Cascade neural network algorithm with analytical connection weights determination for modelling operations and energy applications

Zhengxu Wang[1#], Waqar Ahmed Khan[2*#], Hoi-Lam Ma[3] & Xin Wen[2]

The performance and learning speed of the Cascade Correlation neural network (CasCor) may not be optimal because of redundant hidden units' in the cascade architecture and the tuning of connection weights. This study explores the limitations of CasCor and its variants and proposes a novel constructive neural network (CNN). The basic idea is to compute the input connection weights by generating linearly independent hidden units from the orthogonal linear transformation, and the output connection weights by connecting hidden units in a linear relationship to the output units. The work is unique in that few attempts have been made to analytically determine the connection weights on both sides of the network. Experimental work on real energy application problems such as predicting powerplant electrical energy, predicting seismic hazards to prevent fatal accidents and reducing energy consumption by predicting building occupancy detection shows that analytically calculating the connection weights and generating non-redundant hidden units improves the convergence of the network. The proposed CNN is compared with that of the state-of-the-art machine learning algorithms. The work demonstrates that proposed CNN predicts a wide range of applications better than other methods.

Keywords: **energy management; forecasting; machine learning; neural networks; sustainability**

1. Introduction

The Cascade Correlation learning algorithm (CasCor) has been extensively applied in

[1] School of Business Administration, Institute of Supply Chain Analytics, Dongbei University of Finance and Economics, Dalian, People's Republic of China.

[2] Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Hong Kong.

[3] Department of Supply Chain and Information Management, The Hang Seng University of Hong Kong, Shatin, Hong Kong.

[*] Correponding author.

[#] Zhengxu Wang and Waqar Ahmed Khan contributed equally to this work.

many application areas (Heidari et al. 2018; Chung, Ma, and Chan 2017) due to its self-organizing neural network property, and in many cases, it is considered to be more powerful than the standard multilayer perceptron (Qiao et al. 2016; Hunter et al. 2012). The selection of a neural network (NN) depends upon the application area (Wang et al. 2018; Deng et al. 2019) to achieve better and faster convergence. Learning NNs by gradient algorithms along with too many hyperparameters may make the network more complex, causing the generalization performance to converge at a suboptimal solution (Liew, Khalil-Hani, and Bakhteri 2016; Kapanova, Dimov, and Sellier 2018). Backpropagation (BP) gradient descent is a well-known learning algorithm for NNs (Rumelhart, Hinton, and Williams 1986), but it faces the problem of local minima if the global minima is far away, and the learning speed is highly influenced by gradient iteration and the learning rate hyperparameter (Hecht-Nielsen 1989). To address the backpropagation neural network (BPNN) slowness and topology problem, the self-organizing quick prop (QP) CasCor was formulated (Fahlman and Lebiere 1990).

The QP can reduce the error of CasCor to a small value, but it does not guarantee that the network performance will be satisfactory (Hwarng 2005; Hunter et al. 2012) due to its chaotic behaviour and numerical instability. QP during weight updating takes much larger steps based on previous and current gradients to moves faster towards the minimum of the function (Fahlman 1988). The current gradient may be larger or smaller and in the same or opposite direction to the previous gradient. The larger and opposite gradient will cause the algorithm to cross the minimum of the function and needs to be brought back. This may cause the QP to behave chaotically across the minimum valley of the function. Banerjee et al. (2011) explained that QP becomes numerically unstable if the current gradient is very close or equal to the previous gradient. If the difference between current and previous gradient becomes zero, the weight difference will also become zero and the QP formula will remain zero permanently, even if the gradient changes.

Due to its widespread popularity and the recent increase in interest for self-organizing neural networks (Khan et al. 2019a, 2019b), researchers are extensively focused on improving the existing CasCor. Huang, Song, and Wu (2012) proposed an orthogonal least squares algorithm for training cascade neural networks (OLSCN) by explaining that a larger network size causes lowering the generalization performance of CasCor. Besides, the covariance objective function efforts to adjust the input connection weights cannot assure maximum error reduction on the addition of a new hidden unit. The repeatedly tuning of connection weights, before and after hidden unit generation, causes the network to be more time-consuming. However, Qiao et al. (2016) explained that the new objective function formulated along with the modified Newton method by OLSCN may make mistakes during linear dependencies among variables and results in local minimum with slow convergence. A Faster Cascade Neural Network (FCNN) was proposed to address the CasCor and OLSCN generalization and convergence issues. FCNN selects linearly independent input units one by one by the Gram-Schmidt Orthogonalization method and candidate units by the modified index (MI) formulated objective function. It assures that the selected candidate unit (hidden unit) may have the largest contribution in the existing candidate pools but cannot guarantee that the next expected candidate unit (hidden unit) error reduction will be maximized. For the sake of simplicity, in this paper CasCor, OLSCN and FCNN are referred to as CasCor and its variants because of similar network structure, unless specified.

This paper proposes a novel Cascade Principal Component Least Squares Neural Network Learning Algorithm (CPCLS) to address the convergence limitations of CasCor and its variants. The main contributions are listed below:

- The linear dependence among input units and/or hidden units can be avoided by transforming a set of correlated units orthogonally into linearly independent units. The cascade architecture can be made better by connecting hidden units (or layers) to the output units that may have no linear dependence with each other. Similarly, the input unit’s direct linear connection to the output units can be avoided to get rid of the input unit’s dependency.
- The best least-squares solution can be achieved by connecting only newly added linearly independent (no multicollinearity) hidden layer to the output units and eliminating previous output connections (hidden units).
- Multiple hidden units can be generated in the hidden layer to make the convergence faster.

The advancement in information technology has enabled industries to create a model of products and processes from high dimensional data to benefit production research (Kusiak 2020). Traditional models based on mathematical formulations and physical approaches advantageous to provide a physical understanding of the system. However, in real practices, mathematical models may be inaccurate and difficult to adopt because of ignoring nonlinearities (Wang et al. 2019), unable to understand symbolic data, need of prior expert knowledge, and maybe not well suited to represent relationships among variables (Kuo and Kusiak 2019).

