Ensemble Deep Random Vector Functional Link Neural Network for Regression

Minghui Hu, *Student Member, IEEE*, Jet Herng Chion, Ponnuthurai Nagaratnam Suganthan, *Fellow, IEEE*, and Rakesh Kumar Katuwal

***Abstract*—Inspired by the ensemble strategy of machine learning, deep random vector functional link (dRVFL), and ensemble dRVFL (edRVFL) has shown state-of-the-art results on different datasets. Our present work first fills the gap of dRVFL and edRVFL work in the field of regression. We test and evaluate the performances of the dRVFLs on regression problems. Subsequently, we propose a novel regularization method [boosted factor (BF)], two dRVFLs variants [edRVFL with skip connection (edRVFL-SC) and edRVFL with random skip connections (edRVFL-RSC)] and one strategy [ensemble skip connection edRVFL (esc-edRVFL)] which show significant improvement over the original dRVFL. The BF is a newly introduced hyperparameter to scale the values of the activated hidden neurons to accommodate the diversity of the data, and it is also able to filter the neurons. edRVFL-SC and edRVFL-RSC are the edRVFL variants with skip connections. In edRVFL-SC, we apply dense skip connections to the edRVFL, which is inspired by the residual architecture in the deep learning area. However, due to the specificity of randomized networks, the simple skip connections are probably leading to the reuse of useless features. To address this problem, we propose a random skip connection-based edRVFL, which can keep the diversity in the latent space. esc-RVFL is an ensemble scheme that utilizes several edRVFL-RSC models trained on the different folds of the training dataset. The esc-edRVFL is identified as the best-performing algorithm through a comprehensive evaluation of 31 UCI datasets.**

***Index Terms*—Ensemble, neural networks, random vector functional link (RVFL), regression.**

I. Introduction

ARTIFICIAL neural networks have been around for many decades. Its history dates back to the 40-s when McCulloch and Pitts proposed a mathematical model to model the human brain's neurons [1]. Since then, the neural network has undergone several booming and winter seasons. In 2012, Alex Krizhevsky and his team won the Imagenet competition with their state-of-the-art Alexnet model [2]. The Alexnet achieved an unprecedented performance by neural network and revitalized the neural network field from a long winter period, and the architecture of Alexnet has become the inspiration of different types of models and a lot of progress has been made in various fields, including computer vision, natural language processing, and so on.

Most of the neural network models are trained by applying gradient descent to minimize a loss function defined by the users and using the backpropagation algorithm to update the weights of the hidden layers [3]. There is no doubt that the backpropagation mechanism is one of the most important factors contributing to the booming of the current Artificial Intelligence industry.

However, the backpropagation mechanism has its shortcomings. For instance, the training of such a model is time-consuming. And, the most important problem is that the gradient descent-based model often suffers from the convergence issue where the training progress may be stuck at a local minimum resulting in a suboptimal solution [4], [5].

Randomized neural networks, on the other hand, can address the issues mentioned above. Instead of updating the weights through iterations, the weights of randomized neural networks are randomly initialized and kept fixed throughout the training [4], which means the hidden weights of RNN are data independent. The randomized neural network was first proposed in the 90 s. It has gained attention due to its simplicity and performance with universal approximation capability. Random vector functional link (RVFL), introduced by Pao et al. in 1994 [6], has gained substantial traction due to its simplicity and superior performance in different domains. RVFL is proven as a universal approximator for continuous function with bounded finite dimension and closed-form solution in [7] and [8]. The weights and biases between the enhancement (i.e., hidden) nodes and input nodes in RVFL are randomly generated. The direct link between the input data and the output layer propagates the original data directly to the output layer. The direct link acts as a form of regularization for the randomization [9] and helps to keep the complexity of the model low [10]. Due to its simplicity and efficiency, RVFL is currently used in a variety of applications, such as the short-term load forecasting [11], the rolling bearing fault diagnosis [12], and unsupervised learning [13].

Manuscript received 28 April 2022; revised 31 July 2022 and 1 October 2022; accepted 1 October 2022. This article was recommended by Associate Editor M. Perc. *(Corresponding author: Ponnuthurai Nagaratnam Suganthan.)*

Minghui Hu is with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore (e-mail: e200008@e.ntu.edu.sg).

Jet Herng Chion was with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. He is now with NCS Pte. Ltd., Singapore (e-mail: jchion001@e.ntu.edu.sg).

Ponnuthurai Nagaratnam Suganthan is with the KINDI Center for Computing Research, College of Engineering, Qatar University, Doha, Qatar, and also with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore (e-mail: p.n.suganthan@qu.edu.qa).

Rakesh Kumar Katuwal was with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. He is now with the Services and Products Department, Fusemachines, Kathmandu 44600, Nepal (e-mail: rakeshku001@e.ntu.edu.sg).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TSMC.2022.3213628.

Digital Object Identifier 10.1109/TSMC.2022.3213628

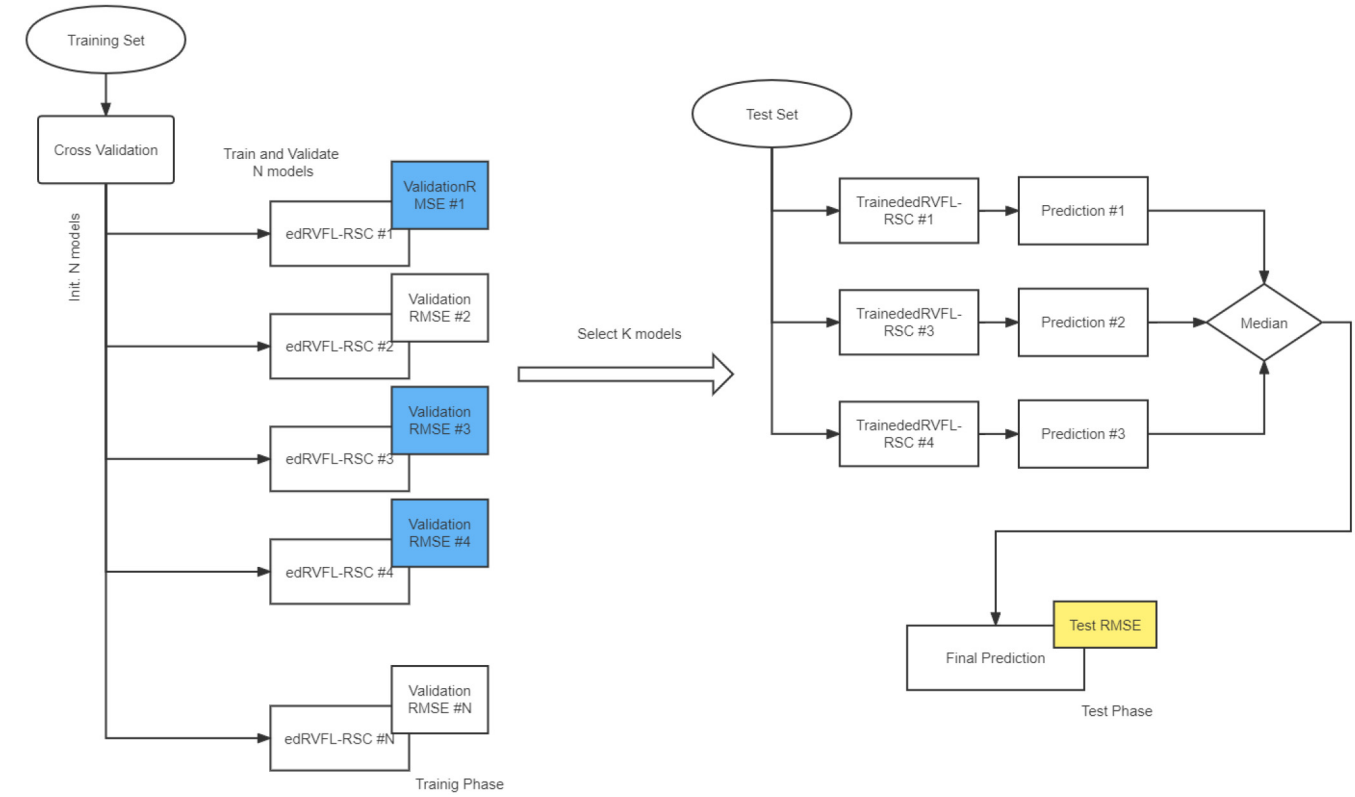


Fig. 1. Pipeline of esc-RVFL. The entire model includes a number of unique edRVFL-RSC models (as shown in 2), each with different random skip connections. *K* models with the lowest RMSE would be filtered from the *N* initialized models and passed to the testing phase. The final predictions of the models are *K* trained. All models are trained under cross-validation. The blue blocks indicate the trained model with the lowest RMSE and the yellow block means the final error.

In the context of regression, this research structurally and algorithmically improve the ensemble deep RVFL (dRVFL) proposed for pattern classification in [10] and [14]. The main contributions of this article can be summarized as follows.

1) The boosted factor (BF) is introduced to cover the latent space more comprehensively and reduce the features diminishing effect occurring in ensemble dRVFL (edRVFL).
2) edRVFL with skip connection (edRVFL-SC) with skip connection (skipping a predefined number of hidden layers) is proposed to improve the predictive power of edRVFL by improving the utilization of latent features and decreasing the corruption of high-dimension information.
3) edRVFL with random skip connection (edRVFL-RSC) with random skip connections between hidden layers is designed to obtain diverse network structures. There are numerous ways to wire the skip connections between hidden layers and it is not practically feasible to exhaustively search the entire search space manually.
4) esc-RVFL is an ensemble scheme which can keep the diversity of the edRVFL-RSC while avoid the duplication of useless features. The pipeline is shown in Fig. 1.

