Deep Random Vector Functional Link Network for Handwritten Character Recognition

Hubert Cecotti
School of Computing and Intelligent Systems, Ulster University
Londonderry, Northern Ireland, UK
email: h.cecotti@ulster.ac.uk

***Abstract*—The field of artificial neural networks has a long history of several decades, where the theoretical contributions have progressed with advances in terms of power and memory in present day computers. Some old methods are now rebranded or represented, taking advantage of the power of present day computers. More particularly, we consider the current trend of Random Vector Functional Link Networks, which suggests that the architecture of a system and the learning algorithm should be properly decoupled. In this paper, we evaluate the performance of multi-layers Random Vector Functional Link Network (RVFL)/ extreme machine learning (EML) on four databases of handwritten characters. Particularly, we evaluate the impact of the architecture (number of neurons per hidden layer), and the robustness of the distribution of the results across different runs. By combining the classifier outputs from different runs, we show that such a maximum combination rule provides an accuracy of 95.97% for Arabic digits, 98.03% for Bangla, 98.64% for Devnagari, and 96.30% for Oriya digits. The results confirm that increasing the size of the hidden layers has a significant impact on the accuracy, and allows to reach state-of-the-art performance; however the performance reaches a plateau after a certain size of the hidden layers.**

I. Introduction

The evolution of machine learning and pattern recognition techniques over the years has taken advantage of both theoretical and hardware progresses. More particularly, classifiers based on artificial neural networks such as multi-layer perceptrons (MLPs) have exploited graphics processing unit (GPU) [1], highlighting the benefit of parallel and distributed systems in applications that require large databases for training models. The different improvements given by theoretical and hardware progresses invite to perpetually reconsider the methods to use for classification. In fact, the large increase of both power and memory can change how methods are perceived over time. While some methods in the 80s could only be used with small databases due to memory and computational issues, the current generation of personal computers, with powerful graphic cards and a large amount of memory, largely driven by the video games industry, invites us to retry old techniques. In addition, the exponential increase of classification problems, with the large amount of available data (Big data), forces researchers to take into account both the time for training and testing, but also the tradeoff between the accuracy and time to achieve this accuracy. It is particularly relevant for applications where it is not possible to dedicate a large amount of time for the optimization of the architecture and the choice of the hyper-parameters. Artificial neural networks have provided state-of-the-art performance since their introduction [2], [3], and they have known different cycles of fame, from the introduction of the perceptron to deep learning techniques, without a priori predefined architecture [4].

The use of GPUs has significantly increased the interest of some methods such as convolutional neural network. A new challenge in machine learning techniques applied on large databases, such as natural images, is to find the best trade-off between the efficiency of the method, the possibility to implement the method on computer clusters and/or GPU, and the possibility to update the system with new data in an incremental way. In fact, as the number of classification problems can change rapidly, it is not possible to dedicate several years of research to develop and tune a system as it was the case for handwritten character recognition and speech recognition, where problems are relevant for the whole population and can be used in many applications. For classification problems that are dedicated to only a particular database, particular classes of images, methods that do not require many labeled trials and hyper-parameters to tune are more suitable. A key issue for classification is the choice of the input features. Some recent and old works, from Random Vector Functional-Link (RVFL) [5], [6] to convolutional neural network [7] and extreme machine learning (EML) [8], suggest that using non-linear representations of the inputs with random weights can provide an efficient mapping of the data. In [7], random filters in a one layer convolutional neural network could impact the performance of only 1.2% compared to unsupervised pretraining and discriminative finetuning of the filters. In [9], it is shown that a part of the performance of certain state-of-the-art techniques can be directly attributed to the architecture alone and not the learning technique. These results show that classifiers based on random weights for the hidden layers, i.e. no learning before the last hidden layer, provides good results and that they must be provided to clearly show the impact of a learning approach.

Single handwritten character recognition is almost a resolved problem thanks to several decades of research in classification and feature extraction algorithms [10], [11], [12], [13]. However, the accuracy remains below 100%, and methods are typically customized for a particular script or database, from the pre-processing steps to the classification. Moreover, the high accuracy is only available for certain

978-1-5090-0620-5/16/$31.00

scripts, *e.g.,* Latin script (MNIST results). In addition, there exist always documents with noisy characters, which cannot be recognized with commercial optical character recognition (OCR) technologies [14].

