The following publication Khan, W. A., Chung, S. H., Awan, M. U., & Wen, X. (2020b). Machine learning facilitated business intelligence (Part II): Neural networks optimization techniques and applications. Industrial Management and Data Systems, 120(1), 128–163 is published by Emerald and is available at https://doi.org/10.1108/IMDS-06-2019-0351.

Machine learning facilitated business intelligence (Part II): Neural networks optimization techniques and applications

Abstract

Purpose- The purpose of this paper include: (i) to review the categories explaining mainly optimization algorithms (techniques) in that needed to improve the generalization performance and learning speed of Feedforward Neural Network (FNN); (ii) discover the change in research trend by analysing all six categories collectively; (iii) recommend new research directions for researchers and facilitate users to understand algorithms real-world applications in solving complex management, engineering and health sciences problems.

Design/methodology/approach- FNN has gained much attention from the researchers to make a more informed decision during the last few decades. The literature survey is focused on the learning algorithms and optimization techniques proposed during the last three decades. Part II is an extension of Part I. To make the study consistent with Part I, the approach and survey methodology in this paper are kept identical to Part I.

Findings- Combining the work performed in Part I, the authors studied in a total of 80 articles through popular keywords searching. The FNN learning algorithms and optimization techniques identified in the selected literature are classified into six categories based on their problem identification, mathematical model, technical reasoning, and proposed solution. Previously, in Part I, the two categories focusing on learning algorithms (i.e., Gradient learning algorithms for Network Training, Gradient free learning algorithms) are reviewed with their real-world applications in management, engineering, and health sciences. Therefore, in the current paper, Part II, the remaining four categories exploring optimization techniques (i.e., Optimization algorithms for learning rate, Bias and Variance (Underfitting and Overfitting) minimization algorithms, Constructive topology Neural Networks, Metaheuristic search algorithms) are studied. The algorithms explanation is made enriched by discussing their technical merits, limitations, and applications in their respective categories. Finally, the authors recommended future new research directions contributing to strengthening the literature.

Research limitations/implications- The FNN contribution is rapidly increasing because of its ability to make reliably informed decisions. Like learning algorithms reviewed in Part I, the authors focus is to enrich the comprehensive study by reviewing remaining categories focusing on optimization techniques. However, future efforts may be needed to incorporate other algorithms into identified six categories or suggest a new category to continuously monitor the shift in the research trend.

Practical implications- The authors studied the shift in research trend for three decades by collectively analysing the learning algorithms and optimization techniques with their applications. This may help researchers to identify future research gaps to improve the generalization performance and learning speed, and user to understand the applications areas of FNN. For instance, research contribution in FNN during the last three decades has changed from complex gradient-based algorithms to gradient free algorithms, trial and error hidden units fixed topology approach to cascade topology, hyperparameters initial guess to analytically calculation, and converging algorithms at a global minimum rather than the local minimum.

Originality/value- The existing literature surveys include comparative study, identifying application areas, and focusing on a specific technique in that it may not be able to identify algorithms categories, a shift in research trend over time, application area frequently analysed, common research gaps and collective future directions. Part I and II attempts to overcome the existing literature surveys limitation by classifying studied articles into six categories covering a wide range of algorithm proposed to improve the FNN generalization performance and convergence rate. The classification of algorithms into six categories helps to analyse the shift in research trend which makes the classification scheme more significant and innovative.

Keywords Feedforward neural network, Machine learning, Optimization techniques, Industrial management, Data analytics

1. Introduction

Feedforward Neural Network (FNN) has gained much attention of researchers in the last few decades (Abdel-Hamid et al., 2014; Babaee et al., 2018; Chen et al., 2018; Chung et al., 2017; Deng et al., 2019; Dong et al., 2016; Ijjina and Chalavadi, 2016; Kastrati et al., 2019; Kummong and Supratid, 2016; Mohamed Shakeel et al., 2019; Nasir et al., 2019; Teo et al., 2015; Yin and Liu, 2018; Zaghloul et al., 2009) because of its ability to extract useful pattern and make a more informed decision from high dimensional data (Kumar et al., 1995; Tkáč and Verner, 2016; Tu, 1996). With modern information technology advancement, the challenging issue of high dimensional, non-linear, noisy and unbalanced data is continuously growing and varying at a rapid rate in that it demands efficient learning algorithms and optimization techniques. The data may become a costly resource if not analyzed properly in the process of business intelligence. Machine learning is gaining significant interest in facilitating business intelligence in the process of data gathering, analyses and extracting knowledge to help users in making better informed decisions (Bottani et al., 2019; Hayashi et al., 2010; Kim et al., 2019; Lam et al., 2014; Li et al., 2018; Mori et al., 2012; Wang et al., 2005; Wong et al., 2018). Efforts are being made to overcome the challenges by building the optimal machine learning FNN that may extract a useful pattern from the data and generate information in real-time for making better-informed decisions. Extensive knowledge and theoretical information are required to build the FNN having characteristics of better generalization performance and learning speed. The generalization performance and learning speed are the two criteria that play an essential role in deciding the use of learning algorithms and optimization techniques to build optimal FNN. Depending upon applications and data structure, the user might prefer either better generalization performance or faster learning speed or combination of both. Some of the drawbacks that may affect the generalization performance and learning speed of FNN includes local minimum, saddle points, plateau surfaces, hyperparameters adjustment, trials and errors experimental work, tuning connection weights, deciding hidden units and layers, and many others. The drawbacks, that limits FNNs applicability, may become worse with inappropriate user expertise and insufficient theoretical information. Several questions were identified in Part I of the study that may become causes of above drawbacks. For instance, how to define network size, hidden units, hidden layers, connection weights, learning rate, topology, and many others. For the sake of simplicity, the paper entitled "Machine learning facilitated business intelligence: Neural networks learning algorithms and applications" is referred to as Part I.

Previously, in Part I of the study, the authors made an effort to answers the key questions by reviewing two categories mainly explaining the learning algorithms. The categories were named as gradient learning algorithms for network training and gradient free learning algorithms. In the current paper, Part II, the authors made an effort to review the remaining four categories mainly explaining optimization techniques (i.e., Optimization algorithms for learning rate, Bias and Variance (Underfitting and Overfitting) minimization algorithms, Constructive topology Neural Networks, Metaheuristic search algorithms). Part II is an extension of Part I. For each category, researchers' efforts to demonstrate the effectiveness of their proposed optimization techniques in solving real-world management, engineering, and health sciences problems are also explained to enrich the content and make users familiar with FNN applications areas. Moreover, all categories are collectively analysed in the current paper to discover the shift in research trend. Based on the review of existing literature, the authors recommended future research directions to contribute in strengthening the literature. In-depth knowledge from the survey will help researchers to design a new, simple and compact algorithm having characteristics of better generalization performance and generating results in the shortest possible time. Similarly, the users may able to decide and select the algorithm that best suits their application area.

The paper is organized as follow: Section 2 is about survey methodology. Section 3 briefly overviews Part I of the study. In Section 4, four categories that focus on optimization techniques are reviewed with a detail description of each algorithm in terms of its merits, limitation, and real-world management, engineering, and health sciences applications. Section 5 is about future directions to improve FNN generalization performance and learning speed. Section 6 concludes the paper.

2. Survey methodology

2.1Source of Literature and philosophy of review work

The source of literature and philosophy of review work is identical to the Part I. Combining the review articles of Part I, the authors studied in total 80 articles, in which 63 (78.75%) were journal papers, 10 (12.50%) conference papers, 3 (3.75%) online arXiv archives, 2 (2.50%) books, 1 (1.25%) technical report and 1 (1.25%) online academic lecture. Previously, in Part I, only 38 articles were reviewed explaining mainly learning algorithms categories. In the current paper, remaining articles are reviewed in section 4 which explains optimization techniques needed to improve the generalization performance and learning speed of FNN. In this section, all 80 articles (Part I and Part II) are collectively analysed to discover the shift in research trend.

2.2Classification schemes

This paper classification is an extension of Part I and focuses on optimization techniques recommended in the last three decades to improve the generalization performance and learning speed of FNN. In this subsection, all categories are collectively analysed to discover the shift in research trend. Combining the categories explained in Part I, in total six categories are:

1. Gradient learning algorithms for Network Training

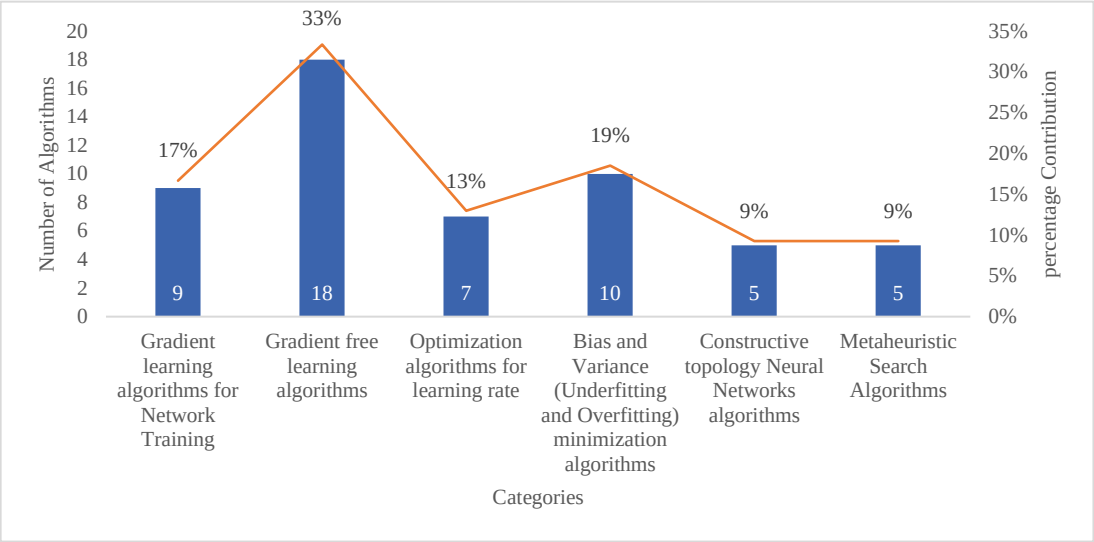

Figure 1(a). Algorithms Distribution Categories Wise

2. Gradient free learning algorithms
3. Optimization algorithms for learning rate
4. Bias and Variance (Underfitting and Overfitting) minimization algorithms
5. Constructive topology Feedforward Neural Network
6. Metaheuristic Search Algorithms

The category one and two are extracted from Part I, whereas, category three to six are reviewed in the current paper. In Part I, the first category explained gradient learning algorithms that need first order or second order gradient information to build FNN and the second category contained gradient free learning algorithms which analytically determine connection weights rather than first or second order gradient tuning. Categories three to six are reviewed in the current paper and are highlighted as: the third category contains optimization algorithms for varying learning rate at different iteration to improve generalization performance by avoiding divergence from minimum of function, fourth category contains algorithms to avoid underfitting and overfitting to improve

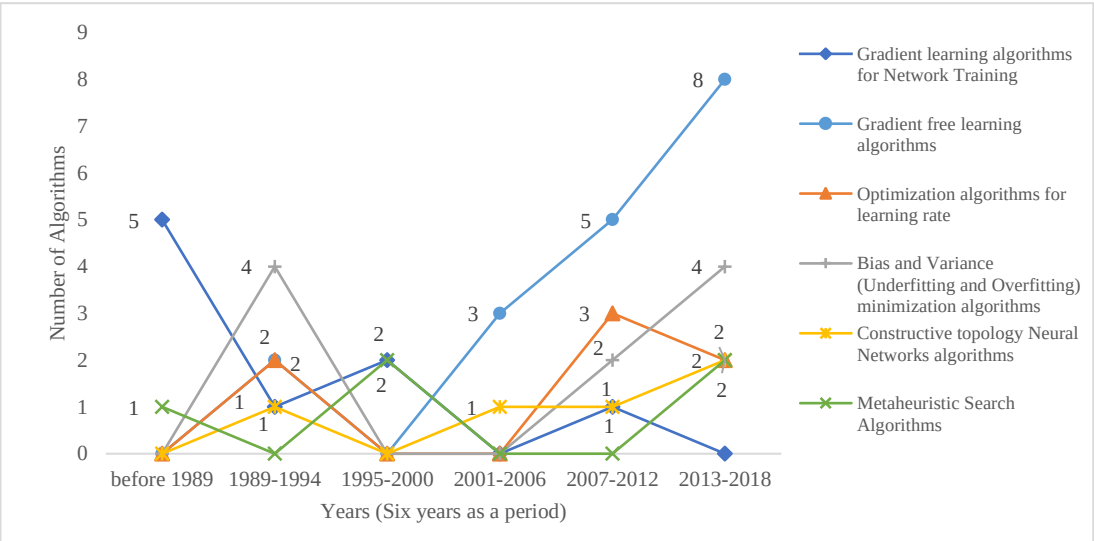

Figure 1(b). Algorithms Proposed over time

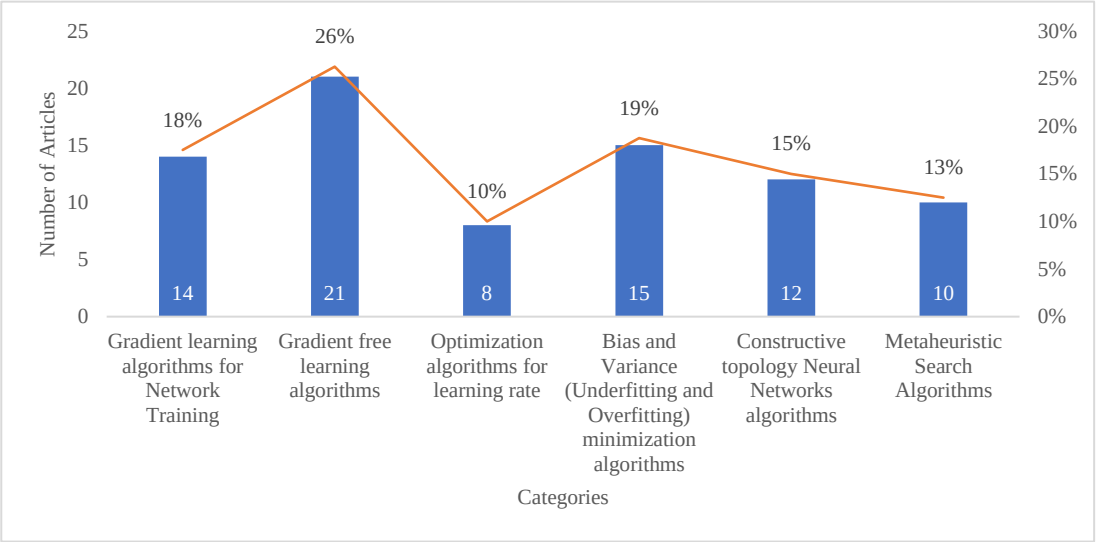

Figure 1(c). Papers Distribution Categories Wise

generalization performance with additional need for training time, fifth category contains constructive topology learning algorithms to avoid the need for trial and error approaches for determining the number of hidden units in fixed-topology networks, and the sixth category contains metaheuristics global optimization algorithms to search for loss functions in global search space instead of local space.