II. Related Works

A. RVFL Variants

Recurrent RVFL (R-RVFL) with higher nonlinearity than the vanilla RVFL was proposed in [15]. This improvement is achieved by adding recurrent feedbacks (outer feedbacks and inner feedbacks) to the network. The feedback connections act as a form of dynamic memory and provide the network with higher modeling capability.

Orthogonal polynomial expanded RVFL neural network (OPE-RVFL) is a variant of the RVFL introduced by Vuković et al. in [16]. OPE-RVFL was formulated based on the fact that orthogonal polynomial is a powerful and efficient method to approximate a nonlinear function. The architecture of OPE-RVFL can be divided into two parts. The first part is a nonlinear transformation while the second part is the same as the vanilla RVFL. The input to OPE-RVFL first undergoes a nonlinear transformation before being fed into the second part for training. In [16], four orthogonal polynomials (Chebyshev, Legendre, Laguerre, and Hermite) are used for the nonlinear transformation to obtain improved performance in regression.

In order to cope with data volatility, an exponentially expanded robust RVFL network (EE-RRVFLNN) was proposed in [17]. The robustness of this algorithm is ensured by a maximum likelihood estimator using Huber's cost function. The architecture of EE-RRVFLNN consists of a direct link and exponentially expanded mapping features that help to cope with positive dynamic volatility.

B. Other Randomized Neural Network

In addition to RVFL, there are many other varieties of randomized neural networks that have received extensive attention from researchers.

1) Extreme Learning Machine: Extreme learning machine (ELM) can be viewed as a simplified version of the RVFL without bias and direct connections from the input to the output. ELM was developed in 2004 [18] and there have been several publications comparing the original RVFL and the original ELM.

Hierarchical ELM (HELM) is a multilayer randomized neural network based on ELM introduced by Tang et al. [19]. The HELM can be divided into two components: 1) feature encoding and 2) classifier. Both the feature encoding and classifier components are based on ELM. Multiple layer HELM can be obtained by stacking the outputs of multiple autoencoders together before feeding it to a one-class or multiple-class classifier. The autoencoder part of HELM learns from the input data and comes out with a representation of the input data with lower dimensions. The autoencoder acts as a feature extraction tool that helps to identify and extract features that best represent the input data. dRVFL [20] was shown to outperform HELM.

2) Stochastic Configuration Networks: Stochastic configuration networks (SCNs) [21] are a type of randomized neural network generated incrementally by stochastic configurations. Instead fixed the number of hidden neurons among the whole training phase, SCN will increase the neurons by stochastic configuration and introduce the addition solution between the hidden neuron and final output. Recent research [22] indicates that algorithms for SCNs can be better off employing hyperparametric optimization techniques, such as the Bayesian optimization, as a better alternative than the stochastic configuration algorithm.

3) Broad Learning System: Broad learning system (BLS) [23] is another RVFL-based model with densed connections. RVFL's output layer connects both the raw input features and the hidden neurons, whereas BLS first maps the input data to the so-called feature layer which is comparable to a hidden layer. The output layer connects both the feature layer and the subsequent hidden layers.

4) Reservoir Computing and Echo State Network: With a strong theoretical foundation, reservoir computation (RC) emerged as a viable alternative to gradient descent for training RNNs. RC refers generically to a class of RNNs, including liquid state machines (LSMs) [24], echo state network (ESN) [25], [26], and other RNNs that use backpropagation decorrelation strategies [27]. In the reservoir, the echo state is constituted by randomly linked neurons [28], which plays an important role in RC and ensures that the initial state's influence vanishes after a brief transitory. This property is analogous to short-term memory [29], and it enables RC to perform well in a variety of sequential tasks. However, as a subtype of RNNs, the vast majority of RC networks and their varieties are suitable solely to sequential tasks, such as time-series classification or forecasting [30]. Deep ESN (DeepESN) [31] has also recently received attention due to the structured state space organization with multitimescale dynamics in deep RNNs which is intrinsic to the compositional nature of recursive neural modules.

The concept of ESN was introduced and evolved in [28] and [32], which is dedicated to processing sequential tasks. The key idea of ESN is randomly generated a sparse matrix as the reservoir, which can be regarded as a directed acyclic graph with weights. The input features at any time step will be map to a latent space by an input weight, which is randomly generated from a uniform distribution. Then, the latent features will be fed to the reservoir. The output will be given by the matrix multiplication between the desired output weight and the concatenation of input features, reservoir features, and the output from the last time step.

III. Preliminary

The vanilla RVFL consists of only one hidden layer. If the training data ($\mathbf{X}$) to the network has M number of instances and N features per instance, the weights ($\mathbf{W}$) randomly generated for the hidden layer will have $N \times Z$ dimension, where Z is the number of neurons per hidden layer. The product of the input data with the weights generated will become the input for a chosen nonlinear activation function (g) in the hidden layer neurons. The outputs of the hidden neurons are collectively denoted as $\mathbf{H}$. The output layer receives data $\mathbf{D}$ which is the concatenation of both input data and outputs of enhancement nodes ($\mathbf{H}$). The final output is obtained by multiplying the data matrix $\mathbf{D}$ with output layer weights β

$$\mathbf{H} = g(\mathbf{XW}) \tag{1}$$

$$\mathbf{D} = \left[\mathbf{X}; \mathbf{H}\right] \tag{2}$$

$$\mathbf{Output} = \beta * \mathbf{D}. \tag{3}$$

Since there is a closed-form solution, the β value could be computed using matrix inversion with regularization (ridge regression) or without (Moore–Penrose pseudoinverse). When using the Moore–Penrose pseudoinverse, the value of β is simply $\beta = \mathbf{D}^{+}Y$. As for regularized least square method (ridge regression), the value of β is given by

$$\text{Primal space: } \beta = \left(\mathbf{D}^T\mathbf{D} + \lambda\mathbf{I}\right)^{-1}\mathbf{D}^T Y \tag{4}$$

$$\text{Dual space: } \beta = \mathbf{D}^T\left(\mathbf{D}\mathbf{D}^T + \lambda\mathbf{I}\right)^{-1} Y \tag{5}$$

where Y is the ground truth of the training dataset while λ is the regularisation parameter. We can choose either equation (4) or (5) depending on the input dataset dimension. By multiplying β with [$\mathbf{X}$; $\mathbf{H}$], we will obtain the final predictions from the model. By comparing the prediction with the ground truth of input data, we can evaluate the performance of the model. Regularization, such as $L1$ Norm and $L2$ Norm may be added to the output layer to prevent over-fitting and improve the generalization of the model. The constraint implemented on the network parameters of RVFL reduces the complexity of the error function, turns the learning process into a quadratic optimization problem, and thus promises very fast optimization [33]. In principle, the global minimum of the cost function could be found within a single step of training if such minimum exists and is well defined. Subsequently, Ignik and Pao proved that as long as the function to be approximated by RVFL is Lipshitz continuous, then the algorithm will exhibit a similar rate of approximation error convergence as MLP, which is $O(1/n)$ [7]. This shows that RVFL is an efficient universal approximator with a closed-form solution.

Deep RVFL (dRVFL) and ensemble deep RVFL (edRVFL) [10] are two recently proposed deep variants of the RVFL. By modifying (1) and (2), the following equations which represent the computations that occur in dRVFL are obtained. For the ease of notation, the bias term in the equations is absorbed into input features and weight matrices

$$\mathbf{H}^{(i)} = \begin{cases} g\left(\mathbf{X}\mathbf{W}^{(1)}\right), & \text{if } i = 1 \\ g\left(\mathbf{H}^{(i-1)}\mathbf{W}^{(i)}\right), & \text{if } i > 1 \end{cases} \tag{6}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{d\times N}$ and $\mathbf{W} \in \mathbb{R}^{N\times N}$ are the weights between the input and 1st hidden layer and intermediate layers, namely. The weights of hidden layers are assigned randomly and kept fixed. The final features fed to the output layer $\mathbf{D}$ in a L-layer dRVFL can be defined as follows:

$$\mathbf{D} = \left[\mathbf{H}^{(1)}; \mathbf{H}^{(2)}; \ldots \mathbf{H}^{(L-1)}; \mathbf{H}^{(L)}; \mathbf{X}\right] \tag{7}$$

where L is the number of layers. Then, the output of the dRVFL can be expressed as follows:

$$\textbf{Output} = \beta * \mathbf{D}. \tag{8}$$

The hidden layers in dRVFL generate randomized internal representations in a cascading manner. The output of the previous layer is nonlinearly transformed and fed into the next layer for subsequent feature extraction. As the input data propagates down the hidden layers, different degrees of feature transformations are carried out at each level. dRVFL differentiates itself from the algorithms discussed in the previous section in such a way that instead of taking only the features from the final hidden layer, dRVFL takes into consideration all features generated from different levels. The features extracted from all levels together with the original input data (direct link) will be concatenated to generate the final outcome. The presence of direct links proves to be beneficial in improving the performance [16], [34], [35] as direct links act as a regularization mechanism to moderate the effect of randomization.

The main distinction between edRVFL and dRVFL is that edRVFL makes L independent sets of predictions [using (4) or (5)] with each set of the prediction made based on the features extracted from each of the L hidden layers, instead of making only one set of prediction as the final output. The L different predictions are ensembled together and the median of the L predictions is taken as the final output of the regression. The direct link is also included in edRVFL for the same reason mentioned above. Additionally, some recent research utilizes model performance enhancement techniques, including weighting and pruning [20].

IV. Methods

In this section, we propose three enhanced variants of the edRVFL networks.

