Neurocomputing 514 (2022) 137–147

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Automated layer-wise solution for ensemble deep randomized feed-forward neural network

Minghui Hu [a], Ruobin Gao [a], Ponnuthurai N. Suganthan [b,*], M. Tanveer [c]

[a] Nanyang Technological University, Singapore, Singapore
[b] KINDI Center for Computing Research, College of Engineering, Qatar University, Doha, Qatar
[c] Department of Mathematics, Indian Institute of Technology Indore, Simrol, Indore, India

ARTICLE INFO

Article history:
Received 2 March 2022
Revised 6 August 2022
Accepted 24 September 2022
Available online 29 September 2022
Communicated by Zidong Wang

Keywords:
Randomized feed-forward neural network
Random vector functional link
Automated machine learning
Bayesian optimization
Ensemble deep random vector functional link

ABSTRACT

The randomized feed-forward neural network is a single hidden layer feed-forward neural network that enables efficient learning by optimizing only the output weights. The ensemble deep learning framework significantly improves the performance of randomized neural networks. However, the framework's capabilities are limited by traditional hyper-parameter selection approaches. Meanwhile, different random network architectures, such as the existence or lack of a direct link and the mapping of direct links, can also strongly affect the results. We present an automated learning pipeline for the ensemble deep randomized feed-forward neural network in this paper, which integrates hyper-parameter selection and randomized network architectural search via Bayesian optimization to ensure robust performance. Experiments on 46 UCI tabular datasets show that our strategy produces state-of-the-art performance on various tabular datasets among a range of randomized networks and feed-forward neural networks. We also conduct ablation studies to investigate the impact of various hyper-parameters and network architectures.

1. Introduction

In the last decade, neural networks have made remarkable strides, with many types of neural networks showing promising outcomes in a number of practical applications, such as classification, segmentation and generation tasks. The multilayer perceptron (MLP) is a straightforward neural network structure made up of numerous hidden layers and one output layer. Despite their simplicity, recent research has shown that MLPs are capable of performing well in large-scale image tasks [1,2]. However, since the MLP optimizes parameters via gradient descent (GD), the computational resources and time demanded for successive iterations are unavoidably large.

Among these, randomized neural networks have attracted great interest for the efficiency and stability [3]. Typically, randomized neural networks relate to single hidden layer feedforward neural networks (SLFN) wherein the weights of the hidden layer are produced randomly and only the weights from the output layer are involved in learning [4]. The Randomized Vector Functional Link (RVFL) [5,6] is the cornerstone of randomized neural networks, in which the hidden weights of hidden neurons are generated randomly and kept fixed while the weights of output neurons are derived by a simple closed-form solution [7]. In RVFL, the raw space features are propagated to the output layer through the direct links, giving an opportunity to be involved in the weight computation. Due to the fact that Occam's Razor argues for simpler and less complex models, RVFL networks are more appealing than other neural networks. Extreme Learning Machine (ELM) [8] is one of the RVFL variants, which omits the direct links and adopts the primitive structure of SLFN. Broad Learning System (BLS) [9] is also one of the RVFL's extensions, as it maps the features from the original domain to a latent space and then propagates them to the output layer for the weight solution.

The Ensemble Deep Randomized Vector Functional Link (edRVFL) [10,35,37] extends the single-layer RVFL network to a multi-layer structure, utilizing the ensemble technique to massively increase the network's efficacy. Each hidden layer in edRVFL has an associated output layer for prediction, or each hidden layer may be regarded as a dependent classifier, and then the ensemble approach, such as majority voting, is used to aggregate all predictions. This Ensemble Deep framework is generic and is applicable to different randomized networks, such as ELM or BLS, in addition to RVFL networks. Each hidden layer in the ensemble deep

* Corresponding author at: KINDI Center for Computing Research, College of Engineering, Qatar University, Doha, Qatar.
E-mail addresses: e200008@e.ntu.edu.sg (M. Hu), gaor0009@e.ntu.edu.sg (R. Gao), p.n.suganthan@qu.edu.qa (P.N. Suganthan), mtanveer@iiti.ac.in (M. Tanveer).

https://doi.org/10.1016/j.neucom.2022.09.148
0925-2312/

framework needs to be trained with an output weight, and utilizing the consistent hyperparameters for all sub-classifiers would lead to less than intended results. The current strategy is to specify the hyperparameters for the options and then use the grid search to sweep through all the possible options and select the best performing one as the result [10]. However, the grid search approach has several obvious drawbacks: first, it is difficult to determine the stride of the given hyperparameter range, and options with varying degrees of granularity can produce very different results; second, there is a difficult trade-off between accuracy and speed, with too many hyperparameter options resulting in lengthy network training times while too few options resulting in a clear drop in performance.

Inspired by the significant process occurring in the field of Automated Machine Learning (AutoML) [11–13] and Hyper-parameter optimization (HPO) [14,15], we propose an automated layer-wise learning pipeline for randomized neural networks with an ensemble deep framework. AutoML introduces techniques that increase machine learning's efficiency and speed up machine learning research. HPO and model structure selection play a significant role in AutoML. The main core of HPO and structure selection consists of Bayesian Optimization (BO) [16]. In our proposed pipeline, we use a BO-based approach to optimize the hyperparameters; we also consider different kinds of randomized networks as the pipeline components, including RVFL, ELM and BLS. In addition, due to the specificity of the ensemble deep framework, where the input of each hidden layer is dependent on the output of the shallow layer, we designed an exclusive training and inference process to demonstrate the benefits of randomized networks. We hope that each individual hidden layer can be responsible for a fraction of the indistinguishable samples and achieve satisfactory results by ensemble. Our contributions are mainly the following:

- We propose an automated layer-wise pipeline for randomized feed-forward neural networks, which integrated different kinds of randomized network architectures.
- We rethink the ensemble deep framework's training and inference phases due to the capacity limitation of randomized networks, and propose a Sample-level Attention Strategy to divide the training set into subsets to solve the capabilities problem.
- Several ablation experiments are conducted to verify the effect of different hyper-parameters on the results.

The remaining part of the article is structured as follows: Section 2 provides a brief overview of related work, including recent advances in RVFL and its variants, AutoML and HPO. Section 3 delves into the specifics of our proposed pipeline. Section 4 describes the experimental setting and dataset description, we also present and analyze the experimental results. Finally, we summarize our paper and suggest some directions for future work.

2. Related Work

In this section, we first demonstrate a brief overview of the fundamental random vector functional link network and some related variants. We then review the ensemble deep framework for randomized feed-forward neural networks and next we present some common approaches for automated machine learning pipelines and hyper-parameter optimization methods.

2.1. Randomized Feed-forward Neural Network

Randomized Feed-forward Neural Network(RFNN) refers to a subclass of SLFNs in which the hidden layer neurons are randomly initialized and fixed, while the output weights are calculated through a simple closed-form solution. There are many variants of RFNN, the mainstream of which is the Random Vector Functional Link (RVFL) proposed by Pao et al. in 1992 [5]. There have been numerous RVFL extensions during the last two decades, including Extreme Learning Machine (ELM), Broad Learning System (BLS), etc.*Random Vector Functional Link*

Random Vector Functional Link (RVFL) was proposed by Pao and Takefuji in [5]. It is similar to SLFN in construction except that it includes a direct link that propagates the input features directly to the output layer, which is composed of an input layer, a single hidden layer and an output layer. Given an input sample $\mathbf{X} \in \mathbb{R}^{k\times d}$ with associated labels $\mathbf{Y} \in \mathbb{N}^k$, random weights $\mathbf{W}$ are used to map the input features to the latent features $\mathbf{A} \in \mathbb{R}^{k\times N}$, where k and d are the numbers and the dimension of the input samples, N is the number of neurons in the hidden layer. The weights of the hidden layers are generated randomly and maintained; only the weights of the output neurons β are tuned during the training phase.