In this paper, we propose to evaluate the performance of four databases of handwritten characters (one with Arabic digits, three with Indian scripts) using RVFL/EML models with different architectures. While random weights can provide good performance, the study aims at evaluating the impact of architecture on the variability that may exist with different runs. Furthermore, as RVFL architectures are computationally efficient, the output results can be combined through a multi-classifier scheme. The remainder of the paper is organized as follows. First, the Random-vector functional links/Extreme Machine Learning frameworks are described in Section II. The four databases are presented in Section III. Then, the experimental results are given in Section IV and discussed in Section VI.

II. Methods

A. Random-vector functional links

Random Vector Functional-Link (RVFL) networks can be seen as artificial feedforward neural networks with only a single hidden layer that can be used for both classification and regression [15], [5], [6]. It corresponds to a linear combination of a number of non-linear representations (e.g. using a sigmoid function) of the input data. A key feature of this type of technique is the way how the parameters are assigned. More particularly, the input weights and biases are set randomly and they do not change, i.e. they are fixed. While this step is simple and seems to not be efficient, RVFLs have the capability of universal approximation if the dimension of the input representation is large enough [16]. The parameters of RVFLs can be obtained with linear regression methods using only matrix inversions and multiplications. These types of operations are particularly well suited for distributed learning [17]. RVFL networks can be estimated with a lead-square approach for learning the weights.

Let us consider first a regression problem with one-dimensional scalar outputs $y \in \mathbb{R}$. A Functional Link Artificial Neural Network (FLANN) with a single output neuron can be defined as a weighted sum of B non-linear transformations of the input $\mathbf{x}$ [5]:

$$f(\mathbf{x}) = \sum_{m=1}^{B} \beta_m h_m(\mathbf{x}; \mathbf{w}_m) = \beta^T h(\mathbf{x}; \mathbf{w}_1, \ldots, \mathbf{w}_B) \quad (1)$$

where the mth transformation is obtained with the parameters $\mathbf{w}_m$, and $\mathbf{x} \in \mathbb{R}^d$. Each function h_m is a functional link, mapping the input data, with the set of parameters, to a real number.

Like many artificial neural networks such as the multilayer perceptron, the sigmoid function is typically used:

$$h_m(\mathbf{x}; \mathbf{w}_m; b) = \frac{1}{1 + \exp^{\sigma}} \quad (2)$$

where $\sigma = -\mathbf{w}^T\mathbf{x} + b$. The set of parameters $\mathbf{w}_m$, $1 \leq m \leq B$, is chosen before the learning process and without any prior assumption about the data. In a RVFL, the parameters are set randomly, in relation to a predefined probability distribution [6]. The capability of universal approximation of the network is guaranteed if there exists a sufficiently large number of representations of the inputs. After the estimation of the set of parameters, the weights $\boldsymbol{\beta}$ must be estimated.

We consider a dataset $\mathcal{X}_{Train} = \{(\mathbf{x}_i, y_i)\}$ of N couples that contain an example $\mathbf{x}_i$ and the expected output y_i, $1 \leq i \leq N$. We denote by $\mathbf{H}$ the following matrix that contains the B representations of the N examples.

$$H = \begin{pmatrix} h_1(\mathbf{x}_1) & \ldots & h_B(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_N) & \ldots & h_B(\mathbf{x}_N) \end{pmatrix} \quad (3)$$

where each function h_m includes also the corresponding set of parameters $\mathbf{w}_m$. The estimation of $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_B]^T$ can be obtained through a regularized least-square problem:

$$\boldsymbol{\beta} = \underset{\beta \in \mathbb{R}^B}{\arg\min} \frac{1}{2} \|\mathbf{H}\boldsymbol{\beta} - \mathbf{Y}\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2 \quad (4)$$

where the vector $\mathbf{Y} = [y_1, \ldots, y_N]^T$ is the ground truth of $\mathcal{X}_{Train}$. As the problem is convex, an estimation of $\hat{\boldsymbol{\beta}}$ can be directly obtained by:

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I}\right)^{-1} \mathbf{H}^T\mathbf{Y} \quad (5)$$

where I is the identity matrix of size $B \times B$. It is worth noting that the computational complexity of RVFL is mainly driven by the $B \times B$ matrix inversion.

For multiclass classification with M classes, $M \geq 2$, the groundtruth $\mathbf{Y}$ is a matrix of size $N \times M$. $\mathbf{Y}(i, j) = 1$ if $\mathbf{x}_i$ belongs to the class j, $1 \leq j \leq M$, $\mathbf{Y}(i, j) = 0$ otherwise.

