An Extreme Learning Machine Architecture Based on Volterra Filtering and PCA

Li CHEN†, Ling YANG†a), Nonmembers, Juan DU†, Member, Chao SUN†, Shenglei DU†, and Haipeng XI†, Nonmembers

SUMMARY Extreme learning machine (ELM) has recently attracted many researchers’ interest due to its very fast learning speed, good generalization ability, and ease of implementation. However, it has a linear output layer which may limit the capability of exploring the available information, since higher-order statistics of the signals are not taken into account. To address this, we propose a novel ELM architecture in which the linear output layer is replaced by a Volterra filter structure. Additionally, the principal component analysis (PCA) technique is used to reduce the number of effective signals transmitted to the output layer. This idea not only improves the processing capability of the network, but also preserves the simplicity of the training process. Then we carry out performance evaluation and application analysis for the proposed architecture in the context of supervised classification and unsupervised equalization respectively, and the obtained results either on publicly available datasets or various channels, when compared to those produced by already proposed ELM versions and a state-of-the-art algorithm: support vector machine (SVM), highlight the adequacy and the advantages of the proposed architecture and characterize it as a promising tool to deal with signal processing tasks.

key words: extreme learning machines, Volterra filter, principal component analysis, unsupervised equalization, support vector machine

1. Introduction

Feedforward neural networks (SLFNs) play key roles in data analytic and have been widely applied in many applications for its promising generalization ability [1]–[3]. Besides, plenty of research work has shown that SLFNs can approximate any function and form decision boundaries with arbitrary shapes if the activation function is chosen properly [4], [5]. However, most of traditional approaches for training SLFNs are slow due to their iterative steps. To overcome this problem, Huang et al. proposed an useful learning scheme called extreme learning machine (ELM) [6]–[8], which has been successfully applied for SLFNs training, leading to fast network training with low human supervision. The main idea in ELM-based approaches is that the network’s hidden layer weights and bias values need not to be learned and can be determined by random assignment, and then ELM transforms the training of the neural network into a linear problem where its output weights can be analytically determined instead of being tuned [9]. This approach is not in line with conventional SLFNs training approaches, like the back-propagation [10] and the Levenberg-Marquardt [11] algorithm, where both the network’s hidden and output parameters are adjusted by following an optimization process, e.g., by applying gradient descend-based optimization. Despite the random hidden layer parameters determination, it has been proven that SLFNs trained by using ELM algorithm have the properties of global approximation [12]. ELMs not only tend to reach the smallest training error, but also the smallest output weight norm, which results in better generalization performance for feedforward networks [13]. Due to its effectiveness and fast learning process, ELM and its many variants proposed in the recent years have been successfully used in various fields, such as forecasting of photovoltaic power [14], face recognition [15], terrain-based navigation [16], time-series data analysis [17] and so on.

Additionally, Suykens and Vandewalle [18] described a training method for SLFNs which applies the hidden layer output mapping as the feature mapping of support vector machine (SVM). However, the drawbacks of this method are the high computational cost and larger number of parameters in the hidden layer. Liu et al. [19] and Frénay et al. [20] showed that the ELM learning approach can be applied to SVMs directly by simply replacing SVM kernels with random ELM kernels and better generalization can be achieved. Recently, based on this and kernel theory, Huang et al. [21] successfully extended ELM to kernel learning, called the kernel extreme learning machine (KELM), which is proposed as an extension of the ELM learning theory, and the results showed that KELM, which can be obtained for feedforward neural networks and radial basis function (RBF) network, not only has universal approximation capability but also has good classification capability. However, the conventional SVM as well as the KELM do not scale well with the big datasets in general. For this problem, Deng et al. [22] presented a fast and accurate kernel-based supervised algorithm referred to as the reduced kernel extreme learning machine (RKELM) which randomly selects a subset of the available data samples as support vectors (or mapping samples) instead of identifying the support vectors (or weight vectors) iteratively with respect to SVM.

Nevertheless, both ELM and RKELM cannot make a fully effective use of the higher-order statistics of the signals coming from the hidden layer, since the output layer still corresponds to a linear combiner, for the sake of keeping the training process as simple as possible. Thus, to overcome this limitation, encouraged by the change of the read-out of echo state network [23]–[26], we introduce a nonlin-
ear architecture as the output layer of ELM.