A. Boosted Factor for edRVFL

We propose the addition of a new hyperparameter to the dRVFLs algorithm to improve their performance. By adding a new hyperparameter e^b to (6) and modifying (7), we obtain (9). We denote the new hyperparameter as "Deep Boosting" coefficient or DB, and the variants with DB added denoted as RVFL with BF from this point onward.

The main objective of proposing this work is twofold: first we want to have different priorities in the different layers of the dRVFLs and second we prefer to have more importance for the front part which is less perturbed by the randomly generated weights. In addition, we also need to take into account the range of values of the randomly generated weights. During the initialization phase, all the random weights take the range [0, 1], however, such weights cover the diversity of the input data, so we set a threshold to scale as e^b

$$\mathbf{H}^{(i)} = \begin{cases} g\left(\mathbf{A}\mathbf{W}^{(1)}e^b\right), & \text{if } i = 1 \\ g\left(\mathbf{H}^{(i-1)}\mathbf{W}^{(i)}e^{b/(L-i+1)}\right), & \text{if } i > 1 \end{cases} \tag{9}$$

where $\mathbf{A} = [\mathbf{X}; \mathbf{1}]$ with $\mathbf{1}$ for bias. The output weights β are then solved independently using (4) or (5). It is worth noting that the BF is applicable for both dRVFL and edRVFL.

B. edRVFL With Skip Connections

Inspired by the concept of deep residual learning framework [36] and Highway network in [37], we propose to add skip connections between different hidden layers in edRVFL to improve its performance. The purpose of this skip connection is not only to avoid gradient exploding or vanishing in back-propagation-based neural networks, but more importantly, to bring the view of identity mapping. In the RVFL network, the direct link is a manifestation of identity mapping, however, this direct link only maps the original input to a different hidden layer. In order to allow the information from the prior hidden layer to be transferred to the posterior layers without corruption, we use skip connections between different hidden layers. We first try edRVFL with skip connections that "skip" only a single layer (skip connection between H_{i-3} and H_{i-1}, where $i \geq 4$). For the sake of simplicity, we will denote edRVFL with skip connections that skip a single layer as edRVFL-SC. The framework of edRVFL-SC is illustrated in Fig. 2 and the pipeline can be found in Fig. 3. The computations of different hidden layers in this improved version of edRVFL (edRVFL-SC) are presented in the following:

$$\mathbf{H}^{(i)} = \begin{cases} g(\mathbf{A}\mathbf{W_1}), & \text{if } i = 1 \\ g\left(\left[\mathbf{A}; \mathbf{H_{i-1}}\right]\mathbf{W_i}\right), & \text{if } 2 \leq i \leq 3 \\ g\left(\left[\mathbf{A}; \mathbf{H_{i-3}}; \mathbf{H_{i-1}}\right]\mathbf{W_i}\right), & \text{if } i > 3 \end{cases} \tag{10}$$

and the output features $\mathbf{D}_{(i)}$ can be obtained by concatenation of input feature $\mathbf{A}$ and $\mathbf{H}^{(i)}$ as $\mathbf{D} = [\mathbf{A}; \mathbf{H}^{(i)}]$.

We can simply fuse DB with edRVFL-SC as follows:

$$\mathbf{H}^{(i)} = \begin{cases} g(\mathbf{A}\mathbf{W_1})e^b, & \text{if } i = 1 \\ g\left(\left[\mathbf{A}; \mathbf{H_{i-1}}\right]\mathbf{W_i}\right)e^{b/(L-i+1)}, & \text{if } 2 \leq i \leq 3 \\ g\left(\left[\mathbf{A}; \mathbf{H_{i-3}}; \mathbf{H_{i-1}}\right]\mathbf{W_i}\right)e^{b/(L-i+1)}, & \text{if } i > 3. \end{cases} \tag{11}$$

We shall refer to the edRVFL with skipping connections and BF as edRVFL-SC in the remainder of this article, and the detail can be found in Algorithms 1 and 2.

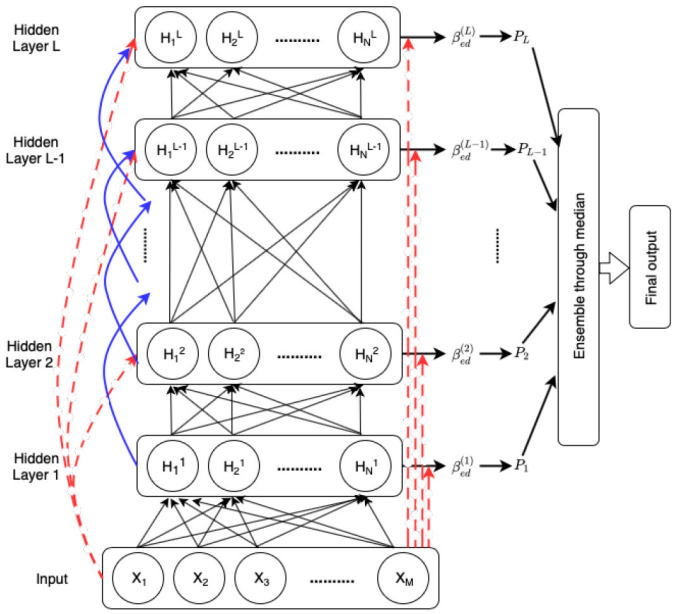


Fig. 2. Framework of an edRVFL-SC network. The arrows in blue color are the newly added skip connections. The framework of edRVFL-RSC resembles edRVFL-SC except the skip connections are connected randomly in edRVFL-RSC. Red dashed lines are the direct links and blue curves are the skip connections.

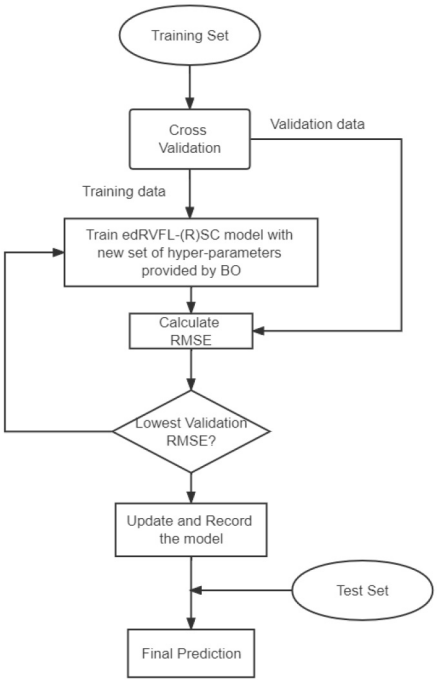


Fig. 3. Pipeline of edRVFL-SC and edRVFL-RSC. The model architecture can be found in Fig. 2. The training process is guided according to the RMSE of the model and BO stands for the Bayesian optimization.

Algorithm 1: Pseudo-Code of edRVFL-SC Training

Input: Training data ($\mathbf{X}$), ground truth (Y), maximum allowed layers ($L_{\max}$)

$\mathbf{A}_1 = [\mathbf{X}\ \mathbf{1}]$ with $\mathbf{1}$ for bias, $L = 1$
while $L \leq L_{\max}$ **do**
  Generate random weight ($\mathbf{W}_L$) with bias included.
  Append $\mathbf{W}_L$ to *weightArray*.
  Multiply $\mathbf{A}_L$ with $\mathbf{W}_L$ to obtain inputs to activation functions.
  Apply activation functions on inputs to obtain $\mathbf{H}_L$ by either equation (11).
  Append $\mathbf{H}_L$ to **H_store**.
  Obtain $\mathbf{D}_L$ by concatenating $\mathbf{H}_L$ with $\mathbf{A}_L$.
  Solve for β using $\mathbf{D}_L$ and Y by either equation (4) or (5).
  Append β to β_computed.
  if $L > 2$ **then**
    $\mathbf{A}_{L+1} = [\mathbf{A_1}\ \ \mathbf{H}_L\ \mathbf{H\_store}_{[L-2]}]$.
  else
    $\mathbf{A}_{L+1} = [\mathbf{A_1}\ \ \mathbf{H}_L]$
  Increase L by 1
return *weightArray, β_computed*

Algorithm 2: Pseudo-Code of edRVFL-SC Testing

Input: Test data ($\mathbf{X}$), ground truth (Y), number of layers learned (L_{maxL}), *weightArray*, β_computed

$\mathbf{A\_test}_1 = [\mathbf{X}\ \ \mathbf{1}]$ with $\mathbf{1}$ for bias, $L = 1$
while $L \leq L_{maxL}$ **do**
  Multiply $\mathbf{A\_test}_L$ with $weightArray_{[L]}$ to obtain inputs to activation functions.
  Apply activation functions on inputs to obtain the output as $\mathbf{H}_L$.
  Append $\mathbf{H}_L$ to $\mathbf{H}\_store$.
  Compute $\mathbf{D\_test}_L$ by concatenating $\mathbf{H}_L$ with $\mathbf{A\_test}_L$.
  if $L > 2$ **then**
    $\mathbf{A\_test}_{L+1} = [\mathbf{A\_test_1}\ \mathbf{H_L}\ \mathbf{H\_store}_{[L-2]}]$
  else
    $\mathbf{A\_test}_{L+1} = [\mathbf{A\_test_1}\ \mathbf{H_L}]$
  Increase L by 1
while $L \leq L_{maxL}$ **do**
  Compute prediction by multiplying $\mathbf{D\_test}_L$ and $\beta\_computed_L$.
  Append the prediction to predictionList.
  Increase L by 1
Compute the final prediction by aggregating the L predictions in predictionList using Median.
Compute the test RMSE using the final prediction and ground truth Y.
return *test RMSE*

C. edRVFL With Random Skip Connections

Due to the attributes of the hidden layers in the randomized neural network, the ability of feature reuse brought by the dense skip connection in [38] is not effectively enhanced. Since the neuron weights in a random neural network are randomly generated, simply doing a high-dimensional mapping of these features to a deeper hidden layer will result in invalid information being used multiple times. Thus, we use a random skip connection strategy to replace the dense skip connection strategy, and we term such network architecture with BF as edRVFL-RSC.