(6)

B. Multi-layer RVFL/EML

Different variations of RVFL networks have had a recent success in different popular classification problems under the name of extreme machine learning (EML) [18], and inspired from other techniques [19], [20]. This name, in addition to the efficiency of the proposed methods, has been a key factor for the reuse of RVFL networks [8], [21], [22]. More particularly, it has been shown that EML can be successful within deep learning architectures (ML-EML) [23], [24]. The learning approach performs layer-by-layer unsupervised learning by using EML auto-encoder (EML-AE), which represents features based on singular values. With EML-AE, the output $\mathbf{Y}$ is similar to the input $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$. In addition, the decoder, i.e. the function that maps the input representation $h_1(x), \ldots, h_B(x)$ of the input x to itself, corresponds to the parameters $\hat{\boldsymbol{\beta}}$ that are estimated. The random weights are constrained to be orthogonal [23]. To create the coder afterward, $\hat{\boldsymbol{\beta}}^T$ is used to map x to the representation that was obtained. Like other deep network architectures, ML-EML stacks on top of EML-AE to create

a multilayer neural network. ML-EML is a greedy approach and does not require finetuning after training the last layer. In RVFL network with a single hidden layer, we denote by $\hat{\boldsymbol{\beta}}^{\mathbf{1}}$ the estimation of the weights for the first hidden layer. Similarly, for ML-EML of L hidden-layers (or $L+1$ layers), we denote by $\hat{\boldsymbol{\beta}}^{\boldsymbol{l}}$ the estimation of the weights for the layer l, $1 \leq l \leq L$. An EML-AE is used for each layer l, and the extracted weights of an EML-AE l are used to generate the inputs of the EML-AE at layer $l+1$. The system for two hidden layers is depicted in Figure 1

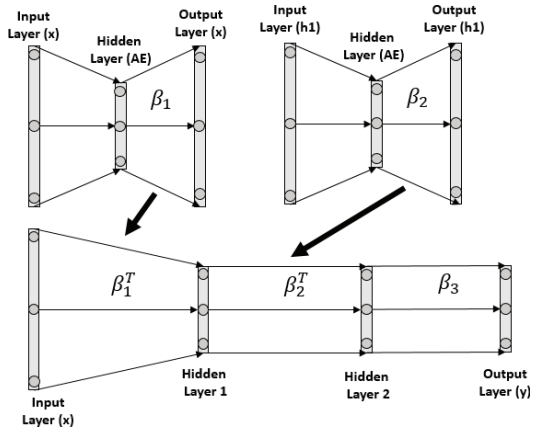

Fig. 1. Example of a deep RVFL/EML with two hidden layers.

C. Multi-classifier system

Because the generation of the RVFL/EML classifiers is based on random weights, it may be possible to extract some complementarities between classifiers. We propose to evaluate the performance that can be obtained by combining the decision scores coming from different runs. Three simple functions are chosen: the mean (the average score from each run is used before selecting the maximum), the median (the median score across runs for each class), and the maximum rules (the maximum score in each class and each run).

D. Performance evaluation

In the subsequent sections, we evaluate two types of architectures. The first type includes a single hidden layer (the number of neurons is set to: 100, 500, 1000, 10000, or 15000). The second type includes two hidden layers, the number of neurons in the first hidden layer is set to: 200, 400, 600, or 800, and the number of neurons in the second hidden layer is set to: 4000, 8000, 12000, and 16000. For each architecture, 10 runs are evaluated.

III. Databases

Four databases of handwritten digits have been chosen for the analysis of the performance. The first database CVL contain images of ten Arabic numerals (Latin script). The next three databases contain images of digits in three Indian scripts: Bangla, Devnagari, and Oriya. Samples of digits are presented in Fig. 2. To facilitate the comparisons with other methods, the images were normalized with the same normalization technique. The images were preprocessed as in the original images in the MNIST database. Because some databases (e.g. Indian digits) have very noisy images and/or images in color, images were first binarized with the Otsu method at their original size [25], then their size were normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain 8 bit gray levels due to the bicubic interpolation for resizing the images. All the images were centered in a 28x28 pixel box field by computing the center of mass of the pixels, and translating the gravity center of the image to the center of the 28x28 field. The number of classes, the total number of images in the database, and the number of images per class, for both training and the test are given in Table I.

TABLE I
Characteristics of the handwritten databases (digits, 10 classes).

Database	CVL	Devnagari	Oriya	Bangla
		Training		
# samples	14000	18783	4970	19392
# per class	1400	1878 ± 15	497 ± 3	360
size (x)	47 ± 12	65 ± 16	73 ± 25	58 ± 16
size (y)	104 ± 30	62 ± 19	73 ± 26	54 ± 16
		Test		
# samples	21780	3763	1000	4000
# per class	2178	376 ± 3	100	400
size (x)	50 ± 12	66 ± 17	75 ± 25	59 ± 17
size (y)	106 ± 31	62 ± 20	74 ± 26	54 ± 18

(a) CVL

(b) Bangla

(c) Devnagari

(d) Oriya

Fig. 2. Representative handwritten digits for the different databases (from zero to nine).