Nowadays, the availability of high dimensional data has made it possible to extract useful information, rather than physical measurement or manual work that may cause subjective judgment or fatigues (Kim et al. 2019), to facilitate in making real-time decisions, time and cost-saving (Q. Liu et al. 2019). It is considered that the application of machine learning compared to mathematical modelling is likely to be more beneficial in improving production research (Kusiak 2020; Kuo and Kusiak 2019; Lv et al. 2020; Y. Liu et al. 2019). The machine learning that has gained significant interest in the literature include NNs and its variants (Kumar, Singh, and Singh 2020; Ertuğrul 2018; Bansal et al. 2019; Zou et al. 2018; Lorencin et al. 2019; Grasso, Luchetta, and Manetti 2018; Nayyeri et al. 2018), support vector machine (SVM) (Bansal et al. 2019), decision tree (DT) (Mantas et al. 2019; Bansal et al. 2019; Candanedo and Feldheim 2016), naïve Bayes (NB) (Bansal et al. 2019), metaheuristics search algorithms and its variants (Bansal et al. 2019; Aljarah, Faris, and Mirjalili 2018), random forest (RF) (Mantas et al. 2019; Candanedo and Feldheim 2016), ensembles (Mantas et al. 2019), gradient boosting machine (Candanedo and Feldheim 2016), regression and its variants (Lorencin et al. 2019), and linear discriminant analysis (LDA) (Candanedo and Feldheim 2016).

Compared to other machine learning algorithms, NNs is widely adopted because of its superior performance and universal approximation ability (Wang et al. 2019). Usually, the application of NNs in production research involves learning of the connection weights by either BP or random generation with a lot of hyperparameter tuning (Chien, Lin, and Lin 2020) which makes learning complicated and challenging (Kusiak 2020; Solimanpur, Vrat, and Shankar 2004). According to the best of our knowledge, insufficient attempts have been made to improve the NNs performance and speed by analytically calculating connection weights on both sides of the network with a small number of hyperparameters initialization. The novelty of the proposed algorithm exists in its improved cascade architecture by connecting linearly independent hidden layer to the output units and analytically calculating connection weights. This may facilitate to predict a wide range of applications with less human intervention.

This work applies the proposed CPCLS algorithm and made a state-of-the-art comparison with other machine learning algorithms to predict health sciences, engineering, marine, food products, forestry, and energy application problems. Better generalization performance and faster learning speed of CPCLS give insight that NNs based model prediction capability can be made better by analytically calculating connection weights rather than BP or random generation. Moreover, in current practice, the majority of the production research is focused on solving problems belonging to a single application. This limits the proposed method, in real practice, to a single industry or single business function. The better performance of CPCLS on a wide range of applications give managerial insight that it can be practiced in general and able to handle industrial and business function problems on an integrated platform. Furthermore, the cascade architecture of CPCLS helps to eliminate the problem of "what-if" of fixed topology BPNNs for determining hidden units and layers that involves human interventions and simultaneously affect decision making. The CPCLS can facilitate in optimizing the operations by providing predictive advice and may derive the decision-making process by building greater confidence in prediction from historical data rather than mathematical formulation or manual work.

This paper is a revised and extended version of that of Khan, Chung, and Chan (2018). In this extended version, the property of maximum error reduction of the CPCLS is explained by supporting statements, lemmas, theoretical analysis, and remarks and further demonstrated by experimental work. The rest of the paper is structured as follows. In Section 2, CasCor and its variants with convergence limitations, Orthogonal linear transformation (OLT) and Ordinary Least Squares (OLS) are briefly explained. Section 3 presents the novel CPCLS. Section 4 describes the state-of-the-art comparison. Section 5 concludes the paper.

2. Existing learning methodologies

2.1. CasCor and its variants with convergence drawbacks

CasCor initializes by linearly connecting the input units to the output units and tuning randomly generated output connection weights by the QP learning algorithm. When training converges, hidden units are added one by one to discover nonlinear patterns in the problem. The candidate units are added to select the hidden unit, having the property of maximum error reduction. The candidate units receive the input connections from input units and any pre-existing hidden units. The aim is to maximize the covariance ***S*** between network error and the candidate units by the gradient ascent. When ***S*** stops improving, the candidate unit with the maximum value of the ***S*** is chosen as the hidden unit and is linked to the output units by the output connection weights, while incoming connections are kept frozen. Again, the output connection weights are trained by the QP and this procedure continues till the error converges. Figure 1 illustrates the architecture of CasCor.

Huang, Song, and Wu (2012) explained that the ***S*** objective function to maximize the correlation between the hidden unit and network error cannot assure a maximum error reduction with the addition of new hidden unit to the network. Secondly, the output training is repeatedly performed after every hidden unit generation which increases the computational burden. OLSCN was proposed to overcome the above disadvantages which lead CasCor to slow convergence and poor generalization performance. The OLSCN reformulated new objective function based on the OLS for input training which was further optimized by the second order modified Newton method. Qiao et al. (2016) supported the work of Huang, Song, and Wu (2012) and concluded that the CasCor objective function cannot guarantee a maximum error reduction and repeatedly output training can be more time-consuming. However, Qiao et al. (2016) argued that the

OLSCN may result in a local minimum, slow convergence, and a computational burden by updating the weights of the hidden units by the modified Newton method. In addition, linear independence of the input units and the hidden units are necessary for QR factorization and the newly formulated objective function, respectively. FCNN was proposed to address the generalization performance and learning speed of CasCor and OLSCN.

In Theorem 3.1 (Qiao et al. 2016) of FCNN, it is explained that one or more candidate units in the pool may be linearly independent to the input and any pre-existing hidden units. However, the column matrix of hidden units may not necessarily full rank due to the random generation of input weights. Therefore, MI was proposed to evaluate the candidate unit among the pool of candidates. The candidate unit with the maximum contribution to the sum of squares error (SSE) is added to the network which specifies linearly independence of the candidate unit, however, network optimal error minimization capability cannot be guaranteed. For instance, if among a pool of candidate units, fewer candidate units are linearly independent than the chances of getting the largest contributed MI also decreases. Secondly, the selected candidate unit (hidden unit) may have the property of maximum error reduction capability among the existing candidate pool which cannot assure that the next expecting candidate unit (hidden unit) error reduction will be maximized. This may cause the network to generate some hidden units with less error minimization capability. Eventually, more hidden units need to be added by randomly generating input weight and bias which may make the network more complex. For better understanding, Figure 9 (Qiao et al. 2016) in experimental work illustrates the same problem of not achieving maximal error reduction by FCNN at each hidden unit. It can be seen that error reduction by adding a new hidden unit is not smooth and the objective of maximum error reduction by next newly added hidden units is not achieved. This may result in redundant hidden units with minor effect on the convergence.