Figure 1(a) illustrates the classification of algorithms into six categories. The authors identified in a total of 54 unique algorithms proposed in 80 articles. Among 54 unique algorithms, 27 learning algorithms were reviewed in Part I and the remaining 27 optimization algorithms are reviewed in Section 4 of the current paper. Figure 1(b) illustrates the number of algorithms identified in each category over time. Other than proposed algorithms, a small number of articles that support or criticize identified algorithms were also included to widen review. The unique algorithms, supportive and criticized articles result in a total of 80 articles. Figure 1(c) illustrates the total number of articles reviewed in each category. Table I provides a detailed summary of the algorithms identified in each category along with references to the total number of papers reviewed. In the table, the references of the article reviewed in each specific category are given, however, the article referenced in more than one category are identified with an asterisk (*). For instance, article (Rumelhart et al., 1986), (Setiono and Hui, 1995), (Hinton et al., 2012), (Zeiler, 2012) and (Hunter et al., 2012) appears in the first, third and fifth category. The major contribution of (Rumelhart et al., 1986) and (Setiono and Hui, 1995) is in the first category, (Hinton et al., 2012) and (Zeiler, 2012) is in the third category, and (Hunter et al., 2012) is in the fifth category. To avoid repetition, Figure 1(c) is plotted showing the major contribution of articles in the respective category. This is a reason that the distribution of gradient learning algorithm category in Part II conflicts with Part I. Initially, in Part I, the authors mentioned 17 articles reviewed for gradient learning algorithm category (Figure 3(c) of Part I). However, in Part II, 14 articles are reported. Reference (Hinton et al., 2012), (Zeiler, 2012) and (Hunter et al., 2012) are cited in gradient learning algorithm category to support relevant literature but their original work relates to optimization algorithms. This implies that (Hinton et al., 2012) and (Zeiler, 2012) major contribution is in the third category, and (Hunter et al., 2012) major contribution is in the fifth category. To avoid the repetition, the

No.	Category	Algorithms Published	References
1	Gradient learning algorithms for Network Training	Gradient descent, stochastic gradient descent, mini-batch gradient descent, Newton method, Quasi-Newton method, conjugate gradient method, Quickprop, Levenberg-Marquardt Algorithm, Neuron by Neuron	(Hecht-Nielsen, 1989), (Bianchini and Scarselli, 2014), (LeCun et al., 2015), (Wilamowski and Yu, 2010), (Rumelhart et al., 1986)*, (Wilson and Martinez, 2003), (Wang et al., 2017), (Hinton et al., 2012)*, (Ypma, 1995), (Zeiler, 2012)*, (Shanno, 1970), (Lewis and Overton, 2013), (Setiono and Hui, 1995)*, (Fahlman, 1988), (Hagan and Menhaj, 1994), (Wilamowski et al., 2008), (Hunter et al., 2012)*
2	Gradient free learning algorithms	Probabilistic Neural Network, General Regression Neural Network, Extreme learning machine (ELM), Online Sequential ELM, Incremental ELM (I-ELM), Convex I-ELM, Enhanced I-ELM, Error Minimized ELM (EM-ELM), Bidirectional ELM, Orthogonal I-ELM (OI-ELM), Driving Amount OI-ELM, Self-adaptive ELM, Incremental Particle Swarm Optimization EM-ELM, Weighted ELM, Multilayer ELM, Hierarchical ELM, No propagation, Iterative Feedforward Neural Networks with Random Weights	(Huang et al., 2015), (Ferrari and Stengel, 2005), (Specht, 1990), (Specht, 1991), (Huang, Zhu, et al., 2006), (Huang, Zhou, et al., 2012), (Liang et al., 2006), (Huang, Chen, et al., 2006), (Huang and Chen, 2007), (Huang and Chen, 2008), (Feng et al., 2009), (Yang et al., 2012), (Ying, 2016), (Zou et al., 2018), (Wang et al., 2016), (Han et al., 2017), (Zong et al., 2013), (Kasun et al., 2013), (Tang et al., 2016), (Widrow et al., 2013), (Cao et al., 2016)
3	Optimization algorithms for learning rate	Momentum, Resilient Propagation, RMSprop, Adaptive Gradient Algorithm, AdaDelta, Adaptive Moment Estimation, AdaMax	(Gori and Tesi, 1992), (Lucas and Saccucci, 1990), (Rumelhart et al., 1986)*, (Qian, 1999), (Riedmiller and Braun, 1993), (Hinton et al., 2012)*, (Duchi et al., 2011), (Zeiler, 2012)*, (Kingma and Ba, 2014)
4	Bias and Variance (Underfitting and Overfitting) minimization algorithms	Validation, n-fold cross-validation, weight decay ($l^1$and l^2 regularization), Dropout, DropConnect, Shakeout, Batch normalization, Optimized approximation algorithm (Signal to Noise Ratio Figure), Pruning Sensitivity Methods, Ensembles Methods	(Geman et al., 1992), (Zaghloul et al., 2009), (Reed, 1993), (Seni and Elder, 2010), (Setiono and Hui, 1995)*, (Krogh and Hertz, 1992), (Srivastava et al., 2014), (Wan et al., 2013), (Kang et al., 2017), (Ioffe and Szegedy, 2015), (Yinyin Liu et al., 2008), (Karnin, 1990), (Kovalishyn et al., 1998)*, (Han et al., 2015), (Hansen and Salamon, 1990), (Krogh et al., 1995), (Islam et al., 2003)
5	Constructive topology Neural Networks	Cascade Correlation Learning, Evolving cascade Neural Network, Orthogonal Least Squares-based Cascade Network, Faster Cascade Neural Network, Cascade Correntropy Network	(Hunter et al., 2012)*, (Kwok and Yeung, 1997), (Fahlman and Lebiere, 1990), (Lang, 1989), (Hwang et al., 1996), (Lehtokangas, 2000), (Kovalishyn et al., 1998)*, (Schetinin, 2003), (Farlow, 1981), (Huang, Song, et al., 2012), (Qiao et al., 2016), (Nayyeri et al., 2018)
6	Metaheuristic Search Algorithms	Genetic algorithm, Particle Swarm Optimization (PSO), Adaptive PSO, Whale optimization algorithm (WOA), Lévy flight WOA	(Mitchell, 1998), (Mohamad et al., 2017), (Liang and Dai, 1998), (Ding et al., 2011), (Eberhart and Kennedy, 1995), (Shi and Eberhart, 1998), (Zhang et al., 2007), (Shaghaghi et al., 2017), (Mirjalili and Lewis, 2016), (Ling et al., 2017)

Table I. Classification of FNN Published Algorithms

distribution (i.e. 17-14= 3) is reported in the third and fifth category reflecting original work. The distribution in Figure 1(a-c) and contents in Table I explain the researchers' interest and trend in a specific category. The distribution and contents explain that research directions are changing from complex gradient-based algorithms to gradient free algorithms, fixed topology to constructive topology, hyperparameters initial guess to analytically calculation, and converging algorithms at a global minimum rather than the local minimum. In another way, the categories may be considered as an opportunity for discovering research gap and further improvement can bring a significant contribution.

3. Part I: An overview

The two categories reviewed in Part I are briefly explained in this section. The section is divided into two subsections. The first subsection gives a concise overview of the FNNs working method and second subsection summarize the two categories in that are explained detailed in Part I.

3.1FNN: An overview

FNN process information in the parallel structure by connecting the input layer to the output layer in a forward direction without forming a cycle or loop. The input layer and output layer are connected through a hidden unit by a series of connection weights (Hecht-Nielsen, 1989). Figure 2 illustrates a simple FNN with three layers. The weight connections linking the input layer consisting of input features x with an added bias b^u to the hidden layers u, are known as input connection weights w^{icw}. Similar, the weight connections linking the u with an added bias b^o and the output layer, are known as output connection weight w^{ocw}. The output layer p is calculated in *forward propagation* step by applying activation function $f(z)$ on the product of w^{ocw} and u, and compared with target vector a to determine loss function E. This can be expressed mathematically as:

$$p = f^{ou}(w^{ocw} f^{hu}(w^{icw} x + b^u) + b^o) \tag{1}$$

Such that:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

The objective is to minimize the loss function E of the network by achieving p approximately equal to a:

$$E = \frac{1}{m} \sum_{h=1}^{m} (p_h - a_h)^2 \tag{3}$$

At each instance h the error e_h can be expressed as:

$$e_h = p_h - a_h \tag{4}$$

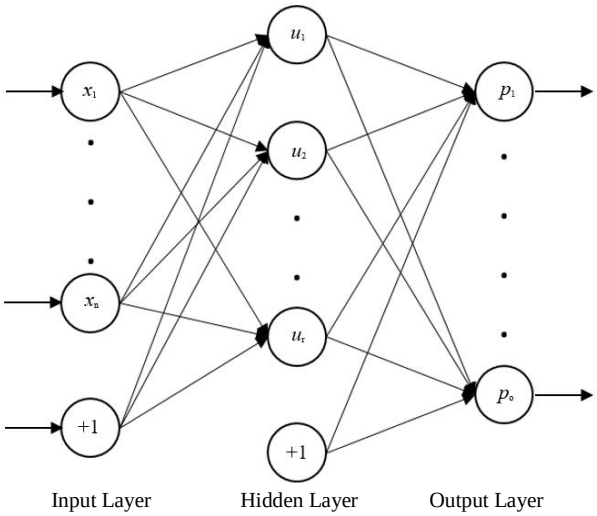

Figure 2. Feedforward Neural Networks with three layers (Input, hidden and Output)

If E is larger than predefined expected error ε, the connection weights are *backpropagated* by taking derivative ∇E of E with respect to each incoming weight in the direction of descending gradient following chain rule.

$$\nabla E = \frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial out_i}\frac{\partial out_i}{\partial net_i}\frac{\partial net_i}{\partial w_i} \tag{5}$$

This continues, and the connection weights are updated at each iteration i in gradient descent direction to bring p closer to a.

$$w_{i+1} = w_i - \propto \nabla E \tag{6}$$

where $\propto$ is a learning rate hyperparameter. A similar method is followed for second order $\nabla^2 E$ learning algorithms. For the sake of simplicity in this study, the connection weights w in the equations refer to all types of connection weights in the network, unless otherwise specified.

3.2Part I Classification

In Part I, the authors identified in a total of 27 unique algorithms proposed in the 38 articles. Other than proposed algorithms, a small number of articles that support or criticize identified algorithms were also incorporated to the broader review. The identified 27 algorithms were classified into two main categories based on their problem identification, mathematical model, technical reasoning, and proposed solution. The categories were named as 1) Gradient learning algorithms for network training, and 2) Gradient free learning algorithms.

Table I illustrates the classification of algorithms into their respective categories. The table summarises the algorithm in each category with paper citation reference. The gradient learning algorithm category includes first order and second order learning algorithms used to train the network following the backpropagation (BP) delta rule. The category was further divided into two subcategories. Subcategory one explained algorithms belonging to the first order derivative rule, whereas, subcategory two explained algorithms belonging to second order derivative rule. The gradient free learning category explained algorithms that do not need gradient information for iterative tuning of connection weights. The category was further classified into three subcategories. Subcategory one explained probability and general regression neural networks, subcategory two explained Extreme Learning Machine (ELM) and its variants algorithms in that they randomly generate hidden unit and analytically calculate output connection weight, and subcategory three explained algorithms adopting the hybrid approach of gradient and gradient-free methods.

4. Optimization Techniques

In continuity to Part I, this section explains the remaining four categories (i.e., Optimization algorithms for learning rate, Bias and Variance (Underfitting and Overfitting) minimization algorithms, Constructive topology Neural Networks, Metaheuristic search algorithms) explaining mainly optimization techniques as summarized in Table I. The optimization algorithms (techniques) are explained along with their merits and technical limitation to understand the FNN drawbacks, answers for questions involving user expertise and theoretical information, to identify research gaps and future directions. The list of applications mentioned in each category gives an intuition

of the successful implementation of optimization techniques in various real-world management, engineering, and health sciences problems. Below sections explain algorithms categories and their subcategories:

4.1Optimization algorithms for learning rate

BP gradient learning algorithms are powerful and have been extensively studied to improve its convergence. The important global hyperparameter that set the position of new weight for BP is the learning rate $\propto$. If $\propto$ is too large, the algorithms may diverge from a minimum of the loss function and will oscillate around it, however, if $\propto$ is too small it will converge very slowly. The suboptimal learning rate can cause FNN to trip in a local minimum or saddle point. The objective of FNN is to determine the global minimum of the loss function that truly represents the domain of function rather than a local minimum. Saddle point can be defined as a point where the function has both minimum and maximum direction (Gori and Tesi, 1992).