Specifically, edRVFL-RSC is to "skip" a random number of layers (skip connection between H_{i-1-k} and H_{i-1}, where $i \geq 4$ and k are randomly generated for each of the layer such that $k > 1$ and $i - k \geq 2$). The same idea can be extended to many-to-one skip connections where the current hidden layer is connected to multiple previous hidden layers.

D. Ensemble Skip Connection edRVFL

Ensemble skip connection edRVFL (esc-edRVFL) is an ensemble of multiple edRVFL-RSC models with skip connections to yield a final prediction. It serves the following purposes: 1) the random skip connection strategy can effectively avoid multiple reuses of useless features and 2) the increased network diversity brought by the ensemble can compensate for the potential loss of information caused by random skip connections.

During the training and validation of stage 1, we train N different edRVFL-RSC models, normally N is generally taken as the fold of cross-validation. In other words, each edRVFL-RSC is trained and tuned using different data partitions. The architectures of all the edRVFL-RSC models are different as the skip connections in each of them are randomly generated. The edRVFL-RSC models with top K performances will be chosen for the testing phase.

In the testing phase, test data is fed into the K chosen models to generate K different predictions. The K predictions are then aggregated together using the median to obtain the final prediction. Finally, the test root mean square error (RMSE) is computed using the final prediction and the ground truth of the test data.

E. Generic to Randomized Neural Networks

Our method is a variation of edRVFL, and the ensemble strategy in edRVFL is generally applicable to other types of random networks, such as the recently published edBLS [39], which is an ensemble-based version of the BLS. Our method may be used in lieu of the deep-ensemble methods in edBLS and improves the performance of the original model by, including data-boosting factor and skip connections.

V. Experimental Setup

The experiment is divided into two phases. In the first phase, the dRVFL and edRVFL proposed in [10], as well as the three newly proposed variants (edRVFL-SC, edRVFL-RSC, and esc-RVFL) are compared against three different variants of randomized neural networks (OPE-RVFL, RVFLNN, and ELM) as done in [16] and a backpropagation-based multilayer perceptron (MLP) with 15 layers. In the second phase of the experiment, the top two performers of the newly proposed randomized neural network algorithms (in the first phase) are further compared against several states of the art randomization methods. The details of the dataset are provided in the Appendix in the supplementary material. It is worth noting that all the datasets used in our experiments are from different fields in the real world [40]—housing price, abalone quality, automobile type and price, noise level, etc.

A. Experiment Design

1) Phase 1: The parameters available for tuning the dRVFLs are the number of neurons (N) in each hidden layer, the number of layers (L), ridge regression parameter (C), and activation function. The number of neurons is chosen among the following values $[32, 256]$. The number of layers for the network is capped at $L_{\max} = 35$. The C parameter is varied as such $C = 2^k$, $k = [0, 20]$ (λ in (4) and (5) is equal to $1/C$). Four different activation functions are tested. They are ReLU, Tansig, Sigmoid, and Tribas. The value of x for the DB coefficient is varied in the range of 0.05 to 3.0. normalization is applied to the hidden layers guided by the validation RMSEs. The layer normalization is added to the model to normalize the input to each of the layers if and only if the validation RMSE shows that it helps to train a better model and yield more accurate predictions than a non-normalized model. Equation (12) shows the formula of normalization being used. The importance of normalization to a model will be further discussed in Section VII

TABLE I
Activation Functions

Activation Function	Formula
ReLU	$h(s) = max(0, s)$
Tansig	$h(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$
Sigmoid	$h(s) = \frac{1}{1+e^{-s}}$
Tribas	$h(s) = max(0, 1 - \lvert s \rvert)$

$$Z = (x_i - \mu)/\sigma$$
$$\mu = \text{Mean}\ ,\ \sigma = \text{Standard Deviation}. \tag{12}$$

As for the MLP, the parameters available for tuning are the batch size, fraction rate of dropout layer, and the arguments of the Adam optimizer. The batch size for a dataset with more than 200 instances is set to 32 instances per batch, else it is set to ten instances per batch. The fraction rate of the dropout layer is fixed at 0.2. Finally, the learning rate and the exponential decay rates for the two-moment estimates are set to 0.001, 0.9, and 0.999, respectively.

Five-fold cross-validation is performed on the 70% data during the training stage. The trained model is then used for making a prediction on the reserved 30% test data and compute the test RMSE. This process is then repeated for 100 independent trials and the mean of the 100 different roots mean square errors is computed and tabulated in Table III. The RMSE is defined as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{n=1}^{N_{\text{test}}} \left(y_n - \hat{y}_n\right)^2}{N_{\text{test}}}}. \tag{13}$$

Lower RMSE implies that the prediction made by a model is closer to the ground truth. Table I shows the equation of the activation functions used in this experiment.

2) Phase 2: Our focus in this part is to compare our best three algorithms in phase 1 against ten other latest algorithms. Following the experiment design in [41], the input attributes of each dataset are rescaled into the interval $[-1, 1]$, while the target value is rescaled into the interval $[0, 1]$. The experimental design is performed over ten runs of five folds cross-validation (total 50 trials). For each run, one partition is used for testing, another one for validation, and the rest are used for training the model. In this phase, the number of neurons in each hidden layer is dropped in $[32, 512]$ while the maximum number of layers is set to $L_{\max} = 50$. The remaining hyperparameters are varied according to the ranges stated

TABLE II
HYPER-PARAMETER CONFIGURATIONS FOR RVFLS IN DIFFERENT PHASES. BAYESIAN OPTIMIZATION METHODS ARE EMPLOYED TO SELECT HYPER-PARAMETERS IN BOTH PHASES 1 AND 2

Hyper-parameters	Search Range		Type
	Phase 1	Phase 2	
N	[32,256]	[32,512]	integer
C	[0, 20]	[-20,20]	float
L	[0,35]	[0,50]	integer
b (Boosted Factor)	[0,3]	[0,3]	float

in phase 1. The considered hyper-parameters are demonstrated as Table II. We use Bayesian Optimization methods to obtain the target hyper-parameters in both Phases 1 and 2.

3) Parameter Sensitivity Analysis: In this phase, we perform ablation experiments and parameter-sensitive analysis. We first conducted a targeted study on the deep boosting methods, especially on the influence of BF, then we consider the impact of other hyper-parameters. The experiments includes: 1) the improvement on deep boosting strategies for different networks, including edRVFL-SC and edRVFL-RSC; 2) the relationship between BF and number of neurons; and 3) the effect of the different number of layers on the final result.

The datasets are selected from the datasets used in Phase 2, which belong to UCI datasets. The search range of hyper-parameters is the same as Phase 1, shown in Table II, while we use grid search to sweep all possible choices in this phase.

VI. RESULTS AND ANALYSES

A. Phase 1

In this phase, we mainly compared the differences between the RVFL series of methods and the basic ELM method. The dRVFLs are compared against the three different randomized neural network stated in [16] and a backpropagation-based MLP. The test RMSEs of the algorithms on the 29 UCI datasets are tabulated in Table III. The values in Table III are the average RMSE of 100 tests. Parametric and nonparametric statistical tests are adopted to conduct pairwise comparisons to identify the best algorithm and activation function. From Table III, it can be clearly seen that dRVFLs outperformed the randomized neural networks in [16] in all 29 datasets. The dRVFLs perform better than OPE-RVFL, RVFL, and ELM by a substantial margin. By using the two-tailed sign test, we have more than 99.99% confidence on dRVFL and dRVFL with BF to perform better than the algorithms in [16], while 100% for the edRVFL, edRVFL-SC, edRVFL-RSC, and esc-edRVFL. Generally, the edRVFL and its variants are performing better than the dRVFL.

The null hypothesis of the two-tailed sign test is that the performances of the paired algorithms are equal, and the value of N is 29. Similarly, the null hypothesis of the Wilcoxon signed-rank test is that the median difference between pairs of algorithms is zero. We will reject the null hypothesis at 0.05 significance level (reject if $p < 0.05$). Detailed discussion on dRVFL with BF, edRVFL-SC, edRVFL-RSC, and esc-edRVFL will be presented in the next section.

Algorithm 3: Pseudo-Code of esc-edRVFL

Input: Training data denoted as **X**
Init. N models on **X**.
for *each model* **do**
    Train edRVFL-RSC network
    Compute validation RMSE
return *K models with lowest validation RMSEs (K≤N)*
Input: K models, test data, ground truth
for *Each model* **do**
    Feed in Test data and generate prediction
Aggregate the K different set predictions together using Median
Compute the test RMSEs with the aggregated predicted values and ground truth
return *Test RMSE*

We compare the difference in performances between dRVFLs and backpropagation-based MLP. Table III shows that dRVFLs have better performances than MLP in most of the datasets. dRVFL, edRVFL, dRVFL with BF, edRVFL-SC, and edRVFL-RSC outperform the MLP in 18 out of 29 datasets while esc-edRVFL outperforms MLP in 22 out of 29 datasets. Comparisons are conducted based on RMSE values rounded up to six decimal places. This implies that dRVFLs have good learning capability that is on par with or better than MLP. Using both the two-tailed sign test and Wilcoxon signed-rank test, we confirm that the esc-edRVFL is significantly better than MLP at a confidence level of 0.05 and the detail of Wilcoxon signed-rank test are provided in Table IV.