The objective equation of RVFL is the following:

$$\min_{\beta} ||\mathbf{E}\beta - \mathbf{Y}||^2 + \lambda||\beta||^2 \tag{1}$$

where $\mathbf{E}$ represents the linear combination of input patterns $\mathbf{X}$ and hidden features $\mathbf{A}$, which we referred to as enhanced features $\mathbf{E} = [\mathbf{A}; \mathbf{X}]$. With the activation function $g(\cdot)$ for non-linear mappings, $g(\mathbf{XW})$ can be used to produce $\mathbf{A}$.

The objective problem can be solved using ridge regression, which is denoted by:

$$\beta = (\mathbf{E}^T\mathbf{E} + \lambda I)^{-1}\mathbf{E}^T\mathbf{Y} \tag{2}$$

$$\beta = \mathbf{E}^T(\mathbf{E}\mathbf{E}^T + \lambda I)^{-1}\mathbf{Y} \tag{3}$$

where I is the identity matrix and λ is the regularization parameter. *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, which is developed in 2004 [8]. The primary theoretical distinction between ELM and RVFL is that the features used to solve for the output layer weights β are different, in another word, the enhanced feature $\mathbf{E}$ in Eq. 1 is replaced by the hidden feature $\mathbf{A}$. It is therefore not difficult to obtain the objective function and the solution for the ELM:

$$\begin{aligned} &\text{Obj.} \quad \min_{\beta} ||\mathbf{A}\beta - \mathbf{Y}||^2 + \lambda||\beta||^2 \\ &\text{Sol.} \quad \beta = (\mathbf{A}^T\mathbf{A} + \lambda I)^{-1}\mathbf{A}^T\mathbf{Y} \\ &\text{or} \quad \beta = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T + \lambda I)^{-1}\mathbf{Y} \end{aligned} \tag{4}$$

Broad Learning System

The Broad Learning System is an extension of the RVFL. The original data is transformed into latent features using random weights. Then, using similar random methods, the latent features are extended to another latent space. Finally, the output weights are connected to both two latent space features, and the appropriate output weights may be derived by solving ridge regression or psudo-inverse. Mathematically, after obtaining $\mathbf{A}$ via $g(\mathbf{X}w)$, we need to procure $\mathbf{A}'$ via similar random mapping $g'(\mathbf{A}w')$. Thus the enhanced feature $\mathbf{E}$ is replaced by two sets of latent features $\mathbf{E} = [\mathbf{A}; \mathbf{A}']$. The solution procedure is same as Eqs. (2, 3) for RVFL.

2.2. Ensemble Deep Randomized Neural Network

edRVFL [10] is a generic framework for randomized feedforward networks that is not only confined to RVFL networks. This framework contains numerous hidden layers, each of which has

an input that is a concatenation of the previous layer's output and the original features from direct link, Each hidden layer separately calculates the output weights, or each hidden layer can be seen of as an individual classifier. Finally, the ensemble approach is used to combine the prediction results from each hidden layer. Recently, there are many works based on the edRVFLin various domain [17,36,38].

In ensemble deep RVFL (edRVFL), the output of every hidden layer $\mathbf{A}^{(i)}$ is defined as follows:

$$\mathbf{A}^{(i)} = \begin{cases} g(\mathbf{X}\mathbf{W}_r^{(1)}) & if\ i = 1 \\ g([\mathbf{A}^{(i-1)}; \mathbf{X}]\mathbf{W}_{re}^{(i)}) & if\ i > 1 \end{cases} \tag{5}$$

where $\mathbf{W}_r^{(1)} \in \mathbb{R}^{d\times N}$ and $\mathbf{W}_{re}^{(i)} \in \mathbb{R}^{(d+N)\times N}$ are the random weights between the input to first hidden layer and inter hidden layers respectively. The weights and biases are randomly generated and kept fixed. For the ease of notation, we omit the bias term in the formulas, as shown in [6], compared with no bias, the network with bias has more freedom, as bias is also a variable parameter that can be adjusted, and the network will have a better approximation to the data. For edRVFL, each layer can be solved according to Eqs. (2, 3) simply substitute the enhancement feature $\mathbf{E}$ for the corresponding layer's $\mathbf{E}^{(i)} = [\mathbf{A}^{(i)}; \mathbf{X}]$.

2.3. Automated Machine Learning and Hyper-parameter Optimization

2.3.1. Automated Machine Learning

Automated Machine Learning (AutoML) is a widely used term, a comprehensive AutoML framework performs a variety of tasks, including data preprocessing, feature selection, model selection, hyperparameter tuning, and results analysis. There are numerous autoML frameworks available, and we will discuss a couple of them briefly.

Auto-WEKA Auto-WEKA [13] was one of the first AutoML frameworks and has remained active in the recent years [18]. Auto-WEKA is based on the WEKA ML library in Java, which employs BO and integrates various ML models via different ensemble strategies.

Auto-sklearn Auto-sklearn [12] uses ML models from the scikit-learn library in Python and has placed first in several competitions. There are various critical aspects that contribute to the success of as follows: 1.Auto-sklearn uses an ensemble model selection strategy [19] for model integration; 2.Auto-sklearn employs a meta-learning approach [20] as warm-up to obtaining more accurate hyperparameters; and 3.Auto-sklearn introduces a time budget system to balance the efficiency and accuracy.

2.3.2. Hyper-parameter Optimization

In comparison to the comprehensive framework for autonomous learning, this paper concentrates on a hyper-parameter optimization strategy. Currently available Hyper-parameter optimization techniques fall into two broad categories: BO and evolutionary algorithms (EA). The following is a sampling of representative work:

Bayesian Optimization Bayesian Optimization (BO) is widely used for hyper-parameter optimization tasks, one of the most commonly used probabilistic models in BO is the Gaussian process (GP) [21]. However, Bayesian optimization based on Gaussian processes has significant flaws. For starters, Gaussian processes are unsuitable for high dimensions and are overly computationally costly; also, it is difficult to adapt to complex configuration spaces such as discrete, polynomial, etc. The Tree Parzen Estimator (TPE) [22] is an alternative choice for GP, which is based on Gaussian mixture model. In each trial, TPE maintains a Gaussian mixture model $L(x)$ for the hyper-parameter associated with the best target value and another Gaussian mixture model $G(x)$ for the remaining hyper-parameters, selecting the hyper-parameter whose minimizing of $G(x)/L(x)$ corresponds to the next set of search values. Mathematically, the common acquisition function Expected Improvement (EI) can be stated as follows:

$$a(x,\alpha) = \int \max(0, \alpha - f(X))\mathrm{d}\ \ p(f|D) \tag{6}$$

where α is the best observation of observed data points $D = \{(x_0, y_0), \ldots, (x_n, y_n)\}$. According to α, TPE uses non-parametric Parzen kernel density estimators (KDE) to model the distribution of different configurations

$$\begin{aligned} L(x) &= p(y < \alpha|x, D) \\ G(x) &= p(y > \alpha|x, D) \end{aligned} \tag{7}$$

It can be derived that maximising the ratio $L(x)/G(x)$ is equal to maximize the acquisition function [22].

Hyperband Hyperband (HB) is proposed in 2017 [23], which utilizes Successive-Halving (SH) [24] to allocate resources iteratively to random configurations. HB selects N configurations for SH on each iteration. In practice, HB performs admirably and frequently outperforms random search and Bayesian optimization approaches when performing full-featured budget evaluations on small to medium overall budgets. However, due to its reliance on randomly chosen configurations, its convergence to the global optimum is limited, and its advantages with high budgets are typically negated by random search.

DEHB DEHB [25] is an HPO approach that combines the benefits of the well-known bandit-based method HB with the evolutionary search strategy Differential Evolution (DE).

3. Methods

Rather than developing a comprehensive framework for automated learning, we will concentrate on developing an automated optimization pipeline for the ensemble deep framework with randomized feed-forward neural networks.