In this work, we propose a novel ELM architecture characterized by employing the structure of a Volterra filter [27] as the output layer. This proposal is encouraged by the fact that the Volterra filter not only enables a better exploitation of the higher-order statistics, but also preserves the simplicity of the training process, since an optimal solution for the free parameters can be determined in the least squares (LS) sense. Additionally, we avoid the occurrence of an excessive growth in the number of weights to be adapted by applying the principal component analysis (PCA) technique [28], [29] before transmitting the hidden layer responses of the network to the output layer.

To provide a comprehensive evaluation of the performance of this novel architecture, we conducted plenty of experiments involving binary classification and multiclassification problems on commonly used datasets, and the results obtained demonstrate that the nonlinear output layer can lead to an excellent generalization performance and efficiency. In addition, we extend its application to a more representative scenario of signal processing problems: unsupervised equalization, where the ELMs, as an unsupervised equalizer, play the role of a nonlinear prediction-error filter according to the elegant idea in [30], [31]. And experimental results show the proposed architecture, in terms of prediction and equalization, can achieve significantly better performance compared with those produced by already proposed ELM versions and conventional SVM.

The rest of the paper is organized as follows: Sect. 2 gives a brief overview of the classic ELM, KELM and RKELM. Section 3 details the proposed architecture. Section 4 presents the performance estimation of the novel ELM architecture in the context of supervised classification, whereas Sect. 5 displays the set of results obtained with the ELMs and classical SVM, confirming the adequacy of the application in the unsupervised equalization problem, as well as highlighting the benefits of the proposed network. Finally, Sect. 6 summarizes the main conclusions.

2. Extreme Learning Machine and Its Kernel Extension Versions

In this section, an overview of ELM and its kernel extension versions: KELM [21] and RKELM [22] is presented. This serves to lay the foundation of the experiments in Sect. 4 and 5.

2.1 ELM

Consider an ELM neural network with \(d, l\) and \(m\) neurons in input layer, hidden layer and output layer respectively, as depicted in Fig. 1. For regression problems \(m = 1\), whereas for classification problems \(m\) is the number of categories, classes or labels.

Given \(G\) different samples \([\mathbf{x}_i, t_i]\), where \(\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{id}]^T \in \mathbb{R}^d\) and \(t_i = [t_{i1}, t_{i2}, \cdots, t_{im}]^T \in \mathbb{R}^m\). On regression problem, \(t_i \in \mathbb{R}^m\) is a continuous real value \((\mathbb{R} = \mathbb{R})\). On \(m\)-class classification problem, \(t_i \in \mathbb{R}^m\) is a \(m\)-dimensional Boolean vector \([0, 1]^m\). The output function of ELM for SLFNs is given by

\[
f_i(x) = \sum_{i=1}^{l} \beta_i h_i(x) = h(x)\beta,
\]

where \(\beta = [\beta_1, \beta_2, \cdots, \beta_m]^T\) is the output weights connected between the \(i\)-th hidden node and output nodes, \(h(x) = [h_1(x), \cdots, h_l(x)]\), \(h_i(x) = g(w_i b_i, x)\), \(w_i = [w_{i1}, w_{i2}, \cdots, w_{id}]^T\) is the input weights connected between input nodes and the \(i\)-th hidden node, \(b_i\) is the bias of the \(i\)-th hidden node, \(g(w_i, b_i, x)\) is a nonlinear piecewise continuous function satisfying ELM universal approximation capability theorems [8].

Standard single-hidden-layer feedforward neural networks with \(l\) hidden nodes can approach to any \(G\) samples with zero error, which can be expressed as

\[
f_i(x_j) = t_j, \quad j = 1, 2, \cdots, G.
\]

For the \(G\) equations of (2) can be simplified as

\[
H\beta = T,
\]

where \(H\) is the response matrix of hidden layer:

\[
H = \begin{bmatrix}
g(w_1 x_1 + b_1) & \cdots & g(w_1 x_1 + b_l) \\
g(w_1 x_2 + b_1) & \cdots & g(w_1 x_2 + b_l) \\
\vdots & \ddots & \vdots \\
g(w_1 x_G + b_1) & \cdots & g(w_1 x_G + b_l)
\end{bmatrix}
\]

and

\[
\beta = \begin{bmatrix}
\beta_1 \\
\beta_2 \\
\vdots \\
\beta_m
\end{bmatrix}, \quad T = \begin{bmatrix}
t_1 \\
t_2 \\
\vdots \\
t_G
\end{bmatrix}.
\]

For the linear system which has the fixed input weights \(w_i\) and hidden bias \(b_i\), the training of the single-hidden-layer feedforward neural networks is simply equivalent to find the LS solution of the linear system \(H\beta = T\), so the least norm and LS solution is

\[
\beta = H^T T,
\]
where $H^*$ is the Moore-Penrose generalized inverse matrix of hidden layer responses.