2.2. OLT and OLS

This section describes the existing methodologies that assist proposed CPCLS to analytically calculate the connection weights for achieving maximum error reduction on each hidden layer generation. Consider a training data sample with $(\boldsymbol{X}, \boldsymbol{Y})$, where $\boldsymbol{X}$ is the input unit matrix of $m\times n$ and $\boldsymbol{Y}$ is the output unit matrix of $m\times q$ with hidden units matrix $\boldsymbol{H}$ of $m\times p$. The input connection weights matrix of $n\times p$ is exemplified by $\boldsymbol{W}$, whereas, the output connection weights matrix of $p\times q$ are exemplified by $\boldsymbol{\beta}$.

OLT generates new p-features space of linearly independent $\boldsymbol{H}$ by orthogonally transforming n-features $\boldsymbol{X}$ (Jolliffe 2002). It helps to reduce the dimensionality of the correlated $\boldsymbol{X}$ by determining the unknown components $\boldsymbol{W}$**,** with each component explaining the amount of variance in the data. OLT initializes by determining the covariance matrix $\boldsymbol{S}$ of equal dimension $n\times n$ matrix, with diagonal numbers indicating covariance for the same feature and each number indicating the covariance between n-features of $\boldsymbol{X}$, to compute the eigenvalue λ and its corresponding eigenvector:

$$\boldsymbol{S} = \frac{1}{m-1}(\boldsymbol{X} - \overline{\boldsymbol{x}})^T(\boldsymbol{X} - \overline{\boldsymbol{x}}) \tag{1}$$

where $\bar{x} = \sum_{i=1}^{m} x_i$, with each quantity indicating the mean of n features.

The eigenvector, explaining the coordinate system for the new p-features by decreasing dimensions equal to or less than n-features, selection is based on the λ value. The λ is computed from the $\boldsymbol{S}$ matrix:

$$|\boldsymbol{S} - \lambda \boldsymbol{I}| = 0 \tag{2}$$

The corresponding eigenvector based on highest λ can be determined by computing the component $\boldsymbol{W}$:

$$(\boldsymbol{S} - \lambda \boldsymbol{I})\boldsymbol{W} = 0 \tag{3}$$

The matrix $\boldsymbol{W}$ linearly transforms *n*-features $\boldsymbol{X}$ into new linearly independent *p*-features $\boldsymbol{H}$:

$$\boldsymbol{H} = \boldsymbol{X}\boldsymbol{W} \tag{4}$$

OLS reduces the estimation error between the predicted $\widehat{\boldsymbol{Y}}$ and the observed $\boldsymbol{Y}$ variables by determining the unknown parameter $\boldsymbol{\beta}$ (Goldberger 1964):

$$\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{\beta} + e \tag{5}$$

OLS theory is used for determining $\boldsymbol{\beta}$ by:

$$\boldsymbol{\beta} = (\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H})^{-\mathbf{1}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{Y} \tag{6}$$

where $(\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H})^{-\mathbf{1}}\boldsymbol{H}^{\boldsymbol{T}}$ is the Moore Penrose pseudo-inverse of matrix $\boldsymbol{H}$. For better convergence, there should be no linear dependence among $\boldsymbol{H}$.

In the last step, the $\widehat{\boldsymbol{Y}}$ is determined:

$$\widehat{\boldsymbol{Y}} = \boldsymbol{H}\boldsymbol{\beta} \tag{7}$$

Better network convergence can be achieved by optimally calculating the connection weights in the forward step. Equations (3) and (6) play a key role in determining the connection weights for the novel CPCLS.

3. Proposed CPCLS learning algorithm

Like CasCor and its variants, which have a similar network structure, CPCLS also works on two concepts of cascade architecture and learning. Figure 2 illustrates CPCLS architecture, which is an improved form of CasCor and its variants. *Firstly*, CPCLS connects input units to the output units by the linearly independent hidden units to avoid the linear dependency of the input units. *Secondly*, more than single hidden units can be generated in the hidden layer to achieve faster convergence. *Thirdly,* the newly generated hidden layer is only linked to the output units, and earlier connections are removed to avoid the linear dependence of the hidden units among the hidden layers. In learning, CasCor repeatedly tunes the connection weights in forward and backward steps by the gradient method, while its variants either perform the gradient method or randomly generate the input weights, which can take more time, and it is equally problematic to control convergence. CPCLS eliminates the need for random generation and gradient methods by analytically computing the connection weights in the forward step.

3.1 Supporting statement and lemma

Statement 1: (Jolliffe 2002) OLT: The $\boldsymbol{X}$ values of n-features are orthogonally transformed into a linearly independent $\boldsymbol{H}$ of p-features by determining the eigenvalue λ and its eigenvector $\boldsymbol{W}$ from the input covariance $\boldsymbol{S}$.

Remark 1: Statement 1 implies that the hidden units generated are linearly independent (uncorrelated) because of the OLT of the input features.

Lemma 1: (Huang, Zhu, and Siew 2006) Given a standard Single hidden Layer Feedforward Network (SLFN) with N hidden nodes and activation function $g: R \to R$, which is infinitely differentiable in any interval, for N arbitrary distinct samples $(\mathrm{x_i}, \mathrm{y_i})$, where $\mathrm{x_i} \epsilon\ \mathbf{R^n}$ and $y_\mathrm{i} \epsilon\ \mathbf{R^m}$, for any $\boldsymbol{w_i}$ and b_i randomly chosen from any intervals of $\mathbf{R^n}$ and $\mathbf{R}$, respectively, according to any continuous probability distribution, then with probability one, the hidden layer output matrix $\boldsymbol{H}$ of the SLFN is invertible and $\left||\boldsymbol{H\beta} - \boldsymbol{Y}|\right| = \mathbf{0}$.

Remark 2: Lemma 1 implies that the hidden units need to be linearly independent with a probability of one to obtain the best least-squares solution of $\boldsymbol{Y} = \boldsymbol{H\beta}$.

Remark 3: (Goldberger 1964) According to ordinary least squares theory, the smallest error $\left\|\widehat{\boldsymbol{Y}}-\boldsymbol{Y}\right\| = \boldsymbol{0}$ can be achieved by calculating $\boldsymbol{\beta} = (\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H})^{-\mathbf{1}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{Y}$ such that there exists no multicollinearity (linearly dependence) among the hidden units.

