

# Random Vector Functional Link Neural Network based Ensemble Deep Learning

Rakesh Katuwal<sup>a</sup>, P.N. Suganthan<sup>a,\*</sup>, M. Tanveer<sup>b</sup>

<sup>a</sup>*School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore*

<sup>b</sup>*Discipline of Mathematics, Indian Institute of Technology Indore, Simrol, Indore, 453552, India*

---

## Abstract

In this paper, we propose a deep learning framework based on randomized neural network. In particular, inspired by the principles of Random Vector Functional Link (RVFL) network, we present a deep RVFL network (dRVFL) with stacked layers. The parameters of the hidden layers of the dRVFL are randomly generated within a suitable range and kept fixed while the output weights are computed using the closed form solution as in a standard RVFL network. We also propose an ensemble deep network (edRVFL) that can be regarded as a marriage of ensemble learning with deep learning. Unlike traditional ensembling approaches that require training several models independently from scratch, edRVFL is obtained by training a single dRVFL network once. Both dRVFL and edRVFL frameworks are generic and can be used with any RVFL variant. To illustrate this, we integrate the deep learning networks with a recently proposed sparse-pretrained RVFL (SP-RVFL). Extensive experiments on benchmark datasets from diverse domains show the superior performance of our proposed deep RVFL networks.

**Keywords:** Random Vector Functional Link (RVFL), deep RVFL, multi-layer RVFL, ensemble deep learning, randomized neural network.

---



---

\*Corresponding author

Email addresses: [rakeshku001@e.ntu.edu.sg](mailto:rakeshku001@e.ntu.edu.sg) (Rakesh Katuwal), [epnsugan@ntu.edu.sg](mailto:epnsugan@ntu.edu.sg) (P.N. Suganthan), [mtanveer@iiti.ac.in](mailto:mtanveer@iiti.ac.in) (M. Tanveer)

## 1. Introduction

Deep Learning, also known as representational learning, has sparked a surging interest in neural networks amongst the machine learning enthusiasts with the state-of-the-art results in diverse applications ranging from image/video classification to segmentation, action recognition and many others. The superiority of a deep learning model emanates from its potential ability to extract meaningful representations at different levels of the hierarchical model while disentangling a complex task into several simpler ones [1].

Deep neural networks typically consist of multiple hidden layers stacked together. Each hidden layer builds an internal representation of the data with the hidden layers closer to the input layer learning simple features such as edges and layers above them learning sophisticated (complex) features [1, 2]. With such stacked layers, deep learning models typically have thousands of model parameters that need to be optimized during the training phase. These networks are typically trained using back-propagation (BP) technique so as to minimize the loss function (cross-entropy or mean square error or others depending on the particular task). In addition to be time-consuming, such models may fail to converge to a global minimum, thus, giving sub-optimal performance or lower generalization [3]. Also, such deep learning models require large amount of training data. While the usual image and speech datasets that are commonly used with deep learning models have abundant data, there are datasets from a wide variety of domains, such as agriculture, credit scoring, health outcomes, ecology and others, with very limited data size. The performance of the state-of-the-art deep learning models on such datasets are far from superior [4].

Apart from the conventional BP-trained neural networks, there has also been a growing interest in the class of randomization based neural networks [5, 6, 7]. Randomization based neural networks with closed form solution avoid the pitfalls of conventional BP-trained neural networks [3, 8, 9]. They are faster to train and have demonstrated good learning performance [10, 11]. Among the randomization based methods, Random Vector Functional Link (RVFL) [12] network

has rapidly gained significant traction because of its superior performance in several diverse domains ranging from visual tracking [13], classification [14, 15], regression [16], to forecasting [17, 18]. RVFL is a single layer feed-forward neural network (SLFN) in which the weights and biases of the hidden neurons are randomly generated within a suitable range and kept fixed while the output weights are computed via a simple closed form solution [12, 19]. Randomization based neural networks greatly benefit from the presence of direct links from the input layer to the output layer as in RVFL network [16, 18, 20]. The original features are reused or propagated to the output layer via the direct links. The direct links act as a regularization for the randomization [21, 22]. It also helps to keep the model complexity low with the RVFL network being thinner and simpler compared to its other counterparts. With the Occam’s Razor principle and PAC learning theory [23] advocating for simpler and less complex models, this makes the RVFL network attractive to use compared to other similar randomized neural networks.

Ensembles of neural networks are known to be much more robust and accurate than individual networks [20, 24, 25, 26]. Because of the existence of several randomization operations in their training procedure, neural networks are regarded as unstable algorithms whose performance greatly vary even when there is a small perturbation in training set or random seed. It is therefore not surprising that two neural networks with identical architectures optimized with different initialization or slightly perturbed training data will converge to different solutions. This diversity can be exploited through ensembling, in which multiple neural networks are trained with slightly different training set or parameters and then combined with majority voting or averaging. Ensembling often leads to drastic reductions in error rates. However, this comes with an obvious trade off: computational cost. While ensembling shallow neural networks doesn’t incur great computational cost, the same is not true for the ensembling of deep networks.

With the current trend of building deep networks, there have also been several attempts in the literature to build deep or multi-layer networks based on

randomized neural networks [27, 28, 29]. Even though there exist several deep learning models with randomized neural networks, there are limited works in the context of RVFL network. In this paper, we investigate the performance of deep learning and ensemble deep learning models based on RVFL networks. To the best of our knowledge, [28] is one of the pioneering paper to propose multi-layer RVFL network. However, the performance of the multi-layer RVFL network compared to a shallow RVFL network (with 1 hidden layer) is sub-optimal and non-persuasive. A deep model enriched with complex feature learning capabilities should achieve good generalization. Thus, in this paper, we propose deep neural networks based on RVFL while maintaining its advantages of lower complexity, training efficiency and good generalization. We also propose an ensemble of such deep networks without incurring any significant training costs. Specifically, we propose an ensemble deep RVFL network which can be regarded as a marriage of ensemble and deep learning that is simple and straight-forward to implement. The key contributions of this paper are summarized as follows:

- We propose a deep RVFL network (dRVFL), an extension of RVFL for representational learning. The dRVFL network consists of several hidden layers stacked on top of each other. The parameters of the hidden layers are randomly generated and kept fixed while only the output weights need to be computed. Thus, the deep RVFL network emanates from the standard RVFL network.
- We also propose an implicit ensembling approach called ensemble deep RVFL framework (edRVFL), a marriage of ensembling learning with deep learning. Instead of training  $L$  neural networks independently from scratch as in traditional ensembling method, we only train a single deep RVFL network. The ensemble consists of  $L$  models equivalent to the number of hidden layers in the single deep RVFL network. The ensemble is trained in such a way that the higher models (equivalent to higher layers in deep RVFL network) utilize both the original features (from direct links as in standard RVFL network) and non-linearly transformed features from the

preceding layers. Thus, the framework is consistent with the tenets of both ensemble learning and deep learning at the same time. The training cost of edRVFL is slightly higher than that of a single dRVFL network while it is significantly lower than that of traditional ensembles.

- The deep learning models proposed in this paper (dRVFL and edRVFL) are generic and are applicable with any RVFL variant. We create deep learning models using both standard RVFL and recently proposed sparse pre-trained RVFL (SP-RVFL) [30].
- With extensive experiments on several real-world classification datasets, we show that our proposed deep RVFL models (dRVFL and edRVFL) have superior performance compared to other relevant neural networks.

The rest of this paper is structured as follows. Section 2 gives a brief overview of related works on shallow randomized neural networks followed by randomization based multi-layer neural network. Section 3 details our proposed deep RVFL method followed by its ensemble. In Section 4, we compare the performance of our proposed methods with other relevant neural networks. Finally, the conclusion is presented in Section 5.

## 2. Related Works

In this section, we present a brief overview of the fundamentals of a standard RVFL network, extreme learning machine (ELM) as a variant of RVFL, state-of-the-art sparse pre-trained RVFL (SP-RVFL) network and hierarchical ELM (HELM), a randomization based multi-layer neural network.

### 2.1. Random Vector Functional Link Network (RVFL)

A basic framework of the standard RVFL network [12] is shown in Fig. 1(a). The inputs to the output layer in RVFL consist of both non-linearly transformed features  $\mathbf{H}$  from the hidden layer and original input features  $\mathbf{X}$ . If  $d$  be the input data features and  $N$  be the number of hidden nodes, then there are total  $d + N$

inputs for each output node. Since the hidden layer parameters are randomly generated and kept fixed during the training phase, only the output weights  $\beta_s$  need to be computed. Thus, the resulting optimization problem can be mathematically represented as:

$$\min_{\beta_s} \|\mathbf{D}\beta_s - Y\|^2 + \lambda\|\beta_s\|^2, \quad (1)$$

where  $\mathbf{D} = [\mathbf{H} \ \mathbf{X}]$  is the concatenation of hidden features and original features,  $\lambda$  is the regularization parameter and  $Y$  is the target vector.

Typically, Eq. 1 can be solved via a closed form solution using either ridge regression (i.e.  $\lambda \neq 0$ ) or Moore-Penrose pseudoinverse (i.e.  $\lambda = 0$ ) [31]. Using Moore-Penrose pseudoinverse, the solution is given by:  $\beta_s = \mathbf{D}^+ Y$  while using the regularized least squares (or ridge regression), the closed form solution is given by:

$$\text{Primal Space: } \beta_s = (\mathbf{D}^T \mathbf{D} + \lambda \mathbf{I})^{-1} \mathbf{D}^T Y, \quad (2)$$

$$\text{Dual Space: } \beta_s = \mathbf{D}^T (\mathbf{D} \mathbf{D}^T + \lambda \mathbf{I})^{-1} Y. \quad (3)$$

The training complexity in the RVFL network introduced by the matrix inversion operation can be circumvented by using either primal or dual solution depending on the sample size or total feature dimensions (i.e. input features plus total number of hidden neurons) [3].