The ELM aiming to reach the smallest norm of output weights $\min_{\beta} ||\beta||_F^2$ and training error $\min_{\beta} ||H\beta - \mathbf{T}||_F^2$ can be solved as a constrained optimization problem [21], [32]:

$$\text{Minimize } \beta : ||H\beta - \mathbf{T}||_F^2 + \frac{C}{2} ||\beta||_1^2,$$  

(7)

where $\alpha_1 > 0$, $\alpha_2 > 0$, $p$, $q = 0$, $\frac{1}{2}$, 1, 2, $\cdots$, $F$, $+ \infty$, and $C$ is control parameter for the tradeoff of structural risk and experimental risk. Numerous efficient methods can be used to attain the output weights $\beta$ including but not limited to orthogonal projection method, iterative methods [33] and eigenvalue decomposition method [34]. When $p$, $q = F$ and $\alpha_1$, $\alpha_2 = 2$, an efficient closed-form solution [21] is

$$\beta = \begin{cases} (C I + H H^*)^{-1} T & G \geq l; \\ (C I + H^* H)^{-1} H^* T & G \leq l. \end{cases}$$  

(8)

2.2 KELM

If $h(\cdot)$ is unknown, i.e., an implicit function, as proposed in [21], one can apply the Mercer’s conditions on ELM, and define a kernel matrix for ELM that takes the form:

$$K_{\text{ELM}} = H H^T : K_{\text{ELM},i,j} = h(\mathbf{x}_i) \cdot h(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j).$$  

(9)

Then, substituting (9) and (8) into (1), we can obtain the kernel form of the output function as follows,

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ k(\mathbf{x}, \mathbf{x}_n) \end{bmatrix} (C I + K_{\text{ELM}})^{-1} \mathbf{T}. \quad \text{(10)}$$

Similar to the SVM and least squares support vector machine (LS-SVM) [35], $h(\cdot)$ need not be known; instead, its kernel $k(u, v)$ (e.g., Gaussian kernel $k(u, v) = \text{exp}(||u - v||^2 / \sigma)$) can be provided. $\sigma$ need not be available beforehand either. The experimental and theoretical analysis of [21] showed that the generalization performance of ELM with Gaussian kernel depends closely on the combination of the control parameter $C$ and kernel parameter $\sigma$; ($C$, $\sigma$), and more importantly, KELM produces improved generalization performance over the SVM/LS-SVM. The work, however, was established only on small datasets. When dealing with big data, the training time and kernel matrix size become a significant concern [37].

2.3 RKELM

To address the issue mentioned in the previous section, based on kernel theory and ELM, [22] presented a fast non-iterative kernel machine, which is referred to as RKELM. The key characteristic of the RKELM algorithm is that support vectors are randomly chosen from the train set as opposed to some sophisticated process which is often compute intensive. In other words, the work is derived from KELM but only uses a reduced kernel matrix instead of the full kernel matrix to build the model. Some crucial details concerning to this algorithm are given below:

The SLFNs with kernel function $k(\cdot, \cdot)$ and $l$ support vectors $\mathbf{x}_i = \{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^l$ can be modeled as

$$\sum_{i=1}^l \beta_i k(\mathbf{x}_i, \mathbf{x}_j) = t_j, \quad j = 1, 2, \ldots, G$$  

(11)

or compressed in the matrix form:

$$K_{\text{GSL}} \beta = \mathbf{T}, \quad \text{(12)}$$

where $K_{\text{GSL}} = k(\mathbf{x}, \mathbf{x}_i)$ is the reduced kernel matrix, $\mathbf{x} = \{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^l$ and $\beta = [\beta_1, \beta_2, \cdots, \beta_l]$ is the output weight vector. Similar to (7), the RKELM with optimization constraints such as the $l_2$ norm minimization can be formulated as

$$\text{Minimize } \frac{C}{2} ||\beta||^2 + \frac{1}{2} ||\xi||^2.$$

Subject to: $\xi = K_{\text{GSL}} \beta - \mathbf{T}$. Based on the KKT theorem, the solution is derived as

$$\beta = (C I + K_{\text{GSL}}^T K_{\text{GSL}})^{-1} K_{\text{GSL}}^T \mathbf{T}. \quad \text{(14)}$$

The results obtained from [22] demonstrated that, RKELM can perform at competitive level of generalized performance as the SVM/LS-SVM at only a fraction of the computational effort incurred in the context of regression, binary classification and multi-classification problems.