The Computer Vision Lab of the Vienna University of Technology, Austria (CVL) database was used during the International Conference on Document Analysis and Recognition (ICDAR) 2013 Competition on Handwritten Digit Recognition [26]. The images in the CVL database are

in RGB color, not size-normalized, and in original size with a resolution of 300 dpi. For this competition, the best methods were based on Finite Impulse Response Multilayer Perceptron (Fir-MLP) partially and fully connected with four layers, and by including affine deformations of the input patterns [27]. In this neural network, the static weights (synapses) were replaced with finite impulse response filters. With one Fir-MLP and an ensemble of four Fir-MLPs, the error rates were 3.28% and 2.26%, respectively.

The databases of Indian digits were created at the Indian Statistical Institute, Kolkata, India [28], [29], [30]. Bangla is the fourth most popular script in the world; it is used by more than 200 million people [31], [32]. In the Bangla database, class-specific feature polynomial classifier with input features [33] based on 8 gradient direction histogram with 5x5 sampling provided an accuracy of 99.40% [34]. The second database corresponds to Devnagari digits, which is part of the Brahmic family of scripts of India, Nepal, Tibet, and South-East Asia [35]. The third database contains Oriya digits [36]. The Oriya script is one of the many descendants of the Brahmi script of ancient India. In [36], Bhowmik et al. obtain an accuracy of 90.50% by using Hidden Markov Models.

IV. Results

The performance for the different architectures is presented in Tables II, III, IV, and V. The tables indicate the number of neurons for the first and second hidden layer (if there is only a single hidden layer, the value for the second hidden layer is 0), the time (in seconds) for training (T_{train}) and testing (T_{test}), the mean, the standard deviation, the maximum, and minimum accuracy across 10 repetitions (in %).

For the CVL database, the best performance, 95.85%, is obtained with 400 and 16000 neurons for the first (L_1) and second (L_2) hidden layers, respectively. For the Bangla database, it is 97.80% for (600,16000); 94.54% with (400,12000) for Devnagari, and 96.13% with (200,16000) for Oriya. The performance that is obtained for the four databases is relatively similar with two hidden layers. However, with only a single hidden layer, the number of transformations has a significant impact on the accuracy. With more than 10000 transformations for the hidden layer, while there is slight improvement in the accuracy, the performance reaches a plateau.

The performance for the different architectures, by using the mean, median, and max rule are presented in Table VI. Pairwise comparisons between the mean accuracy across the different runs and the three combination rules with a Wilcoxon signed rank have been evaluated. A significant improvement is achieved by using the mean, the median, or the max rule. It shows that it is possible to exploit complementarities existing between different runs. Table VII presents the comparison with other methods, k-nearest neighbor (k-nn), using $k = 3$ and the Euclidean distance, and the image distortion model distance (IDMD) using pixels as inputs [37], [38], [39]. The accuracy with the multi-layer RVFL/EML is relatively similar, while the processing time is significantly reduced for both training and the test.

TABLE II
Performance with different architectures (CVL).

	Time		Accuracy		
(L_1, L_2)	T_{train}	T_{test}	Mean±SD	Max	Min
(100,0)	1.85	0.54	75.93 ± 0.54	76.62	74.98
(500,0)	3.04	0.75	83.19 ± 0.23	83.53	82.88
(1000,0)	5.13	1.00	87.49 ± 0.21	87.80	87.04
(10000,0)	9.85	1.81	95.46 ± 0.07	95.59	95.38
(15000,0)	31.41	3.14	95.69 ± 0.10	95.85	95.52
(200,4000)	8.98	1.37	95.39 ± 0.10	95.59	95.28
(400,4000)	9.85	1.81	95.46 ± 0.07	95.59	95.38
(600,4000)	11.52	2.34	95.49 ± 0.07	95.60	95.35
(800,4000)	13.57	2.91	95.41 ± 0.15	95.64	95.19
(200,8000)	30.80	2.38	95.62 ± 0.08	95.73	95.50
(400,8000)	31.41	3.14	95.69 ± 0.10	95.85	95.52
(600,8000)	33.85	4.06	95.54 ± 0.06	95.68	95.48
(800,8000)	37.35	4.89	95.43 ± 0.08	95.57	95.35
(200,12000)	69.91	3.90	95.62 ± 0.11	95.75	95.44
(400,12000)	71.35	4.76	95.83 ± 0.05	95.91	95.76
(600,12000)	73.92	5.77	95.68 ± 0.05	95.78	95.60
(800,12000)	77.19	7.00	95.55 ± 0.07	95.69	95.44
(200,16000)	129.62	4.91	95.52 ± 0.08	95.62	95.37
(400,16000)	127.63	5.73	95.85 ± 0.07	95.93	95.67
(600,16000)	130.75	7.36	95.76 ± 0.05	95.85	95.70
(800,16000)	136.10	8.79	95.68 ± 0.06	95.81	95.62