Algorithm 3 in Section IV-D shows that the esc-edRVFL is validated on N different validation sets for N folds cross-validation. Using N different validation sets for N different models during the training and validation stage allows the N models to learn independently and encourage the models to form characteristics that are distinct from each other. The diversity of the N trained models is beneficial to the final ensemble model as the final model can generalize better and capture the data variations encountered in the testing phase, which in turn results in better performance. Table V shows that esc-edRVFL with N different validation sets performs better than esc-edRVFL with a single validation set.

Another worth mentioning observation is about the pattern of the optimal hyperparameters of dRVFLs. dRVFL and its variants generally require a lower value of ridge regression parameter (C) than edRVFL. We also identified that Sigmoid is the best activation function for all six dRVFLs algorithms. The relationship shows the comparisons between activation functions ($>$ stands for "is better than" or "is more preferable" in this context): Sigmoid > Tansig > ReLU > Tribas.

B. Phase 2

In this phase, we compare the proposed method against some other randomized neural networks. First, we compare the differences between our proposed method and ELM series. The results of the comparison are tabulated in Table VI.

TABLE III
AVERAGE RMSE OF 100 TESTS ON UCI DATASETS

Dataset	OPE-RVFL [16]	RVFL [6]	ELM [18]	dRVFL [10]	edRVFL [10]	dRVFL w/ BF*	edRVFL-SC*	edRVFL-RSC*	esc-RVFL*	MLP
CH	0.2161	0.2181	0.2218	0.1139	0.1158	0.1141	0.1145	0.1152	**0.1126**	0.1123
ABA	0.0954	0.0844	0.0809	0.0741	0.0738	0.0742	0.0739	0.0740	**0.0736**	0.0756
AIL	0.0545	0.0547	0.0697	0.0432	0.0429	0.0431	0.0429	0.0429	**0.0426**	0.0448
AUTO	0.0862	0.0910	0.0859	0.0796	0.0804	0.0747	0.0792	0.0776	**0.0616**	0.0675
WA	0.0171	0.0173	0.0189	0.0173	0.0166	0.0172	0.0166	0.0167	**0.0161**	0.0413
TRE	0.0126	0.0129	0.0323	0.0098	0.0094	0.0096	0.0093	0.0093	**0.0089**	0.0211
MORT	0.0082	0.0098	0.0322	0.0047	0.0045	0.0047	0.0046	0.0046	**0.0044**	0.0224
DELA	0.0389	0.0389	0.0404	0.0390	0.0386	0.0390	0.0386	0.0386	**0.0384**	0.0386
ELE	0.0917	0.0920	0.0939	0.0325	0.0328	0.0324	0.0325	0.0326	**0.0321**	0.0318
WI	0.0190	0.0190	0.0211	0.0180	0.0180	0.0180	0.0180	0.0180	**0.0179**	0.0473
YH	0.0756	0.0918	0.1194	0.0139	0.0121	0.0126	**0.0074**	0.0120	0.0111	0.0425
BNK8	0.0776	0.0557	0.0474	0.0410	0.0409	0.0408	0.0406	0.0405	**0.0394**	0.0393
H16	0.1292	0.2222	0.1964	0.0726	0.0746	0.0729	0.0729	0.0731	**0.0720**	0.0703
DELE	0.0534	0.0534	0.0548	0.0511	**0.0510**	0.0511	**0.0510**	**0.0510**	0.0511	0.0517
CA1	0.0657	0.0623	0.0636	0.0304	0.0298	0.0303	0.0297	0.0296	**0.0293**	0.0373
CA2	0.0552	0.0386	0.0586	0.0261	0.0253	0.0260	0.0254	0.0253	**0.0242**	0.0244
BH	0.0743	0.0764	0.1035	0.0772	0.0751	0.0776	0.0741	0.0742	**0.0716**	0.0792
AMPG	0.0788	0.0790	0.0817	0.0615	0.0610	0.0609	0.0606	0.0609	**0.0573**	0.0685
AIR	0.1522	0.1474	0.1509	0.0737	0.0731	0.0729	0.0688	0.0666	**0.0602**	0.0472
TRI	0.8592	0.9574	0.9025	0.1747	0.1701	0.1722	0.1718	0.1696	**0.1680**	0.1683
PYRI	0.1654	0.2560	0.4828	0.1540	0.1492	0.1523	0.1503	0.1486	**0.1290**	0.2054
DIA	0.3324	0.4218	0.2648	0.1887	0.1910	0.1881	0.1949	0.1897	**0.1788**	0.2128
KIN8	0.1289	0.1282	0.1409	**0.0637**	0.0769	0.0641	0.0694	0.0692	0.0650	0.0557
BNK32	0.1322	0.1279	0.1328	0.1006	0.0993	0.1006	0.0996	0.0995	**0.0984**	0.1036
PUM32	0.1565	0.1565	0.1565	0.1503	0.1484	0.1493	0.1488	0.1489	**0.1471**	0.0512
PUM8	0.1554	0.1574	0.1712	0.1390	0.1391	0.1386	0.1385	0.1383	**0.1358**	0.1426
WIS	0.4206	0.4177	0.3000	0.2773	0.2929	0.2748	0.2889	0.2919	**0.2662**	0.3116
MCPU	0.1394	0.1671	0.1102	0.0963	0.0970	0.0887	0.0943	0.0998	**0.0762**	0.1195
FRI	0.0541	0.0597	0.0953	**0.0343**	0.0401	**0.0343**	0.0382	0.0378	0.0358	0.0366
$+$	29	29	29	26	28	26	27	28		22
$-$	0	0	0	2	1	2	2	1		7
$\approx$	0	0	0	1	0	1	0	0		0

* All values are rounded up to 4 decimal places. The lowest RMSE for each of the dataset is highlighted in green while the algorithm with the best performance among the dRVFLs is highlighted in bold. (+) indicates the number of datasets that the esc-edRVFL outperforms the comparative algorithm. ($-$) denotes the number of datasets that the esc-edRVFL performs worse than the comparative algorithm. ($\approx$) indicates the number of datasets that the esc-edRVFL has the same RMSE as the comparative algorithm. Algorithms with * are the newly proposed variants with boosted factor in this paper.

TABLE IV
PAIRWISE WILCOXON SIGNED-RANK TEST ($\alpha = 0.05$)

Algorithm 2 \ Algorithm 1	OPE-RVFL	RVFL	ELM	dRVFL	edRVFL	dRVFL w/ BF	edRVFL-SC	edRVFL-RSC	esc-edRVFL	MLP
OPE-RVFL		$\approx$	$\approx$	✓	✓	✓	✓	✓	✓	✓
RVFL	$\approx$		$\approx$	✓	✓	✓	✓	✓	✓	✓
ELM	$\approx$	$\approx$		✓	✓	✓	✓	✓	✓	✓
dRVFL	✗	✗	✗		$\approx$	✓	$\approx$	$\approx$	✓	$\approx$
edRVFL	✗	✗	✗	$\approx$		$\approx$	✓	✓	✓	$\approx$
dRVFL w/ BF	✗	✗	✗	✗	$\approx$		$\approx$	$\approx$	✓	$\approx$
edRVFL-SC	✗	✗	✗	$\approx$	✗	$\approx$		$\approx$	✓	$\approx$
edRVFL-RSC	✗	✗	✗	$\approx$	✗	$\approx$	$\approx$		✓	$\approx$
esc-edRVFL	✗	✗	✗	✗	✗	✗	✗	✗		✗
MLP	✗	✗	✗	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	✓	

* The sign tests are based on the average RMSE of 100 tests on UCI datasets rounded up to 4 decimal places. $\approx$ indicates there is no significant difference between algorithm 1 and algorithm 2, ✓ indicates algorithm 1 is significantly better than algorithm 2, ✗ indicates algorithm 1 is significantly worse than algorithm 2. The tests are conducted at $\alpha = 0.05$ significant level.

TABLE V
ESC-EDRVFL DESIGN COMPARISONS

Dataset	N different validation sets	Same validation set
ABA	**0.073621**	0.074700
WA	**0.016114**	0.017550
MORT	**0.004375**	0.006310
BNK8	**0.039368**	0.042920
CA1	**0.029327**	0.032020
BH	**0.071609**	0.091401
KIN8	**0.065012**	0.075783
PUM8	**0.135844**	0.143736
WIS	**0.266206**	0.267718
MCPU	0.076195	**0.069894**

* The values tabulated in the table are the test RMSEs of esc-edRVFL with different designs (esc-edRVFL validated on N different validation sets versus one validation set for N different edRVFL-RSC models in stage 1) for 10 randomly selected datasets. All the values are rounded up to 6 decimal places. The lowest RMSE for each of the dataset is highlighted in bold.

Table VII shows that the esc-edRVFL, edRVFL-SC, and edRVFL-RSC are significantly better than all the ELM-based algorithms. We also explore the performance differences between our proposed approaches and the SCN-based methods. Table VIII shows that our method is superior to both the basic SCN [21] and its recent variant, Bidirectional SCN [51]. We also compare with the fuzzy broad learning system (Fuzzy BLS) [52] approach in Table IX and experimentally demonstrate that our algorithm is significantly better.

Furthermore, we compared the performance of two classical conventional machine learning methods on these datasets, support vector regression (SVR) and K-neighbors regression (KNR). As can be seen, the randomized network generally outperforms classical machine learning algorithms.