3. Proposed Architecture

In Sect. 2, we described ELM and its kernel extension versions: KELM and RKELM. However, in both proposals, the output layer corresponds to a linear output layer, which cannot exploit the higher-order statistics of the information coming from the nonlinear dynamics of the neurons within the hidden layer. Therefore, to circumvent this limitation, using a nonlinear output layer may be an interesting perspective. However, it is crucial that the output layer remains linear with respect to the free parameters, so that it is possible to find a closed-form solution in the LS (or Wiener) sense [38]. Taking into account these facts, our proposal consists in employing a Volterra filter structure [27] in the output layer.

3.1 Volterra Filter and Principal Component Analysis

Consider $\mathbf{x}(n) = [x_1(n), \cdots, x_l(n)]^T$ is the input vector of the Volterra filter, its outputs shall be composed of linear combinations of polynomial terms, as follows:

$$y_s(n) = h_0^{(s)}(n) + \sum_{i=1}^l h_1^{(s)}(i)x_i(n) + \sum_{i=1}^l \sum_{j=1}^l h_2^{(s)}(i, j)x_i(n)x_j(n)$$

$$+ \sum_{i=1}^l \sum_{j=1}^l \sum_{k=1}^l h_3^{(s)}(i, j, k)x_i(n)x_j(n)x_k(n) + \cdots, \quad \text{(15)}$$

IEICE TRANS. INF. & SYST., VOL.E100–D, NO.11 NOVEMBER 2017
where $x_h(n)$ represents the $k$-th hidden node (support vector), $y_s(n)$ is the $s$-th network output and $h^{(z)}_k(k_1, \cdots, k_z)$ denote the filter coefficients called Volterra kernels, with $z = 1, \cdots, M_0$ representing the order of the corresponding polynomial terms. $M_0$ will assume a finite value when the polynomial expansion is truncated.

However, the Volterra structure introduces a new concern: as $l$ increases (for ELM $l$ is the number of hidden nodes, whereas for RKELM it is the number of support vectors), the number of Volterra kernels tends to grow dramatically. In fact, this becomes evident if we observe the expression for the number of non-ambiguous kernels $N_{\text{ker}}$:

$$N_{\text{ker}} = 1 + \frac{l}{1} + \frac{l(l + 1)}{2} + \frac{l(l + 1)(l + 2)}{6} + \cdots.$$  \hfill (16)

As we can see, the respective number of coefficients $N_{\text{ker}}$ is severely increased when either $M_0$ or $l$ grows, which threatens the practical application of this structure along with ELM. To solve this problem, we introduced the classical dimensionality reduction technique PCA [28], [29] which significantly reduces $N_{\text{ker}}$ without necessarily causing a significant loss of information.

Let $X \in \mathbb{R}^{l \times G}$ be the response matrix of hidden layer of ELM considering $G$ input samples. The samples covariance matrix can be estimated as $\hat{C} = \frac{X^T X}{G}$, and $V \in \mathbb{C}^{l \times N_{\text{pc}}}$ is the matrix formed by the eigenvectors of that correspond to its $N_{\text{pc}}$ largest eigenvalues. Thus, the principal components are given by

$$q_i(n) = V_i^T X, \quad i = 1, \cdots, N_{\text{pc}}.$$  \hfill (17)

The perspective of using PCA is encouraged by the observation that there is a non-negligible degree of redundancy between hidden layer responses, which means it is possible to retain a few components of $X$ representing the most significant portion of the dynamics from the hidden nodes.

In this context, the eigenvalues of $\hat{C}$ offer a valuable assistance for choosing how many principal components are necessary: since the sum of the $N_{\text{pc}}$ largest eigenvalues of $\hat{C}$, divided by the total eigenvalue sum, represents the percentage of the cumulative energy covered by the corresponding $N_{\text{pc}}$ directions, one should choose $N_{\text{pc}}$ in order to reach an appropriate trade-off between effective dimensionality reduction and mean squared compression error.

3.2 The ELM Architecture Based on Volterra Filter and PCA

Based on the discussion above, using PCA prior to Volterra Filter not only avoid the soaring computational complexity of Volterra filter coming from enormous growth in the number of Volterra kernels as a result of the increase of $l$, but also unable to causing a significant loss of information.

An important aspect to be emphasized is that the proposed architecture can be flexibly applied in ELM versions with linear output layer. Therefore, we analyzed its performance using the classical ELM and RKELM [22], which are labeled here as PVELM and PVRKELM respectively. According to the previous section, the proposed algorithm can be considered to be a learning process formed by two processing steps as follows.