3.2 Input connection weights W determination

Based on the above supporting statement, lemma, and remarks, the CPCLS can achieve a best least-squares unique solution by the orthogonal transformation of the input and pre-existing hidden units. CPCLS initializes by defining number N of $\boldsymbol{H}$ in the first hidden layer such that $p \leq n$. Initially, $\boldsymbol{X}$ is indirectly connected to $\boldsymbol{Y}$ through $\boldsymbol{H}$ to avoid input feature linear dependence. For $\boldsymbol{W}$ determination, the eigendecomposition of $\boldsymbol{S}$ (1) generates λ (2) and the highest λ values explaining maximum variance in data are used to determine the eigenvectors (3). The determined eigenvectors are referred to as $\boldsymbol{W}$. Knowing $\boldsymbol{X}$ and $\boldsymbol{W}$, the value of $\boldsymbol{H}$ is computed as:

$$\boldsymbol{H} = \emptyset(\boldsymbol{XW}) \tag{8}$$

where $\emptyset(z)$ can be any differentiable or nondifferentiable continuous activation function.

3.3 Output connection weights β determination

The second step is to compute the $\boldsymbol{\beta}$ by considering the linear relationship of $\boldsymbol{H}$ to $\boldsymbol{Y}$. The Moore Penrose pseudo-inverse of $\boldsymbol{H}$ is calculated and its product with $\boldsymbol{Y}$ is used to calculate $\boldsymbol{\beta}$ (6). The linear conversion of $\boldsymbol{H}$ through $\boldsymbol{\beta}$ generates $\widehat{\boldsymbol{Y}}$ (7). The algorithm aims to efficiently converge the network by minimizing the error function E faster:

$$E = \frac{1}{m}\sum_{i=1}^{m}\left(\widehat{\boldsymbol{Y}}_i - \boldsymbol{Y}_i\right)^2 \tag{9}$$

If E is a smaller amount than the described target error e, the CPCLS loop will terminate, else a new $\boldsymbol{H}$ will be generated until the required convergence is reached.

3.4 Newly added hidden layer connection to the output layer

In the proceeding hidden layers, the newly added $\boldsymbol{H_k}$ *(k=1, 2, 3…)* receives all incoming connections from $\boldsymbol{X}$ and any preexisting hidden layers $\boldsymbol{H_{k-1}}$, $\boldsymbol{H_{k-2}}$ and so on, whereas, the output layer receives connections from only the newly added $\boldsymbol{H}_k$ and diminishes its previous connections i.e. $\boldsymbol{H_{k-1}}$, $\boldsymbol{H_{k-2}}$ and so on. Connecting the previously added hidden layers to the output units plays no significant role in the network. It may only add burden to the network by connecting linearly dependent and redundant hidden units which can reduce the generalization performance, as well as learning speed. Each newly added $\boldsymbol{H}_k$ adds its non-linearity based on the variance in $\boldsymbol{X}$ and previously added $\boldsymbol{H_{k-1}}$, $\boldsymbol{H_{k-2}}$ and so on. This can be expressed in term of error minimization as:

$$E^{LHL} = (\boldsymbol{\beta H_k}) - \boldsymbol{Y} \tag{10}$$

where E^{LHL} is network error by connecting only the newly added hidden layer to the output layer. The newly added $\boldsymbol{H}_k$ is of a higher level which has learned from the orthogonal linear transformation of both $\boldsymbol{X}$ and previously added $\boldsymbol{H_{k-1}}$, $\boldsymbol{H_{k-2}}$ and so on, and represents the maximum variance of the network in that it guarantees the convergence of the CPCLS.

Suppose symmetric matrix $\boldsymbol{S}$ has two different eigenvalues λ_1 and λ_2 corresponding to eigenvectors w_1 and w_2 in matrix $\boldsymbol{W}$ respectively. Two vectors can be considered orthogonal if their inner product is zero, such as: $w_1.w_2 = 0$ or $w_1^T w_2 = 0$. where w_1^T is the transpose of w_1.

We have:

$$\boldsymbol{S}w_1 = \lambda_1 w_1 \tag{11}$$

and

$$\boldsymbol{S}w_2 = \lambda_2 w_2 \tag{12}$$

To prove that w_1 and w_2 are orthogonal:

$$\lambda_1(w_1.w_2) = (\lambda_1 w_1).w_2 = (\boldsymbol{S}w_1).w_2 = (\boldsymbol{S}w_1)^{\,T}w_2 = w_1^T\boldsymbol{S}^T w_2$$

$$= w_1^T\boldsymbol{S}w_2 = w_1^T\lambda_2 w_2 = \lambda_2(w_1^T w_2) = \lambda_2(w_1.w_2)$$

$\boldsymbol{S} = \boldsymbol{S^T}$ because $\boldsymbol{S}$ is a symmetric matrix. From mathematical work, we have:

$$\lambda_1(w_1.w_2) = \lambda_2(w_1.w_2) \tag{13}$$

$$(\lambda_1 - \lambda_2)(w_1.w_2) = 0 \tag{14}$$

Since $\lambda_1 - \lambda_2 \neq 0$, because both are different. So, we have:

$$w_1.w_2 = 0 \tag{15}$$

which means that eigenvectors w_1 and w_2 are orthogonal to each other in matrix $\boldsymbol{W}$, i.e., $w_1 \perp w_2$. This orthogonal property of $\boldsymbol{W}$ causes $\boldsymbol{X}$ and preexisting $\boldsymbol{H_{k-1}}$ to orthogonally linearly transform into linearly independent $\boldsymbol{H}_k$. Suppose if two hidden unit vectors are generated in $\boldsymbol{H}_k$ such that the h_{k_1} is generated from w_1 and h_{k_2} is generated from w_2, then they can also be considered orthogonal, i.e., $h_{k_1} \perp h_{k_2}$. The proof supports Lemma 1 and guarantees the convergence of CPCLS because of the $\boldsymbol{H}_k$ generated are invertible and hence $\left||(\boldsymbol{\beta H_k}) - \boldsymbol{Y}|\right| = \boldsymbol{0}$.

However, if all (every previous and newly) hidden layers are connected to the output layer, we have:

$$E^{AHL} = \big(\boldsymbol{\beta}(\boldsymbol{H_k} + \boldsymbol{H_{k-1}} + \boldsymbol{H_{k-2}} + \cdots + \boldsymbol{H_1})\big) - \boldsymbol{Y} \tag{16}$$

where E^{AHL} is the network error by connecting all the hidden layers to the output layer. According to Remarks 1 and Lemma 1, the hidden units in multiple hidden layers may create linear dependency and redundancy in that it will avoid the best least square solution assumption. Suppose if two hidden unit vectors are generated in $\boldsymbol{H_{k-1}}$ such that h_{k-1_1} is generated from w_{k-1_1} and h_{k-1_2} is generated from w_{k-1_2} and two hidden unit vectors are generated in $\boldsymbol{H_k}$ such that h_{k_1} is generated from w_{k_1} and h_{k_2} is generated from w_{k_2} than there is a chance that it may or may not be orthogonal, i.e., $\boldsymbol{H_{k-1}} \perp \boldsymbol{H_k}$ or $\boldsymbol{H_{k-1}} \not\perp \boldsymbol{H_k}$. In the latter case, it may void the assumption that the $\boldsymbol{H}$ generated are invertible and hence $\left|\left|\left(\boldsymbol{\beta}(\boldsymbol{H_k} + \boldsymbol{H_{k-1}} + \boldsymbol{H_{k-2}} + \cdots + \boldsymbol{H_1})\right) - \boldsymbol{Y}\right|\right| \neq \boldsymbol{0}$.