## 2.2. Extreme Learning Machine (ELM)

ELM [32], developed in 2004, can be viewed as a variant of RVFL without direct links and bias term (see Figure 1(b)). Thus, Eq. 1 becomes

$$\min_{\beta_E} \|\mathbf{H}\beta_E - Y\|^2 + \lambda\|\beta_E\|^2. \quad (4)$$

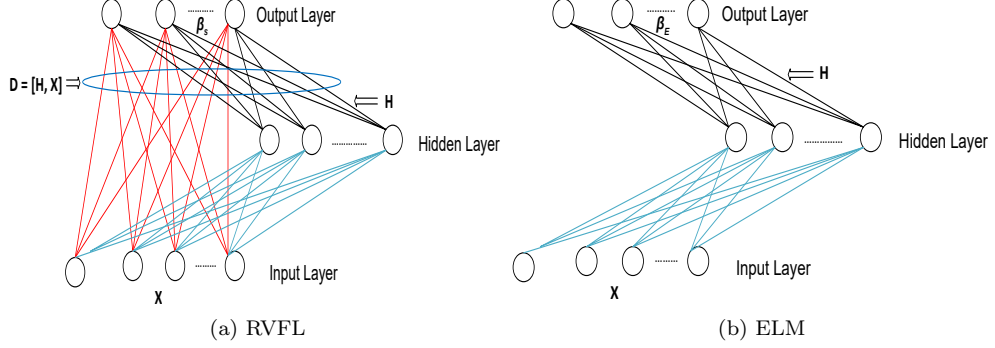


Figure 1: Framework of RVFL (1994) and ELM (2004) networks. The structure of RVFL and ELM differ in the presence (absence) of direct links and bias term (not shown in the figure). The red lines represent the direct links (original features) from the input to the output layer. The weights for the blue lines are randomly generated from a certain range and kept fixed. Only the output weights (associated with red and black lines) need to be computed. Best viewed in color.

Its solution is:

$$\text{Primal Space: } \beta_E = (\mathbf{H}^T \mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{H}^T Y, \quad (5)$$

$$\text{Dual Space: } \beta_E = \mathbf{H}^T (\mathbf{H} \mathbf{H}^T + \lambda \mathbf{I})^{-1} Y. \quad (6)$$

### 2.3. Sparse pre-trained RVFL (SP-RVFL)

In a standard RVFL network, the hidden layer parameters ( $\mathbf{w}$  and  $b$ ) are randomly generated within a suitable range and kept fixed thereafter. Even though RVFL has demonstrated its efficacy in various domains, its performance is often challenged by randomly assigned hidden layer parameters. To alleviate this issue, the authors in [30] proposed an unsupervised parameter learning based RVFL known as sparse pre-trained RVFL (SP-RVFL). In an SP-RVFL network, an autoencoder with  $l_1$  regularization is employed to learn the hidden layer parameters. Specifically, the optimization problem for the autoencoder is given by:

$$\min_{\tilde{\mathbf{H}}} \|\tilde{\mathbf{H}} \boldsymbol{\varpi} - \mathbf{X}\|^2 + \|\boldsymbol{\varpi}\|_1, \quad (7)$$

where  $\mathbf{X}$  is the input,  $\tilde{\mathbf{H}}$  is the hidden layer matrix obtained via random

mapping and  $\varpi$  is the output weight matrix of the autoencoder. The above optimization problem, Eq. 7 is solved using a fast iterative shrinkage-thresholding algorithm (FISTA) [33]. The  $\varpi$  pre-trained by sparse-autoencoder is then used as the weights of the hidden layer of a standard RVFL. The hidden biases are then computed as:

$$\hat{b}_i = \frac{\sum_{j=1}^d \varpi_{ij}}{d}, \quad i = 1, 2, \dots, N. \quad (8)$$

With the pre-trained hidden layer parameters, the output of the hidden layer  $\mathbf{H}$  of RVFL is computed as:

$$\mathbf{H} = g(\mathbf{X}\varpi + \hat{b}), \quad (9)$$

where  $g(\cdot)$  is a non-linear activation function. Only the Eq. 9 in SP-RVFL differs from a standard RVFL network with  $\mathbf{H}$  of a standard RVFL given by  $\mathbf{H} = g(\mathbf{X}\mathbf{w} + b)$  where  $\mathbf{w}$  and  $b$  are randomly generated. The optimization problem of SP-RVFL then becomes similar to the optimization problem given in Eq. 1. Eqs. 2 and 3 are then used to compute the output weights  $\beta_s$  as in a standard RVFL network.

#### 2.4. Hierarchical ELM (HELM)

The HELM [34] is a randomized multi-layer neural network based on ELM. It consists of two components: feature encoding using ELM and an ELM based classifier. For feature extraction, it uses sparse autoencoder as defined by Eq. 7 in the preceding section. Multiple hidden layers are then stacked on top of each other for the feature extraction part. The extracted features are then used by ELM classifier for final decision making.

### 3. Deep RVFL for representational learning

In this section, we introduce our proposed deep learning frameworks based on RVFL. We first describe the deep RVFL network in Section 3.1. We then



elucidate our proposed ensemble deep RVFL network in Section 3.2.

### 3.1. Deep Random Vector Functional Link Network

The Deep Random Vector Functional Link (dRVFL) network is an extension of the shallow RVFL network in the context of representation learning or deep learning. The dRVFL network is typically characterized by a stacked hierarchy of hidden layers as shown in Fig. 2. The input to each layer in the stack is the output of the preceding layer wherein each layer builds an internal representation of the input data. Although the stacked hierarchical organization of hidden layers in dRVFL network allows a general flexibility in the size (both in width and depth) of the network, for the sake of simplicity here we consider a stack of  $L$  hidden layers each of which contains the same number of hidden nodes  $N$ .

For the ease of notation, we omit the bias term in the formulas. The output of the first hidden layer is then defined as follows:

$$\mathbf{H}^{(1)} = g(\mathbf{X}\mathbf{W}^{(1)}), \quad (10)$$

while for every layer  $> 1$  it is defined as:

$$\mathbf{H}^{(L)} = g(\mathbf{H}^{(L-1)}\mathbf{W}^{(L)}), \quad (11)$$

where  $\mathbf{W}^{(1)} \in \mathbb{R}^{d \times N}$  and  $\mathbf{W}^{(L)} \in \mathbb{R}^{N \times N}$  are the weight matrices between the input-first hidden layer and inter hidden layers respectively. These parameters (weights and biases) of the hidden neurons are randomly generated within a suitable range and kept fixed during the training.  $g(\cdot)$  is the non-linear activation function. The input to the output layer is then defined as:

$$\mathbf{D} = [\mathbf{H}^{(1)} \ \mathbf{H}^{(2)} \ \dots \ \mathbf{H}^{(L-1)} \ \mathbf{H}^{(L)} \ \mathbf{X}]. \quad (12)$$

This design structure is very similar to the standard shallow RVFL network wherein the input to the output layer consists of non-linear features from the stacked hidden layers along with the original features. The output of the dRVFL

network is then defined as follows:

$$Y = \mathbf{D}\beta_d. \quad (13)$$

The output weight  $\beta_d \in \mathbb{R}^{(NL+d) \times K}$  ( $K$ : the number of classes) is then solved using Eqs. 2 or 3.

From Eqs. 12 and 13, one can see that in dRVFL there exists a linear combination between the features and the output layer weight matrix  $\beta_d$  i.e. a weighted sum of the features from the hidden layers including the input layer. In the training stage, this directly enables the model to weigh differently the contribution of each type of features originating from different layers.

It is also worth mentioning that our proposed dRVFL network differentiates itself from the deep learning architecture proposed in [35] in threefold: 1) dRVFL is inspired by a shallow RVFL network with the output layer consisting of both non-linearly transformed features and original features via direct links. In contrast, the deep learning architecture proposed in [35] does not consider the original features during the output weight computation, 2) we investigate the performance of the dRVFL network in classification problems while [35] explores time-series problems, and 3) the deep learning framework, dRVFL is generic and can be used with any RVFL variant.

### 3.2. Ensemble Deep Random Vector Functional Link Network

The framework of the ensemble deep RVFL network (edRVFL) is shown in Fig. 3. It serves three purposes: 1) instead of using only the higher level representations (features extracted from the final hidden layer) of the data as in a conventional deep learning model [34] for classification, it employs rich intermediate features also for final decision making. 2) the ensemble is obtained by training a single dRVFL network once with a training cost slightly higher than that of a single dRVFL but cheaper than training several independent models of dRVFL. 3) like dRVFL, the edRVFL framework is generic and any RVFL variant can be used with it.

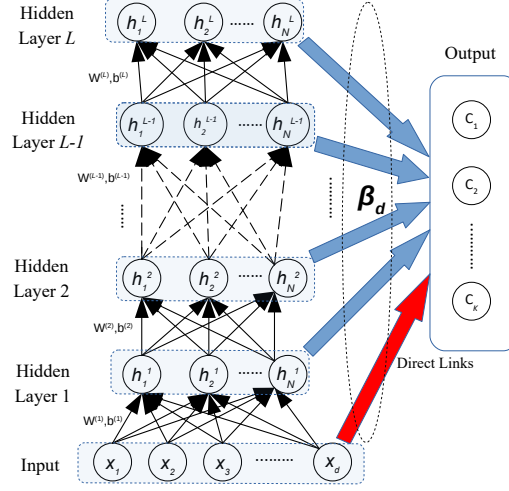


Figure 2: Framework of a dRVFL network. It consists of several hidden layers stacked on top of each other whose parameters (weights and biases between the hidden layers) are randomly generated and kept fixed during the training. Only the output weights  $\beta_d$  need to be computed as in the shallow RVFL network.