We also conduct nonparametric statistical comparison of the algorithms using the Friedman test, and the details are shown in the Appendix in the supplementary material.

TABLE VI
COMPARISON OF AVERAGE RMSE OF BENCHMARK REGRESSION DATASET WITH ELMS AND PROPOSED METHODS

Dataset	ELM	OP-ELM [43]	E-ELM [44]	IPSO-ELM [45]	SaDE-ELM [46]	O-ELM [47]	TAF-ELM [48]	GE-ELM [49]	EE-ELM [50]	Meta-ELM [51]	esc-edRVFL*	edRVFL-SC*	edRVFL-RSC*
QUA	0.1722	0.1731	0.1723	0.1719	0.1720	0.1721	0.1724	0.1719	0.1708	0.1721	**0.1704**	0.1709	0.1706
ANA	0.1359	0.1068	0.0945	0.1506	0.1409	0.0805	0.1028	0.0848	0.0550	0.1296	**0.0368**	0.0397	0.0416
ABA	0.0772	0.3746	0.0783	0.0764	0.0771	0.0775	0.0767	0.0755	**0.0725**	0.0775	0.0738	0.0738	0.0738
DELA	0.0389	0.0399	0.0398	0.0388	0.0387	0.0407	0.0389	0.0385	**0.0375**	0.0387	0.0382	0.0384	0.0384
BNK8	0.0460	0.1014	0.0627	0.0432	0.0429	0.0469	0.0432	0.0431	0.0417	0.0447	**0.0392**	0.0398	**0.0392**
BNK32	0.1117	0.1310	0.1095	0.1101	0.1033	0.1053	0.1061	0.1033	0.1012	0.1049	**0.0985**	0.0986	0.0986
CAI	0.0372	0.5631	0.0454	0.0345	0.0339	0.0581	0.0356	0.0327	0.0310	0.0371	**0.0287**	0.0292	0.0288
CA2	0.0481	4.9265	0.0482	0.0350	0.0327	0.0578	0.0392	0.0316	0.0300	0.0408	**0.0261**	0.0269	0.0266
PUM8	0.1686	0.1777	0.1629	0.1467	0.1416	0.1465	0.1471	0.1518	0.1469	0.1589	**0.1381**	0.1389	0.1384
PUM32	0.1535	0.1554	0.1644	0.1528	0.1528	**0.1232**	0.1543	0.1515	0.1483	0.1524	0.1464	0.1466	0.1469
KIN	0.1109	0.1242	0.1318	0.0899	0.0966	0.1157	0.0957	0.0944	0.0920	0.1000	**0.0591**	0.0616	0.0599
DELE	0.0532	0.0537	0.0539	0.0532	0.0531	0.0542	0.0532	0.0529	0.0522	0.0531	**0.0509**	**0.0509**	**0.0509**
CCPP	0.0541	0.0544	0.0567	0.0541	0.0540	0.0569	0.0538	0.0535	0.0527	0.0541	**0.0494**	0.0496	**0.0494**
COIL	0.2327	0.2340	0.2331	0.2324	0.2321	0.2313	0.2325	0.2307	0.2269	0.2324	**0.1105**	0.2346	0.2360
CBMI	0.2299	0.2932	0.2447	0.0228	0.0105	0.0856	0.0007	0.0410	0.0749	0.1008	**0.0003**	**0.0003**	**0.0003**
CBM2	0.2635	0.2996	0.2798	0.0398	0.0167	0.1252	0.0018	0.1015	0.1469	0.1777	**0.0003**	0.0005	0.0005
AIL	0.0455	0.0529	0.0509	0.0455	0.0452	0.0460	0.0455	0.0447	0.0437	0.0452	0.0426	0.0427	**0.0425**
POL	0.3490	0.4680	0.2971	0.2422	0.2616	0.2655	0.2087	0.2942	0.2901	0.3097	0.1106	0.1029	**0.0976**
ELE	0.0370	0.3568	0.0487	0.0360	0.0356	0.0416	0.0361	0.0346	0.0341	0.0353	**0.0315**	0.0319	0.0316
BIK	0.0130	0.0623	0.0651	0.0042	0.0005	0.0065	0.0045	0.0052	0.0049	0.0063	**0.0000**	**0.0000**	**0.0000**
CH	0.1299	0.1609	0.1396	0.1277	0.1283	0.1445	0.1291	0.1250	0.1219	0.1297	0.1107	0.1096	**0.1093**
CEN8	0.0674	0.1048	0.0751	0.0667	0.0668	0.0761	0.0659	0.0648	0.0632	0.0669	**0.0604**	0.0609	0.0607
CEN16	0.0829	0.1336	0.0926	0.0829	0.0819	0.0860	0.0813	0.0801	0.0789	0.0808	**0.0708**	0.0711	0.0720
FRI	0.0745	0.0839	0.0856	0.0564	0.0569	0.0538	0.0669	0.0643	0.0632	0.0672	**0.0360**	0.0365	0.0361
2DP	0.0692	0.0874	0.0929	0.0524	0.0497	0.0596	0.0598	0.0501	0.0504	0.0551	0.0412	0.0412	**0.0411**
MV	0.0476	0.0882	0.0819	0.0179	0.0225	0.0262	0.0337	0.0228	0.0234	0.0293	0.0082	0.0093	**0.0069**
CASP	0.0535	0.0601	0.0576	0.0535	0.0528	0.0554	0.0526	0.0522	0.0510	0.0540	**0.0466**	0.1976	0.1977
KD	0.0860	0.1014	0.0942	0.0789	0.0879	0.0893	0.0829	0.0754	0.0730	0.0826	**0.0451**	0.0484	0.0512
CT	0.1275	0.1362	0.1571	0.1005	0.1034	0.0957	0.1076	0.0960	0.0948	0.1005	**0.0505**	0.0585	0.0572
BFB	0.0232	0.0236	0.0505	0.0227	0.0221	0.0209	0.0223	0.0220	0.0218	0.0269	0.0179	**0.0177**	**0.0177**
KU	0.0049	0.0187	0.0123	0.0036	0.0040	0.0045	0.0040	0.0036	0.0031	0.0045	**0.0027**	0.0028	**0.0027**

* All values are rounded up to 4 decimal places. The lowest RMSE for each of the dataset is highlighted in bold. Algorithms with * are the newly proposed variants in this paper.

TABLE VII
PAIRWISE WILCOXON SIGNED-RANK TEST ($\alpha = 0.05$)

Algorithm 2 \ Algorithm 1	ELM	OP-ELM	E-ELM	IPSO-ELM	SaDE-ELM	O-ELM	TAF-ELM	GE-ELM	EE-ELM	Meta-ELM	esc-edRVFL	edRVFL-SC	edRVFL-RSC
ELM		✗	✗	✓	✓	≈	✓	✓	✓	✓	✓	✓	✓
OP-ELM	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
E-ELM	✓	✗		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
IPSO-ELM	✗	✗	✗		≈	✗	≈	≈	≈	✗	✓	✓	✓
SaDE-ELM	✗	✗	✗	≈		✗	≈	≈	≈	✗	✓	✓	✓
O-ELM	≈	✗	✗	✓	✓		≈	✓	✓	≈	✓	✓	✓
TAF-ELM	✗	✗	✗	≈	≈	≈		✓	✓	≈	✓	✓	✓
GE-ELM	✗	✗	✗	≈	≈	✗	✗		✓	✗	✓	✓	✓
EE-ELM	✗	✗	✗	≈	≈	✗	✗	✗		✗	✓	✓	✓
Meta-ELM	✗	✗	✗	✓	✓	≈	≈	✓	✓		✓	✓	✓
esc-edRVFL	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗		✗	✗
edRVFL-SC	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓		≈
edRVFL-RSC	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	≈	

* The sign tests are based on the average RMSE of 10 tests on benchmark regression datasets rounded up to 4 decimal places. ≈ indicates there is no significant difference between algorithm 1 and algorithm 2, ✓ indicates algorithm 1 is significantly better than algorithm 2, ✗ indicates algorithm 1 is significantly worse than algorithm 2. The tests are conducted at $\alpha = 0.05$ significant level.

TABLE VIII
COMPARISON OF AVERAGE RMSE OF BENCHMARK REGRESSION DATASET WITH SCN-BASED METHOD, SVR, KNR, AND PROPOSED METHODS

	esc-edRVFL	edRVFL-SC	edRVFL-RSC	BSCN [52]	SCN [21]	SVR	KNR
CBM1	0.0003	0.0003	0.0003	0.0003	0.0005	0.1857	0.0008
AIR	**0.0602**	0.0686	0.0663	0.0652	0.1463	0.1277	0.1133
CBM2	**0.0003**	0.0005	0.0005	**0.0003**	0.0006	0.1333	0.0633
YH	**0.0074**	0.0111	0.0123	0.1281	0.142	0.1531	0.1569

* All values are rounded up to 4 decimal places. The lowest RMSE for each of the dataset is highlighted in bold.

TABLE IX
COMPARISON OF AVERAGE RMSE OF BENCHMARK REGRESSION DATASET WITH FUZZY BLS, SVR, KNR, AND PROPOSED METHODS

	esc-edRVFL	edRVFL-SC	edRVFL-RSC	Fuzzy BLS [53]	SVR	KNR
CH	**0.1114**	0.1145	0.1151	0.1283	0.1377	0.1393
AMPG	**0.0571**	0.0606	0.0608	0.0739	0.1054	0.0794
CEN8	**0.0604**	0.0609	0.0607	0.0655	0.1262	0.0722
2DP	**0.0411**	0.0412	0.0412	0.0412	0.0764	0.0525
BNK8	0.0392	0.0398	0.0398	**0.0365**	0.1293	0.1108
KIN	**0.0591**	0.0616	0.0599	0.0698	0.1085	0.0891
ABA	0.0734	0.0741	**0.0729**	0.0745	0.1436	0.1186

* All values are rounded up to 4 decimal places. The lowest RMSE for each of the dataset is highlighted in bold.