Hidden units are generated from the eigenvalue and corresponding eigenvector; therefore, the new hidden units feature generation will always be less than or equal to the input units and the previously hidden unit features $\boldsymbol{X} = (\boldsymbol{X}, \boldsymbol{H})$, such that $p \leq n$. Jolliffe and Cadima (2016) stated that the eigenvalues having cumulative percentage variance (CPV) of 70% are commonly used to extract eigenvectors. However, Jolliffe and Cadima (2016) further added that there may circumstances in which the last few eigenvalues may be also of interest in explaining more variance in the data. Researchers (Jolliffe and Cadima 2016; Tortorella et al. 2016) in their work recommended selecting eigenvalues giving a CPV greater than 70% to a maximum of 99%. The experimental work has been performed to study the effect of hidden unit selection on generalization performance and learning speed.

3.5 CPCLS hyperparameters

CPCLS initializes with a small number of hyperparameters i.e. $\boldsymbol{H}$ and e, in comparison with other fixed and constructive topology algorithms i.e. learning rate, hidden nodes, candidate units, etc. This makes learning simple.

Algorithm CPCLS

Given a training set $(\boldsymbol{X}, \boldsymbol{Y})$ with input unit matrix $\boldsymbol{X}$ be *m*×*n*, output unit matrix $\boldsymbol{Y}$ be *m*×*q*, hidden unit matrix $\boldsymbol{H}$ be *m*×*p*, and target error *e*:

Step 1) **Initialization:** Define the initial number N of $\boldsymbol{H}$ in a first hidden layer such that $p \leq n$

Step 2) **Learning Step:**

While $E > e$

a) Determine the $\boldsymbol{W}$ matrix of n×p:

1. Calculate the $\boldsymbol{S}$ matrix of n×n from n features $\boldsymbol{X}$:

$$\boldsymbol{S} = \frac{1}{m-1}(\boldsymbol{X} - \overline{\boldsymbol{x}})^T(\boldsymbol{X} - \overline{\boldsymbol{x}})$$

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

2. Select λ with the highest values to calculate the eigenvectors. The calculated N eigenvectors are considered as $\boldsymbol{W}$ for $\boldsymbol{H}$:

$$|\boldsymbol{S} - \lambda\boldsymbol{I}| = 0$$

$$(\boldsymbol{S} - \lambda\boldsymbol{I})\boldsymbol{W} = 0$$

b) Take ∅ of $\boldsymbol{X}$ and $\boldsymbol{W}$ to generate $\boldsymbol{H}$:

$$\boldsymbol{H} = \emptyset(\boldsymbol{XW})$$

c) Determine the $\boldsymbol{\beta}$ matrix of p×q:

$$\boldsymbol{\beta} = (\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H})^{-\mathbf{1}}\boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{Y}$$

d) Calculate $\widehat{\boldsymbol{Y}}$:

$$\widehat{\boldsymbol{Y}} = \boldsymbol{H\beta}$$

e) Calculate E:

```
    $E = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{\boldsymbol{Y}}_i - \boldsymbol{Y}_i\right)^2$
f) Combine the columns of $\boldsymbol{H}$ with $\boldsymbol{X}$:
    $\boldsymbol{X} = (\boldsymbol{X}, \boldsymbol{H})$
g) increase the number of $\boldsymbol{H}$ by $N'$ in the proceeding hidden layers such that $p \leq n$:
    $N = N + N'$
end
```

4. Experimental study

The comparative study of the proposed algorithm CPCLS with state-of-the-art machine learning algorithms was conducted to demonstrate its effectiveness. The experimental work was performed in Netmaker v0.9.5.2 and Anaconda Spyder Python v3.2.6. The experimental work of CPCLS, BPNN, and self-adaptive extreme learning machine (SaELM) (Wang et al. 2016) were performed in Python, whereas, the CasCor work was performed in the built-in powerful Netmaker C-programming code. Generally speaking, experimental work in the two programming codes will not affect the comparative study because C programming is considered much faster than Python. The dataset was normalized in the range [0,1] for both input and output and sigmoid activation function $\emptyset(z) = 1/(1 + e^{-z})$ was used in the hidden units of the algorithms.

The experimental work was divided into three parts: real-world applications prediction, energy applications prediction and studying the CPCLS hidden units and layers characteristics followed by further discussion. Table 1 shows the most popular and widely used dataset in machine learning extracted from UCI (Dua and Graff 2019). The number of hidden units in hidden layers of the CPCLS was set to (2,2), (4,3), (2,1), (2,2), (5,5) for real-world applications such as abalone, airfoil self-noise, forest fires, breast cancer, wine respectively, and (2,2), (2,7), (1,1) for energy applications such as combined cycle power plant, occupancy detection, seismic bumps respectively. The number of CasCor candidate units was set to 3 Nos. The number of hidden units for stochastic gradient descent BPNN was decided by a trial and error approach in the range 5-25 and the hidden units with optimal results are reported. The minimum, maximum and interval hidden units for SaELM was set to 5, 500 and 10 respectively with width factor Q=2 and scale factor L=4.

Tables 2, 3 and 4 show the average best results of 25 trials obtained by the machine learning algorithms. The testing RMSE/accuracy represents the generalization performance, and the learning time represents the learning speed of the algorithms, while the mean and stdev in the table refer to the average and standard deviation results of 25 trials. The performance criteria for regression problems and classification problems are RMSE and percentage accuracy respectively.

4.1 Real-world applications prediction

Table 2 shows the prediction results of real-world applications. The proposed CPCLS algorithm was able to achieve a better generalization performance and learning speed in all cases as compared to CasCor, BPNN, and SaELM. The best results in terms of generalization and learning speed are highlighted in bold and underlined in Table 2. For an in-depth understanding of the convergence rate during each hidden layer, Figure 3 illustrates the CPCLS convergence rate of 25 trials for the Abalone dataset. It can be observed that the convergence rate of CPCLS during each hidden layer addition is smooth and stable.