The computation of the output weight  $\beta_d$  in the dRVFL network described in Section 3.1 requires matrix inversion of size either  $(T \times T)$  or  $((NL + d) \times (NL + d))$ , whichever is small (refer to primal and dual solutions, Eqs. 2 and 3) where  $T$  is the training data size,  $L$  is the number of hidden layers,  $N$  is the number of hidden nodes at each layer and  $d$  is the dimension of the data. For simplicity, we omit the bias terms. In a standard implementation, the matrix inversion of a matrix of size  $(T \times T)$  requires  $\mathcal{O}(T^3)$  time and  $\mathcal{O}(T^2)$  memory [36]. In case of dRVFL, this is equivalent to either  $\mathcal{O}(T^3)$  or  $\mathcal{O}((NL + d)^3)$  time and either  $\mathcal{O}(T^2)$  or  $\mathcal{O}((NL + d)^2)$  memory. Such scaling is prohibitive when all  $N$ ,  $L$ ,  $T$  and  $d$  are large. Inversion of a large matrix can also result in out-of-memory failures thus, requiring powerful and high performance hardware [36]. Depending on the dataset, all these parameters can be actually large. For example, one dataset used in this paper has 7000 training samples with 5000 features. A dRVFL network with 10 hidden layers and 100 hidden nodes in each layer, requires matrix inversion of size  $(6000 \times 6000)$  using primal solution (using dual solution would require matrix inversion of size  $(7000 \times 7000)$ ). Thus, we

decompose the computation of the final output weight  $\beta_d$  of dRVFL into several small  $\beta_{ed}$  in edRVFL. Specifically, each small  $\beta_{ed}$  is independently computed (treated as independent model) and the final output is obtained by using either majority voting or averaging of the models. Each small  $\beta_{ed}$  requires the matrix inversion of size either  $(T \times T)$  or  $((N + d) \times (N + d))$ . Without direct links, it would require an inversion of size either  $(T \times T)$  or  $(N \times N)$ . However, as discussed in the preceding sections, direct links are essential parts of randomized neural networks [16, 18, 20, 21, 22]. The significance of such direct links is also discussed later in Section 4.4.1.

The input to each hidden layer is the non-linearly transformed features from the preceding layer as in dRVFL along with the original input features (direct links) as in standard RVFL. The direct links act as a regularization for the randomization. The input of the first hidden layer is then defined as follows:

$$\mathbf{H}^{(1)} = g(\mathbf{X}\mathbf{W}^{(1)}), \quad (14)$$

while for every layer  $> 1$  it is defined as:

$$\mathbf{H}^{(L)} = g([\mathbf{H}^{(L-1)}\mathbf{X}]\mathbf{W}^{(L)}). \quad (15)$$

The output weights  $\beta_{ed}$  are then solved independently using Eqs. 2 or 3.

The mechanism of obtaining several models while training only a single model (implicit ensembles) as in ensemble deep RVFL network (edRVFL) is related to the snapshot ensembling of [25] which trains a neural network using a cyclic learning rate schedule to converge to different local minima. Instead of training several neural networks independently (true ensembles), the method saves (snapshots the parameters) each time the model converges and adds the corresponding network to the ensemble (also known as implicit ensemble). However, such approach is only applicable to neural networks trained with stochastic gradient descent (SGD). Since RVFL neural networks can be trained using closed form solutions, no learning rate mechanism is required. Like snapshot ensembling, edRVFL can even be ensembled if enough resources are available

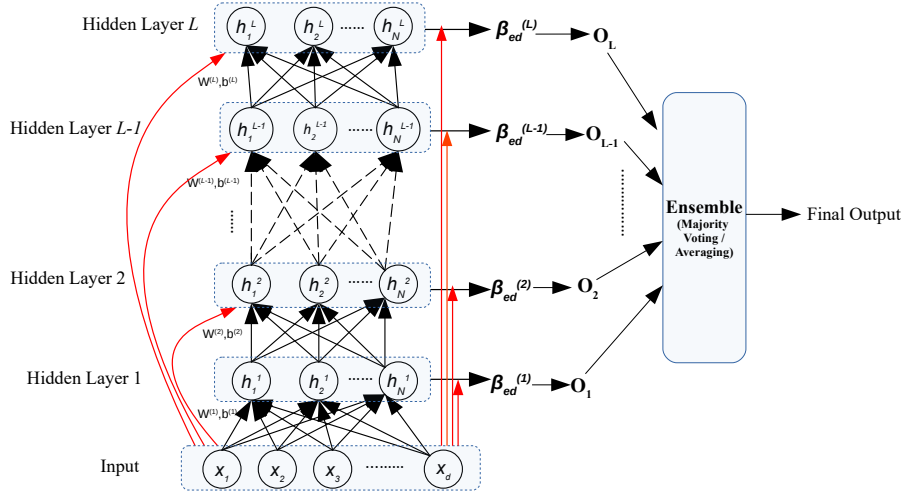


Figure 3: Framework of ensemble deep RVFL network (edRVFL). It differs from dRVFL in that the computation of final output weight  $\beta_d$  is decomposed into several small  $\beta_{ed}$ . Specifically, each small  $\beta_{ed}$  is independently computed (treated as independent model) and the final output is obtained by using either majority voting or averaging of the models. Each higher level model is fed with original input data and the non-linearly transformed features from the preceding model.  $O_1, \dots, O_L$  represents the output of each model.

during training.

## 4. Experiments

### 4.1. Datasets

The experiments are performed on 13 publicly available real-world classification datasets from various domains used in [30] which include two biomedical datasets (Carcinom and Lung), two human face image datasets (ORL and Yale), four hand-written digit datasets (Binary Alphabet(BA), Gisette, a portion of MNIST and USPS), two object recognition datasets (COIL20, COIL100) and three text datasets (BASEHOCK, RCV1 and TDT2). Table 1 gives an overview of the 13 real-world application datasets.

### 4.2. Compared Methods

To verify the effectiveness of our proposed deep learning frameworks, we perform comparisons against relevant algorithms (shallow RVFL networks, ran-

Table 1: Overview of the datasets used in this paper.

Domain	Dataset	#Patterns	#Features	#Class
Biology	Carcinom	174	9182	11
	Lung	203	3312	5
Face	ORL	400	1024	40
	Yale	165	1024	15
Handwritten Digits	BA	1404	320	36
	Gisette	7000	5000	2
	MNIST	4000	748	10
	USPS	1000	256	10
Object	COIL20	1440	1024	20
	COIL100	7200	1024	100
Text	BASEHOCK	1993	1000	2
	RCV1	9625	1000	4
	TDT2	9394	1000	30

For datasets preprocessing and further details, please refer to [30].

domization based multi-layer neural networks and ensembles of RVFL). The compared methods are enumerated as follows:

1. ELM: extreme learning machine [32]; shallow RVFL without direct links and bias.
2. RVFL: standard shallow RVFL network [12].
3. SP-RVFL: sparse pre-trained RVFL [30]; state-of-the-art RVFL network.
4. HELM: hierarchical ELM [34], a multi-layer network based on ELM; has superior performance compared to other relevant deep learning methods such as Stacked Auto-Encoders (SAE) [37], Stacked Denoising Auto-Encoders (SDA) [38], Deep Belief Networks (DBN) [39], Deep Boltzmann Machines (DBM) [40].
5. dRVFL: deep RVFL proposed in this paper.
6. dRVFL(-O): dRVFL without direct links but with bias.
7. edRVFL: ensemble deep RVFL proposed in this paper.
8. edRVFL(-O): edRVFL without direct links but with bias.
9. dSP-RVFL: SP-RVFL based dRVFL
10. edSP-RVFL: SP-RVFL based edRVFL

### 4.3. Experimental Settings

To compare the different algorithms, we follow the experimental settings of [30]. The number of hidden neurons  $N$  is set to 100 [30]. For deep RVFL based methods, the same number of hidden neurons is used at each layer with the maximum number of hidden layers  $L$  set to 10 for each dataset. The HELM algorithm is implemented using the source code<sup>1</sup> available online. Meanwhile, the regularization parameter  $\lambda$  in each layer is set as  $(1/C)$  where  $C$  is tuned over the range  $2^x$ ,  $\{x = -6, -4, -2, \dots, 12\}$ . The widely used sigmoid function is used as the activation function in each type of RVFL network. The experimental results reported are obtained by averaging results from 10-fold cross-validation.

### 4.4. Performance Comparison and Analysis

In this section, we compare our proposed deep RVFL based frameworks against pertinent methods. Specifically, we first compare standard RVFL based methods in Section 4.4.1 and SP-RVFL based methods in Section 4.4.2.

#### 4.4.1. Comparison between standard RVFL based methods

The classification accuracies of each algorithm in each dataset is presented in Table 2. From the table, one can see that the ensemble deep RVFL (edRVFL) proposed in this paper has the best accuracy in 12 out of 13 datasets. The edRVFL has comparable performance to dRVFL in 3 datasets while it outperforms dRVFL in all other datasets. We follow the procedure of [14, 41] and use the Friedman rank of each classifier to assess its performance. Depending on the performance, each classifier is ranked, with the highest performing classifier ranked 1, the second highest ranked 2, and so on in each dataset. From the same table, one can see that edRVFL is the top ranked algorithm followed by dRVFL.