In the first step, PCA is used to reduce the dimension of hidden layer responses matrix $H$ of ELM (or kernel matrix $K$ of RKELM) from $G \times l$ to $G \times N_{\text{pc}}$, and $H_{G \times N_{\text{pc}}}$ is given by

$$H_{G \times N_{\text{pc}}} = \begin{bmatrix} h_{1,1} & \cdots & h_{1,l} & \cdots & h_{1,G} \\ h_{2,1} & \vdots & h_{2,l} & \cdots & h_{2,G} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ h_{N_{\text{pc}},1} & \cdots & h_{N_{\text{pc}},l} & \cdots & h_{N_{\text{pc}},G} \end{bmatrix}^T.$$  \hfill (18)

In the second step, consider $H(n) = [h_1(n), h_2(n), \cdots, h_{N_{\text{pc}}}(n)]^T$ be the input vector of Volterra filter, then let $H_s(n)$ be the vector containing the polynomial cross-products associated with the “linear input” $H(n)$ until the $M_0$-th order, and $h \in \mathbb{R}^{N_{\text{pc}} \times M_0}$ be the matrix containing the Volterra kernels, where the $s$-th column contains the set of coefficients $h^{(z)}(k_1, \cdots, k_z)$, $z = 1, \cdots, M_0$, associated with the $s$-th network output. Thus, the filter outputs (15) can be rewritten as follows:

$$y(n) = h^T H_s(n).$$  \hfill (19)

As a consequence, the Volterra filter, albeit being a nonlinear structure, presents a linear dependence with respect to the adjustable parameters $h^{(z)}(k_1, \cdots, k_z)$, $z = 1, \cdots, M_0$. Then the conceptual framework of optimal filtering and Wiener solution can be readily extended to the Volterra filter according to [39], so that its optimal coefficient $h$ which is the equivalent of the output weight $\beta$ of ELM can be presented as

$$\beta = R_h^T p_{ht, d}.$$  \hfill (20)

where $R_h = E[H_s(n)H_s^T(n)] = \frac{1}{G} \sum_{i=1}^{G} H_s(n)H_s^T(n)$ and $p_{ht, d} = E[H_s(n)d(n)] = \frac{1}{G} \sum_{i=1}^{G} H_s(n)d(n)$, with $d(n)$ representing the output desired signal.

In summary, the final feature obtained by this new architecture $h_{\text{par}} \times M_0$ fully contains the higher-order statistics of the representations of what the hidden layer learned from the data compared with that of the classical ELM $\beta_{\text{par}}$, in other words, it can provide a more effective exploration of the information concerning the hidden layer of ELM, which can be used to illustrate theoretically why the proposed ELM architecture works better.

4. Performance Evaluation: Supervised Classification

In this section, we present the methodology employed for the training and testing of ELMs. Then evaluate the learning performance of the proposed PVELM and PVRKELM by assessing it against their counterparts ELM and RKELM using well established real world benchmark datasets in the field of supervised classification.
4.1 Methodology

To provide a comprehensive investigation on the performance of the different algorithms, a wide range of datasets have been considered in our experimental study, which include six binary classification datasets and five multi-classification datasets. The training and testing sets are defined as described in Table 1, but the order of the dataset is randomly shuffled for each independent trial. Most of these datasets have been taken from the UCI Machine Learning Repository and LIBSVM website.

For the classical ELM and proposed PVELM, the sigmoid activation function \( g(w, b, x) = \frac{1}{1+\exp(-k(w \cdot x+b))} \) has been considered, where \( \lambda \) is set to ‘1’ unless otherwise stated. For RKELM and PVRKELM, the Gaussian kernel function \( k(x, x_i) = \exp(-||x-x_i||^2/\sigma) \) has been used in the experimental study. Besides, all the results reported are the average of 20 independent trials. Another important aspect to be emphasized is that, in the proposed networks, we employed a third-order Volterra filter to compute the outputs, without considering the quadratic terms for the sake of simplicity.

For all networks, the number of hidden nodes (support vectors) \( l \) is chosen by cross-validation method with step size 10, and only the obtained optimized \( l \) is reported. In order to achieve good generalization performance, the cost parameter \( C \) and kernel parameter \( \sigma \) of RKELM and PVRKELM need to be chosen appropriately. We have tried a wide range of \( C \) and \( \sigma \). For each dataset, we have used 20 different values of \( C \) and different values of \( \sigma \), resulting in a total of 400 pairs of \( (C, \sigma) \). The 20 different values of \( C \) and \( \sigma \) are \( \{2^{-5}, 2^{-4}, \ldots, 2^{4}, 2^{10}\} \). And the best average performance is reported. The user-specified parameters considered in the simulation trials are then given in Table 2.