CPCLS performance comparison has also been made with CasCor variants to demonstrate its effectiveness. Due to the limitation caused by the unavailability of the original programming code of OLSCN and FCNN, the simulation results of selected real-world problems representing both algorithms are taken from their original source papers. To make the comparison more valuable and to get better insights, the CPCLS simulation is carried out by considering all test conditions mentioned in the original paper of OLSCN and FCNN. Table 3 shows the dataset description, algorithms comparison in terms of generalization performance and learning speed. It can be observed that CPCLS generalization performance and learning speed averaged over 25 trials are better with more improved results compared to FCNN and OLSCN.

4.2 Energy applications prediction

To further validate the performance, a comparative study was performed on energy-based problems. The most demanding energy applications are:

(1) ***Combined cycle power plant:*** A combined cycle power plant is used to generate electricity from gas turbines and consequently uses the waste energy in a steam turbine to improve the efficiency of the electrical output. The attributes that considerably affect the performance of gas turbine are atmospheric pressure (millibar), temperature (°C) and relative humidity (%), whereas, the attributes that affect the performance of the steam turbine are exhaust steam pressure (cm Hg). The dataset contains an hourly average of attributes (atmospheric pressure, temperature and relative humidity, exhaust steam pressure) to predict the net hourly electrical energy (MW) of the powerplant.

(2) ***Seismic bumps:*** Seismic hazard prediction is a challenging application area in coal mining. The purpose is to detect the possibilities of the occurrence of rock bursts from seismic activity. The task is to classify high energy seismic bumps as “hazardous” and “non-hazardous” from attributes such as possible seismic hazard,

seismic energy, pulses, energy deviation, number of seismic bumps with different energy levels, total and maximum energy recorded for seismic bumps.

(3) ***Occupancy detection:*** Predicting occupancy detection in an office building is attracting significant interest in reducing energy consumption. Various measurements of light energy (Lux), temperature (°C), relative humidity (%), humidity ratio (kgwater-vapor/kg/air), and CO_2 (ppm) along with the time are used to classify whether the room is occupied or not.

Table 4 shows the performance of various machine learning algorithms for energy application prediction. For the combined cycle power plant, CPCLS was able to achieve a better performance of 0.0545 RMSE in a learning time of 2.96s compared to CasCor of 0.0573 in 29.69s, BPNN of 0.0577 in 59.54s, and SaELM of 0.0547 in 7.09s. For seismic bumps, the generalization accuracy of CPCLS and BPNN is the same with the advantage of CPCLS in that it took 0.01s compared to BPNN of 1.06s. The CPCLS demonstrated its effectiveness by achieving a performance accuracy of 93.83% in a learning time of 0.01s compared to CasCor of 92.98% in 29.86s, BPNN of 93.83% in 1.06s, and SaELM of 93.44% in 1.87s respectively. Similar to the combined cycle power plant and seismic bumps, CPCLS also efficiently predicted occupancy detection. CPCLS achieved a better performance accuracy of 99.05% in learning time of 3.95s compared to CasCor of 98.97% in 31.54s, BPNN of 98.98% in 75.33s, and SaELM of 99.03% in 17.64s respectively. The standard deviation of the generalization performance and learning time are also lower which demonstrates the stable results of CPCLS.

4.3 Connecting hidden layers to the output layer and varying hidden unit size in the hidden layer of CPCLS

4.3.1 Varying hidden unit sizes in the hidden layers

For CPCLS, the selection of hidden units in the first hidden layer and proceeding hidden layers is only a single hyperparameter that needs to be defined based on the eigenvalue and corresponding eigenvector. For illustration, experimental work has been performed by taking the example of the abalone dataset. The abalone dataset consists of 9 input attributes with bias. This implies that a lower and higher combination can be (1,1) and (9,9) respectively with a total of 81 combinations.

Figures 4, 5 and 6 show the generalization performance, learning speed and number of hidden layers for different combinations. The horizontal axis concerns the addition of hidden units in the first layer and the right legend concerns the addition of hidden units in the proceeding layers. Figure 4 illustrates that the generalization performance is stable for a maximum number of combinations. The minimum 0.0748RMSE and maximum 0.0774RMSE were achieved by (5,2) and (4,2) combinations respectively. Furthermore, a lower combination (1,1) achieved 0.0765RMSE and higher combination (9,9) achieved 0.0755RMSE. The (5,2), (4,2), (1,1) and (9,9) hidden units are generated from the eigenvalue CPV of (99.65%,96.94%), (99.19%,96.94%), (71.26%,72.92%) and (100%,100%) respectively. The minimum and maximum RMSE combination, and lower and higher hidden unit combinations give insight that hidden units generated based on eigenvalue explaining CPV $\lambda > 70\%$ are helpful in achieving better generalization performance. However, as shown in Figure 5, the learning time was 2.03s with (1,1) as compared to 0.03s for (9,9). The increase in learning time happens because of the higher computational burden by hidden layers. Figure 6 illustrates that hidden layers reach to 45 Nos. for lower combination (1,1) compared to 4 Nos. for higher combination (9,9). The findings support the existing work and recommend generating hidden units in the layers having eigenvalue explaining CPV $\lambda > 70\%$. Based on our experimental work, it is recommended that the CPV should not be greater than 99% because many of the last few eigenvalues may have approximately zero variability. The zero variability eigenvalues may create a problem of overfitting which needs to be avoided.

4.3.2 The effect of hidden layers connection to the output layer

Experimental work has been performed to study the effect of hidden layers connection to output layer by considering both cases for CPCLS:

(1) Connecting the last hidden layer to the output layer (LHL)

(2) Connect all hidden layers to the output layer (AHL)

The work was performed on artificial nonlinear SinC function regression task, generating 4,000 observations in the range [-20,20], by changing the data random state from 0 to 100 with an interval of 5 and data test size from 30% to 70% with an interval of 5%. This makes a total of 21 trials with different random states and 9 trials with different test sizes. The 21 trials with different random states were performed by keeping the constraint of test size equal to 50%. The best result by the random state was selected to perform 9 trials by varying the test sizes.

Table 5 shows the generalization performance and learning speed of both cases. Figures 7 and 8 illustrate the generalization performance and learning speed of both cases for each random state and for each test size respectively. Both figures show that the generalization performance becomes worse in most cases for AHL. Compared to AHL, the generalization results of LHL are more stable with minimal deviation. Similarly, the learning time increases for AHL compared to LHL. To avoid an increase in further learning time, the algorithm for AHL needs to stop early when there is no further decrease in error, and the training time is about five times more than LHL.

