623.
Roodschild, M., Sardiñas, J. G., Will, A., 2020. A new approach for the vanishing gradient problem on sigmoid activation. Progress in Artificial Intelligence, 9(4), 351-360.
Rusiecki, A., 2012. Robust learning algorithm based on iterative least median of squares. Neural Process. Lett., 36:145–160. https://doi.org/10.1007/s11063-012-9227-z.
Scales, L.E., 1985. Introduction to Non-Linear Optimization. Springer-Verlag, New York.
Sexton, R.S., Gupta, J.N.D., 2000. Comparative evaluation of genetic algorithm and backpropagation for training neural networks. Inf. Sci. , 129:45–59. https://doi.org/10.1016/S0020-0255(00)00068-2.
Sewak, M., Sahay, S.K., Rathore, H., 2018. Comparison of deep learning and the classical machine learning algorithm for the malware detection, in: Proc. - 2018 IEEE/ACIS 19th Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed

Comput. SNPD 2018, Institute of Electrical and Electronics Engineers Inc., pp.293–296. https://doi.org/10.1109/SNPD.2018.8441123.
Schmidhuber, J., 2015. Deep Learning in neural networks: An overview. Neural Networks 61:85–117. https://doi.org/10.1016/j.neunet.2014.09.003.
Shrestha, A., Mahmood, A., 2019. Review of deep learning algorithms and architectures. IEEE Access 7:53040-53065. https://doi.org/10.1109/ACCESS.2019.2912200
Stanley, K.O., Clune, J., Lehman, J., Miikkulainen, R., 2019. Designing neural networks through neuroevolution. Nature Machine Intelligence, 1(1), 24-35.
Stanujkic, D., Zavadskas, E.K., 2015. A modified Weighted Sum method based on the decision-maker's preferred levels of performances. Stud. Informatics Control. 24:461-469. https://doi.org/10.24846/v24i4y201510.
Strang, G., 2018. The functions of deep learning. SIAM news., 51:1-4.
Suganuma, M., Shirakawa, S., Nagao, T., 2017. A genetic programming approach to designing convolutional neural network architectures. In: Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '17. ACM Press, New York, New York, USA, 497–504.
Sussillo, D., Abbott, L.F., 2014. Random Walk Initialization for Training Very Deep Feedforward Networks. arXiv preprint arXiv:1412.6558. http://arxiv.org/abs/1412.6558 (accessed June 11, 2021).
Tagluk, M.E., Isik, I., 2019. Communication in nano devices: Electronic based biophysical model of a neuron. Nano Commun. Netw. , 19:134–147. https://doi.org/10.1016/j.nancom.2019.01.006.
Thakkar, A., Mungra, D., Agrawal, A., 2020. Sentiment analysis: An empirical comparison between various training algorithms for artificial neural network, Int. J. Innov. Comput. Appl., 11:9–29. https://doi.org/10.1504/IJICA.2020.105315.
Winkler, D.A., Le, T.C., 2017. Performance of Deep and Shallow Neural Networks, the Universal Approximation Theorem. Activity Cliffs, and QSAR, Mol. Inform,36. https://doi.org/10.1002/minf.201600118.
Wong, W.K., Ming, C.I., 2019. A Review on Metaheuristic Algorithms: Recent Trends, Benchmarking and Applications, in: 2019 7th Int. Conf. Smart Comput. Commun. ICSCC 2019, Institute of Electrical and Electronics Engineers Inc., 1-5. https://doi.org/10.1109/ICSCC.2019.8843624.
Vogl, T.P., Mangis, J.K., Rigler, A.K., Zink, W.T., Alkon, D.L., 1988. Accelerating the convergence of the back-propagation method. Biol. Cybern., 59:257–263. https://doi.org/10.1007/BF00332914.
Zamri, N.B.A., Bhuvaneswari, T., Aziz, N.A.B.A., Aziz, N.H.B.A., 2018. Feature selection using simulated Kalman filter (SKF) for prediction of body fat percentage. In Proceedings of the 2018 International Conference on Mathematics and Statistics, 23–27. https://doi.org/10.1145/3274250.3274264.
Zeugmann, T., Poupart, P., Kennedy, J., Jin, X., Han, J., Saitta, L., Sebag, M., Peters, J., Bagnell, J.A., Daelemans, W., Webb, G.I., Ting, K.M., Ting, K.M., Webb, G.I., Shirabad, J.S., Fürnkranz, J., Hüllermeier, E., Matwin, S., Sakakibara, Y., Flener, P., Schmid, U., Procopiuc, C.M., Lachiche, N., Fürnkranz, J., 2011. Particle Swarm Optimization. in: Encycl. Mach. Learn., Springer US, Boston, MA, 760–766. https://doi.org/10.1007/978-0-387-30164-8_630.
Zhao, X., Xia, L., Zhang, J., Song, W., 2020. Artificial neural network based modeling on unidirectional and bidirectional pedestrian flow at straight corridors. Phys. A Stat. Mech. Its Appl, 547:123825. https://doi.org/10.1016/j.physa.2019.123825.
Zhao, Z., Xin, H., Ren, Y., Guo, X., 2010. Application and comparison of BP neural network algorithm in MATLAB, in: 2010 Int. Conf. Meas. Technol. Mechatronics Autom. ICMTMA, 2010: 590–593. https://doi.org/10.1109/ICMTMA.2010.492.
Zhu, H., Akrout, M., Zheng, B., Pelegris, A., Jayarajan, A., Phanishayee, A., Schroeder, B., Pekhimenko, G., 2018. Benchmarking and Analyzing Deep Neural Network Training, in: 2018 IEEE Int. Symp. Workload Charact. IISWC 2018, Institute of Electrical and Electronics Engineers Inc., 2018:88–100. https://doi.org/10.1109/IISWC.2018.8573476.