4.2 Performance Evaluation

The performance efficacy of the ELMs has been considered on binary classification and multi-classification problems. Training accuracy (Tr. rate) and testing accuracy (Ts. rate) are employed to measure the performance of the algorithms. In all experiments, in order to observe the impact of the number of principal components \( N_{pc} \) on classification performance, \( l \) was set to a fixed value, whereas the \( N_{pc} \) value of the proposed architecture assumed different values. Table 3 and 4 report the results on the binary classification and multi-classification problems respectively. Besides, to conduct fair comparison, in terms of both training time (Tr. time) and testing accuracy (Ts. rate), Table 5 presents the results of all algorithms mentioned above with the same number of features 286 on each dataset. In other words, for PVELM and PVRKELM \( N_{pc} \) is set to 286 (\( N_{pc} = 10 \)), while for ELM and RKELM consider the case of \( l = 286 \).

Some interesting remarks can be drawn from Table 3 and 4. Firstly, it is possible to observe that, as expected, the performance of the proposed PVELM and PVRKELM is improved when the number of principal components \( N_{pc} \) is increased. Secondly, whether binary classification or multi-classification problems, by comparing the testing accuracy of ELM and RKELM to that of their counterparts PVELM and PVRKELM respectively, the benefits acquired with the use of a nonlinear output layer become evident. In fact, both PVELM and PVRKELM outperformed classical ELM and RKELM with 2%-10% improvement. Taking the dataset vowel as an example, consider the case where \( N_{pc} = 20 \), PVELM and PVRKELM achieved testing accuracy performance of 87.85% and 92.30%, respectively, which are greatly superior to that of the classical ELM and RKELM at 83.64% and 85.63%, respectively. This means that the additional flexibility acquired with the use of a nonlinear output layer has directly contributed to a better approximation of the desired behavior.

As shown in Table 5, it is possible to notice that PVELM and PVRKELM showcased significant improvements in the test performance with respect to classical ELM and RKELM while the training time taken by the proposed networks is a little higher than their counterparts, the latter is attributed to the fact that the Volterra filtering operation of PVELM and PVRKELM is originally thought to be more computationally intensive than the linear solution of output layer in ELM and RKELM. For example on dataset australian, PVELM took 0.3000 s to achieve the 88.10% testing accuracy while ELM took 0.2105 s to achieve a testing accuracy of 77.03%. This indicates the more efficiency of the proposed architecture compared to just putting more hidden neurons (support vectors) in a standard ELM (RKELM).
5. Application Analysis: Unsupervised Equalization

In this section, first, the principle of prediction-based unsupervised equalization is given, then the methodology employed for the training and testing of the ELMs and SVM is described. Finally we present the analysis on application of the novel architecture in the context of unsupervised equalization problem.

5.1 Prediction-Based Unsupervised Equalization

Traditional techniques for communication channel equalization are based on linear transversal equalizers (LTE’s), PVELM, RKELM and PVRKELM.
whose coefficients are being adjusted to match the channel characteristics. Depending on whether the equalizer knows an originally transmitted sequence or not, it is characterized as trained adaptation or blind equalizer respectively. Unsupervised (Blind) equalization is a particularly useful and difficult type of equalization, as for example in the case of multipoint communication networks.

Some classical strategies for unsupervised equalization are the following related algorithms: decision directed (DD), Sato, Godard [40] and Benveniste-Goursat [41]. Considering the transmitted symbols to be uncorrelated, it is possible to deal with the unsupervised equalization problem by means of prediction [42].

The representation of a prediction-based equalizer is shown in Fig. 2, where \( x(n) \) is the noisy channel output sequence, \( b(n) \) is the noise sequence, \( \hat{x}(n) \) is the predicted signal, \( e(n) \) is the prediction error, \( \mathbf{P} \) is a prediction filter (in this paper, we use the ELMs and SVM) and \( g \) is an Automatic Gain Control (AGC) [30].