The difference in Figure 9 illustrates that AHL is unable to correctly predict the SinC function, whereas LHL, (the original CPCLS), has predicted accurately all data points of the function.

4.4 Further discussion

The better generalization performance and faster learning speed of CPCLS on real-world and energy problems compared to CasCor, BPNN, SaELM, OLSCN, and FCNN demonstrate its effectiveness. However, comparison with state-of-the-art machine learning algorithms is important to build greater confidence in the application of CPCLS. Table 6 shows the comparison of CPCLS with popular machine learning algorithms. The comparative study gives an important insight that CPCLS generalization performance in solving various real-world and energy problems is better compared to other machine learning results, that are published recently in the literature. This finding supports that CPCLS is a promising machine learning tool that can be practiced in general to improve various operations of production research.

In real practice, the work is beneficial in numerous manners. Taking the example of breast cancer, the CPCLS correctly classified its class as malignant or benign. It is important to avoid misclassification of malignant cancer as benign because it can cause human death. In engineering, the aviation sector works on zero-defect philosophy. Better prediction of airfoils noise by CPCLS can facilitate in improving aircraft efficiency and reduce environmental pollution. CPCLS efficiently prediction of marine species ages rather than a microscope measurement can facilitate in avoiding subjective judgment and fatigue. Besides, the application of CPCLS in predicting possible future hazards can help to protect food products and the wastage of natural resources.

The better prediction results of CPCLS for energy applications such as predicting electrical energy of powerplant and reducing energy consumption by accurately predicting building occupancy detection can help in designing better energy management systems. Moreover, predicting seismic hazards by CPCLS as hazardous and non-hazardous can prevent fatal accidents.

5. Conclusions

In this paper, a novel learning algorithm called CPCLS is proposed. Unlike other cascade algorithms, in this approach, hidden units are linearly generated by orthogonal linear transformation and only the last hidden layer is connected to the output layer. It was theoretically and experimentally verified that the hidden units generated in the respective hidden layer are inevitable (i.e. linearly independent) which guarantees CPCLS convergence. Connecting only the last hidden layer to the output layer eventually improves the performance and increase the learning speed because all the hidden units are orthogonal.

Compared to the state-of-the-art machine learning algorithms, the proposed CPCLS achieved better generalization performance and learning speed in various prediction tasks. Experimental work also demonstrated that connecting only the last hidden layer rather than all the hidden layers to the output layer creates less burden on the network and significantly improves convergence.

The major contributions and findings are: i) The CPCLS provides new insight into existing algorithms by analytically calculating connection weights on both sides of the network rather than gradient iteration or random generation, ii) In CPCLS, the generated hidden units are inevitable ensuring that convergence will be optimal, iii) CPCLS initialize with small number of hyperparameters, such as only defining number of hidden units in the layer, iv) Compared to the existing works, this study provides insight that avoiding direct linear connection of the input layer to the output layer and connecting only newly added hidden layer to the output layer reduces network burden and improves convergence, and v) In current practice, majority of research or models are proposed for specific applications. The better performance of CPCLS, on various applications, in comparison with state-of-the-art machine learning algorithms demonstrate that CPCLS can be practiced in general for prediction of regression and classification tasks to make better-informed decisions.

The implications are: i) In the proposed CPCLS, the experimental work was performed on the OLT of the covariance matrix. Other than the covariance matrix, single value decomposition and the correlation matrix can also be applied for OLT. Future work may include studying the application of single value decomposition and correlation matrix and their performance on the CPCLS, ii) Besides, the experimental work is limited to the application of commonly used sigmoid activation function. Other than sigmoid function, the effect of various other activation functions on the performance of CPCLS needs to be explored in future work.

References

Aljarah, I., H. Faris, and S. Mirjalili. 2018. “Optimizing connection weights in neural networks using the whale optimization algorithm.” *Soft Computing* 22 (1): 1-15. doi: 10.1007/s00500-016-2442-1.

Banerjee, P., V. S. Singh, K. Chatttopadhyay, P. C. Chandra, and B. Singh. 2011. "Artificial neural network model as a potential alternative for groundwater salinity forecasting." *Journal of Hydrology* 398 (3-4):212-20.

Bansal, P., S. Gupta, S. Kumar, S. Sharma, and S. Sharma. 2019. “MLP-LOA: A metaheuristic approach to design an optimal multilayer perceptron.” *Soft Computing* 23 (23): 12331-45.

Candanedo, L. M., and V. Feldheim. 2016. “Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models.” *Energy and Buildings* 112: 28-39.

Chien, C. F., Y. S. Lin, and S. K. Lin. 2020. “Deep Reinforcement Learning for Selecting Demand Forecast Models to Empower Industry 3.5 and an Empirical Study for a Semiconductor Component Distributor.” *International Journal of Production Research* (in press) doi: 10.1080/00207543.2020.1733125.

Chung, S. H., H. L. Ma, and H. K. Chan. 2017. "Cascading delay risk of airline workforce deployments with crew pairing and schedule optimization." *Risk Analysis* 37 (8):1443-58. doi: 10.1111/risa.12746.

Deng, C., J. Miao, Y. Ma, B. Wei, and Y. Feng. 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* (in press). doi: 10.1080/00207543.2019.1636327.

Dua, D., and C. Graff. 2019. "UCI Machine Learning Repository." Accessed 2019-03-28. http://archive.ics.uci.edu/ml.

Ertuğrul, Ö. F. 2018. “A novel type of activation function in artificial neural networks: Trained activation function.” *Neural Networks* 99: 148-57.

Fahlman, S. E., and C. Lebiere. 1990. The cascade-correlation learning architecture. Paper presented at the Advances in neural information processing systems.

Fahlman, S. E. 1988. "An empirical study of learning speed in back-propagation networks." In. Pittsburgh PA 15213: School of Computer Science, Carnegie Mellon University.

Goldberger, A. S. 1964. "Classical linear regression." In *Econometric theory*, 156-212. New York: John Wiley & Sons.

Grasso, F., A. Luchetta, and S. Manetti. 2018. “A multi-valued neuron based complex ELM neural network.” *Neural Processing Letters* 48 (1): 389-401.

Hecht-Nielsen, R. 1989. Theory of the backpropagation neural network. Paper presented at the International Joint Conference on Neural Networks.

Heidari, A., V. G. Agelidis, J. Pou, J. Aghaei, and A. M. Y. M. Ghias. 2018. "Reliability worth analysis of distribution systems using cascade correlation neural networks." *IEEE Transactions on Power Systems* 33 (1):412-20.