The algorithms in this category, that improves the convergence of BP by optimizing learning rate, are based on knowledge of exponential moving weighted average (EMA) statistical technique (Lucas and Saccucci, 1990) and its bias correction (Kingma and Ba, 2014). Before moving into the detail of each optimization algorithm it is important to understand the basic intuition behind EMA and the role of bias correction for better optimization. EMA is best suitable when the data is noisy, and it helps to make it denoise by taking a moving average of previous values to define the next sequence in the data. With an initial guess of $\Delta w_o = 0$, the next sequence can be computed from the expression:

$$\Delta w_i = \beta \Delta w_{i-1} + (1-\beta)\nabla E_i \tag{7}$$

Where β is moving exponential hyperparameter with value [0,1]. Hyperparameter β plays an important role in EMA. Firstly, it approximates the moving average Δw_i by $1/(1-\beta)$ which means higher the value of β, the more values will be averaged, and data trend will be less noisy, whereas, the lower value of β will average less values and trend will be more fluctuating. Secondly, for a higher value of β, more importance will be given to the previous weights as compared to the derivative values. Moreover, it is discovered that during the initial sequence, the trend is biased and is more away from the original function due to the initial guess $\Delta w_o = 0$. This results in much lower values which can be improved by computing bias such that:

$$\Delta w_i = \frac{\beta \Delta w_{i-1} + (1-\beta)\nabla E_i}{1-\beta^i} \tag{8}$$

The effect of $1-\beta^i$ in dominator decreases with increasing iteration i. Therefore, $1-\beta^i$ has more influence on the starting iteration and will generate a sequence with better results. The EMA and its bias correction techniques has been extensively utilized in FNN research to optimize learning and convergence towards error with a minimum number of iterations.

When the learning rate is small, the gradient slowly moves towards the minimum, whereas, a large learning rate will oscillate the gradient along the long and short axis of minimum error valley. This reduces gradient movement towards the long axis which faces towards the minimum. The *momentum* minimizes the short axis of oscillation

while add more contributions along the long axis to move gradient in larger steps towards minimum with less iteration (Rumelhart et al., 1986). Equation (7) can be modified for momentum:

$$\Delta w_i^a = \beta^a \Delta w_{i-1}^a + (1 - \beta^a)\nabla E_i \tag{9}$$

$$w_{i+1} = w_i - \propto \Delta w_i^a \tag{10}$$

Where β^a is momentum hyperparameter controlling the exponential decay. The new weight is a linear function of both the current gradient and weight change during the previous step (Qian, 1999). Riedmiller and Braun (1993) argue that in practical experience it is always not true that momentum will make learning more stable. The momentum parameter β^a is equally problematic as the learning parameter $\propto$ and that no general improvement can be achieved. Despite $\propto$, the second factor that effect the Δw_i is the unforeseeable behaviour of derivative ∇E_i which magnitude will be different for different weights. *Resilient Propagation (RProp) algorithm* was developed to avoid the problem of blurred adaptivity of ∇E_i and change the size of Δw_i directly without considering ∇E_i. For each weight w_{i+1}, the size of the update value Δ_i is computed based on the local gradient information:

$$\Delta_i = \begin{cases} \eta^+ * \Delta_{i-1}, & if\ \nabla E_{i-1} * \nabla E_i > 0 \\ \eta^- * \Delta_{i-1}, & if\ \nabla E_{i-1} * \nabla E_i < 0 \\ \Delta_{i-1}, & else \end{cases} \tag{11}$$

where η^+ and η^- are increasing and decreasing hyperparameters. The basic intuition is that when algorithm jumped over the minimum the Δ_i is decreased by η^- factor and if it retains sign it is accelerated by η^+ increasing factor. The weight update Δw_i is decreased if the derivative is positive and increased if the derivative is negative such that:

$$\Delta w_i = \begin{cases} -\Delta_i, & if\ \nabla E_i > 0 \\ +\Delta_i, & if\ \nabla E_i < 0 \\ 0, & else \end{cases} \tag{12}$$

$$w_{i+1} = w_i + \Delta w_i \tag{13}$$

RProp has an advantage over BP that it gives equal importance to all weights during learning. In BP the size of derivative decreases for weights that are a far distance from the output and they are less modified compared to other weights which make it slower. Whereas, RProp is dependent on the sign rather than derivative magnitude which give equal importance to far away weights. The simulation results demonstrate that RProp is four times and one time faster than popular gradient descent (GD) and quick prop (QP) respectively. RProp is effective for batch learning and it does not work with mini-batches. Hinton et al. (2012) addressed that RProp does not work with mini-batches because it treats all mini-batches equivalent. The possibility exists that weight will grow highly which may not notice by RProp. They proposed *RMSProp* by combining the robustness of RProp, the efficiency of mini-batches and gradient average over mini-batches. RMSprop is a minibatch version of RProp by keeping a moving average of square gradient such that:

$$\Delta w_i^b = \beta^b \Delta w_{i-1}^b + \left(1 - \beta^b\right)\nabla E_i{}^2 \tag{14}$$

$$w_{i+1} = w_i - \left(\frac{\propto}{\sqrt{\Delta w_i^b}} \right) \nabla E_i \tag{15}$$

Like β^a, β^b is hyperparameter that controls the exponential decay. When the gradient is in the short axis the moving average will be large which will slow down gradient, whereas, in the long axis, the moving average will be small which will accelerate gradient. It will reduce the depth of gradient oscillation and move it faster along the long axis. Another advantage of this algorithm is that the larger learning rate can be used to move it more quickly towards the minimum of the loss function. For optimal results, the author recommended using $\beta^b = 0.9$.

On the contrary, *Adaptive Gradient Algorithm (AdaGrad)* performs informative gradient-based learning by incorporating the knowledge of the geometry of the data observed during each iteration (Duchi et al., 2011). The feature that occurs infrequently is assigned a larger learning rate because they are more informative, while features that occur frequently are given a small learning rate. Standard BP follows predetermined procedural that weights are updated at once with a similar learning rate which makes convergence poor. AdaGrad compute matrix G as l^2 norm of all previous gradient:

$$\Delta w_i = \left(\frac{\propto}{\sqrt{G_i}} \right) \nabla E_i \tag{16}$$

$$w_{i+1} = w_i - \Delta w_i \tag{17}$$

This algorithm is more suitable for high spare dimensionality data. The above equations increase the learning rate for more spare data and decrease the learning rate for low spare data. The learning rate is scaled based on G and ∇E_i to give a parameter specific learning rate. The main drawback of AdaGrad is that it accumulates the squared gradient which grows after each iteration. This shrinks the learning rate and it becomes too infinitesimally small and limits the purpose of gaining additional information (Zeiler, 2012). Therefore, *AdaDelta* is proposed based on idea extracted from AdaGrad to improve two main issues in learning: 1) shrinkage of learning rate, and 2) manually selecting the learning rate (Zeiler, 2012). Accumulating sum of squared gradients shrink the learning rate in AdaGrad and this can be controlled by restricting the previous gradient to a fixed size s. This restriction will not accumulate gradient to infinity and more importance will go to the local gradient. The gradient can be restricted by accumulating an exponential decaying average of the square gradient. Assume at i the running average of the gradient is $E[\nabla E^2]_i$ then:

$$E[\nabla E^2]_i = \beta^c E[\nabla E^2]_{i-1} + (1 - \beta^c) \nabla E_i{}^2 \tag{18}$$

Like AdaGrad, taking the square root of the parameter $E[\nabla E^2]_i$:

$$RMS_i = \sqrt{E[\nabla E^2]_i + \epsilon} \tag{19}$$

Where constant ϵ is added to condition the denominator for approaching zero. The update parameter Δw_i can be expressed as:

$$\Delta w_i = \left(\frac{\propto}{RMS_i}\right)\nabla E_i \tag{20}$$

$$w_{i+1} = w_i + \Delta w_i \tag{21}$$

This method performance on FNN is better with no manual setting of the learning rate, insensitive to hyperparameter, and robust to large gradient and noise. However, it requires some extra computation per iteration over GD and requires expertise to select the best hyperparameters.

Similarly, the concept of both first and second moments of gradients was incorporated in *Adaptive Moment Estimation (Adam).* It requires first order gradient information to compute individual learning rate for parameters from an exponential moving average of first and second moments of gradients (Kingma and Ba, 2014). For the parameter update, it combines the idea of both the exponential moving average of gradients like momentum and an exponential moving average of square gradients like RMSProp and AdaGrad. The moving average of the gradient is an estimate of the 1st moment (mean) and the square gradient is an estimate of the 2nd moment (the uncentered variable) of the gradient. Like momentum and RMSProp, 1st-moment estimation and 2nd-moment estimation can be expressed from Equations (9) and (14) respectively. The above estimations are initialized with 0's vectors which make it biased towards zero during the initial iterations. This can be corrected by computing bias correction such as:

$$\Delta w_i^{a\prime} = \frac{\Delta w_i^a}{1-(\beta^a)^i} \tag{22}$$

$$\Delta w_i^{b\prime} = \frac{\Delta w_i^b}{1-(\beta^b)^i} \tag{23}$$

The parameters are updated by:

$$\Delta w_i = \frac{\propto}{\sqrt{\Delta w_i^{b\prime}} + \epsilon}\Delta w_i^{a\prime} \tag{24}$$

$$w_{i+1} = w_i - \Delta w_i \tag{25}$$

The parameter update rule in Adam is based on scaling their gradient inversely proportional to the l^2 norm of their individual present and past gradient. In *AdaMax* the l^2 norm based update rule is extended to l^p norm based update rule (Kingma and Ba, 2014). For a more stable solution and avoiding instability of large p, if let $p \to \infty$ then:

$$\Delta w_i^d = \max\left(\beta^d \Delta w_{i-1}^d, |\nabla E_i|\right) \tag{26}$$

where Δw_i^d is exponentially weighted infinity norm. The update parameter can be expressed as:

$$\Delta w_i = \left(\frac{\propto}{1-(\beta^a)^i}\right)\frac{\Delta w_i^a}{\Delta w_i^d} \tag{27}$$

$$w_{i+1} = w_i - \Delta w_i \tag{28}$$

The advantages of Adam and AdaMax is that it combines characteristics of both AdaGrad while dealing with sparse gradient and RMSProp dealing with non-stationary objectives. Adam and its extension AdaMax require less memory and suitable for both convex and non-convex optimization problems.

4.1.1 Application of optimization algorithms for learning rate

The learning algorithms are found to have application in a complex domain problem solving. The high dimensional data with a lot of features may make the task difficult to solve with an initial guess and manual settings of the parameter such as learning rate. Deciding large learning rate may make network unstable causing the generalization performance to be poor, whereas, small learning rate may reduce the learning speed. The algorithms in this category help to adjust the learning rate on each iteration and move the gradients faster towards the long axis of the valley for better performance. Table II contains some of the application on complex task problem solving. AdaGrad solved the problem of newspaper articles classification by improving various categories performance by an average 9%-109%. Similarly, for the subcategory image classification problem, AdaGrad was able to improve the precision 2.09% - 9.8%. The graphical representation of AdaDelta results showed that it gained highest accuracy on speech recognition of English data. The application of Adam on multiclass logistics regression, multilayer neural network and convolutional neural network for classification of handwritten images, object images, and movie reviews graphically showed that the Adam get a better generalization performance by yielding smaller convergence per iteration. The above problems illustrate that the application of optimization algorithms for learning rate are more suitable for high dimension complexed dataset to avoid manual adjustment of learning rate.

4.2Bias and Variance (Underfitting and Overfitting) Minimization algorithms

The best FNN should be able to truly approximate training and testing data. Researchers extensively studied FNN to avoid high bias and variance to improve convergence approximation. The high bias is referred to a problem known as underfitting and high variance is referred to as overfitting. Underfitting occurs when the network is not properly trained and the patterns in training data are not fully discovered. It occurs before the convergence point and, in this case, the generalization performance (also known as test data performance) does not deviate much from training data performance. That the reason that the underfitting region has high bias and low variance. In opposite to this, overfitting occurs after the convergence point when the network is overtrained. In this scenario, the training error is in a continuous state of decreasing whereas testing error starts increasing. The overfitting reason is known as having the property of low bias and high variance. The high variance in training and testing error occurs because the network trains the noise in the training data which might be not part of the actual model data. The best choice is to choose a model which balance bias and variance which is known as bias and variance trade-off. Bias and variance trade-off is a point on which FNN can discover all pattern in training data and simultaneously give better generalization performance (Geman et al., 1992).

For simplicity, the bias-and-variance trade-off point is referred to as the convergence point. The FNN should be stopped from further training when it approaches convergence point to balance bias (underfitting) and variance

Application	Description
Game decision rule	Predicting decision rules for the Nine Men's Morris game (Riedmiller and Braun, 1993)
Census	Predicting whether the individual has income above or below the average income based on certain demographic and employment-related information (Duchi et al., 2011)
Newspaper articles	Classifying the newspapers articles into four major categories such as economics, commerce, medical, and government with multiple more specific categories (Duchi et al., 2011)
Subcategories image classification	Classifying thousands of images in each of their individual subcategory (Duchi et al., 2011)
Handwritten images classification	Classifying images of the handwritten digits (Duchi et al., 2011; Kingma and Ba, 2014; Zeiler, 2012)
English data recognition	Recognizing speech from several hundred hours of the US English data collected from voice IME, voice search and read data (Zeiler, 2012)
Movie reviews	Classifying review of the movies either positive or negative to know the sentiment of the reviewers (Kingma and Ba, 2014)
Digital images classification	Classifying the images into one of a category such as an airplane, deer, ship, frog, horse, truck, cat, dog, bird and automobile (Kingma and Ba, 2014)

Table II. Applications of optimization algorithms for learning rate

(overfitting). The most popular method to identify and stop the FNN at convergence point is to select suitable *validation data*. Validation dataset and test dataset are used interchangeably and are unseen data which is held back from training data to check the network performance. The error estimate by training data is not a useful estimator and comparing its performance with validation dataset generalization performance can help to identify convergence point (Zaghloul et al., 2009). One of the drawbacks associated with the validation technique is that the dataset should be large enough to split it (Reed, 1993). This makes it unsuitable for small dataset especially complicated data with few instances and many variables. When enough data is not available, the *n-fold cross-validation technique* can be considered as an alternative to the validation technique (Seni and Elder, 2010). The *n*-fold cross-validation technique split data randomly into *n*-equal sized data subsample. The FNN is repeated *n* times by allocating one subsample for testing and remaining *n-1* for the training model. The *n*-subsample is allocated once for each testing. Finally, the *n*-results are averaged to get a single accurate estimation. The suitable selection of *n*-fold size is based on dataset complexity and network size which make this technique challenging for complicated applications.