We also perform a statistical comparison of the algorithms using the Friedman test [42, 43]. The Friedman test compares the average ranks of the classifiers,  $R_j = \sum_i r_i^j$  where,  $r_i^j$  is the rank of the  $j$ -th of the  $m$  classifier on the  $i$ -th

---

<sup>1</sup>[http://www.ntu.edu.sg/home/egbhuang/elm\\_codes.html](http://www.ntu.edu.sg/home/egbhuang/elm_codes.html)

Table 2: Accuracy (%) of the standard RVFL based methods on all the datasets

Dataset	ELM[30]	RVFL[30]	HELM[34]	dRVFL <sup>†</sup>	dRVFL(-O) <sup>†</sup>	edRVFL(-O) <sup>†</sup>	edRVFL <sup>†</sup>
Carcinom	62.94±12.46	97.05±3.42	90.85±7.13	<b>98.86±2.41</b>	80.46±10.94	97.12±4.01	<b>98.86±2.41</b>
Lung	88.5±9.44	95.5±4.38	95.57±5.45	<b>97.05±4.19</b>	92.52±8.91	96.57±4.03	<b>97.05±4.19</b>
ORL	69±6.69	94.5±2.43	91.25±3.58	<b>99±1.75</b>	89±3.57	<b>99±1.75</b>	<b>99±1.75</b>
Yale	59.38±12.5	77.5±11.51	71.51±5.78	87.17±7.83	70.85±12.83	86.03±7.96	<b>88.58±6.92</b>
BA	52.21±5.21	57.42±4.27	<b>67.09±3.31</b>	65.33±4.11	55.13±2.17	59.97±3.48	66.11±3.34
Gisette	83.04±2.11	92.07±1.17	95.17±0.93	98.16±0.48	83.39±2.1	98.17±0.55	<b>98.21±0.55</b>
MNIST	79.13±1.74	88.11±1.15	87.6±1.35	88.12±1.32	84.07±1.71	86.52±2.06	<b>92.8±1.92</b>
USPS	91.1±1.6	91.9±2.57	91.35±2.2	93.3±2.18	91.45±3	92.3±1.93	<b>94.1±2.07</b>
COIL20	92.78±1.43	93.41±2.18	98.54±0.51	99.65±0.49	98.54±0.69	98.82±0.93	<b>99.86±0.29</b>
COIL100	64.07±2.09	85.16±1.4	75.28±1.07	90.06±0.91	81.65±0.72	87.51±0.77	<b>90.5±0.82</b>
BASEHOCK	73.37±3.63	88.19±3.18	96.39±1.22	98.04±1.07	83.59±2.54	97.34±1.11	<b>98.04±0.76</b>
RCV1	79.51±1.74	87.57±0.73	88.69±1.25	93.75±0.71	79.52±2.18	92.34±0.76	<b>93.86±0.69</b>
TDT2	61.32±2.21	81.92±0.83	85.6±0.68	96.51±0.6	80.31±1.28	94.73±0.43	<b>96.51±0.53</b>
<b>Mean Acc.</b>	73.56±4.83	86.95±3.01	87.29±2.65	92.69±2.15	82.34±4.09	91.26±2.29	<b>93.35±2.01</b>
<b>Avg. Friedman Rank</b>	7	4.54	4.35	2	5.73	3.08	<b>1.3</b>

The results for ELM and RVFL are directly copied from [30]. <sup>†</sup> are the methods introduced in this paper. The best results for each dataset is given in bold. Lower rank reflects better performance.

of  $M$  datasets. The null hypothesis is that the performance of all the classifiers are similar with their ranks  $R_j$  being equal.

Let  $M$  and  $m$  denote the number of datasets and classifiers respectively. When  $M$  and  $m$  are large enough, the Friedman statistic

$$\chi_F^2 = \frac{12M}{m(m+1)} \left[ \sum_j R_j^2 - \frac{m(m+1)^2}{4} \right], \quad (16)$$

is distributed according to  $\chi_F^2$  with  $m-1$  degrees of freedom under the null hypothesis. However, in this case,  $\chi_F^2$  is undesirably conservative. A better statistics is given by

$$F_F = \frac{(M-1)\chi_F^2}{M(m-1) - \chi_F^2}, \quad (17)$$

which is distributed according to  $F$ -distribution with  $(m-1)$  and  $(m-1)(M-1)$  degrees of freedom. If the null-hypothesis is rejected, the Nemenyi post-hoc test [44] can be used to check whether the performance of two among  $m$  classifiers is significantly different. The performance of two classifiers is significantly different if the corresponding average ranks of the classifiers differ by at least the critical



difference (CD)

$$CD = q_\alpha \sqrt{\frac{m(m+1)}{6M}}, \quad (18)$$

where critical values  $q_\alpha$  are based on the Studentized range statistic divided by  $\sqrt{2}$ .  $\alpha$  is the significance level and is equal to 0.05 in this paper.

Based on simple calculations we obtain,  $\chi_F^2 = 68.11$  and  $F_F = 82.64$ . With 7 classifiers and 13 datasets,  $F_F$  is distributed according to the  $F$ -distribution with  $7 - 1 = 6$  and  $(7 - 1)(13 - 1) = 72$  degrees of freedom. The critical value for  $F_{(6,72)}$  for  $\alpha = 0.05$  is 2.22, so we reject the null-hypothesis. Based on the Nemenyi test, the critical difference is  $CD = q_\alpha \sqrt{(m(m+1))/(6M)} = 2.948 * \sqrt{7 * 8 / (6 * 13)} \simeq 2.49$ . From Fig. 4, we can see that edRVFL is statistically significantly better than ELM, RVFL, HELM and dRVFL(-O) while dRVFL is statistically significantly better than ELM, RVFL, and dRVFL(-O). The difference of ranks between the randomized multi-layer networks, HELM and dRVFL is 2.35 (0.14 less than CD). The dRVFL has superior performance (around 5.4% times more accurate) in almost all the datasets compared to HELM except BA dataset. Some of the biggest improvements of dRVFL over HELM in average are in Carcinom (8.01%), ORL (7.75%), Yale (15.66%), COIL100 (14.7%), RCV1 (5.06%) and TDT2 (10.91%) datasets. This indicates the superior generalization ability (representational capability) of dRVFL over HELM.

In addition, we select 5 datasets from each domain (Lung from biology, ORL from face, USPS from digits, COIL20 from object and RCV1 from text), and report the experimental results for different number of hidden nodes in dRVFL and edRVFL in Fig. 5. The number of hidden nodes in each dataset is varied from 10 to 100 with a step-size of 10. One can see from Fig. 5, setting  $N = 100$  is appropriate for both dRVFL and edRVFL. Increasing the number of hidden nodes generally increases the generalization of the network until some point after which it becomes stable. Similarly, we also compare the training and testing times of dRVFL and edRVFL in these 5 datasets in Fig. 6. As can be seen

from the figure, the training times of both dRVFL and edRVFL increase with the increase in the number of hidden nodes while there is only a slight increase in the testing time for both cases.

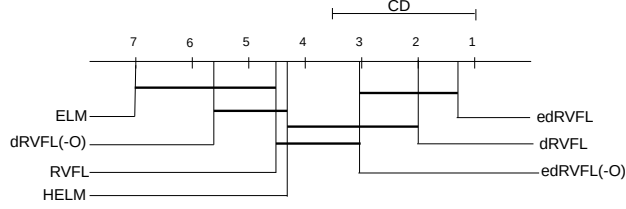


Figure 4: Statistical comparison of classifiers against each other based on Nemenyi test. Groups of classifiers that are not significantly different (at  $\alpha = 0.05$ ) are connected.

*Comparison between RVFL and dRVFL.* From Table 2, it can be observed that the deep learning framework proposed in this paper, dRVFL, is more accurate than its baseline (shallow) RVFL network by approx. 5.74%. Some of the biggest improvements are in Yale (9.67%), BA (7.91%), Gisette (6.09%), COIL20 (6.24%), BASEHOCK (9.85%), RCV1 (6.18%) and TDT2 (14.59%) datasets. We also compare dRVFL and RVFL with the same number of hidden nodes in each dataset in Fig. 7. The dRVFL network is on average 3.61% more accurate than shallow RVFL network with the same number of hidden nodes. This accentuates the benefits of representational learning in case of multi-layer (deep) networks wherein each hidden layer extracts meaningful feature representation from its input.