TABLE III
Performance with different architectures (Bangla).

	Time		Accuracy		
(L_1, L_2)	T_{train}	T_{test}	Mean±SD	Max	Min
(100,0)	2.49	0.10	80.30 ± 0.41	81.28	79.85
(500,0)	4.01	0.15	86.54 ± 0.26	87.13	86.23
(1000,0)	6.64	0.19	90.17 ± 0.31	90.53	89.60
(10000,0)	13.14	0.37	97.23 ± 0.14	97.43	97.00
(15000,0)	42.47	0.63	97.64 ± 0.09	97.78	97.50
(200,4000)	11.76	0.26	97.08 ± 0.13	97.38	96.95
(400,4000)	13.14	0.37	97.23 ± 0.14	97.43	97.00
(600,4000)	15.10	0.45	97.31 ± 0.16	97.63	97.10
(800,4000)	17.45	0.56	97.24 ± 0.16	97.50	97.00
(200,8000)	40.33	0.46	97.33 ± 0.13	97.53	97.20
(400,8000)	42.47	0.63	97.64 ± 0.09	97.78	97.50
(600,8000)	43.56	0.76	97.67 ± 0.11	97.83	97.48
(800,8000)	47.52	0.93	97.59 ± 0.14	97.80	97.40
(200,12000)	89.95	0.65	97.26 ± 0.14	97.45	97.00
(400,12000)	90.13	0.86	97.73 ± 0.10	97.90	97.60
(600,12000)	92.43	1.09	97.80 ± 0.11	97.93	97.55
(800,12000)	97.84	1.34	97.79 ± 0.07	97.88	97.68
(200,16000)	159.85	0.83	97.16 ± 0.08	97.28	97.03
(400,16000)	158.92	1.07	97.71 ± 0.08	97.83	97.60
(600,16000)	165.28	1.40	97.80 ± 0.06	97.85	97.70
(800,16000)	170.25	1.69	97.80 ± 0.08	97.90	97.68

V. Computational performance

The evaluation was conducted on a desktop with a Intel®Core™i7-3770 running Matlab 2015b with 16GB of memory. The median time for training and testing for the different architecture is presented in Tables II, III, IV, and V. It shows it is possible to train the classifiers with two hidden layers and a large number of hidden neurons below 3 minutes, and the testing stage is also fast.

TABLE VI
PERFORMANCE WITH DIFFERENT THE MULTI-CLASSIFIER STRATEGY USING MEAN AND MAX FUNCTIONS TO COMBINE THE DECISION SCORES.