The underfitting problem is easily detectable and therefore literature is much focused on proposing techniques for improving overfitting. Overfitting is a common problem in nonparametric models and it greatly influences generalization performance. Overfitting mainly results from hyperparameters initialization and adjustments. For instance, the size of hidden units for the successful training of the network is not always evident. A large size network has a greater possibility to overfit as compared to a smaller size (Setiono and Hui, 1995). Similarly, overfitting chances increase when the network degree of freedom (such as weights) increases largely from the training sample (Reed, 1993). These two problems have gain greater attention which led to the development of techniques explained below to avoid overfitting analytically rather than trial and error approaches. The algorithms for this category are subcategorized into *regularization, pruning, and ensembles*. The algorithms and techniques in this category are discussed by considering their contribution scope in FNN only. For instance, the dropout technique, as discussed below, was not limited to FNN but also applied to graphical models such as Boltzmann Machines.

4.2.1 Regularization Algorithms

Overfitting occurs when the training examples information does not match with network complexity. The network complexity depends upon the free parameters such as a number of weights and added bias. Often it is desirable to keep the network complexity less by keeping less weight to train networks. This can be done by limiting the growth of weight by techniques known as *weight decay* (Krogh and Hertz, 1992). It penalizes the large weights from growing too large by adding penalize term in loss function:

$$\sum_{h=1}^{m}(a_h - p_h)^2 + \lambda \sum_{j=1}^{p}\left(w_j\right)^2 \tag{29}$$

Where $\sum_{h=1}^{m}(a_h - p_h)^2$ is the error function and w_j is the weight in a j position which will be penalized by parameter λ. The selection of λ depends upon how large to penalize weight. The above equation expressed that if the training examples are more informative than network complexity, the regularization influence will be weak and vice versa. The above weight decay method is also known as l^2 *regularization*. Similarly, for l^1 *regularization*:

$$\sum_{h=1}^{m}(a_h - p_h)^2 + \lambda \sum_{j=1}^{p}\left|w_j\right| \tag{30}$$

l^1 regularization add absolute value, while l^2 regularization add the squared value of weight to the loss function. Krogh and Hertz (1992) experimental work demonstrated that FNN with weight decay technique can avoid overfitting and improve the generalization performance. The improvement as mentioned in their paper was less significant. Several high-level techniques, as discussed below, was proposed to achieve better results.

A renowned regularization technique *Dropout* was proposed for large fully connected network to overcome the issues of overfitting and the need for FNN ensembles (Srivastava et al., 2014). With a limited dataset, the estimation of training data with noise will not approximate testing data which will lead to overfitting. Firstly, this happens because each parameter during the gradient update influences the other parameter, which creates complex coadoption among hidden units. This coadoption can be broken by removing some hidden units from the network. Secondly during FNN ensembles, combining the average of many outputs of separately trained FNN is an expensive task. Training many FNN requires many trial and error approaches to find the best possible hyperparameters which make it a daunting task and need a lot of computation efforts. Dropout avoids overfitting and provides a method of approximately combining exponentially several FNN. The technique temporary drop the units (hidden and visible) along with all connections with certain random probability during network forward steps. Training FNN with dropout means training 2^n thinned networks with n is a number of units in FNN. During test time, it is not recommended to take an average of prediction from all thinned trained FNNs. It is best to use a single FNN during test time without dropout. The weights of this single FNN will be scaled down version of all trained weights. If the unit output is denoted by y_i', input by x_i' with weight by w', than dropout is expressed as:

$$y_i' = r * f(w' x_i') \tag{31}$$

Where r is a vector of independent Bernoulli random variable with * denotes element wise product. The downside of dropout is that its training time is 2-3 times longer than standard FNN for the same architecture. However, the experimental results achieved by dropout compared to FNN are remarkable. *DropConnect*, a generalization of Dropout, is another regularization technique for large fully connected layer FNN to avoid overfitting (Wan et al., 2013). The key difference between Dropout and DropConnect is their dropping mechanism. The Dropout temporary drops the units along with their connection weights, whereas, DropConnect randomly drop a subset of weights connection with a random probability. The technique is like Dropout and the only key difference is that DropConnect drops connection weights during forward propagation of the network. The output y_i' in Dropconnect can be expressed as:

$$y_i' = f\big((r * w')x_i'\big) \tag{32}$$

Experimental results on MNIST, CIFAR-10, SVHN, and NORB demonstrates that DropConnect outperforms compared to No-Drop (backpropagation) and Dropout. The training time of DropConnect is slightly higher than No-Drop and Dropout due to feature extractor bottleneck in large models. The advantage of DropConnect is that it allows training a large network without overfitting with better generalization performance. The limitation of both Dropout and DropConnect are that they are suitable only for fully connected layer FNN. Other regularization technique includes *Shakeout* and was proposed to remove unimportant weights to enhance FNN prediction performance (Kang et al., 2017). The performance of FNN may not severely be affected if most of the unimportant connection weights are removed. One technique is to train a network, prune the connections and finetune the weights. However, this technique can be simplified by imposing sparsity-inducing penalties during the training process. During implementation, Shakeout randomly chooses to reverse or enhance unit contribution to the next layer in the training forward stage to avoid overfitting. Dropout can be considered a special case of shakeout by keeping enhance factor to one and reverse factor to zero value. Shakeout induced l^0 and l^1 regularization which penalize the magnitude of weight and leads to the sparse weight that truly represents a connection among units. This sparsity induced penalty (l^0 and l^1 regularization) is combined with l^2 regularization to get a more effective prediction. It randomly modifies the weight based on r and can be expressed as:

$$\begin{cases} w_{i+1}' = -cs & if\ r = 0 \\ w_{i+1}' = \dfrac{w_i' + c\tau s}{1-\tau} & otherwise \end{cases} \tag{33}$$

Where $s = \text{sgn}(w_i')$ with ±1 or 0 if w_i'=0, constant $c > 1$ and $\tau \in [0,1]$. When hyperparameter c is set to zero, shakeout can be considered a special case of Dropout. Experimental results from MNIST, CIFAR-10, and ImageNet demonstrate that Shakeout Outperform standard No-Drop and Dropout, especially on much scarce data.

Beside dropping hidden units or weights, *batch normalization* method avoids overfitting and improves the learning speed by preventing the formulation of internal covariant shift problem, which occurs due to change in parameters during the training process of FNN (Ioffe and Szegedy, 2015). Updating the parameters during training affects the layers of input distribution in that it makes it deeper and slower with require for more careful

hyperparameters initialization. Batch normalization technique eliminates the internal covariant shift and accelerates network training. This simple technique improves the layers distribution by normalizing each layer activation with mean zero and unit variance. Unlike BP with higher learning rate which results in gradient overshoot or diverges and stuck in local minima, the stable distribution of activation values by batch normalization during training can tolerate for larger learning rate. The effective results can be achieved by performing the normalization of mini-batches and backpropagate the error through normalized parameters. During the training process of batch normalization, the goal is to accomplish stable activation function during training which results in either the reduction or elimination of overfitting from the network. This makes it a regularization technique and eliminates the need for Dropout in most cases.

Similarly, besides using the validation and its variant techniques which may not capture overfitting, and might go undetected, which cost more resources. *Optimized approximation algorithm (AOA)* avoids overfitting without any need for a separate validation set (Yinyin Liu et al., 2008). It uses stopping criteria formulated in an original paper known as *Signal to Noise Ratio Figure (SNRF)* to detect overfitting during the training error measurement. It calculates SNRF during each iteration and if it is less than the defined threshold value, the algorithm is stopped. Experimental works on a different number of hidden nodes and iteration validate AOA effectiveness.

4.2.2 Pruning FNN

Training a network with larger size requires more time because of many hyperparameters adjustment which can lead to overfitting if not initialized and controlled properly, whereas, the smaller network will learn fast but may converge to poor local minima or unable to learn data. Therefore, it is often desirable to get an optimal network size with no overfitting, better generalization performance, less hyperparameter, and fast convergence. The effective way is to train a network with large possible architecture and remove the parts that are insensitive to performance. This approach is known as *pruning* because it prunes unnecessary units from the large network and reduces its depth to the optimal possible small network with few parameters. This technique is considered effective in eliminating the overfitting problem and improving generalization performance but at the same time, it requires a lot of efforts and training time to construct a large network and reduce it. The simplest technique of pruning involves training a network, pruning unnecessary part and retrain. The training and retraining of weights in pruning will sustainably increase the training time of FNN. The training time can be reduced to some extent by calculating the sensitivity of each weight with respect to global loss function (Karnin, 1990). The weight sensitivity can be calculated from the expression:

$$S_i = \sum_{i=0}^{n-1} [\Delta w_i]^2 \frac{w_{i+1}}{\propto (w_{i+1} - w_i)} \tag{34}$$

The idea is to calculate sensitivity concurrently during the training process without interfering with the learning process. At the end of the training process, the sensitivity will be sort in descending order and weight with fewer sensitivity values will be pruned. This method advantages over traditional approaches that instead of training and retraining which consume more training time, it prunes one-time weights that are insensitive and are less important

for error reduction. Reed (1993) explained that besides sensitivity methods, penalty methods (weight decay or regularization) is also another type of pruning. In sensitivity methods, weight or units are removed, whereas, in penalty method weights or units are set to zero which is like removing them from FNN. Kovalishyn et al. (1998) applied the pruning technique to the constructive FNN algorithm and demonstrate that constructive algorithms (also known as cascade algorithms) generalization performance improves significantly compared to fixed topology FNN. Han et al. (2015) demonstrated the effectiveness of the pruning method by performing experimental work on AlexNet and VGG-16. Pruning method reduced the parameters from 61 million to 6.37 million and 138 million to 10.3 million respectively without losing accuracy. They used three stages iterative pruning technique: learning with normal FNN, prune low weight connections and retrain the pruned network to learn final weights. Using pruned weights without retraining may significantly impact the accuracy of FNN.

4.2.3 Ensembles of FNNs

Another effective method of avoiding overfitting is the *ensembles of FNNs*, which means training multiple networks and combining their prediction as opposed to single FNN. The combination is done through averaging in regression and majority in classification or weighted combination of FNNs. The more advanced techniques include stacking, boosting and bagging with many others. The error and its variance obtained by the collective decision of ensembles is considered to be less as compared to individual networks (Hansen and Salamon, 1990). Many ensembles combine three stages: defining the structure of FNN in ensembles, train each FNN, and combine the results. Krogh et al. (1995) explained that the generalization error of ensembles is less than network individual error for uniform weight. If the ensembles are strongly biased than ensembles generalization error will be equal to individual network error, whereas, if the variance is high, the ensemble generalization error will be higher than individual network error. Therefore, during combination, it reduces the variance (overfitting) by reducing the uncertainty in prediction. Ensembles are best suitable to deal with complex large problems which are difficult to solve by the single FNN. Islam et al. (2003) explained that drawback with existing ensembles is that the number of FNNs in the ensemble and the number of hidden units in individual FNN need to be predefined for FNNs ensemble training. This makes it suitable for application where prior rich knowledge and FNN experts are available. This traditional method involves a lot of trial and error approaches with tremendous efforts. The above-mentioned drawbacks can be overcome by the *constructive NN ensemble (CNNE)* by determining both the number of FNNs in the ensemble and their hidden units in individual FNN based on negative correlation learning. Negative correlation learning promotes diversity among individual FNN by learning different patterns of training data which will cause the ensemble to learn better as a whole. The stopping criteria were defined as an increase in ensemble error rather than individual error increase. FNN ensembles are still being used in many applications with different strategies by researchers. The above discussion covers some of the techniques for effective ensembles and others can be more explored in the literature.

4.2.4 Applications of bias and variance minimization algorithms

The algorithms in this category improve the generalization performance of the network and include applications that cannot be solved by the simple network and more sophisticated knowledge is needed to avoid overfitting for gaining better performance. The regularization, pruning, and ensembles methods are implemented by training many networks simultaneously to get the average best results. The limitation of this category is that it requires more learning time compared to the standard networks for convergence. The limitation can be minimized by adopting a hybrid approach of learning optimization algorithms and this category. Besides, the limitation of regularization technique such as dropout, DropConnect, and shakeout are that they are suitable to work with fully connected neural networks.

Table III shows some of the applications in the area such as classification, recognition, segmentation, and forecasting. Shakeout work on the classification of high dimensional complex handwritten digits images and digital images, comprising of more than 50,000 images, improved the accuracy about 0.95%-2.85% and 1.63%-4.55% respectively for fully connected neural networks. For classifying more than 50,000 house number images generated from google street view, DropConnect slightly enhanced the accuracy to 0.02%-0.03%. Similarly, DropConnect work on 3D object recognition achieved better accuracy of 0.34%. Shakeout and DropConnect show promising results on highly complex data, however, the popularity and successes of dropout technique in many applications is comparatively large. Some applications of dropout include classifying species images into their class and subclasses, recognizing the speech of different dialects, classifying newspaper articles into different categories, human diseases by predicting alternative splicing and classifying hundred and thousands of images into different categories. The dropout, because of its property of dropping hidden unit to avoid overfitting, achieved an average more than 5% better accuracy for the mentioned applications.

The application of ensemble technique CNNE on various problems, such as deciding credit card request, diagnosing breast cancer, diagnosing diabetes, identifying object used in conducting crime, categorizing heart diseases, identifying images as English letter, and determining types of defects in soybean, performed an average 1-3 times better in generalization performance compared to other popular ensembles techniques. Some other problems that have gained popularity in this category are predicting energy consumption, estimating the angular acceleration of the robot arm, analysing semiconductor manufacturing by examining the number of dies, and predicting credit defaulter.