*Comparison of edRVFL with True Ensembles.* Here, we compare the implicit ensembles of dRVFL (edRVFL) with its true ensemble (TedRVFL) in terms of performance and training complexities. The true ensemble method, TedRVFL averages dRVFL methods trained independently as in [14]. The training and testing accuracies of edRVFL and TedRVFL is presented in Fig. 8. For edRVFL, the number of models corresponds to the number of hidden layers  $L$  while for TedRVFL, this corresponds to an ensemble of  $L$  dRVFL models with  $L$  hidden layers. As can be observed from the figure, the training accuracies of both

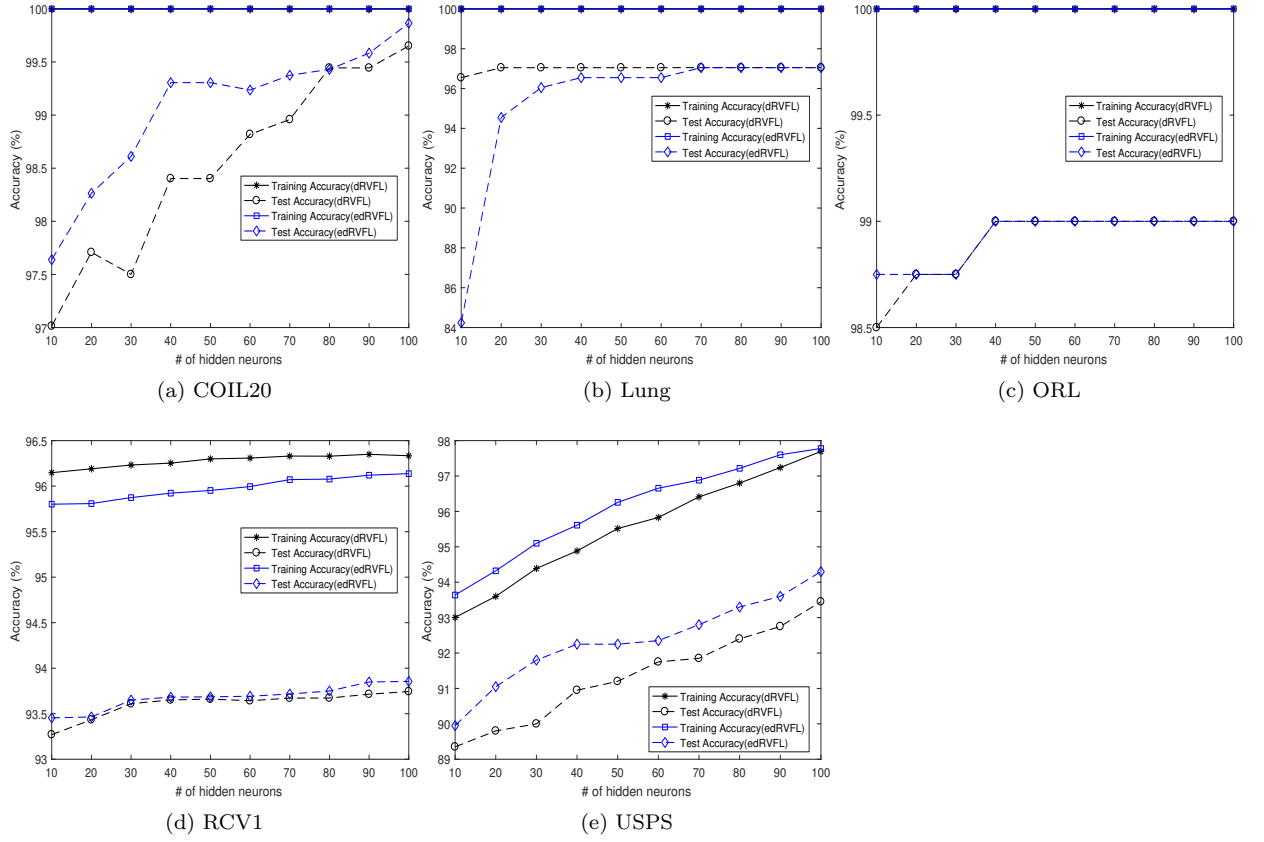


Figure 5: Comparison of dRVFL and edRVFL in terms of accuracy (%) w.r.t different number of hidden nodes.

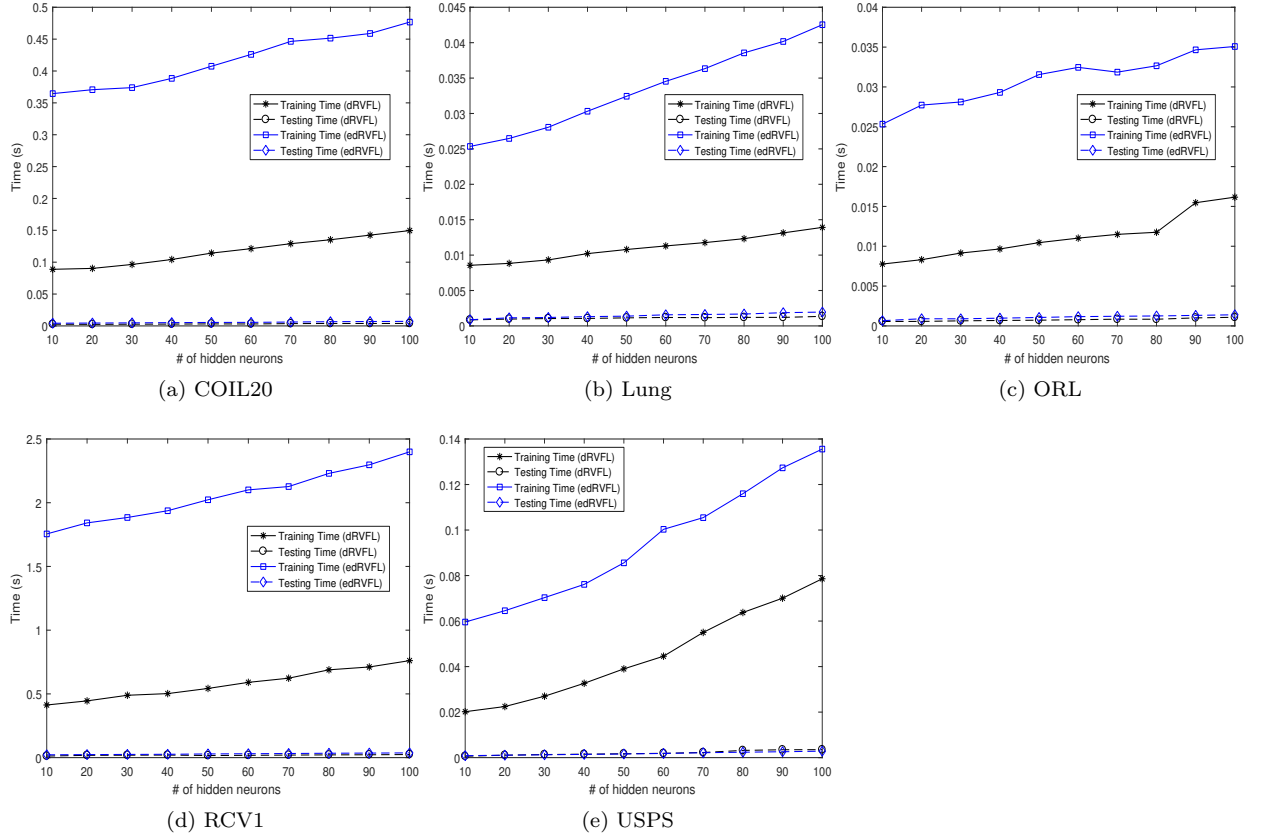


Figure 6: Training and testing times comparison of dRVFL and edRVFL w.r.t different number of hidden nodes. For the same number of hidden nodes, the training time of edRVFL is in average 3 times more than that of dRVFL.

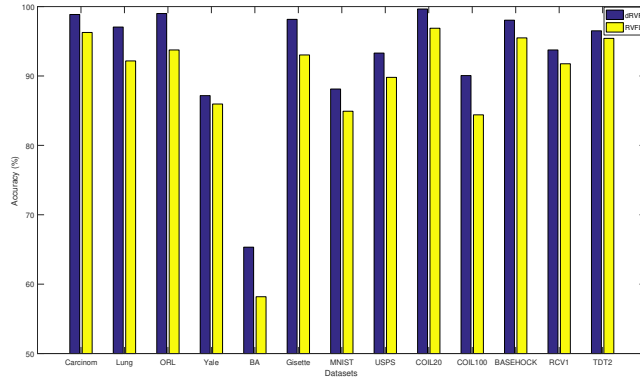


Figure 7: Comparison of dRVFL and RVFL in terms of accuracy (%) with the same number of hidden nodes.

edRVFL and TedRVFL increase with the increase in the number of models. However, this isn't the case for the test accuracies thus, requiring the best parameter search (in this case  $L$ ). However, from Fig. 8, one can see that with proper selection of  $L$ , the edRVFL can achieve either comparable or even better test accuracies compared to TedRVFL. Similarly, we also compare the training and testing times of edRVFL and TedRVFL in 5 datasets in Fig. 9. As can be seen from the figure, the training times of both edRVFL and TedRVFL increase with the increase in the number of models while there is only a slight increase in the testing times for both cases. As the number of models increases, the training time of TedRVFL increases sharply while that for edRVFL only increases slightly. In a nutshell, edRVFL has comparable or better performance than TedRVFL while requiring significantly less training time.