TABLE X
PERFORMANCE COMPARISON BETWEEN ALGORITHMS WITH/WITHOUT DEEP BOOSTING

	dRVFL w/o BF	dRVFL w/ BF	edRVFL-SC w/o BF	edRVFL-SC w/ BF	edRVFL-RSC w/o BF	edRVFL-RSC w/ BF
AIL	0.0432	0.0431	0.043	0.0429	0.0429	0.0429
TRE	0.0098	0.0096	0.0094	0.0093	0.0092	0.0093
YH	0.0139	0.0126	0.008	0.0074	0.0121	0.012
TRI	0.1747	0.1722	0.1719	0.1718	0.1699	0.1696
PUM8	0.139	0.1386	0.1386	0.1385	0.1383	0.1383
Δ	0.0045		0.001		0.0006	

Δ means the difference between the RMSE of algorithm with BF and the one without BF. The value is the average of the experiment datasets.

C. Parameter Sensitivity Analysis

1) Impact of Deep Boosting: We conduct the experiments on the algorithms include dRVFL, dRVFL w/ BF, edRVFL-SC w/o BF, edRVFL-SC, edRVFL-RSC w/o BF, and edRVFL-RSC to evaluate the performance improvement provided by the deep boosting method.

From Table X, we can see the deep boosting methods improve the performance on all three deep variants. The improvement of dRVFL is the most significant, as the output weight of dRVFL is calculated by the concatenation of all the intermediate features, and the features in the shallow area play a much more important role than the deeper ones. In edRVFL series, as each layer involve the input patterns from the direct link, The loss of information is not evident at the deep area.

2) Connection Between the Boosted Factor and the Layer Scale: In this part, we choose the dRVFL method to test the BF for achieving the best-performing network with different numbers of neurons. The results are given in Table XI. We can see the optimized BF will increase along the increase of neurons. This is another way of verifying that simply increasing the width of the network does not improve the performance of the model.

3) Sensitivity of Layer Numbers: We show the performance of proposed methods at different depths in the Appendix in the

TABLE XI
CONNECTION BETWEEN THE BF AND THE LAYER SCALE

Dataset	AIL			TRE			YH		
#Neurons	64	128	512	64	128	512	64	128	512
b	2.313	2.098	1.255	2.663	1.422	1.389	1.952	1.886	1.425
RMSE	0.0436	0.0433	0.0441	0.011	0.01	0.0099	0.0126	0.0126	0.0128
Dataset	TRI			PUM8					
#Neurons	64	128	512	64	128	512			
b	2.708	2.554	2.153	1.415	1.201	1.09			
RMSE	0.1738	0.1727	0.1725	0.1386	0.1388	0.1389			

supplementary material. We can observe that the performance of the model gets progressively better as the depth increases in the early stages, however, after the network reaches 30 layers, the performance of the network stops improving or decreases. The reason for this is that the random weights of dRVFL introduce an excessive amount of noise to the information propagated to the deeper layers, making it difficult for the excessively deeper parts to effectively capture feature information, thereby degrading the overall network performance.

VII. DISCUSSION

As mentioned earlier, all the weights in the randomized neural network are randomly generated and kept constant throughout the whole training process. This has led to a lot of debates and doubts on the learning capabilities of RNN. The work by Giryes et al. managed to shed some light on this long debate. Data fed into any neural network at the input stage can be adequately viewed as points with different angles in a multidimensional space. Giryes et al. [53] showed that for a classification problem, the random weights in each layer of neural network distorts the Euclidean distances between different classes of input data proportionally to the angle of the input points. Thus, the smaller the angle of the input, the greater is the shrinkage of the distance and vice versa. Larger separation among classes makes the data easier to be classified. The model is able to learn useful patterns from the data by prioritizing their distorted angles.

However, excessive distortion may even reduce the network's performance. Due to the nature of RVFL, randomly generated weights may contain many redundant and invalid information, and a proper neuron eliminate approach may mitigate excessive distortion effects, such as the neuron selection according to importance of each neuron. In some classification tasks, ranking neurons according to the magnitude of their weights and sequentially discarding low-weight neurons can effectively improve the accuracy of the network.

At the same depth of randomized networks (equal L), the ensemble strategy has a significant advantage in terms of both time complexity and space complexity, which means edRVFL is generally faster than dRVFL in terms of training consumption. In the dRVFL, as we concatenate all intermediate outputs to calculate the final output weight β, the weight matrix will be huge as $\beta \in \mathbb{R}^{(d+N)^2}$, and the algorithm to obtain such matrix inverse needs either $\mathcal{O}((d+N)^3)$ time or $\mathcal{O}((d+N)^2)$ memory, which contradicts the simple and low-consumption concept of RVFL series. In the edRVFL, each sublayer requires an output weight β_l, but this weight matrix is only $\beta_l \in \mathbb{R}^{(d+N)^2}$, and the time for all β_l calculation only requires $\mathcal{O}(L(d+N)^3)$. The testing stage does not involve matrix inversion, but only matrix multiplication, which is why the testing is always much faster than training. Our result shows that the training of RVFLs is much shorter than those of MLP. This is expected as the design and internal computation of dRVFLs are generally much simpler than MLP. Unlike MLP, dRVFLs do not use backpropagation and gradient descent to fine-tune the weight of the hidden layers. This explains why dRVFLs are much faster than the MLP in both training phases.

The addition of the newly proposed deep boosting coefficient is inspired by the notion that the features extracted from the lower layer will be diminished gradually as they are passing down to the higher layer. To counteract this diminishing effect, we introduced the deep boosting coefficient (e^b, $b > 0$) to "boost" the features extracted in the previous layers. This boosting technique has successfully boosted the overall performance of dRVFL variants.

The result presented in the previous section shows that the addition of skip connections between different hidden layers improves the performance of edRVFL by a large margin in most of the datasets. The skip connections provide extra paths for more information to flow from the lower layers to the upper layers. In a deep neural network, different types of information will be extracted at different levels of layers. The skip connections coming from the lower layers to the upper layers are beneficial to the model as the model can now generate more diverse features leading to a better decision-making capability.

Versatility and extendability are also features of our approach. For ensemble strategies and skip connection methods, they can be applied to any type of randomized neural network with direct links to improve performance [10], [54]. Although the datasets we used are drawn from open-source UCI repository [40], the raw signals are collected from different fields in the real world. Furthermore, our approach can also be applied to other practical areas to solve the general regression problems, such as tidal turbine vibration [55], lateral force [56], and driver workload prediction [57].

VIII. CONCLUSION

In this article, we presented a detailed evaluation of the deep RVFL (dRVFL) and ensembled deep RVFL (edRVFL) which were proposed recently in [10] on regression problems. In addition, we introduced four novel dRVFL variants (dRVFL with BF, edRVFL-SC, edRVFL-RSC, and esc-edRVFL) and one deep boosting methods. Through a comprehensive evaluation, we concluded that the dRVFLs perform better than the other randomized neural networks in most circumstances. Furthermore, we compared the dRVFLs with MLPs and found out that the performances of the dRVFLs are actually on par or better than MLP. The newly proposed variants show significant improvement when compare with the original dRVFL variants. The findings of this article can serve as a guideline for designing a proper dRVFL network for solving regression problems. Different ensemble approaches will be investigated in the future to produce more accurate final predictions, and

the introduction of additional neural networks to process the output of subnetworks in addition to the usage of the median will be investigated in the future. While this work only focuses on regression tasks, our methods may also be migrated to the classification tasks. By substituting multiple neurons for single floating neurons, the model could handle the classification tasks. We leave the exploration of the method for classification tasks, and the application of our algorithms to other practical problems are promising future directions.

ACKNOWLEDGMENT

Open Access funding provided by the Qatar National Library.

REFERENCES

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[4] L. Zhang and P. N. Suganthan, "A survey of randomized algorithms for training neural networks," *Inf. Sci.*, vols. 364–365, pp. 146–155, Oct. 2016.

[5] A. K. Malik, R. Gao, M. Ganaie, M. Tanveer, and P. N. Suganthan, "Random vector functional link network: Recent developments, applications, and future directions," 2022, *arXiv:2203.11316*.

[6] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.

[7] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, Nov. 1995.

[8] D. Needell, A. A. Nelson, R. Saab, and P. Salanevich, "Random vector functional link networks for function approximation on manifolds," 2020, *arXiv:2007.15776*.

[9] L. Zhang and P. N. Suganthan, "A comprehensive evaluation of random vector functional link networks," *Inf. Sci.*, vol. 367, pp. 1094–1105, Nov. 2016.

[10] Q. Shi, R. Katuwal, P. Suganthan, and M. Tanveer, "Random vector functional link neural network based ensemble deep learning," *Pattern Recognit.*, vol. 117, Sep. 2021, Art. no. 107978.

[11] R. Gao, L. Du, P. N. Suganthan, Q. Zhou, and K. F. Yuen, "Random vector functional link neural network based ensemble deep learning for short-term load forecasting," *Expert Syst. Appl.*, vol. 206, Nov. 2022, Art. no. 117784.

[12] X. Li, Y. Yang, N. Hu, Z. Cheng, and J. Cheng, "Discriminative manifold random vector functional link neural network for rolling bearing fault diagnosis," *Knowl.-Based Syst.*, vol. 211, Jan. 2021, Art. no. 106507.