	CVL			Bangla			Devnagari			Oriya		
(L_1, L_2)	Mean	Med	Max	Mean	Med	Max	Mean	Med	Max	Mean	Med	Max
(100,0)	95.68	95.65	95.69	97.25	97.28	97.23	98.43	98.41	98.41	96.20	96.20	96.00
(500,0)	95.86	95.81	95.83	97.48	97.43	97.40	98.64	98.62	98.56	96.20	96.30	96.40
(1000,0)	95.83	95.81	95.86	97.38	97.43	97.45	98.54	98.59	98.56	96.40	96.30	96.20
(5000,0)	95.71	95.69	95.68	97.35	97.35	97.43	98.49	98.54	98.46	96.20	96.30	96.30
(10000,0)	95.98	95.92	95.83	97.43	97.40	97.53	98.49	98.46	98.46	95.80	95.70	95.80
(15000,0)	95.97	95.97	95.88	97.85	97.83	97.80	98.62	98.54	98.56	95.90	95.80	95.80
(200, 4000)	95.68	95.65	95.69	97.25	97.28	97.23	98.43	98.41	98.41	96.20	96.20	96.00
(400, 4000)	95.98	95.92	95.83	97.43	97.40	97.53	98.49	98.46	98.46	95.80	95.70	95.80
(600, 4000)	95.94	95.92	95.93	97.45	97.48	97.65	98.51	98.46	98.51	95.60	95.50	95.50
(800, 4000)	95.86	95.79	95.77	97.45	97.53	97.48	98.51	98.59	98.46	95.90	96.20	95.40
(200, 8000)	95.86	95.81	95.83	97.48	97.43	97.40	98.64	98.62	98.56	96.20	96.30	96.40
(400, 8000)	95.97	95.97	95.88	97.85	97.83	97.80	98.62	98.54	98.56	95.90	95.80	95.80
(600, 8000)	95.91	95.95	95.81	97.83	97.78	97.83	98.51	98.54	98.51	95.80	95.70	95.80
(800, 8000)	95.91	95.87	95.95	97.93	97.90	97.98	98.43	98.46	98.56	95.60	95.60	95.60
(200,12000)	95.83	95.81	95.86	97.38	97.43	97.45	98.54	98.59	98.56	96.40	96.30	96.20
(400,12000)	96.01	96.02	95.92	97.83	97.85	97.80	98.56	98.54	98.64	95.80	95.80	95.80
(600,12000)	95.86	95.87	95.83	97.80	97.88	97.78	98.46	98.46	98.49	95.80	95.70	95.70
(800,12000)	95.72	95.73	95.77	97.95	97.93	98.03	98.49	98.51	98.41	95.60	95.70	95.40
(200,16000)	95.71	95.69	95.68	97.35	97.35	97.43	98.49	98.54	98.46	96.20	96.30	96.30
(400,16000)	95.97	95.93	95.97	97.83	97.78	97.78	98.64	98.56	98.64	95.90	95.80	95.90
(600,16000)	95.86	95.85	95.83	97.80	97.85	97.78	98.54	98.54	98.54	95.90	95.90	95.70
(800,16000)	95.81	95.80	95.79	97.78	97.78	97.85	98.54	98.54	98.41	95.60	95.60	95.60

TABLE IV
PERFORMANCE WITH DIFFERENT ARCHITECTURES (DEVNAGARI).

	Time		Accuracy		
(L_1, L_2)	T_{train}	T_{test}	Mean±SD	Max	Min
(100,0)	2.39	0.10	80.90 ± 0.39	81.56	80.33
(500,0)	3.87	0.14	86.73 ± 0.43	87.51	86.05
(1000, 0)	6.48	0.18	90.51 ± 0.33	91.12	90.06
(10000,0)	13.13	0.34	98.32 ± 0.06	98.38	98.22
(15000,0)	41.14	0.60	98.45 ± 0.13	98.67	98.22
(200,4000)	11.61	0.27	98.29 ± 0.08	98.38	98.17
(400,4000)	13.13	0.34	98.32 ± 0.06	98.38	98.22
(600,4000)	14.97	0.42	98.33 ± 0.10	98.46	98.17
(800,4000)	16.94	0.53	98.28 ± 0.04	98.35	98.25
(200,8000)	39.27	0.44	98.49 ± 0.06	98.54	98.38
(400,8000)	41.14	0.60	98.45 ± 0.13	98.67	98.22
(600,8000)	43.05	0.73	98.43 ± 0.09	98.56	98.33
(800,8000)	46.13	0.88	98.40 ± 0.14	98.59	98.22
(200,12000)	86.50	0.69	98.46 ± 0.08	98.56	98.33
(400,12000)	88.51	0.80	98.54 ± 0.06	98.64	98.46
(600,12000)	90.54	1.06	98.45 ± 0.06	98.51	98.33
(800,12000)	94.78	1.25	98.39 ± 0.11	98.56	98.25
(200,16000)	152.22	0.87	98.41 ± 0.06	98.49	98.35
(400,16000)	156.59	1.03	98.53 ± 0.07	98.62	98.43
(600,16000)	160.60	1.29	98.50 ± 0.08	98.64	98.38
(800,16000)	166.74	1.61	98.44 ± 0.08	98.56	98.35

TABLE V
PERFORMANCE WITH DIFFERENT ARCHITECTURES (ORIYA).