*Effect of direct links.* The significance of direct links in case of randomized shallow neural networks has been extensively articulated in the literature [16, 18, 20, 21, 22]. In this paper, we investigate the effect of direct links in the case of randomization based deep neural networks, specifically in the dRVFL and edRVFL methods introduced in this paper. The dRVFL(-O) and edRVFL(-O) are the deep learning methods equivalent to dRVFL and edRVFL respectively without direct links. The experimental results of dRVFL(-O) and edRVFL(-O) on the real-world classification datasets are presented in Table 2. The dRVFL(-O) dif-

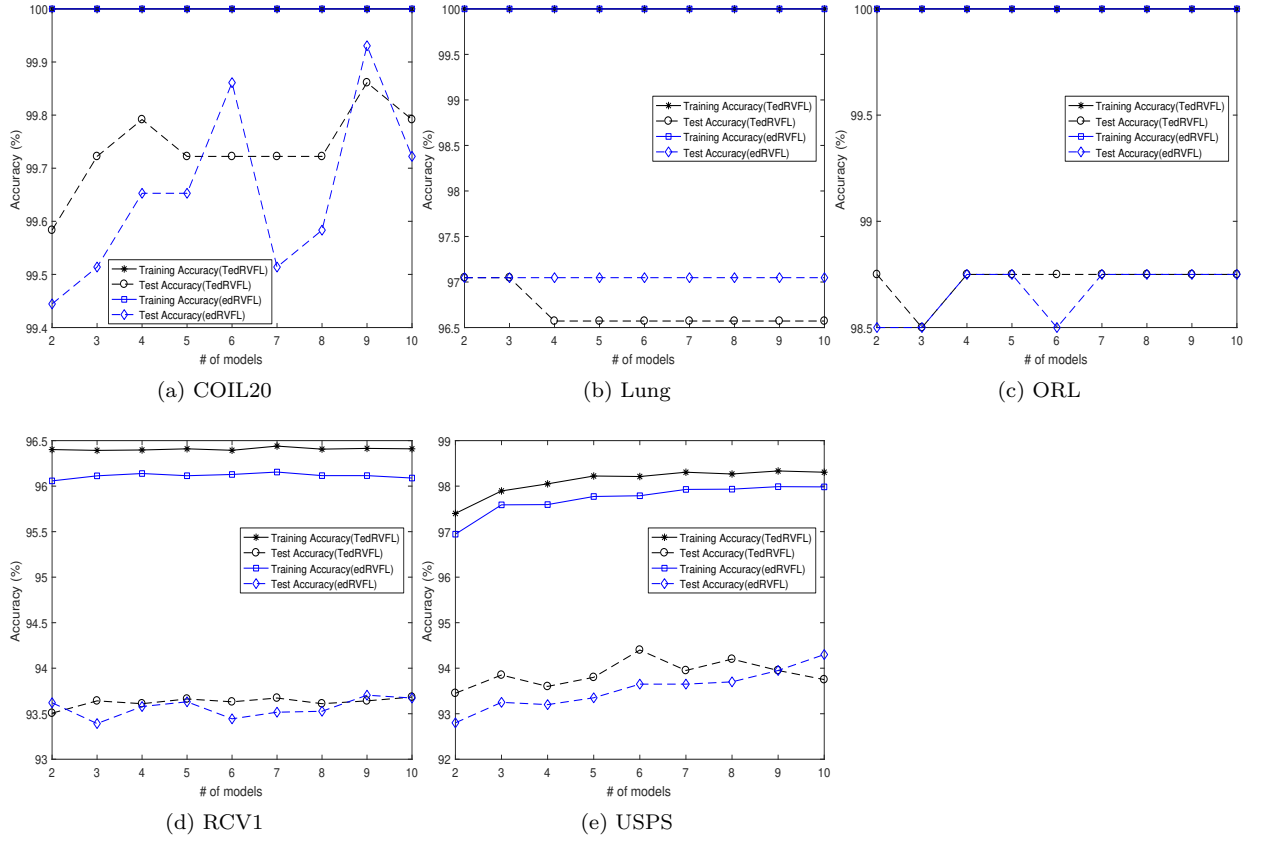


Figure 8: Comparison of edRVFL (implicit ensemble) and TedRVFL (true ensemble) in terms of accuracy (%).

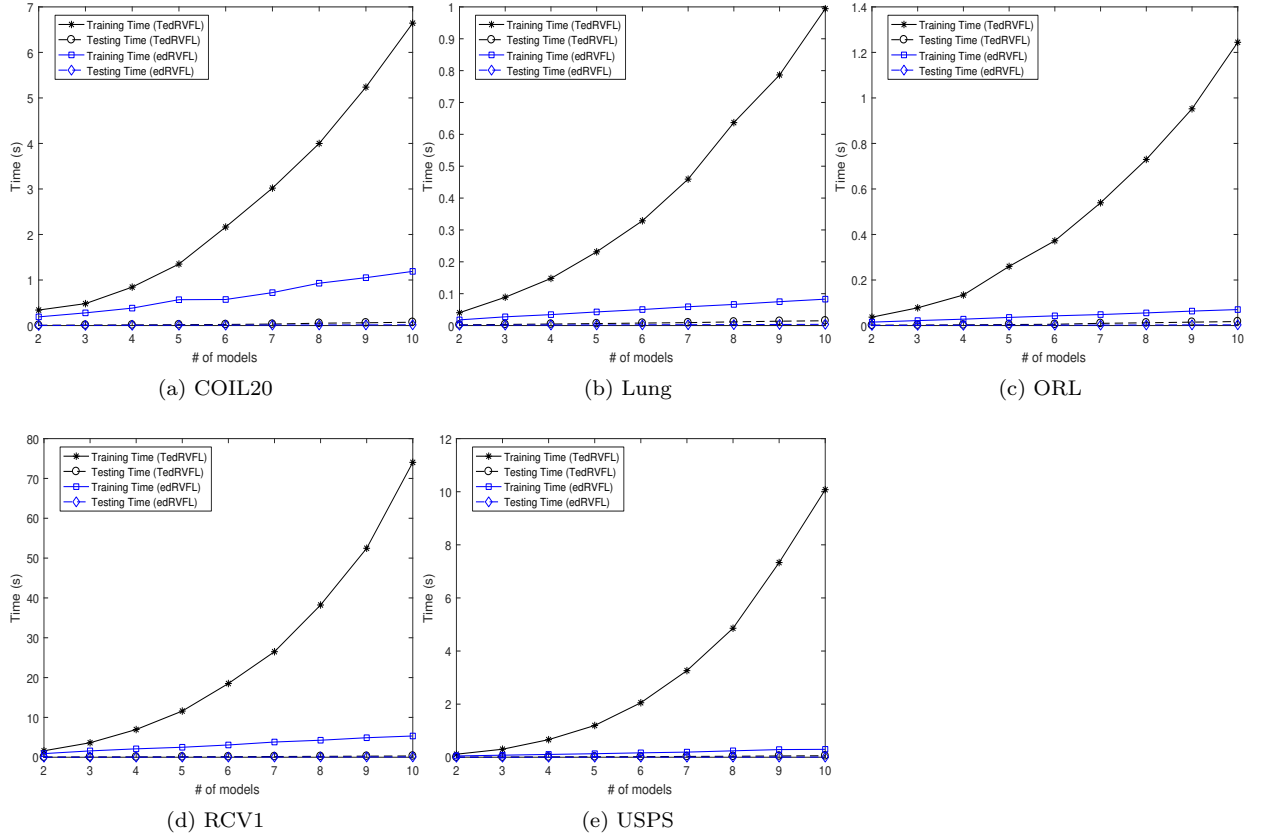


Figure 9: Training and testing times comparison of edRVFL (implicit ensemble) and TedRVFL (true ensemble).

fers from HELM [34] in that HELM uses only the last layer features extracted from the feature extractor part for final classification while dRVFL uses all the hidden layer features. From the table, we can see that the difference in accuracies between dRVFL and dRVFL(-O) is approx. 10.35% while that between edRVFL and edRVFL(-O) is approx. 2.09%. In dRVFL (refer to Fig. 2), in addition to the randomly generated hidden layer features, the input to the output nodes contains original features via the direct links. This enables the network to weigh higher the discriminative features while ignoring the redundant or less-important features. As in the case of randomized shallow neural networks, the direct links act as regularization for the randomization. Without direct links, the inputs to the output of dRVFL is simply the hidden layer features (generated randomly) resulting in a sub-optimal performance. To avoid such issue (sub-optimal performance) in case of edRVFL, the input to each hidden layer is a concatenation of original features via direct links and the hidden layer features from the preceding layer (except in case of the first hidden layer).

*Parameter Sensitivity Analysis.* In dRVFL and edRVFL, the number of hidden layers  $L$ , the number of hidden nodes  $N$  and the regularization parameter  $C$  need to be properly selected for each dataset. To further analyze the dRVFL and edRVFL methods, we conduct the sensitivity study for the parameters  $L$ ,  $H$  and  $C$  in case of two datasets COIL20 and USPS. The parameters  $H$  and  $C$  are common parameters for both shallow and deep RVFL based methods while only parameter  $L$  is relevant for deep learning methods. We show the experimental results in two different settings: varying  $L$  while keeping  $H$  fixed and varying  $H$  while keeping  $L$  fixed. Specifically, we employ a grid search strategy to vary these parameters. The parameter  $L$  is varied from 2:10 with a step-size of 1, the parameter  $H$  is varied from 10:100 with a step-size of 10. Similarly, the  $C$  parameter is varied within  $2^x$ ,  $\{x = -6, -4, -2, 0, 2, 4, 6, 8, 10, 12\}$ . As can be seen from Figs. 10 and 11, different combination of the parameters result in different performance. Therefore, it is necessary to determine the suitable values of the parameters  $L$ ,  $H$  and  $C$  for each dataset.