[13] M. Hu and P. N. Suganthan, "Representation learning using deep random vector functional link networks for clustering," *Pattern Recognit.*, vol. 129, Sep. 2022, Art. no. 108744.

[14] M. Hu, R. Gao, P. N. Suganthan, and M. Tanveer, "Automated layer-wise solution for ensemble deep Randomized feed-forward neural network," *Neurocomputing*, vol. 514, pp. 137–147, Dec. 2022.

[15] Ö. F. Ertuğrul, "A novel randomized recurrent artificial neural network approach: Recurrent random vector functional link network," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 27, no. 6, pp. 4246–4255, 2019.

[16] N. Vuković, M. Petrović, and Z. Miljković, "A comprehensive experimental evaluation of orthogonal polynomial expanded random vector functional link neural networks for regression," *Appl. Soft Comput.*, vol. 70, pp. 1083–1096, Sep. 2018.

[17] L. Priyadarshini, P. Dash, and S. Dhar, "A new exponentially expanded robust random vector functional link network based MPPT model for local energy management of PV-battery energy storage integrated microgrid," *Eng. Appl. Artif. Intell.*, vol. 91, May 2020, Art. no. 103633.

[18] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, 2004, pp. 985–990.

[19] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016.

[20] Q. Shi, M. Hu, R. Katuwal, and P. N. Suganthan, "Weighting and pruning based ensemble deep random vector functional link network for tabular data classification," *Pattern Recognit.*, vol. 132, Dec. 2022, Art. no. 108879.

[21] D. Wang and M. Li, "Stochastic configuration networks: Fundamentals and algorithms," *IEEE Trans. Cybern.*, vol. 47, no. 10, pp. 3466–3479, Oct. 2017.

[22] M. Hu and P. N. Suganthan, "Experimental evaluation of stochastic configuration networks: Is SC algorithm inferior to hyper-parameter optimization method?" *Appl. Soft Comput.*, vol. 126, Sep. 2022, Art. no. 109257.

[23] C. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.

[24] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, no. 11, pp. 2531–2560, 2002.

[25] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 15, 2002, pp. 609–616.

[26] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Comput. Sci. Rev.*, vol. 3, no. 3, pp. 127–149, 2009.

[27] U. D. Schiller and J. J. Steil, "Analyzing the weight dynamics of recurrent learning algorithms," *Neurocomputing*, vol. 63, pp. 5–23, Jan. 2005.

[28] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," German Nat. Res. Center Inf. Technol., Bonn, Germany, GMD Rep. 148, 2001.

[29] S.-X. Lun, X.-S. Yao, and H.-F. Hu, "A new echo state network with variable memory length," *Inf. Sci.*, vol. 370, pp. 103–119, Nov. 2016.

[30] C. Sun, M. Song, S. Hong, and H. Li, "A review of designs and applications of echo state networks," 2020, *arXiv:2012.02974*.

[31] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep reservoir computing: A critical experimental analysis," *Neurocomputing*, vol. 268, pp. 87–99, Dec. 2017.

[32] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[33] D. Husmeier, "Random vector functional link (RVFL) networks," in *Neural Networks for Conditional Probability Estimation*. London, U.K.: Springer, 1999, pp. 87–97.

[34] Y. Dash, S. K. Mishra, S. Sahany, and B. K. Panigrahi, "Indian summer monsoon rainfall prediction: A comparison of iterative and non-iterative approaches," *Appl. Soft Comput.*, vol. 70, pp. 1122–1134, Sep. 2018.

[35] R. Katuwal, P. N. Suganthan, and L. Zhang, "An ensemble of decision trees with random vector functional link networks for multi-class classification," *Appl. Soft Comput.*, vol. 70, pp. 1146–1153, Sep. 2018.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[37] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2377–2385.

[38] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[39] C. Zhang, S. Ding, L. Guo, and J. Zhang, "Broad learning system based ensemble deep model," *Soft Comput.*, vol. 26, pp. 7029–7041, Apr. 2022.

[40] D. Dua and C. Graff. "UCI machine learning repository." 2017. [Online]. Available: http://archive.ics.uci.edu/ml

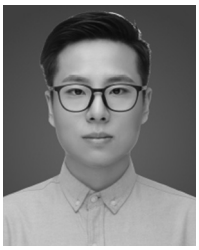
[41] P. Musikawan, K. Sunat, Y. Kongsorot, P. Horata, and S. Chiewchanwattana, "Parallelized metaheuristic-ensemble of heterogeneous feedforward neural networks for regression problems," *IEEE Access*, vol. 7, pp. 26909–26932, 2019.

[42] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally pruned extreme learning machine," *IEEE Trans. neural Netw.*, vol. 21, no. 1, pp. 158–162, Jan. 2010.

[43] Q.-Y. Zhu, A. K. Qin, P. N. Suganthan, and G.-B. Huang, "Evolutionary extreme learning machine," *Pattern Recognit.*, vol. 38, no. 10, pp. 1759–1763, 2005.

[44] F. Han, H.-F. Yao, and Q.-H. Ling, "An improved evolutionary extreme learning machine based on particle swarm optimization," *Neurocomputing*, vol. 116, pp. 87–93, Sep. 2013.

[45] J. Cao, Z. Lin, and G.-B. Huang, "Self-adaptive evolutionary extreme learning machine," *Neural Process. Lett.*, vol. 36, no. 3, pp. 285–305, 2012.
[46] T. Matias, F. Souza, R. Araújo, and C. H. Antunes, "Learning of a single-hidden layer feedforward neural network using an optimized extreme learning machine," *Neurocomputing*, vol. 129, pp. 428–436, Apr. 2014.
[47] B. Li, Y. Li, and X. Rong, "The extreme learning machine learning algorithm with tunable activation function," *Neural Comput. Appl.*, vol. 22, no. 3, pp. 531–539, 2013.
[48] X. Xue, M. Yao, Z. Wu, and J. Yang, "Genetic ensemble of extreme learning machine," *Neurocomputing*, vol. 129, pp. 175–184, Apr. 2014.
[49] D. Wang and M. Alhamdoosh, "Evolutionary extreme learning machine ensembles with size control," *Neurocomputing*, vol. 102, pp. 98–110, Feb. 2013.
[50] S. Liao and C. Feng, "Meta-ELM: ELM with ELM hidden nodes," *Neurocomputing*, vol. 128, pp. 81–87, Mar. 2014.
[51] W. Cao, Z. Xie, J. Li, Z. Xu, Z. Ming, and X. Wang, "Bidirectional stochastic configuration network for regression problems," *Neural Netw.*, vol. 140, pp. 237–246, Aug. 2021.
[52] S. Feng and C. L. P. Chen, "Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 414–424, Feb. 2020.
[53] R. Giryes, G. Sapiro, and A. M. Bronstein, "Deep neural networks with random gaussian weights: A universal classification strategy?" *IEEE Trans. Signal Process.*, vol. 64, no. 13, pp. 3444–3457, Jul. 2016.
[54] Z. Zhang, Y. Cai, and W. Gong, "Evolution-driven randomized graph convolutional networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Mar. 17, 2022, doi: 10.1109/TSMC.2022.3158276.
[55] G. S. Galloway, V. M. Catterson, C. Love, A. Robb, and T. Fay, "Modeling and interpretation of tidal turbine vibration through weighted least squares regression," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 4, pp. 1252–1259, Apr. 2020.
[56] B. H. G. Barbosa, N. Xu, H. Askari, and A. Khajepour, "Lateral force prediction using gaussian process regression for intelligent tire systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 8, pp. 5332–5343, Aug. 2022.
[57] D. Yi, J. Su, C. Liu, and W.-H. Chen, "New driver workload prediction using clustering-aided approaches," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 1, pp. 64–70, Jan. 2019.



Minghui Hu (Student Member, IEEE) received the B.Eng. degree in electrical engineering from Dalian Maritime University, Dalian, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

His research interest includes randomized neural networks, deep learning, and computer vision.

Jet Herng Chion received the B.Eng. degree in electrical and electronics engineering from Nanyang Technological University, Singapore, in 2020.

Ponnuthurai Nagaratnam Suganthan (Fellow, IEEE) received the B.A. and M.A. degrees in electrical and information engineering from the University of Cambridge, Cambridge, U.K., in 1990 and 1992, respectively, and the Honorary Doctorate (i.e., Doctor Honoris Causa) degree in computer science from the University of Maribor, Maribor, Slovenia, in 2020.

He served as a Predoctoral Research Assistant with The University of Sydney, Sydney, NSW, Australia, from 1995 to 1996, and a Lecturer with the University of Queensland from 1996 to 1999. Since August 2022, he has been with the KINDI Centre for Computing Research, Qatar University, Doha, Qatar.

Dr. Suganthan's coauthored SaDE paper won the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award in 2012. He was selected as one of the Highly Cited Researchers by Thomson Reuters Science Citations yearly from 2015 to 2022 in computer science. He has been a founding Co-Editor-in-Chief of Swarm and Evolutionary Computation since 2010, an *SCI Indexed Journal* (Elsevier). He served as the General Chair of the IEEE SSCI 2013. He was an elected AdCom Member of the IEEE Computational Intelligence Society (CIS) from 2014 to 2016. He is/was an Associate Editor of nine journals. He was an IEEE CIS Distinguished Lecturer (DLP) from 2018 to 2021.

Rakesh Kumar Katuwal received the B.Eng. degree in electrical engineering from Kathmandu University, Dhulikhel, Nepal, in 2014, and the Ph.D. degree in electrical engineering with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2019.

His research interest includes various ensemble learning methods, deep learning, and computer vision.