4.3Constructive topology FNN

FNN with a fixed number of hidden layers and units are known as fixed-topology networks which can be either shallow (single) or deep (more than one) depending upon application task. Whereas, the FNN that starts with the simplest network and then adds hidden units until the error convergence is known as constructive (or cascade) topology network. The drawbacks associated with fixed typology FNN is that hidden layers and units need to be predefined before training initialization. This needs a lot of trial and error to find optimal hidden units in the layers. If too many hidden layers and units are selected the training time will increase, whereas, if fewer layers and hidden unit are selected might result in poor convergence. Constructive topology networks have advantages

Application	Description
Handwritten images classification	Classifying the images of hand-written numerals with 20% flip rate (Geman et al., 1992)
Text to speech conversion	Learning to convert English text to speech (Krogh and Hertz, 1992)
Credit card	Deciding to approve or reject credit card request based on the available information such as credit score, income level, gender age, sex, and many others (Islam et al., 2003)
Breast cancer	Diagnosing breast cancer as a malignant or benign based on the feature extracted from the cell nucleus (Islam et al., 2003)
Diabetes	Diagnosing whether the patient has diabetes based on certain diagnostic measurements (Islam et al., 2003)
Crime	Identifying glass type used in crime scene based on chemical oxide content such as sodium, potassium, calcium, iron and many others (Islam et al., 2003)
Heart diseases	Diagnosing and categorizing the presence of heart diseases in a patient by studying the previous history of drug addiction, health issues, blood tests, and many others (Islam et al., 2003)
English letters	Identifying black and white image as one of the English letters among twenty-six capital letters (Islam et al., 2003)
Soybean defects	Determining the type of defect in soybean based on physical characteristics of the plant (Islam et al., 2003)
Energy consumption	Predicting the hourly consumption of building electricity and associated cost based on environmental and weather conditions (Yinyin Liu et al., 2008)
Robot arm acceleration	Estimating the angular acceleration of the robot arm based on a position, velocity, and torque (Yinyin Liu et al., 2008)
Semiconductor manufacturing	Analysing semiconductor manufacturing by examining the number of dies in a wafer that pass electrical tests (Seni and Elder, 2010)
Credit defaulter	Predicting credit defaulters based on the credit score information (Seni and Elder, 2010)
3D object recognition	Recognizing 3D object by classifying the images into generic categories (Wan et al., 2013)
Google street images classification	Classifying the images containing information of house numbers collected by google street view (Srivastava et al., 2014; Wan et al., 2013)
Class and superclass	Classifying the images into class (for example: shark) and their superclass (for example: fish) (Srivastava et al., 2014)
Speech recognition	Recognizing speech from different dialects of the American language (Srivastava et al., 2014)
Newspaper articles	Classifying the newspapers articles into major categories such as finance, crime, and many others (Srivastava et al., 2014)
Ribonucleic acid	Understanding human disease by predicting alternative splicing based on ribonucleic acid features (Srivastava et al., 2014)
Subcategories image classification	Classifying thousands of images in each of their individual subcategory (Han et al., 2015; Ioffe and Szegedy, 2015; Srivastava et al., 2014)
Handwritten digits classification	Classifying images of the handwritten digits (Han et al., 2015; Kang et al., 2017; Srivastava et al., 2014; Wan et al., 2013)
Digital images classification	Classifying the images into one of a category such as an airplane, deer, ship, frog, horse, truck, cat, dog, bird and automobile (Kang et al., 2017; Srivastava et al., 2014; Wan et al., 2013)

Table III. Applications of bias and variance minimization algorithms

over fixed topology that it starts with a minimal simple network and then add hidden layers with some predefined hidden units until error convergence. This eliminates the need for trial and error approach and automatically construct the network. Recent studies have shown that constructive algorithms are more powerful than fixed algorithms. Hunter et al. (2012) in their FNNs comparative study explained that a constructive network can solve the Parity-63 problem with only 6 hidden units compared to a fixed network which required 64 hidden units. The training time is proportional to the hidden unit's quantity. Adding more hidden units will cause the network slow because of more computational work. The computational training time of constructive algorithms is much better than fixed topology. However, it does not mean that the generalization performance of constructive networks will be always superior to the fixed network. A more careful approach is needed to handle hidden units with a lot of connections and parameters to stop the addition of further hidden units for better generalization performance (Kwok and Yeung, 1997).

The most popular algorithm for constructive topology network known in literature is the *Cascade-Correlation neural network(CNN)* (Fahlman and Lebiere, 1990). CNN was developed to address the slowness of BP based

fixed topology FNN (BPFNN). The factor that contributes to the slowness of BPFNN is that each hidden unit faces a constantly changing environment when all weights in the network are changed collectively. CNN initializes with a simple network by linearly connecting input x to output p by the output connection weight w^{ocw}. Most often, the QP learning algorithm is applied for repetitively tuning of w^{ocw}. When training converges, and the E is more than ε, then a new hidden unit u is added with receiving all input connections w^{icw} from the x and any pre-existing hidden units. The u is trained to maximize it covariance S^{cov} magnitude with E as expressed below:

$$S_r^{cov} = \sum_o \left| \sum_r (u_r - \bar{u})\left(E_{r,o} - \overline{E_o}\right) \right| \tag{35}$$

where $\bar{u}$ and $\overline{E_o}$ are mean values of hidden unit u_r and output error $E_{p,o}$, o and r are the network output and the hidden unit respectively. The magnitude is maximized by computing derivative of S with respect to w^{icw} and performing gradient ascent. When there is no more increase in S^{cov}, the $w^{icw}$is kept frozen and u is connected to p through the w^{ocw}. Again, the w^{ocw} are retrained, and E is calculated. If the E is less than ε the algorithm is stopped, otherwise, u are added one by one till acceptable error. CNN suggests using QP learning algorithm for training because of its characteristics to take larger steps toward minimum rather than GD infinitesimal small steps. Experimental work on nonlinear classification benchmarking two spiral problem (Lang, 1989) demonstrates the training speed effectiveness of CNN over BPFNN. The major advantage of CNN is that it learns quickly and determine its own network size. However, recent studies show that CNN is more suitable for classification task (Hwang et al., 1996). This narrow the scope of CNN in other application areas. Some studies have shown that the generalization performance of CNN may not be optimal and even more larger network may require (Lehtokangas, 2000). Kovalishyn et al. (1998) experimental work on quantitative structure activity relationship (QSAR) data did not find any model where the performance of CNN is significantly better than BPFNN. They optimized the performance of CNN by introducing a pruning algorithm.

Evolving cascade Neural Network (ECNN) was proposed to select the most informative features from the data to resolve the issue of overfitting in CCN (Schetinin, 2003). Overfitting in CNN results from the noise and redundant features in the training data which affects its performance. ECNN select the most informative features by adding initially one input unit and evolve network by adding new input unit as well as a hidden unit. The final ECNN has a minimal number of hidden units and most informative input features in the network. The selection criteria for a neuron is based on the regularity criterion C extracted from the Group Method of Data Handling (GMDH) algorithm (Farlow, 1981). The higher the value of C, the less informative the hidden unit will be. The selection criteria for hidden units can be expressed as:

$$if\ C_r < C_{r-1}, accept\ r^{th}\ hidden\ unit \tag{36}$$

The selection criteria illustrate that if C_r calculated for the current hidden unit is less than the previous hidden unit C_{r-1}, it means that the current hidden unit is more informative and relevant than the previous unit and then will be added in the network.

Huang, Song, et al. (2012) explain that CNN covariance objective function is maximized by training w^{icw} which cannot guarantee a maximum error reduction when the new hidden unit is added. This will slow down the convergence and more hidden units will be needed which will reduce the generalization performance. In addition to this, the repetitive tuning of w^{ocw}after each hidden unit addition is more time consuming. They proposed algorithm named as *Orthogonal Least Squares-based Cascade Network (OLSCN)*, which implemented an orthogonal least squares technique and derived a new objective function S^{ols} for input training expressed as below:

$$S_r^{ols} = E_{r-1} - E_r = (\gamma_r^T a)^2/(\gamma_r^T \gamma_r) \tag{37}$$

where γ are the element of the orthogonal matrix obtained by performing QR factorization of the output of the r hidden unit. This objective function is optimized iteratively by using second order algorithm as expressed below:

$$w_{i+1\,(r)}^{icw} = w_{i\,(r)}^{icw} - [\mathbf{H}_r - \mu I_r]^{-1} g_r \tag{38}$$

where **H** is a Hessian matrix, *I* is an identity matrix and μ is damping hyperparameter factor, g_r is the gradient of the new objective function S_r with respect to w^{icw} for hidden unit r. The information generated from input training are further continued to calculate w^{ocw} using back substitution method for linear equations. The w^{icw} , after randomly initializing, are trained based on the above objective function, however, w^{ocw} are calculated after all u are added and thus does not need any repeatedly training. The benefit of OLSCN is that it needs less hidden units as compared to original CCN for the same training example with some improvement in generalization performance.

In addition, the *Faster Cascade Neural Network (FCNN)* was proposed to improve the generalization performance of CCN and improve the existing OLSCN drawbacks: Firstly, the linear dependence of candidate units with existing u can cause the new objective function in OLSCN to make mistake. Secondly, the w^{icw} modification by modified Newton's Method is based on a second-order **H** which may result in a local minimum and convergence slow due to heavy computation (Qiao et al., 2016). In FCNN, hidden nodes are generated randomly and remain unchanged as inspired by randomly mapping algorithms. The w^{ocw} connecting both input and hidden units to output units are calculated after addition of all necessary input and hidden units. The FCNN initializes with no input and hidden unit in the network. The bias unit is added to network and input units are added one by one with error calculation for each input unit. When there are no input units to be added, a pool of candidate units is randomly generated. The candidate unit with the maximum capability to error reduction computed from the reformulated modified index is added as a hidden unit to the network. When a maximum number of hidden units or defined threshold has achieved, the addition of hidden units is stopped. Finally, the output units are calculated by computing back substitution like OLSCN. The experimental comparison among FCNN, OLSCN, and CNN demonstrated that FCNN achieved better generalization performance and fast learning, however, the network architecture size increases many times.

Nayyeri et al. (2018) proposed to use correntropy based objective function with a sigmoid kernel to adjust the w^{icw} of the newly added hidden unit rather than covariance (correlation) objective function. The success of correlation is heavily relied on the Gaussianity and linearity assumptions, whereas, properties of correntropy is to make the network more optimal in nonlinear and non-Gaussian. The new algorithm with correntropy objective function based on a sigmoid kernel is named as *Cascade Correntropy Network (CCOEN)*. Similar to other cascade algorithms described above, w^{icw} is optimized by using correntropy objective function S^{ct} with sigmoid:

$$S_r^{ct} = \max_{U_r} \left(\sum_{i=1}^{r} \left(\tanh(\mathrm{a}_r \langle E_{r-1}, U_r \rangle + c) \right) - C \left\| w_r^{icw} \right\|^2 \right) \tag{39}$$

Where $\tanh(\mathrm{a}_r \langle ., . \rangle + c)$ is defined as a sigmoid kernel with a and c hyperparameters. $C \in \{0,1\}$ is a constant. If the new hidden unit and residual error are orthogonal than $C = 0$ to ensure algorithm convergence, otherwise $C = 1$. The w^{ocw} is adjusted as:

$$w_r^{ocw} = \frac{E_{r-1} U_r^T}{U_r U_r^T} \tag{40}$$

Application	Description
Biological activity	Predicting of the biological activity of molecules such as benzodiazepine derivatives with anti-pentylenetetrazole activity, antimycin analogues with antifilarial activity and many others from molecular structure and physicochemical properties (Kovalishyn et al., 1998)
Communication channel	Equalizing burst of bits transferred through a communication channel (Lehtokangas, 2000)
Clinical electroencephalograms	Classifying artifacts and a normal segment in clinical electroencephalograms (Schetinin, 2003)
Vowel recognition	Recognizing vowel of different or same languages in the speech mode (Huang, Song, et al., 2012)
Cars fuel consumption	Determining the fuel consumption of cars in terms of engine specification and car characteristics (Huang, Song, et al., 2012)
Beverages quality	Determining quality of same class of the beverages based on relevant ingredients (Huang, Song, et al., 2012; Qiao et al., 2016)
House price	Estimating the price of houses based on the availability of clean quality air (Huang, Song, et al., 2012; Nayyeri et al., 2018)
Species	Determining the age of species from their known physical measurements (Huang, Song, et al., 2012; Nayyeri et al., 2018)
Soil classification	Classifying image according to a different type of soil such as grey soil, vegetation soil, red soil and many others based on a database consisting of the multi-spectral images (Qiao et al., 2016)
English letters	Identifying black and white image as one of the English letters among twenty-six capital letters (Qiao et al., 2016)
Crime	Identifying glass type used in crime scene based on chemical oxide content such as sodium, potassium, calcium, iron and many others (Qiao et al., 2016)
Silhouette vehicle images classification	Classifying image into different types of vehicle based on the feature extracted from the silhouette (Qiao et al., 2016)
Outdoor objects segmentation	Segmenting the outdoor images into many different classes such as window, path, sky and many others (Huang, Song, et al., 2012; Qiao et al., 2016)
Human body fats	Determining the percentage of human body fats from key physical factors such as weight, age, chest size and other body parts circumference (Nayyeri et al., 2018)
Automobile prices	Determining the prices of automobile based on various auto specifications, the degree to which auto is risky than price, and an average loss per auto per year (Nayyeri et al., 2018)
Presidential election	Estimating the proportion of voter in the presidential election based on key factors such as education, age, and income (Nayyeri et al., 2018)
Weather forecasting	Forecasting weather in terms of cloud appearance (Nayyeri et al., 2018)
Basketball winning	Predicting basketball winning team based on players, team formation and actions information (Nayyeri et al., 2018)
Industrial strike volume	Estimating the industrial strike volume for the next fiscal year considering key factors such as unemployment, inflation and labor unions (Nayyeri et al., 2018)
Earthquake strength	Forecasting the strength of earthquake given its latitude, longitude and focal point (Nayyeri et al., 2018)
Heart diseases	Diagnosing and categorizing the presence of heart diseases in a patient by studying the previous history of drug addiction, health issues, blood tests, and many others (Nayyeri et al., 2018)

Table IV. Applications of constructive algorithms

An experimental study was performed on regression problems with and without noise and outliers by comparing CCOEN with six other objective functions as defined in (Kwok and Yeung, 1997), CNN covariance objective function as shown in Equation (35), and one hidden layer feedforward neural network. The study demonstrates that CCOEN correntropy objective function, in most cases, achieved better generalization performance and increase robustness to noise and outliers compared to other objective functions. The network size of CCOEN shows slightly less compactness compared to other objective function, however, no specific objective function was found to have a better compact size in general.