Through our experiments we found that the ensemble deep framework has numerous significant drawbacks, including the following:

1. The hyper-parameters for all sub-classifiers are identical, including the number of neurons, the type of activation function, the regression coefficients, etc. This significantly reduces the network's diversity.
2. Each sub-classifier is responsible for the whole training space. However, the capacity of the random network is limited. Hence, the prediction of all layers is a more moderate result and the performance of the whole model relies heavily on the effect of the final ensemble.
3. Due to the grid search methods, all viable hyper-parameter alternatives must be artificially given prior to training the framework. However, such an approach might result in biased findings. Simultaneously, a finer hyper-parameter stride will result in increased training time and make it difficult to enjoy the advantages of randomized networks.

To address the aforementioned issues, we introduce an automated layer-wise optimization pipeline that incorporates the following enhancement methods:

1. We propose a layer-wise optimization strategy, where the hyper-parameters of each hidden layer are independent. Moreover, to add diversity to the hidden layer features, we introduce additional randomized networks for backbone choice.

2. We introduce a Sample-level Attention Strategy that enable distinct hidden layers to concentrate their efforts on distinct training subspaces. Additionally, we suggest a nearest neighbor partitioning strategy for the inference phase to account for weighted subspaces strategy.
3. Inspired by the recent rapid progress of hyper-parameter optimization methods, we use BO and HB based methods to optimize the hyper-parameters of the pipeline.

The pipeline can be found in Algorithm 1 and Fig. 1. The details of proposed strategies are demonstrated below.

Algorithm 1: Automated Layer-wise Optimization

input non-test set $\mathscr{X}_{nt} = \{\mathbf{X}_{nt}, y_{nt}\}$, test set $\mathscr{X}_t = \{\mathbf{X}_t\}$
given number of layers N
output predictions y_t of $\mathscr{X}_t$
for i $\in N$ **do**
  if $i = 0$ **then**
    random split the $\mathscr{X}_{nt}$ into train $\mathscr{X}^{(0)}_{train}$ and validation $\mathscr{X}^{(0)}_{val}$
    search the best configurations via BOHB to
      obtain $\mathbf{A}^{(\mathbf{0})}$ via Eq. 8
      generate attention matrix $\mathbb{W}^{(0)}$ by Eq. 9
      calculate β^0 by Eq. 2 or Eq. 3
    obtain prediction y^0_p of $\mathscr{X}_t$
  end if
  split the $\mathscr{X}_{nt}$ into train $\mathscr{X}^{(i)}_{train}$ and validation $\mathscr{X}^{(i)}_{val}$ w.r.t.$\mathbb{W}^{(i)}$
  search the best configurations via BOHB
    obtain $\mathbf{A}^{(\mathbf{i})}$ via Eq. 8
    generate attention matrix $\mathbb{W}^{(i)}$ by Eq. 9
    calculate β^i by Eq. 10
  obtain prediction y^i_p of $\mathscr{X}_t$
end for
pick top T layers for each non-test sample in $\mathscr{X}_{nt}$
find t nearest neighbors in $\mathscr{X}_{nt}$ for each test sample in $\mathscr{X}_t$
ensemble $T \times t$ predictions and get the final decision y_t

3.1. Layer-wise Solution

Inspired by the greedy training strategy [26,27], we introduce a layer-wise optimization method for ensemble deep randomized neural network. Instead of obtaining the local minimum of the whole network, each hidden layer in the ensemble deep framework can be regarded as an independent classifier, and we aim to find the optimal configuration for each layer. The subsequent hidden layers will not be trained until the training of preceding layers has been completed. Mathematically, the output of each hidden layer can be expressed by:

$$\mathbf{A}^{(i)} = \begin{cases} g^{(1)}(\mathbf{X}\mathbf{W}^{(1)}) & \text{if } i = 1 \\ g^{(i)}([\mathbf{A}^{(i-1)}; \mathbf{X}]\mathbf{W}^{(i)}) & \text{if } i > 1 \ \& \ \text{w/ direct link} \\ g^{(i)}(\mathbf{A}^{(i-1)}\mathbf{W}^{(i)}) & \text{if } i > 1 \ \& \ \text{w/o direct link} \\ g^{(i)}([\mathbf{A}^{(i-1)}; \mathbf{A}\prime^{(i-1)}]\mathbf{W}^{(i)}) & \text{if } i > 1 \ \& \ \text{w/} 2^{nd} \text{mapping} \end{cases} \tag{8}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times N^{(1)}}$ and $\mathbf{W}^{(i)} \in \mathbb{R}^{(\eta^{(i)} + N^{(i-1)}) \times N^{(i)}}$. $N^{(i)}$ and $g^{(i)}(\cdot)$ represent the neuron number and the activation function type of ith layer. $\eta^{(i)}$ depends on the ith layer type: if the ith layer is with direct link (RVFL type), $\eta^{(i)}$ means the dimension of the raw input $\mathbf{X}$, else if it has another random mapping (BLS type), $\eta^{(i)}$ is subject to the dimension of $\mathbf{A}\prime^{(i-1)}$, otherwise, $\eta^{(i)} = 0$ (ELM type). Fig. 2 depicts the weight shape of the various kinds of hidden layers. For the activation function $g^{(i)}(\cdot)$, it can be selected from the Table 1.

It is worth noting that all the layers are independent. Specifically, the configuration of each layer can be different, e.g. different layers can have different numbers of neurons and these configurations are optimised by Bayesian methods. In contrast, for the traditional random neural network ensemble framework, the number of neurons in each layer is predefined and fixed. Our layerwise solution provides stronger diversity for ensemble random neural networks.

After the acquisition of $\mathbf{A}^{(i)}$, we can generate enhanced features $\mathbf{E}^{(i)}$ based on the type of current layer. The enhanced features $\mathbf{E}^{(i)}$ can be $[\mathbf{A}^{(i)}; \mathbf{X}]$ or $[\mathbf{A}^{(i)}]$ or $[\mathbf{A}^{(i)}; \mathbf{A}\prime^{(i)}]$ for RVFL type, ELM type and BLS type separately.

3.2. Sample-level Attention Strategy

Our sample-level Attention strategy consists of three main sections. First, based on the results of the previous layer, we determine the relevance and importance of the various samples; next, we divide the training set for the current layer based on the relevance and importance. Finally, we select the hidden layers that will be used for the ensemble based on the inclusiveness of training samples.

3.2.1. Sample Attention

The performance of a randomized neural network is limited by its capacity [30]. Instead of increasing the number of neurons, we prefer to make training phase more purposeful. We use a sample-level attention strategy to implicitly assign the focus of training phase, which is based on an adaptive weighting mechanism. Specifically, the attention matrix of the ith layer is based on the score of all previous layers, which can be expressed as:

$$\mathbb{W}^{(i)} = \frac{\sum_i \text{sigmoid}\left(\max(\mathbf{A}^{(\mathbf{i})}_{\mathbf{N}}) - \mathbf{A}^{(i)}_{\mathbf{P}}\right)}{\sum_i \sum_j^k \text{sigmoid}\left(\max(A^{(i,j)}_N) - A^{(i,j)}_P\right)} \tag{9}$$

where i is the layer index and j is the index of k training samples. $\mathbf{A}^{(i)}_{\mathbf{P}}$ and $\mathbf{A}^{(i)}_{\mathbf{N}}$ are the score matrix in $\mathbb{R}^k$ of the prediction score of ground truth and all the remaining scores separately. After obtaining $\mathbb{W}^{(i-1)}$ and $\mathbf{E}^{(i)}$, the weight of ith layer output can be calculated via the equations similar to Eqs. (2, 3):

$$\begin{aligned} \text{PrimalSpace}: \quad & \beta = \left([\mathbb{W}^{(i-1)} * \mathbf{E}^{(\mathbf{i})}]^T [\mathbb{W}^{(i-1)} * \mathbf{E}^{(\mathbf{i})}] + \lambda I\right)^{-1} [\mathbb{W}^{(i-1)} * \mathbf{E}^{(\mathbf{i})}]^T \mathbf{Y} \\ \text{or Dual Space}: \quad & \beta = [\mathbb{W}^{(i-1)} * \mathbf{E}^{(\mathbf{i})}]^T \left([\mathbb{W}^{(i-1)} * \mathbf{E}^{(\mathbf{i})}][\mathbb{W}^{(i-1)} * \mathbf{E}^{(\mathbf{i})}]^T + \lambda I\right)^{-1} \mathbf{Y} \end{aligned} \tag{10}$$

As Eq. 9, when samples are successfully classified, their weights are reduced. And when samples are incorrectly predicted, or the probability gap is not significant, these samples are given higher weights in the following layer. This assures that distinct hidden layers are accountable for different subsets.