The channel is modeled as a linear filter with finite impulse response represented by

\[
F(z) = \sum_{i=0}^{N-1} f_i z^{-i},
\]

where \( N \) is the channel length and \( f_i \) are the channel coefficients. Equation (21) can be also represented in a vectorial form: \( \mathbf{f} = [f_0, f_1, f_2, \ldots, f_{N-1}]^T \), the same as this, \( \mathbf{a}(n) = [a(n), a(n-1), \ldots, a(n-N+1)]^T \) which is the original signal sequence, \( \mathbf{b}(n) = [b(n), b(n-1), \ldots, b(n-N-M+1)]^T \) and \( \mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-N+M+1)]^T \), where \( M \) is the length of equalizer. Accordingly, the noiseless channel outputs, which we call channel states, can be revealed by the following expression:

\[
\begin{align*}
\bar{x}(n) &= a(n)f_0 + \cdots + a(n-N+1)f_{N-1} \\
\bar{x}(n-1) &= a(n-1)f_0 + \cdots + a(n-N+2)f_{N-1} \\
\bar{x}(n-2) &= a(n-2)f_0 + \cdots + a(n-N+3)f_{N-1} \\
\vdots 
\end{align*}
\]

then the prediction error corresponds to

\[
e(n) = x(n) - \mathbf{P}(x(n-1)),
\]

where \( \mathbf{x}(n) = \bar{x}(n) + \mathbf{b}(n) \), \( \mathbf{x}(n-1) = [x(n-1), x(n-2)]^T \) and \( \mathbf{P}(\cdot) \) is predictive filter function. For the purpose of removing the temporal redundancies of the received signal, (23) can be rewritten as

\[
e(n) = a(n)f_0 + a(n-1)f_1 + \cdots + b(n) \\
- \mathbf{P}(\bar{x}(n-1) + b(n-1) + \bar{x}(n-2) + b(n-2) + \cdots).
\]

We can represent the predictor device in a linear filter with discrete finite impulse response form: \( \mathbf{P} = [p_1, p_2, \ldots, p_k] \), then

\[
e(n) = a(n)f_0 + a(n-1)f_1 + \cdots + b(n) \\
- \mathbf{P}(\bar{x}(n-1) + b(n-1) + \bar{x}(n-2) + b(n-2) + \cdots).
\]

Combining (25) and (26) leads to

\[
e(n) = a(n)f_0 + b(n) + a(n-1)f_1 + \cdots + b(n-1)p_1 \\
+ a(n-2)f_0 + a(n-3)f_1 + \cdots + b(n-2)p_2 + \cdots \\
+ a(n-k)f_0 + a(n-k+1)f_1 + \cdots + b(n-k)p_k.
\]

The object of equalization is to retain \( a(n)f_0 \) of (27) and remove other parts, then restore \( a(n)f_0 \) to \( a(n) \) using AGC. Since the linear mapping of a linear predictor cannot remove all the redundancy of (27), many researchers choose to use a nonlinear predictor as the filter.

Supposing \( \psi_{NN} \) is the function of predictive filter, (27) can be rewritten as

\[
e(n) = x(n) - \psi_{NN}(x(n-1)) \\
= a(n)f_0 + a(n-1)f_1 + \cdots + b(n) \\
- \psi_{NN}(x(n-1), x(n-2), \ldots).
\]

For the purpose of equalization, we can train the predictive filter to achieve

\[
\psi_{NN}(x(n-1)) = a(n-1)f_1 + a(n-2)f_2 \\
+ \cdots + a(n-N+1)f_{N-1} + b(n). \quad (29)
\]

According to this, in this paper, we employ nonlinear prediction-error filter to unsupervised equalization problem, in which the ELMs and SVM are employed to predict the received signal \( x(n) \), so that the error between the prediction value \( \hat{x}(n) \) and the actual value \( x(n) \) of the received signal contains the desired information regarding the original source \( a(n) \).

5.2 Methodology

In all simulation of this part, the source signal assumes the values \( \{+1, -1\} \) with equal probability (BPSK modulation). All networks aim to estimate the original information \( a(n) \) using solely the received signal \( x(n) \) at the same time instant, meaning that there is no equalization delay. Besides, we do
not consider the presence of noise, except in one particular scenario, where we analyzed the influence of the signal-to-noise ratio (SNR) over the performance of the proposed architectures and their counterpart: ELM and RKELM.

The first hundred samples of 20000 sample data from the BPSK modulation discarded to eliminate transient effects. Half of the remaining samples are used as the training set and the other half as the testing set. Equation (30) was used to assess the quality of the equalization performed by each network:

\[
\text{AMSE} = \frac{1}{N_e G} \sum_{n=1}^{G} [x(n) - \hat{x}(n)]^2, \tag{30}
\]

where AMSE is the average of mean square error, \(N_e\) is the number of simulations of every group experiment (to consistently evaluate the performance of the networks, we considered \(N_e = 20\)), and \(G\) is the number of samples. Additionally, we present the standard deviation of the MSE values obtained in such experiments, which offers an interesting view over the variability of the performance achieved with the ELMs and SVM in the unsupervised equalization. It is important to stress that, even though there is a reference signal, which resembles a supervised training framework, it essentially involves samples of the received signal \(x(n)\), which are clearly available at the receiver. Thus, since no explicit information regarding the original source is used during training, it can be adequately classified as unsupervised.