4.3.1 Application of Constructive FNN

The benefit of constructive over fixed topology FNN, which make it more favorable, is the addition of a hidden unit in each hidden layer until error convergence. This eliminates the problem of finding an optimal network for fixed topology FNN by doing a lot of experimental work. The learning speed of constructive algorithms are better

than fixed topology, however, the generalization performance is not always guaranteed to be optimal. Table IV shows some of the applications of constructive FNN.

The application of CCOEN on regression prediction of human body fats, automobile prices, voters, weather, winning team, strike volume, earthquake strength, heart diseases, house price, and species age showed that algorithm gives generalization performance an average 4 times better for most cases. CCOEN theoretically guaranteed the solution will be global minimum, however, the improvement in learning speed is not clear. The prediction accuracy and learning speed of FCNN is considered to be 5% better and more than 20 times faster respectively on predicting beverages quality, classifying soil in to different types, identifying black and white image as one of the English letters, identifying object used in crime, classifying image into different types of vehicle and segmenting outdoor images. The problem of vowel recognition and car fuel consumption are considered to be better solved by OLSCN. The vowel was recognized with improvement in accuracy of 8.61% and car fuel consumption was estimated with performance improvement of 0.15 times. Some other problems demonstrating the applications of this category include molecular biological activities prediction, equalizing burst of bits and classifying artifacts and a normal segment in clinical electroencephalograms.

4.4Metaheuristic Search Algorithms

The major drawback of FNN is that it stuck at a local minimum with a poor convergence rate and it became worse at the plateau surface where the rate of change in error is very slow at each iteration. This increases the learning time and coadaptation among hyperparameters. Therefore, the trial and error approaches are applied to find optimal hyperparameters, this makes gradient learning algorithms even more complex and decision become difficult to select the best FNN. Moreover, the unavailability of gradient information in some applications makes FNN hopeless. To solve the two main issues of FNN that need to find best optimal hyperparameters and to make it useable with no gradient information, the application of metaheuristic algorithms such as genetic algorithm, particle swarm optimization, and whale optimization algorithm are implemented in combination with FNN. The major contribution of the applications of metaheuristic algorithms is that it may converge at a global minimum rather than the local minimum by moving from local search to global search. Therefore, these are more suitable for global optimization problems. The metaheuristic algorithms are good for identifying best hyperparameters and convergence at a global minimum but has drawbacks of high memory requirement and processing time.

4.4.1 Genetic Algorithm

Genetic Algorithm (GA) is a metaheuristic technique belonging to the class of evolutionary algorithms (EA) inspired by Darwinian theory about evolution and natural selection and was invented by John Holland in the 1960s (Mitchell, 1998). It is used to find a solution for optimization problems by performing chromosome encoding, fitness selection, crossover, and mutation. Like BP, it is used to train FNN to find the best possible hyperparameters. *Genetic algorithm based FNN (GAFNN)* initialized with a population of serval possible solution of FNN with randomly assigned hyperparameters. The hyperparameters are encoded in chromosome string of 0 and 1. The fitness function is reciprocal of the loss function in FNN to evaluate each chromosome in the

population. The higher the value of fitness means lower will be its loss function and better chances of reproduction. Initially, two chromosomes are selected from a population with either higher fitness value or using selection techniques such as roulette wheel, tournament, steady state and ranked position. Reproduction of selected chromosomes is performed by crossover and mutation. Crossover produces new offspring's and possible techniques include single point crossover, multiple point crossovers, shuffle crossover and uniform crossover. Mutation slightly alters the new offspring to create a better string and possible methods include bit string, flip bit, shift, inverse, and Gaussian. The two newly generated offspring are added in population and repeat the cycle of selection and reproduction until a maximum number of generations has achieved. The old population will be deleted to create a space for new offspring. When the algorithm is stopped it will return a chromosome that representing best hyperparameters for FNN. The above-explained GAFNN computational method is simple and adopted by many researchers with different techniques for selection, crossover, and mutation. Researchers demonstrated that the generalization performance of GA is better than BP (Mohamad et al., 2017) with the additional need for training time. Liang and Dai (1998) highlighted the same issue by applying GA to FNN. They demonstrated that the performance of GA is better than BP, but the training time increases. Ding et al. (2011) proposed that GA performance can be made better by integrating the concept of both BP gradient information and GA genetic information to train an FNN for superior generalization performance. GA is good for global search, whereas, BP is appropriate for local search. The algorithm proposed to first use GA to optimize initial weights by searching better search space and then use BP to fine tune and search for an optimal solution. Experimental results demonstrate that FNN trained on the hybrid approach of GA and BP achieved better generalization performance than individual BPFNN and GAFNN. Moreover, in term of learning speed, a hybrid approach is better than GAFNN compared to BPFNN.

4.4.2 Particle Swarm Optimization

Eberhart and Kennedy (1995) argue that GA used to find hyperparameters may not be suitable for crossover operator. The two chromosomes selected with high fitness values might be very different from one another and reproduction will be not effective. The *Particle Swarm Optimization (PSO) based FNNs (PSOFNN)* can overcome the issue in GA by searching the best optimal solution through the movement of particles in space. The concept of particle movement is motivated by the flocking of birds. Like GA, the particles in PSO can be considered as a hyperparameter. Each particle represents the hyperparameter that needs to be optimized for the global search rather than the local search. PSO is an iterative algorithm in which particles w adjust its velocity v during each iteration based on momentum, its best possible position achieved p^{pbest} so far and the best possible position achieved by global search p^{gbest}. This can be expressed in mathematically as:

$$v_{i+1} = v_i + c_1 * r * \left(p_i^{pbest} - w_i\right) + c_2 * r * \left(p_i^{gbest} - w_i\right) \tag{41}$$

$$w_{i+1} = w_i + v_{i+1} \tag{42}$$

Application	Description
Ripping production	Predicting ripping production, used as an alternative to blasting for ground loosening and breaking in mining and civil engineering, from experimental collected input such as weather zone, joint spacing, point load strength index and sonic velocity (Mohamad et al., 2017)
Crime	Identifying glass type used in crime scene based on chemical oxide content such as sodium, potassium, calcium, iron and many others (Ding et al., 2011)
Flowers species	Classifying the flowers into different species from available information on the width and length of petals and sepals (Ding et al., 2011)
River width	Estimating the width of a river, to minimize erosion and deposition, from fluid discharge rate, bed sediments of the river, shields parameter and many other (Shaghaghi et al., 2017)
Beverages	Identifying the type of beverages in term of its physical and chemical characteristics (Ding et al., 2011)
Silhouette vehicle images classification	Classifying image into different types of vehicle based on the feature extracted from the silhouette (Ding et al., 2011)

Table V. Applications of metaheuristic search algorithms

where r is a random number with value [0,1] and c is acceleration constant. The more robust generalization may be achieved by balancing global search and local search, PSO is modified to *Adaptive PSO (APSO)* by multiplying the current velocity with inertia weight $w^{interia}$ such that (Shi and Eberhart, 1998):

$$v_{i+1} = w^{interia} * v_i + c_1 * r * \left(p_i^{pbest} - w_i\right) + c_2 * r * \left(p_i^{gbest} - w_i\right) \quad (43)$$

Zhang et al. (2007) argue that PSO converge faster in global search but around global optimum, it becomes slow, whereas, BP converges faster at global optimum due to the property of efficient local search. The faster property of APSO of global search at initial stages of learning and BP of local search motives to use a hybrid approach known as particle swarm optimization backpropagation (PSO-BP) to train hyperparameters of FNNs. Experimental results indicate that PSO-BP generalization performance and learning speed are better than both PSOFNN and BPFNN. The critical point in PSO-BP is to decide when to shift learning from PSO to BP during learning. This can be done by a heuristic approach that if the particle has not changed from many iterations than the learning should be shifted to gradient BP. One of the difficulties associated with PSO algorithms is a selection of optimal hyperparameters such as r, c and $w^{interia}$. The performance of the PSO algorithm is highly influenced on those hyperparameters adjustment and is currently investigating by many researchers. Shaghaghi et al. (2017) detailed comparative analysis of GMBH neural network optimized by GA and PSO concluded that GA is more efficient than PSO. However, in literature, the performance advantage of GAFNN over PSOFNN and vice versa are still unclear.

4.4.3 Whale Optimization Algorithm

Although the application of GAs and APSO are investigated widely in literature, however, according to no free lunch (NFL) theorem, there is no algorithm that is superior for all optimization problems. Theorem of the NFL is the motivation for proposing *whale optimization algorithm (WOA)* to solve the problem of local minima, slow convergence, and dependency on hyperparameters. The WOA is inspired by the bubble net hunting strategy of humpback whales (Mirjalili and Lewis, 2016). The idea of humpback whale hunting is incorporated in WOA, which creates a trap by bubbles and moving in a spiral way around the pray. It can be expressed mathematically as:

$$w_{i+1} = \begin{cases} w_i^* - AD & p < 0.5 \\ D'^{e^{bl}} \cos(2\pi i) + w_i^* & p \geq 0.5 \end{cases} \quad (44)$$

Where $A = 2ar - a$, $C = 2r$ such that a is linearly decreased from 2 to 0 and r is a random number in [0,1], b is constant and represent the shape of a spiral, and l is a random number in [-1,1]. $D' = |w_i^* - w_i|$ is the distance between the best solution obtained so far and current position and $D = |Cw_i^* - w_i|$. The first part in the above equation represents the encircling mechanism and the second part represents the bubble net technique. Like GAs and APSO, the search process starts with candidate solutions and then improve it until defined criteria is achieved. In WOA search agents are assigned and their fitness function is calculated. The position of the search agent is updated by computing Equation (44). The process repeats until the maximum number of iterations are achieved. The experimental results demonstrate that WOA outperforms compared to other known metaheuristics.

Ling et al. (2017) mentioned that WOA due to its better performance compared to other known metaheuristics has not only gain the attention of researchers but also been increasingly applied in many areas such as neural networks and optimization. However, it has demerits of slow convergence, low precision and trapped at a local minimum because of lack of population diversity. They argue that WOA is more useful for solving low dimensional problems, however, when dealing with high dimensional and multi-model problems its solution is not quite optimal. To overcome WOA drawbacks, an extension known as Lévy flight WOA was proposed to enhance the convergence by avoiding local minimum. The finding suggests that the adoption of the Lévy flight trajectory in WOA helps to obtain a better trade-off between exploration and exploitation and lead to global optimization.

4.4.4 Application of metaheuristic search algorithms

The basic purpose of metaheuristics search algorithms was to use with applications having incomplete gradient information. However, because of its successful application on many problems attracted the attention of authors. At present, the application is not limited to the incomplete gradient information but also to other problems searching for best hyperparameter to converge at a global minimum. Table V shows some of the application researchers used to demonstrate the effectiveness of metaheuristic search algorithm. Mohamad et al (2017) recommended GAFNN to be a reliable technique for solving a complex problem in the field of excavatability. Their working on first time predicting ripping production, from experimental collected input such as weather zone, joint spacing, point load strength index and sonic velocity, shows that GAFNN achieved generalization performance 0.42 times better comparative to FNN. Shaghaghi et al. (2017) applied GA optimized neural network to estimate the width of a river and found better generalization performance of 0.40 times. Besides, it was reported that neural network optimized by GA is more efficient than PSO. Ding et al. (2011) work on predicting flower species, beverages type, identifying object used in crime and classifying images summarize that hybrid approach of GA and BP achieve prediction accuracy an average 2%-3% better than GA and BP. The study further explained that accuracy of BP for above problems is better than GA. Furthermore, it was concluded that GA needs more

learning time compared to BP and this learning speed can be increased to some extent by using hybrid approach. The hybrid approach achieved learning speed 0.05 times better than GA but still slower than BP.

5. Discussion of future research directions

The classification of learning algorithms and optimization techniques into six categories provide insight that the trend in FNNs research work is changing with time. In a broad perspective, the categories can be considered large research gaps and further improvement may bring a significant contribution. Based on extensive review and discussion on the optimization algorithms in the current paper, Part II, the authors recommend further three new future research directions to add knowledge to the existing literature:

5.1Data structure

Future research may include designing FNN architecture with less complexity with characteristics of learning noisy data without overfitting. Little attempt has been made to study the effect of data size and features on FNN algorithms. The availability of training data in many applications (for instance, medical sciences and so on) is limited (small data rather than big data) and costly. Once trained, the same model may become unstable for lesser or greater data within the same application and may loss generalization ability. Every time, for the same application area and algorithm, a new model is needed to be trained with a different set of parameters. Future research on designing an algorithm that can approximate problem task equally regardless of data size (instance) and shape (features) will be a breakthrough achievement.

5.2Eliminating the need for data pre-processing and transformation

The better results of machine learning algorithms are highly dependent on data pre-processing and transformation steps. In pre-processing, various algorithms/techniques are adapted to clean data to reduce noise, outliers, and missing values, whereas, in transformation, various algorithms/techniques are adapted to transform the data into formats and forms by encoding or standardization, that are appropriate for machine learning. Both steps are performed prior to feeding data into FNN and are adopted by many algorithms. Insufficient knowledge or inappropriate application of pre-processing and transformation techniques may cause the algorithm to wrongly conclude the findings. Future algorithms may be designed in that are lesser sensitive to noise, outliers, missing values, does not need a specific form of reducing data features magnitude to a common scale.

5.3A hybrid approach for real-world applications

Researchers demonstrate the effectiveness of the proposed algorithm either on benchmarking or on real-world problems or a combination of both. The study of all six categories applications gives a clear indication that the researcher's interest in studying their proposed algorithms on real-world problems is rapidly increasing. Traditionally, the year 2000 and before, researchers experimental work for demonstrating the effectiveness of proposed algorithms were limited to artificial benchmarking problems. The possible reasons might be unavailability of database sources and less user preference of using FNN. In the literature survey, it is noticed that

nowadays researchers most often use real-world data to ensure consistency and compare results with other popular published known algorithms. This may become an issue of causing specific data as a benchmark. The same issue has been observed in all categories and especially in the second category. It is important to avoid causing specific data to become a benchmark and even which may become unsuitable for practical application in the future with the passage of time. The role of high dimensional, nonlinear, noisy and unbalance big data is changing at a rapid rate. The best practice may be to use a hybrid approach of well-known data in the field along with new data applications during performing algorithms comparative study. This may increase and maintain users' interest in FNN over the passage of time.