3.2.2. Data Division

To further manifest the effects of the attention mechanism, we also propose a specialized scheme for the division of the training and validation sets. Typically, a dataset will be divided into three parts in the data pre-process phase: a training set, a validation set, and a test set. Without considering cross-validation, we only divide the entire dataset into a test set and a non-test set before training in our experiment. We partitioned the non-test set into

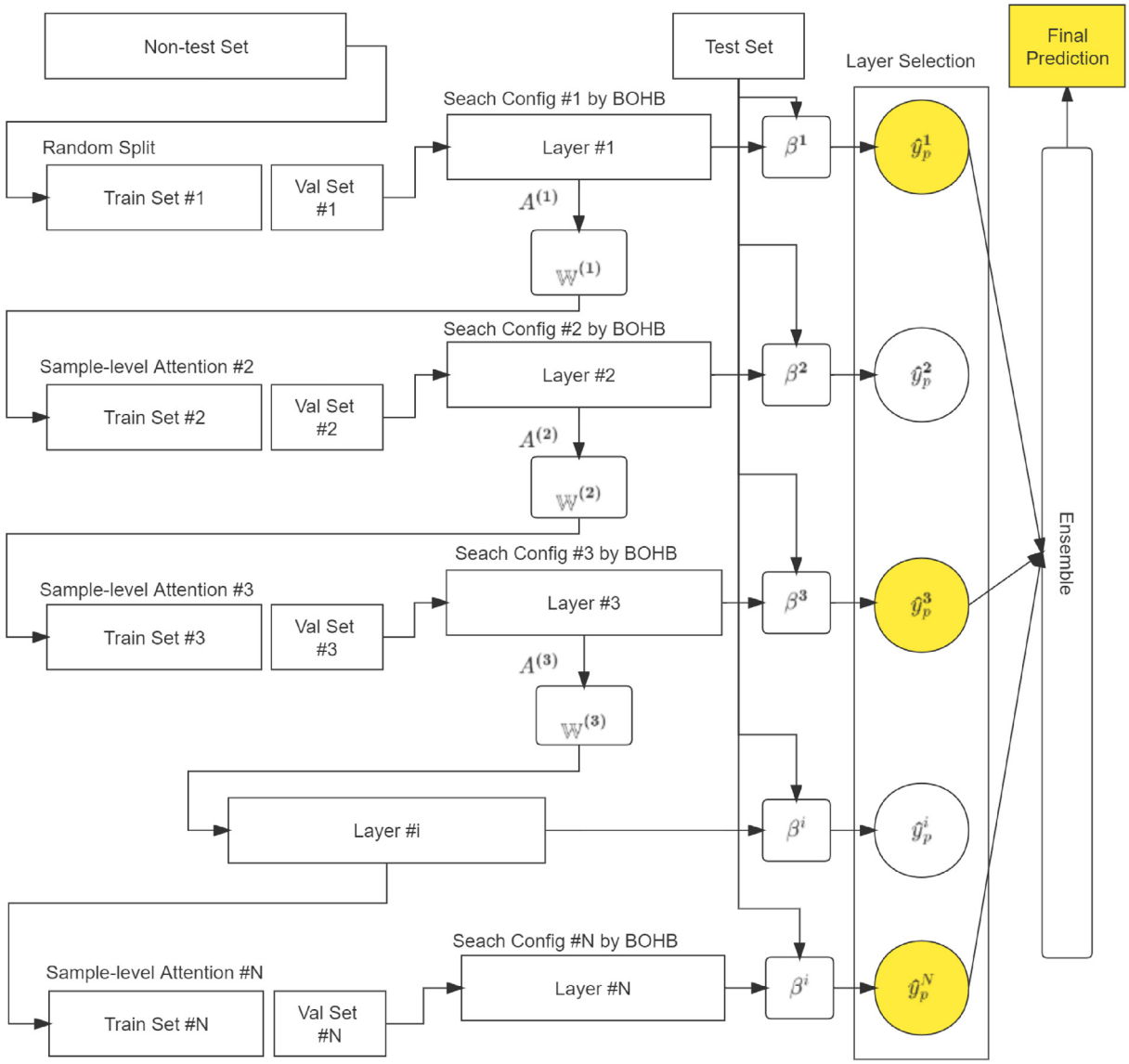

Fig. 1. The pipeline of automated layer-wise solution for ensemble deep randomized feed-forward neural network. The hyper-parameters of each layer are independent, e.g., the number of neurons in different layers is not uniform. The partitioning of the training and validation sets is also different for each layer, and the division is based on the output of the previous layer. At the end, top T feasible layers will be screened to utilize for the ensemble to get the final outcome.

a training set and a validation set at each layer during the layer-wise training procedure afterwards. In contrast to random split, we introduce an novel division strategy.

The new dataset split strategy is inspired by the nearest neighbor clustering algorithm. The motivation for this is that we want each layer to be responsible for a few clusters of hard-to-train data. Prior to beginning training in any layer, we sort the features by their weights $\mathbb{W}^{(i)}$ and use the top M enhanced features $\mathbf{E}^{(i)}$ as clustering centers. For each clustering centre, we take the m nearest enhanced features to them and form clusters, where m is a fixed ratio of the sample number of non-test set. Following the acquisition of $M+1$ clusters (M clusters with centre and one cluster include all remainings), the training set for this layer contains 90% of the samples from M clusters and 10% of the samples from the remaining clusters.

3.2.3. Layer Selection

However, such training process will lead a diminishing prediction results as the hidden layers have difficulties with samples without weights in inference phase. If we simply adopt an ensemble approach, we will introduce a multitude of unexpected predictions and thus significantly reduce the accuracy. Therefore, we also need to assign matching classifiers to the test set during the inference phase. We first rank the performance of all instances at each layer based on cross-entropy loss during the training phase, the loss can be expressed in Eq. 11:

$$Loss = -\sum_{i=0}^{k} \mathrm{OneHot}(Y_i) \log \left(\frac{\exp(A_i)}{\sum_{j=0}^{k} \exp(A_j)} \right) \tag{11}$$

where $\mathrm{OneHot}(\cdot)$ is an operator which transform the input from scalars to $\{0,1\}^c$ vectors, and c is the number of classes. We then select the best performing T sub-classifiers based on these rankings and record them. In test phase, we find t training samples nearest to the test features and ensemble the corresponding $T \times t$ predictions by voting. We measure the distance between features by the L2 distance, where the features are the enhanced $\mathbf{E}^{(i)}$ feature and depends on the ith layer type.