	Time		Accuracy		
(L_1, L_2)	T_{train}	T_{test}	Mean±SD	Max	Min
(100,0)	1.00	0.03	83.88 ± 0.61	84.70	82.80
(500,0)	1.48	0.04	89.20 ± 0.61	90.00	88.40
(1000,0)	2.49	0.05	91.60 ± 0.52	92.90	91.10
(10000,0)	4.75	0.10	95.34 ± 0.41	96.00	94.90
(15000,0)	16.97	0.17	95.83 ± 0.22	96.30	95.60
(200,4000)	4.08	0.07	95.86 ± 0.23	96.20	95.60
(400,4000)	4.75	0.10	95.34 ± 0.41	96.00	94.90
(600,4000)	5.31	0.13	95.14 ± 0.39	95.80	94.50
(800,4000)	6.29	0.15	94.93 ± 0.37	95.50	94.40
(200,8000)	15.44	0.12	96.04 ± 0.28	96.50	95.40
(400,8000)	16.97	0.17	95.83 ± 0.22	96.30	95.60
(600,8000)	17.42	0.22	95.70 ± 0.24	96.20	95.30
(800,8000)	18.78	0.25	95.49 ± 0.28	95.80	94.80
(200,12000)	39.33	0.18	96.11 ± 0.28	96.30	95.60
(400,12000)	41.27	0.25	95.91 ± 0.20	96.10	95.50
(600,12000)	40.92	0.30	95.73 ± 0.22	96.10	95.40
(800,12000)	41.94	0.36	95.65 ± 0.19	95.80	95.30
(200,16000)	74.41	0.27	96.13 ± 0.09	96.30	96.00
(400,16000)	75.02	0.31	95.83 ± 0.19	96.20	95.50
(600,16000)	80.93	0.41	95.84 ± 0.21	96.30	95.50
(800,16000)	78.57	0.45	95.71 ± 0.13	95.90	95.50

VI. DISCUSSION AND CONCLUSION

The revival of Random Vector Functional-Link networks through its different variations and acronyms shows the key importance of the classifier architecture, and how researchers from previous decades established robust frameworks that lacked the power of present day computers to be fully exploited. The number of layers and the number of projections in each layer (e.g. the number of neurons) have a major impact on the accuracy. We have shown that it is possible to exploit different runs and improve the performance by combining the classifiers outputs. As the classifiers are fast to train, it is possible to combine a large number of decisions and increase the mean accuracy.

TABLE VII
COMPARISON OF THE ACCURACY (IN %) WITH OTHER METHODS.

Method	CVL	Bangla	Devnagari	Oriya
k-nn	89.33	93.70	96.25	92.00
IDMD	96.13	97.80	99.18	96.50
This paper	96.02	97.98	98.64	96.40

This type of network has several benefits, first it provides a good accuracy that is close to the state of the art, second it provides for a predefined architecture the minimum accuracy that can be obtained without learning the weights of the hidden layer(s) (e.g. by using backpropagation). RVFL/EML

models allow to properly evaluate the impact of learning. Because the architecture has such an effect on the accuracy, learning the architecture, i.e. setting the number of layers and neurons, seems to be more useful than learning the weights themselves if the number of available data is large enough.

In this paper, we have evaluated the performance of multi-layer RVFL/EML architectures on four handwritten databases. The results confirm the interest of this type of approach for both the accuracy and the processing time. This type of classifier can be useful in applications with large labeled instances where a classifier has to be deployed rapidly. Further work will include convolutional RVFL/EML models to evaluate the choice of the pooling and convolution techniques.

REFERENCES

[1] D. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 12, pp. 3207–3220, 2010.

[2] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems (NIPS) 2*, 1990, pp. 396–404.

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[4] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, July 2006.

[5] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, 1992.

[6] W. F. Schmidt, M. A. Kraaijveld, and R. P. W. Duin, "Feedforward neural networks with random weights," in *Proc. of 11th IAPR Int. Conf. on Pattern Recog., Conf. B: Pattern Recognition Methodology and Systems*, vol. 2, 1992, pp. 1–4.

[7] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. of the 12th Int. Conf. on Computer Vision (ICCV'09)*, 2009, pp. 2146–2153.

[8] G. B. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *IEEE Trans. on Neural Networks*, vol. 16, no. 1, pp. 57–67, 2005.

[9] A. M. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, "On random weights and unsupervised feature learning," in *Proc. of the 28th Int. Conf. on Machine Learning*, 2011, pp. 1–8.

[10] N. Nagy, "Twenty years of document image analysis in PAMI," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 38–62, 2000.

[11] D. Ghosh, T. Dube, and A. Shivaprasad, "Script recognition: A review," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2142–2161, Dec. 2010.

[12] R. Plamondon and N. S. Srihari, "On-line and off-line handwritten recognition: a comprehensive survey," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, Jan. 2000.

[13] C. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: Benchmarking of state-of-the-art techniques," *Pattern Recognition*, vol. 36, no. 10, pp. 2271–2285, Oct. 2003.

[14] N. Arica and F. T. Yarman-Vural, "An overview of character recognition focused on off-line handwriting," *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 31, no. 2, pp. 216–233, May 2001.