5.4Hidden units' analytical calculation

The successful application of learning and regularization algorithms on the complex high dimensional big data application areas, involving more than 50,000 instances, to improve the convergence of deep neural networks is noteworthy. The maximum error reduction property of hidden units in deep neural networks is dependent on connection weights and an activation function determination. More research work needs to focus on hidden units' analytical calculation to avoid the trial and error approach. Calculating optimal number of hidden units and connection weights for single or multiple hidden layers in the network such that no linear dependence exist may help in achieving better and stable generalization performance. Similarly, future research work may give clear direction to users about the application of different activation functions for enhancing business decisions.

6. Conclusions

Feedforward Neural Network (FNN) is gaining much popularity because of its ability to solve complex high dimensional nonlinear problems more accurately. The growing and varying nature of big data demand to design and propose compact and efficient learning algorithms and optimization techniques in the area of the FNN. A comprehensive literature review was carried out to understand the reason causing FNNs drawbacks and effort was made to answers the question by studying learning algorithms and optimization techniques merits, technical limitations, and applications. The selective database was searched with popular keywords and the screened articles were classified into six broad categories: 1) Gradient learning algorithms for network training, 2) Gradient free learning algorithms, 3) Optimization algorithms for learning rate, 4) Bias and variance (underfitting and overfitting) minimization algorithms, 5) Constructive topology Feedforward Neural Network, and 6) Metaheuristic search algorithms. Reviewing and discussing all the six categories in one paper is too long in length. Therefore, the authors further divided the six categories into two parts (i.e., Part I and Part II). Part I reviewed mainly two categories explaining learning algorithms and Part II reviewed the remaining four categories greatly explaining optimization techniques. The main conclusion of the six categories is:

1. First order gradient learning algorithms may be more time consuming compared to the second order gradient. The increase in a number of iterations for the first-order gradient may stuck the network at the local minimum. The second order algorithms are considered to more efficient than first order with the additional need for more computational memory to store a Hessian matrix. However, the second order algorithm maybe not

suitable for all types of application. For instance, the Levenberg-Marquardt Algorithm (LM) gradient algorithm is considered faster than gradient descent (GD) with limitation in that it can only be applied with least square loss function and fixed topology FNN. The wide range applications of gradient learning algorithm to facilitate business intelligence in improving business decisions is explored in Part I.

2. The use of gradient information may make the task difficult to solve. Lot of experimental work may be needed to search for an optimal network which may be more time-consuming. This can be made better by randomly generating hidden units and analytically calculate connection weights using gradient free learning algorithms. Gradient learning algorithms can be avoided, and connection weights can be calculated more analytically by gradient free learning algorithms. The better learning speed and generalization performance (in most cases) of gradient free learning algorithms are evident. However, the network complexity may increase because of an increase in a number of hidden units compared to gradient learning algorithms compact network size which may increase chances of overfitting. The wide range applications of gradient free learning algorithm to facilitate business intelligence in improving business decisions is explored in Part I.
3. Finding a minimum of the function by taking steps proportional to the negative of the gradient is influenced by the learning rate. Various proposed algorithms work in that it increases convergence by assigning a higher learning rate at initial stages of training and lower learning rate when a minimum of the function is nearer. Traditional, learning rate was fixed to a certain value based on user expertise which encounters many difficulties. Keeping the learning rate at a lower value is better to approach downward slope and reaches a minimum of a function, but it can also mean to converge very slowly and may stuck in the plateau region. Whereas, a large learning rate will oscillate the gradient along the long and short axis of minimum error valley. This may reduce gradient movement towards the long axis which faces towards the minimum. The learning optimization algorithms minimize the short axis of oscillation while add more contributions along the long axis to move gradient in larger steps towards the minimum. The category range of application include problems having a high dimension complex data. The category shows an improvement of 2.09% to 109% in prediction accuracy by adjusting the learning rate to make the network stable.
4. The regularities in training examples and the generalization performance of unseen data can be improved by bias and variance trade-off. Underfitting the network will result in higher bias and low variance in that it will not discover regularities in training data, whereas, overfitting the network will result in low bias and high variance between training dataset and testing dataset. Overfitting decreases the generalization performance because of network complexity (the number of connection weights) increases compared to the training examples. Therefore, it is often desirable to make the network simple by keeping a number of weight less and limiting their growth by suitable weight decay algorithms. Other algorithms to make the network simpler includes limiting the number of hidden units or normalize the input distribution to hidden units, pruning, and ensembles. The major application includes the area such as classification, recognition, segmentation, and forecasting. The algorithms achieved improvement in accuracy 0.02%-5% for predicting high dimensional featured data having instance more than 50,000. The limitation of this category is that it requires more

learning time compared to standard networks for convergence. The possible solution is to use a hybrid approach of learning optimization algorithms and this category to minimize limitation.

5. The issue with fixed topology FNN is that it needs many experimental trials to determine the number of hidden units in the hidden layer. For deep architecture, the experimental trial is more time consuming and difficult to determine the ideal combination of hidden units in multiple hidden layers. The constructive topology FNN algorithms advantages over fixed topology in that it determines network topology by adding hidden units' step by step in the hidden layers. The category was able to improve regression performance 0.5-4 times and classification prediction accuracy 5%-8.61% with an average learning speed 20 times faster in solving various real-world problems.
6. The unavailability of gradient information, the problem of local minimum, and finding optimal hyperparameters make FNN hopeless. The metaheuristic global optimization algorithms in combination with FNN can overcome such problems. It assists to search for a global minimum of the loss function with best hyperparameters but with an additional need for memory and learning time. The metaheuristic algorithms application on regression problems finds a better performance of 0.40 times, whereas, for application on classification problems the prediction accuracy is enhanced by 2%-3%. Mostly, the metaheuristic algorithms learning speed is less than traditional FNN, however, the learning speed can be improved to some extent by using the hybrid approach of metaheuristic and gradient learning algorithms for training FNN.

The categories explain the noteworthy contribution made by researchers in improving FNNs generalization performance and learning speed. The study shows a major change in research trend and future research survey may include adding knowledge to the existing categories or identifying a new category by studying further algorithms. The successful application of FNN learning algorithms and optimization techniques on real-world management, engineering, and health sciences problems demonstrate the advantages of algorithms in enhancing decision making for practical operations. Based on existing literature and research trend, the authors suggested in total six research direction (Part I and II) that may contribute to enhancing existing literature. The three future research direction suggested in Part I was studying the role of various activation functions, designing efficient and compact algorithm with fewer hyperparameters, and determining optimal connection weights. The new four future research direction proposed in current paper (Part II) is to design algorithms with the ability to approximate any problem task equally regardless of data size (instance) and shape (features), eliminating the need of data pre-processing and transformation, using hybrid approach of well-known data and new data to demonstrate the effectiveness of algorithms, and analytical calculating hidden units.

References

Abdel-Hamid, O., Mohamed, A., Jiang, H., Deng, L., Penn, G. and Yu, D. (2014), "Convolutional neural networks for speech recognition", *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 22 No. 10, pp. 1533–1545.

Babaee, M., Dinh, D.T. and Rigoll, G. (2018), "A deep convolutional neural network for video sequence

background subtraction", *Pattern Recognition*, Elsevier Ltd, Vol. 76, pp. 635–649.

Bianchini, M. and Scarselli, F. (2014), "On the complexity of neural network classifiers: A comparison between shallow and deep architectures", *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, Vol. 25 No. 8, pp. 1553–1565.

Bottani, E., Centobelli, P., Gallo, M., Kaviani, M.A., Jain, V. and Murino, T. (2019), "Modelling wholesale distribution operations: an artificial intelligence framework", *Industrial Management & Data Systems*, Emerald Publishing Limited, Vol. 119 No. 4, pp. 698–718.

Cao, F., Wang, D., Zhu, H. and Wang, Y. (2016), "An iterative learning algorithm for feedforward neural networks with random weights", *Information Sciences*, Elsevier Inc., Vol. 328, pp. 546–557.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K. and Yuille, A.L. (2018), "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, Vol. 40 No. 4, pp. 834–848.

Chung, S.H., Ma, H.L. and Chan, H.K. (2017), "Cascading delay risk of airline workforce deployments with crew pairing and schedule optimization", *Risk Analysis*, Wiley Online Library, Vol. 37 No. 8, pp. 1443–1458.

Deng, C., Miao, J., Ma, Y., Wei, B. and Feng, Y. (2019), "Reliability analysis of chatter stability for milling process system with uncertainties based on neural network and fourth moment method", *International Journal of Production Research*, Taylor & Francis, pp. 1–19.

Ding, S., Su, C. and Yu, J. (2011), "An optimizing BP neural network algorithm based on genetic algorithm", *Artificial Intelligence Review*, Vol. 36 No. 2, pp. 153–162.

Dong, C., Loy, C.C., He, K. and Tang, X. (2016), "Image super-resolution Using deep convolutional networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 38 No. 2, pp. 295–307.

Duchi, J., Hazan, E. and Singer, Y. (2011), "Adaptive subgradient methods for online learning and stochastic optimization", *Journal of Machine Learning Research*, Vol. 12, pp. 2121–2159.

Eberhart, R. and Kennedy, J. (1995), "A new optimizer using particle swarm theory", *Proceedings of the Sixth International Symposium on Micro Machine and Human Science MHS'95.*, pp. 39–43.

Fahlman, S.E. (1988), *An Empirical Study of Learning Speed in Back-Propagation Networks*, School of Computer Science, Carnegie Mellon University, Pittsburgh PA 15213.

Fahlman, S.E. and Lebiere, C. (1990), "The cascade-correlation learning architecture", *Advances in Neural Information Processing Systems*, pp. 524–532.

Farlow, S.J. (1981), "The GMDH algorithm of Ivakhnenko", *The American Statistician*, Vol. 35 No. 4, pp. 210–

215.

Feng, G., Huang, G.-B., Lin, Q. and Gay, R.K.L. (2009), "Error minimized extreme learning machine with growth of hidden nodes and incremental learning", *IEEE Trans. Neural Networks*, Vol. 20 No. 8, pp. 1352–1357.

Ferrari, S. and Stengel, R.F. (2005), "Smooth function approximation using neural networks", *IEEE Transactions on Neural Networks*, Vol. 16 No. 1, pp. 24–38.

Geman, S., Doursat, R. and Bienenstock, E. (1992), "Neural networks and the bias/variance dilemma", *Neural Computation*, Vol. 4 No. 1, pp. 1–58.

Gori, M. and Tesi, A. (1992), "On the problem of local minima in backpropagation", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, IEEE, No. 1, pp. 76–86.

Hagan, M.T. and Menhaj, M.B. (1994), "Training feedforward networks with the Marquardt algorithm", *IEEE Transactions on Neural Networks*, Vol. 5 No. 6, pp. 989–993.

Han, F., Zhao, M.-R., Zhang, J.-M. and Ling, Q.-H. (2017), "An improved incremental constructive single-hidden-layer feedforward networks for extreme learning machine based on particle swarm optimization", *Neurocomputing*, Elsevier, Vol. 228, pp. 133–142.

Han, S., Pool, J., Tran, J. and Dally, W.J. (2015), "Learning both weights and connections for efficient neural networks", *Advances in Neural Information Processing Systems*, pp. 1135–1143.

Hansen, L.K. and Salamon, P. (1990), "Neural network ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12 No. 10, pp. 993–1001.

Hayashi, Y., Hsieh, M.-H. and Setiono, R. (2010), "Understanding consumer heterogeneity: A business intelligence application of neural networks", *Knowledge-Based Systems*, Elsevier, Vol. 23 No. 8, pp. 856–863.

Hecht-Nielsen. (1989), "Theory of the backpropagation neural network", *International Joint Conference on Neural Networks*, Vol. 1, IEEE, pp. 593–605 vol.1.

Hinton, G.E., Srivastava, N. and Swersky, K. (2012), "Lecture 6a- overview of mini-batch gradient descent", *COURSERA: Neural Networks for Machine Learning*.

Huang, G.-B. and Chen, L. (2007), "Convex incremental extreme learning machine", *Neurocomputing*, Elsevier, Vol. 70 No. 16–18, pp. 3056–3062.

Huang, G.-B. and Chen, L. (2008), "Enhanced random search based incremental extreme learning machine", *Neurocomputing*, Vol. 71 No. 16–18, pp. 3460–3468.

Huang, G.-B., Chen, L. and Siew, C.K. (2006), "Universal approximation using incremental constructive

feedforward networks with random hidden nodes", *IEEE Transactions on Neural Networks*, Vol. 17 No. 4, pp. 879–892.

Huang, G.-B., Zhou, H., Ding, X. and Zhang, R. (2012), "Extreme learning machine for regression and multiclass classification", *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics*, Vol. 42 No. 2, pp. 513–29.

Huang, G.-B., Zhu, Q.-Y. and Siew, C.-K. (2006), "Extreme learning machine: theory and applications", *Neurocomputing*, Vol. 70 No. 1–3, pp. 489–501.

Huang, G., Huang, G.-B., Song, S. and You, K. (2015), "Trends in extreme learning machines: A review", *Neural Networks*, Elsevier, Vol. 61, pp. 32–48.

Huang, G., Song, S. and Wu, C. (2012), "Orthogonal least squares algorithm for training cascade neural networks", *IEEE Transactions on Circuits and Systems I: Regular Papers*, Vol. 59 No. 11, pp. 2629–2637.

Hunter, D., Yu, H., Pukish III, M.S., Kolbusz, J. and Wilamowski, B.M. (2012), "Selection of proper neural network sizes and architectures—A comparative study", *IEEE Transactions on Industrial Informatics*, Vol. 8 No. 2, pp. 228–240.

Hwang, J.-N., You, S.-S., Lay, S.-R. and Jou, I.-C. (1996), "The cascade-correlation learning: A projection pursuit learning perspective", *IEEE Transactions on Neural Networks*, Vol. 7 No. 2, pp. 278–289.