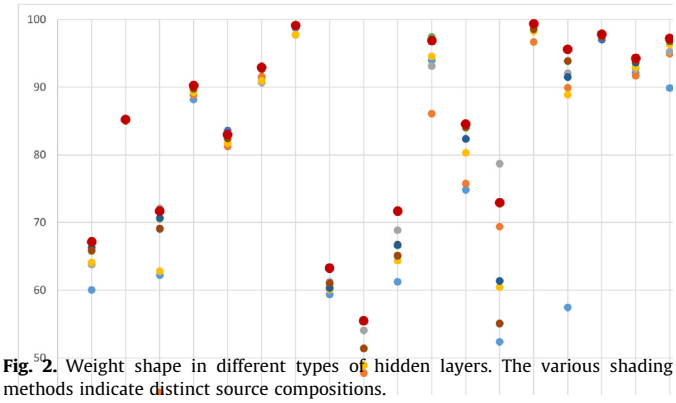

Fig. 2. Weight shape in different types of hidden layers. The various shading methods indicate distinct source compositions.

3.3. BO & HB for HPO

Following [14], we adopted a similar BOHB technique for our model. We first utilize Hyperband(HB) [23] to choose the budget and then use Successive-Halving(SH) [24] to choose the ideal parameters. In contrast to the original HB, the SH process is led by BO (based on a model), rather than random sampling. The BO part is similar to the TPE in that the Gaussian mixture model is constructed by a single multidimensional KDE. The whole sampling process [14] is given in Algorithm 2:

Algorithm 2: BOHB sampling process

input Observations D, fraction of random runs ρ, number of sample N_s, min number of points N_{min}
output next configuration to evaluate
if $rand() < \rho$ **then**
return random configuration
end if
find largest budget with at least $N_{min}+1$ observations
if no such B exists **then**
return random configuration
end if
fit KDEs according to Eq. 7
draw N_s samples $\sim L(x)$
return Sample with highest ratio $L(x)/G(x)$

Table 1
Activation functions considered for the model.

Activation Type	Expression
Rectified Linear Units (ReLU)	$f(x) = max(0, x)$
Sigmoid Activation	$f(x) = 1/(1+\exp(-x))$
Scaled Exponential Linear Unit (SELU) [28]	$f(x) = \lambda \cdot x$ if $x \geqslant 0$[1]
	$f(x) = \lambda \cdot \alpha(\exp(x) - 1)$ if $x < 0$[2]
Swish [29]	$f(x) = x \cdot \text{sigmoid}(\beta x)$ [3]
Self-normalizing Swish	$f(x) = \lambda \cdot x \cdot \text{sigmoid}(\beta x - \gamma)$[4]

[1] $\lambda \approx 1.0507$
[2] $\alpha \approx 1.6732$
[3] In most cases, $\beta = 1$
[4] This activation function has not appeared in any literature, however, according to the proof in [28], we can easily obtain $\gamma \approx 0.2066$.

4. Experiments

In this section, we perform several experiments to demonstrate the superiority of our proposed method. It is worth noting that all the data are extracted from the real world.

4.1. Datasets and Comparison Methods

The experiments are conducted on public UCI datasets from various domains. Table 2 gives an overview of the 46 real-world application datasets. We used the same pre-processing method as [28] for splitting the dataset into test and non-test groups.

It's worth mentioning that all datasets have been cross-validated fourfold. The entire data was split into a non-test set and a test set four times, each time called one fold. During the training process, we will use the non-test to train the network

Table 2
Overview of the UCI datasets.

Name	#Samples	#Features	#Classes	Name	#Samples	#Features	#Classes
abalone	4177	9	3	ozone	2536	73	2
adult	48842	15	2	page-blocks	5473	11	5
arrhythmia	452	263	13	ot_nucleus_5b	912	25	2
bank	4521	17	2	pendigits	10992	17	10
car	1728	7	4	plant-margin	1600	65	100
cardio-10	2126	22	10	plant-shape	1600	65	100
cardio-3	2126	22	3	plant-texture	1599	65	100
chess-krvk	28056	7	18	ringnorm	7400	21	2
chess-krvkp	3196	37	2	semeion	1593	257	10
connect-4	67557	43	2	spambase	4601	58	2
congressional-voting	435	16	2	monks-3	3190	6	2
contrac	1473	10	3	german-credit	1000	25	2
hill-valley	1212	101	2	statlog-image	2310	19	7
sementation	2310	19	7	statlog-landsat	6435	37	6
led-display	1000	8	10	statlog-shuttle	58000	10	7
letter	20000	17	26	steel-plates	1941	28	7
magic	19020	11	2	thyroid	7200	22	3
minboone	130064	51	2	titanic	2201	4	2
molec-biol-splice	3190	61	2	twonorm	7400	21	2
mushroom	8124	22	2	wall-following	5456	25	4
musk-2	6598	167	2	waveform	5000	22	3
nursery	12960	9	5	waveform-noise	5000	41	3
om_nucleus_4d	1022	42	3	wine-quality-red	1599	12	6
om_states_2f	1022	26	2	wine-quality-white	4898	12	7
optical	5620	63	10	yeast	1484	9	10
glass	214	9	6				

Table 3
Hyper-parameter configurations for RVFL networks.

HPs	Description	Search Range	Type
N	The number of hidden layer neurons	[256,2048]	integer
C	Regularization factor coefficient	[-12, 12]	float
S	Neuron Normalization Scale	[0,10]	float
g	Activation Function type, details in Table 1	N/A	categorical
T	The number of selected best performing sub-classifiers set	[0,5]	integer
t	The number of selected training features nearest to the test set	[0,5]	integer

and will seek the ideal hyperparameters based on the lowest validation loss, the details are in Section 3.2. Then, using the entire non-test set, we retrain the current layer using the optimized configurations. Finally, we provide the mean accuracy of test set and their standard deviation. Each set of trials is repeated 10 times with different random seeds.

In order to demonstrate the superior performance of our model, comparison with several deep neural network approaches and other shallow networks methods is performed. All relevant algorithms used are shown below:

1. ResNet: Residual networks [31].
2. SNN: Self normalizing networks with SELUs [28]

Table 4
Accuracy (%) and Average Ranks of basic approaches and ablation results.

	BLS	SCN	HELM	RVFL	dRVFL	edRVFL	AL-RVFL†	edRandom	ALR†
abalone	60.06	64.00	63.77	64.12	66.33	65.81	66.34	65.87	**67.14**
adult	85.15	85.01	85.05	85.01	85.12	**85.25**	85.14	85.21	**85.25**
arrhythmia	62.23	44.91	**72.12**	62.83	69.03	70.58	70.67	69.12	71.68
bank	88.19	88.83	89.20	89.29	89.87	89.76	89.90	89.80	**90.26**
cardio-10	83.47	81.26	82.39	81.73	**83.62**	82.39	82.99	82.44	83.00
cardio-3	91.33	91.57	90.68	90.96	92.84	92.66	92.91	92.69	**92.93**
chess-krvkp	98.75	97.77	99.00	97.80	99.03	**99.12**	**99.12**	**99.12**	**99.12**
congressional-voting	59.40	60.09	61.24	60.09	60.33	61.01	60.35	61.03	**63.30**
contrac	41.78	47.75	54.08	48.91	55.37	51.36	55.41	51.43	**55.50**
glass	61.23	66.79	68.87	64.38	65.09	66.58	66.67	65.13	**71.70**
letter	93.99	86.11	93.15	94.55	97.23	**97.46**	96.84	96.88	96.88
molec-biol-splice	74.84	75.75	82.40	80.32	82.34	84.00	82.37	84.09	**84.57**
monks-3	52.37	69.42	**78.70**	60.45	55.07	61.35	61.39	55.10	72.92
musk-2	98.77	96.71	98.32	98.29	99.04	98.57	99.12	98.62	**99.39**
OMN_4d	57.46	89.91	92.06	88.92	91.47	93.86	91.53	93.87	**95.61**
pendigits	97.45	97.05	97.41	97.23	97.01	97.46	97.78	97.47	**97.83**
spambase	92.15	91.71	92.67	92.95	93.61	93.87	93.63	93.95	**94.26**
statlog-image	89.90	94.97	95.28	96.36	96.79	96.84	96.83	96.91	**97.23**
statlog-landsat	83.47	90.25	**91.22**	89.85	90.70	91.20	90.72	91.18	91.20
statlog-shuttle	96.82	99.79	99.88	98.77	99.88	99.91	99.87	99.90	**99.92**
wall-following	89.53	85.41	89.46	87.84	90.71	90.30	90.72	90.34	**90.98**
waveform	83.48	84.76	86.16	84.64	86.34	85.90	86.44	85.96	**86.88**
noise	82.44	83.70	**86.08**	84.48	84.48	85.72	**86.08**	85.73	**86.08**
wine-quality-white	55.15	55.96	55.49	58.48	62.36	63.30	62.40	63.31	**65.60**
Average Accuracy	78.31	80.40	83.53	81.60	83.15	83.51	83.55	83.13	**84.97**
Average Rank	7.67	7.54	5.17	7.21	4.54	3.04	3.56	3.81	**1.46**

† means the method includes automated layer-wise training.