Huang, G., S. Song, and C. Wu. 2012. "Orthogonal least squares algorithm for training cascade neural networks." *IEEE Transactions on Circuits and Systems I: Regular Papers* 59 (11):2629-37.

Huang, G. B., Q. Y. Zhu, and C. K. Siew. 2006. "Extreme learning machine: theory and applications." *Neurocomputing* 70 (1-3):489-501.

Hunter, D., H. Yu, M. S. Pukish III, J. Kolbusz, and B. M. Wilamowski. 2012. "Selection of proper neural network sizes and architectures—A comparative study." *IEEE Transactions on Industrial Informatics* 8 (2):228-40.

Hwarng, H. B. 2005. "Simultaneous identification of mean shift and correlation change in AR(1) processes." *International Journal of Production Research* 43 (9):1761-83. doi: 10.1080/00207540512331311822.

Jolliffe, I. T., and J. Cadima. 2016. "Principal component analysis: a review and recent developments." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374 (2065):20150202.

Jolliffe, I. T. 2002. *Principal component analysis*. New York: Springer-Verlag.

Kapanova, K. G., I. Dimov, and J. M. Sellier. 2018. "A genetic approach to automatic neural network architecture optimization." *Neural Computing and Applications* 29 (5):1481-92.

Khan, W. A., S. H. Chung, M. U. Awan, and X. Wen. 2019a. “Machine learning facilitated business intelligence (Part I): Neural networks learning algorithms and applications.” *Industrial Management & Data Systems* 120 (1):164-95.

Khan, W. A., S. H. Chung, M. U. Awan, and X. Wen. 2019b. “Machine learning facilitated business intelligence (Part II): Neural networks optimization techniques and applications.” *Industrial Management & Data Systems* 120 (1):128-63.

Khan, W. A., S. H. Chung, and C. Y. Chan. 2018. Cascade Principal Component Least Squares Neural Network Learning Algorithm. Paper presented at the 2018 24th International Conference on Automation and Computing (ICAC).

Kim, B., Y. S. Jeong, S. H. Tong, and M. K. Jeong. 2019. “A generalised uncertain decision tree for defect classification of multiple wafer maps.” *International Journal of Production Research* (in press). doi: 10.1080/00207543.2019.1637035.

Kumar, S., J. Singh, and O. Singh. 2020. “Ensemble-based extreme learning machine model for occupancy detection with ambient attributes.” *International Journal of System Assurance Engineering and Management* (in press).

Kuo, Y. H., and A. Kusiak. 2019. “From data to big data in production research: The past and future trends.” *International Journal of Production Research* 57 (15-16): 4828-53. doi: 10.1080/00207543.2018.1443230.

Kusiak, A. 2020. “Convolutional and generative adversarial neural networks in manufacturing.” *International Journal of Production Research* 58 (5): 1594-1604. doi: 10.1080/00207543.2019.1662133.

Kwok, T. Y., and D. Y. Yeung. 1997. "Constructive algorithms for structure learning in feedforward neural networks for regression problems." *IEEE Transactions on Neural Networks* 8 (3):630-45.

Liew, S. S., M. Khalil-Hani, and R. Bakhteri. 2016. "An optimized second order stochastic learning algorithm for neural network training." *Neurocomputing* 186:74-89.

Liu, Q., H. Zhang, J. Leng, and X. Chen. 2019. “Digital Twin-Driven Rapid Individualised Designing of Automated Flow-Shop Manufacturing System.” *International Journal of Production Research* 57 (12): 3903–19. doi: 10.1080/00207543.2018.1471243.

Liu, Y., L. Wang, X. V. Wang, X. Xu, and L. Zhang. 2019. “Scheduling in Cloud Manufacturing: State-of-the-Art and Research Challenges.” *International Journal of Production Research* 57 (15–16): 4854–79. doi: 10.1080/00207543.2018.1449978.

Lorencin, I., N. Anđelić, V. Mrzljak, and Z. Car. 2019. “Genetic algorithm approach to design of multi-layer perceptron for combined cycle power plant electrical power output estimation.” *Energies* 12 (22): 435201-26.

Lv, J., T. Peng, Y. Zhang, and Y. Wang. 2020. “A novel method to forecast energy consumption of selective laser melting processes.” *International Journal of Production Research* (in press). doi: 10.1080/00207543.2020.1733126.

Mantas, C. J., J. G. Castellano, S. M. García, and J. Abellán. 2019. “A comparison of random forest based algorithms: Random credal random forest versus oblique random forest.” *Soft Computing* 23 (21): 10739-54.

Nayyeri, M., H. S. Yazdi, A. Maskooki, and M. Rouhani. 2018. “Universal approximation by using the correntropy objective function.” *IEEE Transactions on Neural Networks and Learning Systems* 29 (9): 4515-21.

Qiao, J., F. Li, H. Han, and W. Li. 2016. "Constructive algorithm for fully connected cascade feedforward neural networks." *Neurocomputing* 182:154-64. doi: 10.1016/j.neucom.2015.12.003.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams. 1986. "Learning representations by back-propagating errors." *Nature* 323 (6088):533-6. doi: 10.1038/323533a0.

Solimanpur, M., P. Vrat, and R. Shankar. 2004. “Feasibility and Robustness of Transiently Chaotic Neural Networks Applied to the Cell Formation Problem.” *International Journal of Production Research* 42 (6): 1065–82. doi: 10.1080/00207543.2004.10750072.

Tortorella, G. L., G. A. Marodin, D. D. C. Fettermann, and F. S. Fogliatto. 2016. "Relationships between lean product development enablers and problems." *International Journal of Production Research* 54 (10):2837-55.

Wang, G. G., M. Lu, Y. Q. Dong, and X. J. Zhao. 2016. "Self-adaptive extreme learning machine." *Neural Computing and Applications* 27 (2):291-303.

Wang, H., S. Ding, D. Wu, Y. Zhang, and S. Yang. 2018. "Smart connected electronic gastroscope system for gastric cancer screening using multi-column convolutional neural networks." *International Journal of Production Research* (in press). doi: 10.1080/00207543.2018.1464232.

Wang, Z., H. Ma, H. Chen, B. Yan, and X. Chu. 2019. “Performance degradation assessment of rolling bearing based on convolutional neural network and deep long-short term memory network.” *International Journal of Production Research* (in press). doi: 10.1080/00207543.2019.1636325.

Zou, W., F. Yao, B. Zhang, and Z. Guan. 2018. “Improved meta-ELM with error feedback incremental ELM as hidden nodes.” *Neural Computing and Applications* 30 (11): 3363-70.