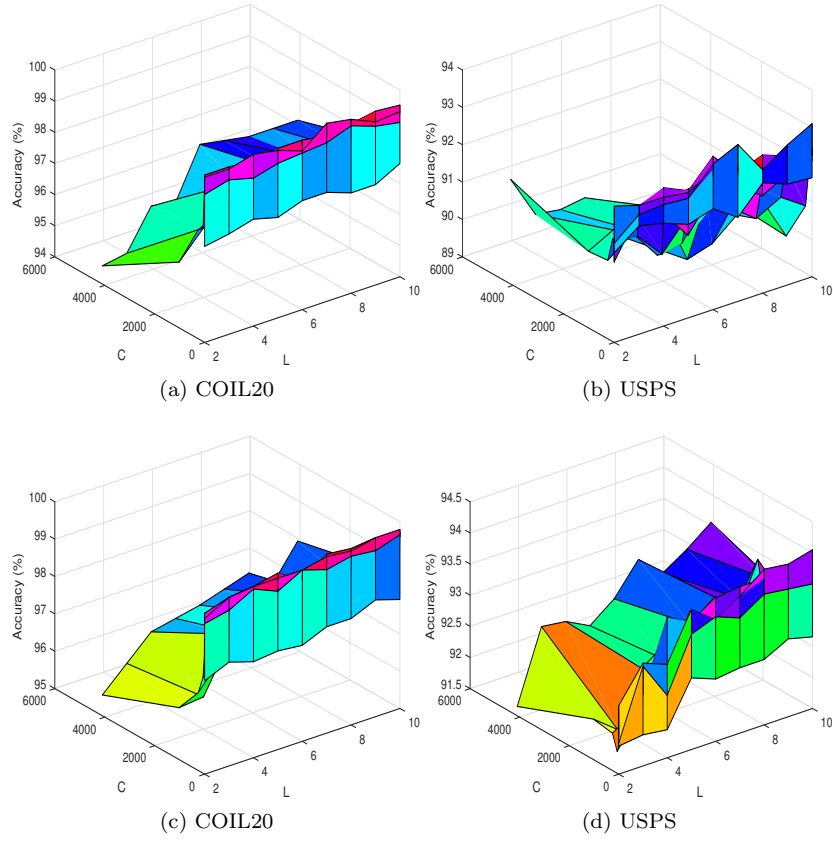


Figure 10: Performance variation of the proposed dRVFL (first row) and edRVFL(second row) methods in terms of accuracy (%) for fixed H. Different parameter combinations may result in different performance.

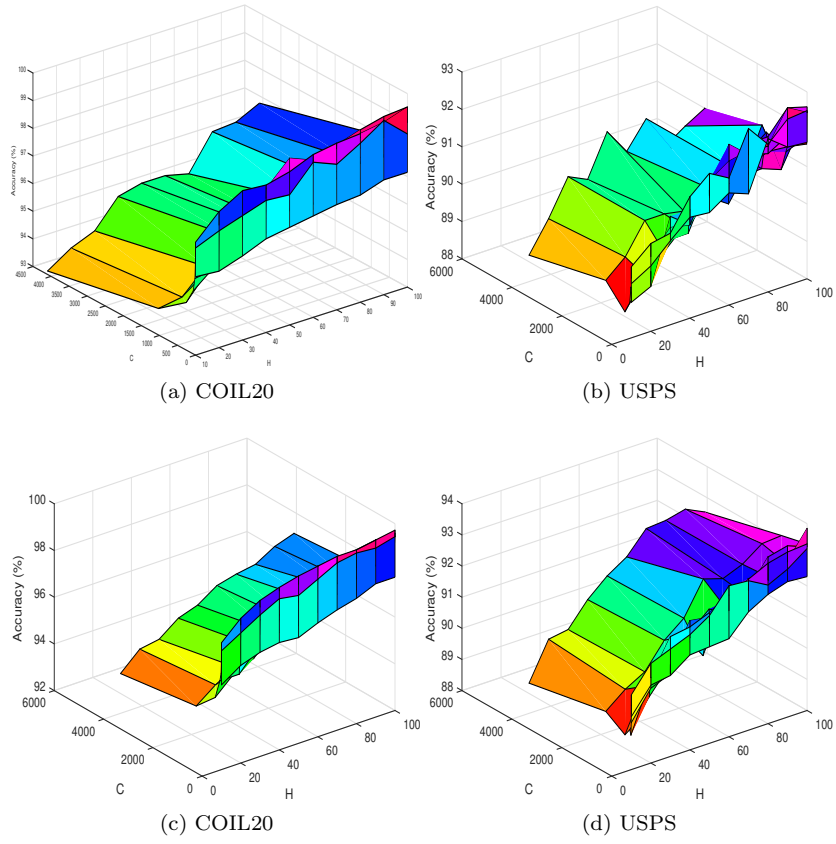


Figure 11: Performance variation of the proposed dRVFL (first row) and edRVFL(second row) methods in terms of accuracy (%) for fixed L. Different parameter combinations may result in different performance.

#### 4.4.2. Comparison between SP-RVFL based methods

The deep learning frameworks dRVFL and edRVFL proposed in Sections 3.1 and 3.2 respectively are generic and any RVFL network can be used as a base model with both dRVFL and edRVFL. Here, we use SP-RVFL [30], a recently proposed state-of-the-art RVFL network to create dRVFL and edRVFL networks. Specifically, we term the deep architecture using SP-RVFL as deep sparse pre-trained RVFL (dSP-RVFL) and its ensemble as edSP-RVFL. The SP-RVFL is described in Section 2.3. We run the experiments on all the datasets presented in Table 1 and report the experimental results of dSP-RVFL and edSP-RVFL in Table 3. From the table, one can see that our deep learning frameworks dSP-RVFL and edSP-RVFL have better performance compared to SP-RVFL. Also, edSP-RVFL has either same or better performance than dSP-RVFL.

Table 3: Comparison between SP-RVFL based methods in terms of accuracy (%)

Dataset	SP-RVFL[30]	dSP-RVFL	edSP-RVFL
Carcinom	97.64±3.04	<b>98.27±2.79</b>	<b>98.27±2.79</b>
Lung	96±4.38	<b>97.05±4.19</b>	<b>97.05±4.19</b>
ORL	97.25±2.3	<b>99.5±1.58</b>	<b>99.5±1.58</b>
Yale	86.87±9.88	<b>87.17±7.38</b>	<b>87.17±7.38</b>
BA	65.57±5.66	68.52±2.65	<b>73.08±2.67</b>
Gisette	98.13±1.45	<b>98.21±0.41</b>	<b>98.21±0.41</b>
MNIST	91.26±0.82	93.43±1.4	<b>95.02±1.5</b>
USPS	92.9±1.97	93.5±1.68	<b>95±1.6</b>
COIL20	98.75±1.39	98.96±0.67	<b>99.72±0.36</b>
COIL100	89.76±1.24	93.38±1.26	<b>94.26±0.86</b>
BASEHOCK	91.4±2.77	<b>97.79±0.98</b>	<b>97.79±0.98</b>
RCV1	93.38±0.61	93.9±0.61	<b>94.6±0.5</b>
TDT2	93.29±0.86	<b>96.17±0.63</b>	<b>96.17±0.63</b>
<b>Mean Acc.</b>	91.7±2.58	93.52±2.01	<b>94.29±1.95</b>
<b>Avg. Friedman Rank</b>	3	1.73	<b>1.26</b>

The results for SP-RVFL are directly copied from [30] for all the datasets except for the RCV1 dataset where we were unable to replicate the reported result (94.84±0.64) within the given level of variability. Thus, SP-RVFL’s performance in RCV1 is based on our implementation.

We also perform a statistical comparison of the algorithms using the Fried-

Table 4: Average Friedman rank based on classification accuracy of each method

Algorithm	Ranking
ELM	10
dRVFL(-O) <sup>†</sup>	8.73
RVFL	7.54
HELM	7.19
edRVFL(-O) <sup>†</sup>	5.61
SP-RVFL	5.46
dRVFL <sup>†</sup>	3.46
dSP-RVFL <sup>†</sup>	2.65
edRVFL <sup>†</sup>	2.3
edSP-RVFL <sup>†</sup>	<b>2.03</b>

<sup>†</sup> are the methods introduced in this paper. Lower rank reflects better performance.

man test as in Section 4.4.1. Based on simple calculations we obtain,  $\chi_F^2 = 20.54$  and  $F_F = 71.23$ . With 3 classifiers and 13 datasets,  $F_F$  is distributed according to the  $F$ -distribution with  $3 - 1 = 2$  and  $(3 - 1)(13 - 1) = 24$  degrees of freedom. The critical value for  $F_{(2,24)}$  for  $\alpha = 0.05$  is 3.4, so we reject the null-hypothesis. Based on the Nemenyi test, the critical difference is  $CD = q_\alpha \sqrt{(m(m+1))/(6M)} = 2.344 * \sqrt{3 * 4 / (6 * 13)} \simeq 0.92$ . From Table 3, one can see that both dSP-RVFL and edSP-RVFL are statistically significantly better than their baseline method, SP-RVFL.

#### 4.4.3. Overall Comparison

We present an overall comparison of the algorithms using the Friedman rank in Table 4. From the table, one can see that the deep learning frameworks introduced in this paper have the best ranks compared to other algorithms. Specifically, the ensemble deep learning frameworks (edSP-RVFL and edRVFL) obtain the top ranks followed by the single deep learning frameworks (dSP-RVFL and dRVFL).