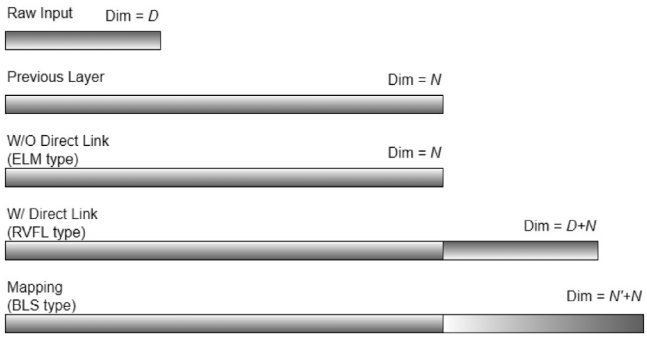

Fig. 3. The accuracy of basic approaches and ablation results. The horizontal axis represents the different data sets and the vertical axis indicates the accuracy rate. The different coloured scatters represent different algorithms. The proposed algorithm (ALR) are highlighted with larger size.

3. RVFL: Randomized Feedforward Neural Network with functional link [5].
4. SCN: Stochastic configuration networks [32]
5. HELM: Hierarchical ELM with different regularization methods [33]
6. BLS: Broad learning system [9], a paradigm shift version based on RVFL.
7. deep RVFLs: Deep RVFL network and its ensemble version [10]

To conduct the analysis, we use the method described in [10] and additionally apply statistical comparison methods described in [34] to demonstrate the method's validity. We use the Wilcoxon signed rank test method to do statistical analysis, which is a non-parametric statistical hypothesis test based on the observations.

4.2. Experiment Setting

In our experiments, there are six hyper-parameters to be tuned and two meta parameters need to be assigned manually. All the hyper-parameters are optimized by the BOHB framework as described in Section 3.3. The hyper-parameters are summarized in the Table 3, include the number of hidden neurons N, the coefficient of regularization factor C where $\lambda = 1/2^C$ in Eqs. 2, 3 and their extensions, the activation function type g and the hyper-parameters T and t for sample-level attention strategy, and the neuron normalization factor in Eq. 12:

$$\mathbf{W} = s \cdot \frac{\mathbf{v}}{||\mathbf{v}||} \tag{12}$$

where $\mathbf{v}$ is randomly generated and s can be considered as a hyperparameter.

The meta parameters are the number of network layers L and the desired budget of BOHB. In our experiments, L is set to 20, and the budget is set to 50.

4.3. Results and Analysis

We begin by comparing our proposed methods Automated Layer-wise Randomized Feedforward Networks (ALR) to several classical randomized feed-forward neural networks on smaller datasets, and also conduct ablation experiments to illustrate the influence of various parameters on the final performance. The abla-

Table 5
Accuracy (%) and Average Ranks of various deep approaches on 46 UCI datasets.

	dRVFL	edRVFL	SNN	Resnet	ALR
abalone	66.33	65.81	**66.57**	64.66	**67.14**
adult	85.12	**85.25**	84.76	84.84	**85.25**
bank	**89.87**	89.60	89.03	87.96	**90.26**
car	97.97	98.04	**98.38**	92.82	**99.12**
cardio-10	83.62	83.24	**83.99**	81.73	83.00
cardio-3	**92.84**	92.66	91.53	90.21	**92.93**
chess-krvk	68.39	70.07	**88.05**	85.43	84.44
chess-krvkp	99.03	**99.12**	**99.12**	**99.12**	**99.12**
connect-4	83.94	84.77	**88.07**	87.16	82.51
contrac	**55.37**	54.01	51.90	51.36	**55.50**
hill-valley	58.75	**67.00**	52.48	53.96	**76.16**
image-sementation	89.10	88.52	**91.14**	89.19	**92.19**
led-display	73.90	74.40	**76.40**	71.60	**76.80**
letter	97.23	97.46	97.26	**97.62**	96.68
magic	86.55	86.81	86.92	**87.23**	86.83
minboone	92.33	92.72	**93.07**	92.54	**93.07**
molec-biol-splice	82.34	84.00	**90.09**	85.57	84.57
mushroom	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
musk-2	99.04	98.57	98.91	**99.64**	99.39
nursery	98.32	98.70	**99.78**	99.40	98.70
oocytes_merluccius_nucleus_4d	82.75	**84.41**	82.35	80.00	**85.45**
oocytes_merluccius_states_2f	91.86	93.63	**95.29**	93.73	**95.69**
optical	98.16	**98.27**	97.11	96.27	**98.27**
ozone	97.12	**97.20**	97.00	96.69	**97.48**
page-blocks	**96.60**	96.56	95.83	96.05	**97.08**
pendigits	**97.77**	97.46	97.06	97.08	**97.83**
plant-margin	81.88	**81.88**	81.25	79.75	80.50
plant-shape	71.75	72.31	**72.75**	51.50	**73.25**
plant-texture	84.06	**85.25**	81.25	80.00	84.50
ringnorm	**98.26**	97.97	97.51	98.11	**98.38**
semeion	**93.47**	92.96	91.96	91.46	**93.97**
spambase	93.61	93.87	94.09	**94.61**	94.26
statlog-german-credit	76.20	**77.70**	75.60	77.20	**78.80**
statlog-image	96.79	**96.84**	95.49	95.84	**97.23**
statlog-landsat	90.70	**91.20**	91.00	90.55	**91.20**
statlog-shuttle	99.88	99.91	99.90	**99.92**	**99.92**
steel-plates	58.66	76.44	**78.35**	76.29	77.11
thyroid	95.92	95.65	**98.16**	97.99	96.18
titanic	78.82	**78.82**	78.36	77.27	**79.09**
twonorm	97.78	97.81	**98.05**	97.35	**98.11**
wall-following	90.71	90.30	90.98	**92.23**	90.98
waveform	86.34	**86.44**	84.80	83.60	**86.88**
waveform-noise	**86.32**	85.70	86.08	85.84	86.00
wine-quality-red	61.75	**65.63**	63.00	61.50	**66.00**
wine-quality-white	62.36	63.30	**63.73**	63.07	**65.60**
yeast	60.24	61.66	**63.07**	54.99	58.22
Average Accuracy	85.43	86.30	86.47	85.02	**87.21**
Average RANK	3.50	2.95	2.97	3.72	**1.87**