[15] M. Alhamdoosh and D. Wang, "Fast decorrelated neural network ensembles with random weights," *Information Sciences*, vol. 264, pp. 104–117, 2014.

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

[17] S. Scardapane, D. Wang, M. Panella, and A. Uncini, "Distributed learning for random vector functional-link networks," *Information Sciences*, vol. 301, pp. 271–284, 2015.

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

[19] C. L. P. Chen and J. Z. Wan, "A rapid learning and dynamic stepwise updating algorithm for flat neural networks and the application to time-series prediction," *IEEE Trans on Systems, Man, and Cybernetics, Part B*, vol. 29, no. 1, pp. 62–72, 1999.

[20] C. L. P. Chen, "A rapid supervised learning neural network for function interpolation and approximation," *IEEE Trans on Neural Networks*, vol. 7, no. 5, pp. 1220–1230, 1996.

[21] G. B. Huang and C. K. Siew, "Extreme learning machine with randomly assigned rbf kernels," *Int. J of Information Technology*, vol. 11, no. 1, pp. 16–24, 2005.

[22] L. P. Wang and C. R. Wan, "Comments on the extreme learning machine," *IEEE Trans on Neural Networks*, vol. 19, no. 8, pp. 1494–1495, 2008.

[23] L. L. C. Kasun, H. Zhou, G. B. Huang, and V. C. M., "Representational learning with extreme learning machine for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, Dec. 2013.

[24] J. Tang, C. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Networks and Learning Systems*, vol. 27, no. 4, pp. 809–21, 2016.

[25] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Sys. Man. Cyber.*, vol. 9, no. 1, pp. 62–66, 1979.

[26] M. Diem, S. Fiel, A. Garz, M. Keglevic, F. Kleber, and R. Sablatnig, "ICDAR 2013 competition on handwritten digit recognition (HDRC 2013)," in *Proc. of the 12th Int. Conf. on Document Analysis and Recognition (ICDAR)*, 2013, pp. 1454–1459.

[27] A. Back and A. Tsoi, "A time series modeling methodology using FIR and IIR synapses," in *Proc. of the Workshop on Neural Networks for Statistical and Economic Data*, 1990, p. 187194.

[28] B. B. Chaudhuri and U. Pal, "A complete printed Bangla OCR system," *Pattern Recognition*, vol. 31, pp. 531–549, 1998.

[29] U. Pal and B. B. Chaudhuri, "Indian script character recognition: a survey," *Pattern Recognition*, vol. 37, no. 9, pp. 1887–1899, Sept. 2004.

[30] U. Bhattacharya and B. Chaudhuri, "Databases for research on recognition of handwritten characters of indian scripts," in *Proc. of the 8th Int. Conf. on Document Analysis and Recognition (ICDAR'05)*, 2005, pp. 789–793.

[31] B. Chaudhuri, "A complete handwritten numeral database of Bangla - a major Indic script," in *Proc. of the 10th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR'10)*, 2006, pp. 1–6.

[32] S. Vajda, K. Roy, U. Pal, B. Chaudhuri, and A. Belaïd, "Automation of Indian postal documents written in Bangla and English," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 8, pp. 1599–1632, Dec. 2009.

[33] C.-L. Liu and H. Sako, "Class-specific feature polynomial classifier for pattern classification and its application to handwritten numeral recognition," *Pattern Recognition*, vol. 39, no. 4, pp. 669–681, 2006.

[34] C.-L. Liu and C. Y. Suen, "A new benchmark on the recognition of handwritten bangla and farsi numeral characters," *Pattern Recognition*, vol. 42, pp. 3287–3295, 2009.

[35] I. K. Sethi and B. Chatterjee, "Machine recognition of constrained hand printed devanagari," *Pattern Recognition*, vol. 9, pp. 69–75, July 1977.

[36] T. Bhowmick, S. Parui, U. Bhattacharya, and B. Shaw, "An HMM based recognition scheme for handwritten oriya numerals," in *Proc. of the 9th Int. Conf. on Information Technology (ICIT 2006)*, 2006, pp. 105–110.

[37] D. Keysers, T. Deselaers, C. Gollan, and H. Ney, "Deformation models for image recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 8, pp. 1422–1435, Aug. 2007.

[38] H. Cecotti, "Handwritten digit recognition of indian scripts: a cascade of distances approach," in *Int. Joint Conf. on Neural Networks*, 2015, pp. 1–7.

[39] ——, "Active graph based semi-supervised learning using image matching: application to handwritten digit recognition," *Pattern Recognition Letters*, pp. 1–8, 2016.