## 5. Conclusion

In this paper, we first proposed a deep learning model (dRVFL) based on random vector functional link network. As in a standard RVFL network, the parameters of the hidden layers were randomly generated and kept fixed with the output weights computed analytically using a closed form solution. The dRVFL network while extracting rich feature representations through several hidden layers also acts as a weighting network thereby, providing a weight to features from all the hidden layers including the original features obtained via direct links. We then proposed an ensemble dRVFL, edRVFL, which combines ensemble learning with deep learning. Instead of training several models independently as in traditional ensembles, edRVFL can be obtained by training a deep network only once. The training cost of edRVFL is slightly greater than that of a single dRVFL network but significantly lower than that of the traditional ensembles. Both dRVFL and edRVFL are generic and any RVFL variant can be used with them. To demonstrate this generic nature, we developed sparse-pretrained RVFL (SP-RVFL) based deep RVFL networks (dSP-RVFL and edSP-RVFL). The SP-RVFL uses an sparse-autoencoder to learn the hidden layer parameters of RVFL as opposed to randomly generating them as in standard RVFL. Extensive experiments on several classification datasets showed that the our deep learning RVFL networks achieve better generalization compared to pertinent randomized neural networks. As our future work, we will consider other applications (datasets) related to but not limited to regression, time-series forecasting and other learning tasks such as semi-supervised learning, and incremental learning.

## References

## References

- [1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.

- [2] J. Schmidhuber, Deep learning in neural networks: An overview, *Neural Networks* 61 (2015) 85 – 117. doi:10.1016/j.neunet.2014.09.003.
- [3] P. N. Suganthan, On non-iterative learning algorithms with closed-form solution, *Applied Soft Computing* 70 (2018) 1078 – 1082. doi:10.1016/j.asoc.2018.07.013.
- [4] M. Olson, A. Wyner, R. Berk, Modern neural networks generalize on small data sets, in: *Advances in Neural Information Processing Systems* 31, 2018, pp. 3623–3632.
- [5] P. Guo, C. Chen, Y. Sun, An exact supervised learning for a three-layer supervised neural network, in: *Proceedings of the International Conference on neural Information Processing (ICONIP’95)*, 1995, pp. 1041–1044.
- [6] P. Guo, A vest of the pseudoinverse learning algorithm, in: *arXiv*, <https://arxiv.org/pdf/1805.07828>, 2018, pp. 1–5. doi:<https://arxiv.org/pdf/1805.07828>.
- [7] H. Berry, M. Quoy, Structure and dynamics of random recurrent neural networks, *Adaptive Behavior* 14 (2) (2006) 129–137.
- [8] W. F. Schmidt, M. A. Kraaijveld, R. P. Duin, Feedforward neural networks with random weights, in: *Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*, IEEE, 1992, pp. 1–4.
- [9] H. A. T. Braake, G. V. Straten, Random activation weight neural net (rawn) for fast non-iterative training, *Engineering Applications of Artificial Intelligence* 8 (1) (1995) 71 – 80. doi:10.1016/0952-1976(94)00056-S.
- [10] B. Widrow, A. Greenblatt, Y. Kim, D. Park, The no-prop algorithm: A new learning algorithm for multilayer neural networks, *Neural Networks* 37 (2013) 182 – 188, twenty-fifth Anniversary Commemorative Issue. doi:10.1016/j.neunet.2012.09.020.

- [11] H. White, Chapter 9 approximate nonlinear forecasting methods, Vol. 1 of Handbook of Economic Forecasting, Elsevier, 2006, pp. 459 – 512. doi: 10.1016/S1574-0706(05)01009-8.
- [12] Y. H. Pao, Y. Takefuji, Functional-link net computing: theory, system architecture, and functionalities, IEEE Computer 25 (5) (1992) 76–79. doi: 10.1109/2.144401.
- [13] L. Zhang, P. N. Suganthan, Visual tracking with convolutional random vector functional link network, IEEE Transactions on Cybernetics 47 (10) (2017) 3243–3253. doi:10.1109/TCYB.2016.2588526.
- [14] L. Zhang, P. N. Suganthan, Benchmarking ensemble classifiers with novel co-trained kernel ridge regression and random vector functional link ensembles [research frontier], IEEE Computational Intelligence Magazine 12 (4) (2017) 61–72. doi:10.1109/MCI.2017.2742867.
- [15] R. Katuwal, P. Suganthan, L. Zhang, An ensemble of decision trees with random vector functional link networks for multi-class classification, Applied Soft Computing 70 (2018) 1146 – 1153. doi:10.1016/j.asoc.2017.09.020.
- [16] N. Vukovi, M. Petrovi, Z. Miljkovi, A comprehensive experimental evaluation of orthogonal polynomial expanded random vector functional link neural networks for regression, Applied Soft Computing 70 (2018) 1083 – 1096. doi:10.1016/j.asoc.2017.10.010.
- [17] L. Tang, Y. Wu, L. Yu, A non-iterative decomposition-ensemble learning paradigm using rvfl network for crude oil price forecasting, Applied Soft Computing 70 (2018) 1097 – 1108. doi:10.1016/j.asoc.2017.02.013.
- [18] Y. Dash, S. K. Mishra, S. Sahany, B. K. Panigrahi, Indian summer monsoon rainfall prediction: A comparison of iterative and non-iterative approaches, Applied Soft Computing 70 (2018) 1122 – 1134. doi:10.1016/j.asoc.2017.08.055.

- [19] 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. doi:10.1016/0925-2312(94)90053-1.
- [20] R. Katuwal, P. N. Suganthan, Enhancing multi-class classification of random forest using random vector functional neural network and oblique decision surfaces, in: 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–8. doi:10.1109/IJCNN.2018.8489738.
- [21] L. Zhang, P. N. Suganthan, A comprehensive evaluation of random vector functional link networks, *Information Sciences* 367-368 (2016) 1094 – 1105. doi:10.1016/j.ins.2015.09.025.
- [22] Y. Ren, P. N. Suganthan, N. Srikanth, G. Amaratunga, Random vector functional link network for short-term electricity load demand forecasting, *Information Sciences* 367-368 (2016) 1078 – 1093. doi:10.1016/j.ins.2015.11.039.
- [23] M. J. Kearns, U. V. Vazirani, U. Vazirani, An introduction to computational learning theory, MIT press, 1994.
- [24] A. Veit, M. J. Wilber, S. Belongie, Residual networks behave like ensembles of relatively shallow networks, in: *Advances in Neural Information Processing Systems*, 2016, pp. 550–558.
- [25] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, K. Q. Weinberger, Snapshot ensembles: Train 1, get m for free, *arXiv preprint arXiv:1704.00109*.
- [26] Y. Ren, L. Zhang, P. N. Suganthan, Ensemble classification and regression-recent developments, applications and future directions [review article], *IEEE Computational Intelligence Magazine* 11 (1) (2016) 41–53. doi:10.1109/MCI.2015.2471235.
- [27] C. Gallicchio, A. Micheli, L. Pedrelli, Design of deep echo state networks, *Neural Networks* 108 (2018) 33 – 47. doi:10.1016/j.neunet.2018.08.002.



- [28] P. A. Henrquez, G. A. Ruz, Twitter sentiment classification based on deep random vector functional link, in: 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–6. doi:10.1109/IJCNN.2018.8489703.
- [29] Ö. F. Ertuğrul, A novel type of activation function in artificial neural networks: Trained activation function, Neural Networks 99 (2018) 148 – 157. doi:10.1016/j.neunet.2018.01.007.
- [30] Y. Zhang, J. Wu, Z. Cai, B. Du, P. S. Yu, An unsupervised parameter learning model for RVFL neural network, Neural Networks 112 (2019) 85 – 97. doi:10.1016/j.neunet.2019.01.007.
- [31] B. K. Verma, J. J. Mulawka, A modified backpropagation algorithm, in: Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94), Vol. 2, 1994, pp. 840–844 vol.2. doi:10.1109/ICNN.1994.374289.
- [32] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: Theory and applications, Neurocomputing 70 (1) (2006) 489 – 501, neural Networks. doi:10.1016/j.neucom.2005.12.126.
- [33] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, Siam Journal on Imaging Sciences 2 (1) (2009) 183–202.
- [34] J. Tang, C. Deng, G. Huang, Extreme learning machine for multilayer perceptron, IEEE Transactions on Neural Networks and Learning Systems 27 (4) (2016) 809–821. doi:10.1109/TNNLS.2015.2424995.
- [35] C. Gallicchio, A. Micheli, L. Pedrelli, Deep reservoir computing: A critical experimental analysis, Neurocomputing 268 (2017) 87 – 99, advances in artificial neural networks, machine learning and computational intelligence. doi:10.1016/j.neucom.2016.12.089.

- [36] Y. Zhang, J. Duchi, M. Wainwright, Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates, *The Journal of Machine Learning Research* 16 (1) (2015) 3299–3340.
- [37] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, *science* 313 (5786) (2006) 504–507.
- [38] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: *Proceedings of the 25th International Conference on Machine Learning, ICML '08*, ACM, New York, NY, USA, 2008, pp. 1096–1103. doi:10.1145/1390156.1390294.
- [39] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, *Neural Computation* 18 (7) (2006) 1527–1554, pMID: 16764513. doi:10.1162/neco.2006.18.7.1527.
- [40] R. Salakhutdinov, G. Hinton, Deep boltzmann machines, in: *Artificial intelligence and statistics*, 2009, pp. 448–455.
- [41] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, *Journal of Machine Learning Research* 15 (2014) 3133–3181.
- [42] J. Demšar, Statistical comparisons of classifiers over multiple data sets, *Journal of Machine learning research* 7 (Jan) (2006) 1–30.
- [43] K. Rakesh, P. N. Suganthan, An ensemble of kernel ridge regression for multi-class classification, *Procedia Computer Science* 108 (2017) 375 – 383, international Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland. doi:10.1016/j.procs.2017.05.109.
- [44] P. Nemenyi, Distribution-free multiple comparisons, in: *Biometrics*, Vol. 18, International Biometric SOC 1441 I ST, NW, SUITE 700 Washington, DC, 20005-2210, 1962, p. 263.