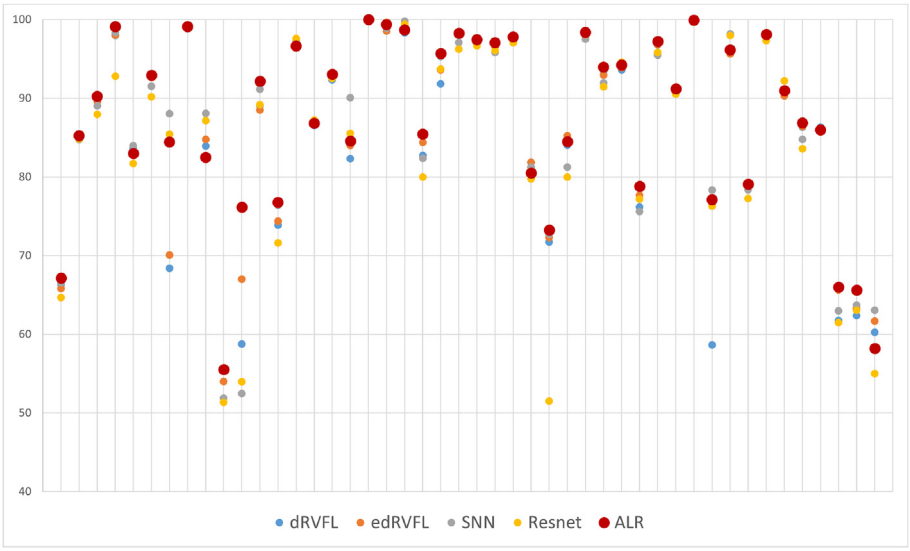

Fig. 4. The accuracy of various deep approaches and ablation results. The horizontal axis represents the different data sets and the vertical axis indicates the accuracy rate. The different coloured scatters represent different algorithms. The proposed algorithm (ALR) are highlighted with larger size.

Table 6
Pairwise statistical comparison of all mentioned algorithms.

	BLS	SCN	RVFL	HELM	ResNet	SNN	dRVFL	edRVFL	edRandom	AL-RVFL	ALR
SCN			-	-	-	-	-	-	-	-	-
BLS				-	-	-	-	-	-	-	-
RVFL	+					-	-	-	-	-	-
HELM	+	+							-	-	-
ResNet	+	+							-	-	-
dRVFL	+	+	+							-	-
edRVFL	+	+	+							-	-
SNN	+	+	+								-
edRandom	+	+	+	+	+						-
AL-RVFL	+	+	+	+	+	+	+				-
ALR	+	+	+	+	+	+	+	+	+	+	

* + means the method in the corresponding row is statistically better than the method in the corresponding column. Similarly, - indicates that the method in the corresponding row is statistically worse than the method in the corresponding column.

tion experiment is divided into two sections. The results are given in Table 4 and Fig. 3. From the figure, we can visualise that our method is useful in achieving the highest accuracy rates on most data sets. The first is built on the ensemble deep framework and contains a variety of randomized networks topology situations, called as edRandom. We do not employ layer-wise training or offer sample-level attention in this experiment, the architecture of each hidden layer is randomly picked. The other approach, termed AL-RVFL, uses just RVFL as the base architecture but employs automated layer-wise training.

We can deduce from this basic experiment that the proposed ALR model is viable. From the results of AL-RVFL, we can conclude that the layer-wise training method is significantly effective, the average ranking and average accuracy ranked second out of nine. We can reasonably speculate that having separate hyper-parameters for the different hidden layers is beneficial to the effectiveness of the framework. As seen by the final findings of ALR, the integration of different backbone networks guided by BO significantly improves the results. This demonstrates the need of considering whether to establish direct linkages and alter the output of certain hidden layers.

Moreover, we compare the proposed ALR approach against other neural networks on the broader datasets, and all training sets in this experiment have sample sizes greater than 1000. The elaborated results are given in Table 5 and Fig. 4. Our method is superior to the other two neural network baselines.

Additionally, we present results from pairwise comparisons in Table 6. The absence of an item indicates that there is no statistically significant difference noticed. The sign + indicates that the technique in the adjacent row is statistically superior to the method in the adjacent column; the symbol – indicates that the method in the adjacent row is statistically inferior to the method in the adjacent column.

5. Conclusion

We suggested an Automated Layer-wise pipeline in this study that may be used to replace the original ensemble randomized neural network training process with enhancing performance. By the help of sample-wise attention, we may increase the capacity of the randomized neural network implicitly and therefore improve performance. And to the best of our knowledge, this is the first work integrated various randomized feed-forward neural network architectures into one framework. We outlined the distinctions between randomized networks, the main difference between which lies in the representation of the direct link. The fusion of different networks can improve the stability of the model, especially in the ALR pipeline. Additionally, an efficient hyper-parameter optimization tool, such as Hyperband with Bayesian Optimization, can also have a significant effect on the model's performance. Based on empirical evidence evaluated on a diverse col-

lection of benchmark tabular datasets, it has been shown that deep ensemble randomized networks provides improved performance with automated layer-wise solution.

Our further research will first concentrate on more robust methods of hyper-parameter optimization. The current pipeline is quite sensitive to the range of hyper-parameters, and we need a more reliable method for optimizing the range of hyper-parameters based on the property of datasets. We are also exploring how the ALR pipeline can be adapted to other tasks such as regression, time series prediction, etc. Simultaneously, there are other details of our model that need consideration, such as the selection of similar samples. A good selection strategy enables us to distinguish samples more accurately, hence determining the hidden layer's inputs and optimizing performance.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

Open Access funding was provided by the Qatar National Library.