Ijjina, E.P. and Chalavadi, K.M. (2016), "Human action recognition using genetic algorithms and convolutional neural networks", *Pattern Recognition*, Elsevier, Vol. 59, pp. 199–212.

Ioffe, S. and Szegedy, C. (2015), "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", available at:https://doi.org/10.1007/s13398-014-0173-7.2.

Islam, M., Yao, X. and Murase, K. (2003), "A constructive algorithm for training cooperative neural network ensembles", *IEEE Transactions on Neural Networks*, Vol. 14 No. 4, pp. 820–834.

Kang, G., Li, J. and Tao, D. (2017), "Shakeout: A New Approach to Regularized Deep Neural Network Training", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40 No. 5, pp. 1245–1258.

Karnin, E.D. (1990), "A simple procedure for pruning back-propagation trained neural networks", *IEEE Transactions on Neural Networks*, Vol. 1 No. 2, pp. 239–242.

Kastrati, Z., Imran, A.S. and Yayilgan, S.Y. (2019), "The impact of deep learning on document classification using semantically rich representations", *Information Processing & Management*, Elsevier, Vol. 56 No. 5, pp. 1618–1632.

Kasun, L.L.C., Zhou, H., Huang, G. and Vong, C. (2013), "Representational Learning with Extreme Learning

Machine for Big Data", *IEEE Intelligent System*, Vol. 28 No. 6, pp. 31–34.

Kim, Y.-S., Rim, H.-C. and Lee, D.-G. (2019), "Business environmental analysis for textual data using data mining and sentence-level classification", *Industrial Management & Data Systems*, Emerald Publishing Limited, Vol. 119 No. 1, pp. 69–88.

Kingma, D.P. and Ba, J. (2014), "Adam: A Method for Stochastic Optimization", pp. 1–15.

Kovalishyn, V. V, Tetko, I. V, Luik, A.I., Kholodovych, V. V, Villa, A.E.P. and Livingstone, D.J. (1998), "Neural network studies. 3. variable selection in the cascade-correlation learning architecture", *Journal of Chemical Information and Computer Sciences*, Vol. 38 No. 4, pp. 651–659.

Krogh, A. and Hertz, J.A. (1992), "A simple weight decay can improve generalization", *Advances in Neural Information Processing Systems*, Vol. 4, pp. 950–957.

Krogh, A., Krogh, A., Vedelsby, J. and Vedelsby, J. (1995), "Neural Network Ensembles, Cross Validation, and Active Learning", *Advances in Neural Information Processing Systems*, pp. 231–238.

Kumar, A., Rao, V.R. and Soni, H. (1995), "An empirical comparison of neural network and logistic regression models", *Marketing Letters*, Vol. 6 No. 4, pp. 251–263.

Kummong, R. and Supratid, S. (2016), "Thailand tourism forecasting based on a hybrid of discrete wavelet decomposition and NARX neural network", *Industrial Management and Data Systems*, Vol. 116 No. 6, pp. 1242–1258.

Kwok, T.-Y. and Yeung, D.-Y. (1997), "Constructive algorithms for structure learning in feedforward neural networks for regression problems", *IEEE Transactions on Neural Networks*, Vol. 8 No. 3, pp. 630–645.

Lam, H.Y., Ho, G.T.S., Wu, C.-H. and Choy, K.L. (2014), "Customer relationship mining system for effective strategies formulation", *Industrial Management & Data Systems*, Emerald Group Publishing Limited, Vol. 114 No. 5, pp. 711–733.

Lang, K.J. (1989), "Learning to tell two spirals apart", *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, pp. 52–59.

LeCun, Y., Bengio, Y. and Hinton, G. (2015), "Deep learning", *Nature*, Nature Publishing Group, Vol. 521 No. 7553, pp. 436–444.

Lehtokangas, M. (2000), "Modified cascade-correlation learning for classification", *IEEE Transactions on Neural Networks*, Vol. 11 No. 3, pp. 795–798.

Lewis, A.S. and Overton, M.L. (2013), "Nonsmooth optimization via quasi-Newton methods", *Mathematical Programming*, Vol. 141 No. 1–2, pp. 135–163.

Li, M., Ch'ng, E., Chong, A.Y.L. and See, S. (2018), "Multi-class Twitter sentiment classification with emojis",

Industrial Management & Data Systems, Emerald Publishing Limited, Vol. 118 No. 9, pp. 1804–1820.

Liang, H. and Dai, G. (1998), "Improvement of cascade correlation learning", *Information Sciences*, Vol. 112 No. 1–4, pp. 1–6.

Liang, N.-Y., Huang, G.-B., Saratchandran, P. and Sundararajan, N. (2006), "A fast and accurate online sequential learning algorithm for feedforward networks", *IEEE Transactions on Neural Networks*, Vol. 17 No. 6, pp. 1411–1423.

Ling, Y., Zhou, Y. and Luo, Q. (2017), "Lévy Flight Trajectory-Based Whale Optimization Algorithm for Global Optimization.", *IEEE Access*, Vol. 5 No. 99, pp. 6168–6186.

Lucas, J.M. and Saccucci, M.S. (1990), "Exponentially Weighted Moving Average Control Schemes: Properties and Enhancements", *Technometrics*, IEEE, Vol. 32 No. 1, pp. 1–12.

Mirjalili, S. and Lewis, A. (2016), "The Whale Optimization algorithm", *Advances in Engineering Software*, Elsevier Ltd, Vol. 95, pp. 51–67.

Mitchell, M. (1998), "Genetic Algorithms: An Overview", *An Introduction to Genetic Algorithms*, MIT press, London, England, pp. 1–15.

Mohamad, E.T., Faradonbeh, R.S., Armaghani, D.J., Monjezi, M. and Majid, M.Z.A. (2017), "An optimized ANN model based on genetic algorithm for predicting ripping production", *Neural Computing and Applications*, Springer, Vol. 28 No. 1, pp. 393–406.

Mohamed Shakeel, P., Tobely, T.E. El, Al-Feel, H., Manogaran, G. and Baskar, S. (2019), "Neural Network Based Brain Tumor Detection Using Wireless Infrared Imaging Sensor", *IEEE Access*, IEEE, Vol. 7, pp. 5577–5588.

Mori, J., Kajikawa, Y., Kashima, H. and Sakata, I. (2012), "Machine learning approach for finding business partners and building reciprocal relationships", *Expert Systems with Applications*, Elsevier, Vol. 39 No. 12, pp. 10402–10407.

Nasir, M., South-Winter, C., Ragothaman, S. and Dag, A. (2019), "A comparative data analytic approach to construct a risk trade-off for cardiac patients' re-admissions", *Industrial Management & Data Systems*, Emerald Publishing Limited, Vol. 119 No. 1, pp. 189–209.

Nayyeri, M., Yazdi, H.S., Maskooki, A. and Rouhani, M. (2018), "Universal Approximation by Using the Correntropy Objective Function", *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, Vol. 29 No. 9, pp. 4515–4521.

Qian, N. (1999), "On the momentum term in gradient descent learning algorithms", *Neural Networks*, Vol. 12 No. 1, pp. 145–151.

Qiao, J., Li, F., Han, H. and Li, W. (2016), "Constructive algorithm for fully connected cascade feedforward neural networks", *Neurocomputing*, Elsevier, Vol. 182, pp. 154–164.

Reed, R. (1993), "Pruning algorithms-a survey", *IEEE Transactions on Neural Networks*, Vol. 4 No. 5, pp. 740–747.

Riedmiller, M. and Braun, H. (1993), "A direct adaptive method for faster backpropagation learning: the RPROP algorithm", *IEEE International Conference on Neural Networks*, Vol. 1993-Janua, IEEE, pp. 586–591.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986), "Learning representations by back-propagating errors", *Nature*, Vol. 323 No. 6088, pp. 533–536.

Schetinin, V. (2003), "A learning algorithm for evolving cascade neural networks", *Neural Processing Letters*, Springer, Vol. 17 No. 1, pp. 21–31.

Seni, G. and Elder, J. (2010), "Model Complexity, Model Selection and Regularization", *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*, Morgan & Claypool Publishers, pp. 21–40.

Setiono, R. and Hui, L.C.K. (1995), "Use of a quasi-Newton method in a feedforward neural network construction algorithm", *IEEE Transactions on Neural Networks*, Vol. 6 No. 1, pp. 273–277.

Shaghaghi, S., Bonakdari, H., Gholami, A., Ebtehaj, I. and Zeinolabedini, M. (2017), "Comparative analysis of GMDH neural network based on genetic algorithm and particle swarm optimization in stable channel design", *Applied Mathematics and Computation*, Elsevier, Vol. 313, pp. 271–286.

Shanno, D.F. (1970), "Conditioning of quasi-Newton methods for function minimization", *Mathematics of Computation*, Vol. 24 No. 111, pp. 647–647.

Shi, Y. and Eberhart, R. (1998), "A modified particle swarm optimizer", *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, IEEE, pp. 69–73.

Specht, D.F. (1990), "Probabilistic neural networks", *Neural Networks*, Vol. 3 No. 1, pp. 109–118.

Specht, D.F. (1991), "A general regression neural network", *IEEE Transactions on Neural Networks*, Vol. 2 No. 6, pp. 568–576.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), "Dropout: A simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, Vol. 15 No. 1, pp. 1929–1958.

Tang, J., Deng, C. and Huang, G.-B. (2016), "Extreme Learning Machine for multilayer perceptron", *IEEE*

Transactions on Neural Networks and Learning Systems, Vol. 27 No. 4, pp. 809–821.

Teo, A.-C., Tan, G.W.-H., Ooi, K.-B., Hew, T.-S. and Yew, K.-T. (2015), "The effects of convenience and speed in m-payment", *Industrial Management & Data Systems*, Vol. 115 No. 2, pp. 311–331.

Tkáč, M. and Verner, R. (2016), "Artificial neural networks in business: Two decades of research", *Applied Soft Computing*, Vol. 38, pp. 788–804.

Tu, J. V. (1996), "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes", *Journal of Clinical Epidemiology*, Vol. 49 No. 11, pp. 1225–1231.

Wan, L., Zeiler, M., Zhang, S., LeCun, Y. and Fergus, R. (2013), "Regularization of neural networks using dropconnect", *International Conference on Machine Learning*, pp. 109–111.

Wang, G.-G., Lu, M., Dong, Y.-Q. and Zhao, X.-J. (2016), "Self-adaptive extreme learning machine", *Neural Computing and Applications*, Springer, Vol. 27 No. 2, pp. 291–303.

Wang, J., Wu, X. and Zhang, C. (2005), "Support vector machines based on K-means clustering for real-time business intelligence systems", *International Journal of Business Intelligence and Data Mining*, Citeseer, Vol. 1 No. 1, pp. 54–64.

Wang, L., Yang, Y., Min, R. and Chakradhar, S. (2017), "Accelerating deep neural network training with inconsistent stochastic gradient descent", *Neural Networks*, Elsevier, Vol. 93, pp. 219–229.

Widrow, B., Greenblatt, A., Kim, Y. and Park, D. (2013), "The No-Prop algorithm: A new learning algorithm for multilayer neural networks", *Neural Networks*, Elsevier Ltd, Vol. 37, pp. 182–188.

Wilamowski, B.M., Cotton, N.J., Kaynak, O. and Dundar, G. (2008), "Computing gradient vector and Jacobian matrix in arbitrarily connected neural networks", *IEEE Transactions on Industrial Electronics*, Vol. 55 No. 10, pp. 3784–3790.

Wilamowski, B.M. and Yu, H. (2010), "Neural network learning without backpropagation", *IEEE Transactions on Neural Networks*, Vol. 21 No. 11, pp. 1793–1803.

Wilson, D.R. and Martinez, T.R. (2003), "The general inefficiency of batch training for gradient descent learning", *Neural Networks*, Elsevier, Vol. 16 No. 10, pp. 1429–1451.

Wong, T.C., Haddoud, M.Y., Kwok, Y.K. and He, H. (2018), "Examining the key determinants towards online pro-brand and anti-brand community citizenship behaviours: a two-stage approach", *Industrial Management & Data Systems*, Emerald Publishing Limited, Vol. 118 No. 4, pp. 850–872.

Yang, Y., Wang, Y. and Yuan, X. (2012), "Bidirectional extreme learning machine for regression problem and its learning effectiveness", *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, Vol. 23 No. 9, pp. 1498–1505.

Yin, X. and Liu, X. (2018), “Multi-Task Convolutional Neural Network for Pose-Invariant Face Recognition”, *IEEE Transactions on Image Processing*, Vol. 27 No. 2, pp. 964–975.

Ying, L. (2016), “Orthogonal incremental extreme learning machine for regression and multiclass classification”, *Neural Computing and Applications*, Springer, Vol. 27 No. 1, pp. 111–120.

Yinyin Liu, Starzyk, J.A. and Zhen Zhu. (2008), “Optimized Approximation Algorithm in Neural Networks Without Overfitting”, *IEEE Transactions on Neural Networks*, Vol. 19 No. 6, pp. 983–995.

Ypma, T.J. (1995), “Historical Development of the Newton–Raphson Method”, *SIAM Review*, Society for Industrial and Applied Mathematics, Vol. 37 No. 4, pp. 531–551.

Zaghloul, W., Lee, S.M. and Trimi, S. (2009), “Text classification: neural networks vs support vector machines”, *Industrial Management & Data Systems*, Vol. 109 No. 5, pp. 708–717.

Zeiler, M.D. (2012), “ADADELTA: An Adaptive Learning Rate Method”, available at:https://doi.org/http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503.

Zhang, J.R., Zhang, J., Lok, T.M. and Lyu, M.R. (2007), “A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training”, *Applied Mathematics and Computation*, Vol. 185 No. 2, pp. 1026–1037.

Zong, W., Huang, G.-B. and Chen, Y. (2013), “Weighted extreme learning machine for imbalance learning”, *Neurocomputing*, Elsevier, Vol. 101, pp. 229–242.

Zou, W., Xia, Y. and Li, H. (2018), “Fault Diagnosis of Tennessee-Eastman Process Using Orthogonal Incremental Extreme Learning Machine Based on Driving Amount”, *IEEE Transactions on Cybernetics*, IEEE, Vol. 48 No. 12, pp. 3403–3410.