References

[1] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A.P. Steiner, D. Keysers, J. Uszkoreit, et al., Mlp-mixer: An all-mlp architecture for vision, in: Thirty-Fifth Conference on Neural Information Processing Systems, 2021.
[2] Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao, C. Zheng, Synthesizer: Rethinking self-attention for transformer models, in: International Conference on Machine Learning, PMLR, 2021, pp. 10183–10192.
[3] P.N. Suganthan, R. Katuwal, On the origins of randomization-based feedforward neural networks, Applied Soft Computing 105 (2021) 107239.
[4] W.F. Schmidt, M.A. Kraaijveld, R.P. Duin, et al., Feed forward neural networks with random weights, in: International Conference on Pattern Recognition, IEEE Computer Society Press, 1992, pp. 1–1.
[5] Y.-H. Pao, Y. Takefuji, Functional-link net computing: theory, system architecture, and functionalities, Computer 25 (5) (1992) 76–79.
[6] Y.-H. Pao, G.-H. Park, D.J. Sobajic, Learning and generalization characteristics of the random vector functional-link net, Neurocomputing 6 (2) (1994) 163–180.
[7] P.N. Suganthan, On non-iterative learning algorithms with closed-form solution, Applied Soft Computing 70 (2018) 1078–1082.
[8] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[9] C.P. Chen, Z. Liu, Broad learning system: An effective and efficient incremental learning system without the need for deep architecture, IEEE transactions on neural networks and learning systems 29 (1) (2017) 10–24.
[10] Q. Shi, R. Katuwal, P. Suganthan, M. Tanveer, Random vector functional link neural network based ensemble deep learning, Pattern Recognition 117 (2021) 107978.
[11] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, F. Hutter, Efficient and robust automated machine learning, in: C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 28, Curran Associates Inc, 2015.
[12] M. Feurer, A. Klein, K. Eggensperger, J.T. Springenberg, M. Blum, F. Hutter, Auto-sklearn: efficient and robust automated machine learning, in: Automated Machine Learning, Springer, Cham, 2019, pp. 113–134.
[13] C. Thornton, F. Hutter, H.H. Hoos, K. Leyton-Brown, Auto-weka: Combined selection and hyperparameter optimization of classification algorithms, in: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, 2013, pp. 847–855.
[14] S. Falkner, A. Klein, F. Hutter, Bohb: Robust and efficient hyperparameter optimization at scale, in: International Conference on Machine Learning, PMLR, 2018, pp. 1437–1446.
[15] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, R. Sass, F. Hutter, Smac3: A versatile bayesian optimization package for hyperparameter optimization (2021). arXiv:2109.09831.
[16] M. Pelikan, D.E. Goldberg, E. Cantú-Paz, et al., Boa: The bayesian optimization algorithm, in: Proceedings of the genetic and evolutionary computation conference GECCO-99, Vol. 1, Citeseer, 1999, pp. 525–532.
[17] Q. Shi, M. Hu, P.N. Suganthan, R. Katuwal, Weighting and pruning based ensemble deep random vector functional link network for tabular data classification, Pattern Recognition 108879 (2022).
[18] L. Kotthoff, C. Thornton, H.H. Hoos, F. Hutter, K. Leyton-Brown, Auto-weka: Automatic model selection and hyperparameter optimization in weka, in: Automated Machine Learning, Springer, Cham, 2019, pp. 81–95.
[19] R. Caruana, A. Niculescu-Mizil, G. Crew, A. Ksikes, Ensemble selection from libraries of models, in: Proceedings of the twenty-first international conference on Machine learning, 2004, p. 18.
[20] M. Feurer, J.T. Springenberg, F. Hutter, Using meta-learning to initialize bayesian optimization of hyperparameters., MetaSel@ ECAI, Citeseer (2014) 3–10.
[21] J. Snoek, H. Larochelle, R.P. Adams, Practical bayesian optimization of machine learning algorithms, Advances in neural information processing systems 25 (2012).
[22] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, Advances in neural information processing systems 24 (2011).
[23] L. Li, K.G. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: Bandit-based configuration evaluation for hyperparameter optimization, in: ICLR (Poster), 2017.
[24] K. Jamieson, A. Talwalkar, Non-stochastic best arm identification and hyperparameter optimization, in: Artificial Intelligence and Statistics, PMLR, 2016, pp. 240–248.
[25] N. Awad, N. Mallik, F. Hutter, Dehb: Evolutionary hyberband for scalable, robust and efficient hyperparameter optimization, arXiv preprint arXiv:2105.09821 (2021).
[26] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural computation 18 (7) (2006) 1527–1554.
[27] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in neural information processing systems, 2007, pp. 153–160.
[28] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, in: Proceedings of the 31st international conference on neural information processing systems, 2017, pp. 972–981.
[29] P. Ramachandran, B. Zoph, Q.V. Le, Searching for activation functions, arXiv preprint arXiv:1710.05941 (2017).
[30] P. Baldi, R. Vershynin, The capacity of feedforward neural networks, Neural networks 116 (2019) 288–311.
[31] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[32] D. Wang, M. Li, Stochastic configuration networks: Fundamentals and algorithms, IEEE transactions on cybernetics 47 (10) (2017) 3466–3479.
[33] J. Tang, C. Deng, G.-B. Huang, Extreme learning machine for multilayer perceptron, IEEE transactions on neural networks and learning systems 27 (4) (2015) 809–821.
[34] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (Jan) (2006) 1–30.
[35] M. Hu, P.N. Suganthan, Experimental evaluation of Stochastic Configuration Networks: Is SC algorithm inferior to hyper-parameter optimization method?, Applied Soft Computing 126 (2022) 109257
[36] M. Hu, P.N. Suganthan, Representation learning using deep random vector functional link networks for clustering, Pattern Recognition 129 (2022) 108744.
[37] M.A. Ganaie, M. Hu, A.K. Malik, M. Tanveer, P.N. Suganthan, Ensemble deep learning: A review, Engineering Applications of Artificial Intelligence 115 (2022) 105151.
[38] M. Hu, Q. Shi, P.N. Suganthan, M. Tanveer, Adaptive Ensemble Variants of Random Vector Functional Link Networks, International Conference on Neural Information Processing (2020) 30–37.

Minghui Hu received his B. Eng degree from Dalian Maritime University, China in 2018. Currently, he is a PhD student in the School of Electrical and Electronic Engineering in Nanyang Technological University, Singapore. His research interest includes randomized neural networks, machine learning and computer vision.

Ruobin Gao received his BEng degree from Jilin University, China, MSc and PhD degrees from Nanyang Technological University, Singapore. He is a research fellow at Nanyang Technological University. His research interests are time series forecasting and statistical learning. He also has extensive research experience in fuzzy time series and echo state neural network.

Ponnuthurai Nagaratnam Suganthan (or P N Suganthan) received the B.A degree, Postgraduate Certificate and M.A degree in Electrical and Information Engineering from the University of Cambridge, UK in 1990, 1992 and 1994, respectively. He received an honorary doctorate (i.e. Doctor Honoris Causa) in 2020 from University of Maribor, Slovenia. After completing his PhD research in 1995, he served as a predoctoral Research Assistant in the Dept. of Electrical Engineering, University of Sydney in 1995-96 and a lecturer in the Dept of Computer Science and Electrical Engineering, University of Queensland in 1996-99. He is currently a Research Professor at Qatar University. He is an Editorial Board Member of the Evolutionary Computation Journal, MIT Press (2013-2018). He is an associate editor of the IEEE Trans. on Cybernetics (2012 -2018), IEEE Trans. on Evolutionary Computation (2005 -), Information Sciences (Elsevier) (2009 -), Pattern Recognition (Elsevier) (2001 -), Applied Soft Computing (2018-), Neurocomputing (2018-) and Engineering Applications of Artificial Intelligence (2022 -) Journals. He is a founding co-editor-in-chief of Swarm and Evolutionary Computation (2010 -), an SCI Indexed Elsevier

Journal. His co-authored SaDE paper (published in April 2009) won the "IEEE Trans. on Evolutionary Computation outstanding paper award" in 2012. His research interests include swarm and evolutionary algorithms, pattern recognition, big data, deep learning and applications of swarm, evolutionary & machine learning algorithms. He was selected as one of the highly cited researchers by Thomson Reuters every year between 2015 and 2020 in computer science. He served as the General Chair of the IEEE SSCI 2013. He has been a member of the IEEE since 1991 and Fellow since 2015. He was an elected AdCom member of the IEEE Computational Intelligence Society (CIS) in 2014-2016. He is an IEEE CIS distinguished lecturer (2018-2021).

Dr. M. Tanveer is currently working as Associate Professor and Ramanujan Fellow at the Department of Mathematics of the Indian Institute of Technology Indore. Prior to that, he worked as a Postdoctoral Research Fellow at the Rolls-Royce@NTU Corporate Lab of the Nanyang Technological University, Singapore. He received the Ph.D degree in Computer Science from the Jawaharlal Nehru University, New Delhi, India. Prior to that, he received the M.Phil degree in Mathematics from Aligarh Muslim University, Aligarh, India. His research interests include support vector machines, optimization, machine learning, deep learning, randomized neural networks, applications to Alzheimer's disease and dementias. He has published over 100 referred journal papers of international repute. His publications have over 2700 citations with h index 28 (Google Scholar, September 2022). Recently, he has been listed in the world's top 2% scientists in the study carried out by Stanford University, USA. He is the recipient of the 2017 SERB-Early Career Research Award in Engineering Sciences and the only recipient of 2016 DST-Ramanujan Fellowship in Mathematical Sciences. He is currently the Associate Editor - IEEE Transactions on Neural Networks and Learning Systems, Associate Editor - Pattern Recognition (Elsevier), Action Editor - Neural Networks (Elsevier), Board of Editors - Engineering Applications of Artificial Intelligence (Elsevier), Associate Editor - Neurocomputing (Elsevier), Associate Editor - Cognitive Computation (Springer), Editorial Board - Applied Soft Computing (Elsevier), International Journal of Machine Learning and Cybernetics (Springer), He has also co-edited one book in Springer on machine intelligence and signal analysis. He has organized many international/national conferences/symposium/workshop as General Chair/Organizing Chair/Coordinator, and delivered talks as Keynote/Plenary/invited speaker in many international conferences and Symposiums. He has organized several special sessions in top-ranked conferences including WCCI, IJCNN, IEEE SMC, IEEE SSCI, ICONIP. Amongst other distinguished, international conference chairing roles, he is the General Chair for 29th International Conference on Neural Information Processing (ICONIP2022). Tanveer is currently the Principal Investigator (PI) or Co-PI of 11 major research projects funded by Government of India including DST, SERB, CSIR, MHRD-SPARC and ICMR.