As performed in the experiments for supervised classification, we employed a third-order Volterra filter to compute the outputs. And in all communication channel used in our experiments involving the proposed architecture, \(N_{pc}\) takes the value of 1, which means the matrix after dimension reduction through PCA is \(H_{G \times 1} = [h_1, h_2, \ldots, h_G]\). The input vector of Volterra filter in the equalization problem is \(\hat{H}(n) = [h_1(n), h_2(n), \ldots, h_p(n)]^T = [h_0, h_{n+1}, \ldots, h_{M+n-1}]^T\), therefore, (16) can be rewritten as

\[
N_{\text{ker}} = 1 + M + \frac{M(M+1)}{2} + \frac{M(M+1)(M+2)}{6} + \cdots, \tag{31}
\]

With the proposed architectures, the number of hidden nodes (support vectors) was set to \(l = 40\), whereas \(M\) assumed different values. With RKELM, PVRKELM and SVM, as performed in the experiments for supervised classification, we tried 400 pairs of \((C, \sigma)\) and the best average performance is reported. Besides, the obtained optimized \(SV\) which is the number of support vectors of SVM is also given. The details about the optimized parameter for each channel are reported in Table 6.

### 5.3 Application Analysis

To present a comprehensive analysis on application to the unsupervised equalization, four groups of experiments are considered in our work which include seven channels.

<table>
<thead>
<tr>
<th>Channels</th>
<th>RKELM</th>
<th>PVRKELM</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>(H_1(z))</td>
<td>((2^2, 2^1))</td>
<td>((2^2, 2^1))</td>
<td>((2^2, 2^1), 1019)</td>
</tr>
<tr>
<td>(H_2(z))</td>
<td>((2^2, 2^2))</td>
<td>((2^1, 2^0))</td>
<td>((2^1, 2^1), 9329)</td>
</tr>
<tr>
<td>(H_3(z))</td>
<td>((2^2, 2^2))</td>
<td>((2^1, 2^1))</td>
<td>((2^1, 2^0), 9850)</td>
</tr>
<tr>
<td>(H_4(z))</td>
<td>((2^2, 2^2))</td>
<td>((2^2, 2^0))</td>
<td>((2^2, 2^0), 9241)</td>
</tr>
<tr>
<td>(H_5(z))</td>
<td>((2^2, 2^2))</td>
<td>((2^1, 2^2))</td>
<td>((2^1, 2^2), 3939)</td>
</tr>
<tr>
<td>(H_6(z))</td>
<td>((2^2, 2^2))</td>
<td>((2^2, 2^2))</td>
<td>((2^2, 2^0), 8607)</td>
</tr>
<tr>
<td>(H_7(z))</td>
<td>((2^2, 2^2))</td>
<td>((2^2, 2^2))</td>
<td>((2^2, 2^0), 7607)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(H_1(z))</th>
<th>AMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELM</td>
<td>5.236(0.103)e-02</td>
</tr>
<tr>
<td>PVELM</td>
<td>5.184(0.060)e-02</td>
</tr>
<tr>
<td>RKELM</td>
<td>5.541(0.076)e-02</td>
</tr>
<tr>
<td>PVRKELM</td>
<td>5.541(0.030)e-02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(H_1(z))</th>
<th>AMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELM</td>
<td>5.236(0.103)e-02</td>
</tr>
<tr>
<td>PVELM</td>
<td>5.184(0.060)e-02</td>
</tr>
<tr>
<td>RKELM</td>
<td>5.541(0.076)e-02</td>
</tr>
<tr>
<td>PVRKELM</td>
<td>5.541(0.030)e-02</td>
</tr>
</tbody>
</table>

5.3.1 The First Group of Experiment

In this part, we shall consider the typical telephone channel which is characterized by the following transfer function: \(H_1(z) = 0.005 + 0.009z^{-1} - 0.024z^{-2} + 0.854z^{-3} - 0.218z^{-4} + 0.049z^{-5} - 0.016z^{-6}\). The ELMs have been trained and tested according to the methodology described in the previous section. Exceptionally, both the noiseless case and noisy case are considered in this channel and the obtained results are presented in Table 7 and 8 respectively. The values presented inside parentheses correspond to the standard deviation.

Firstly, we shall analyze the performance of each network considering the noiseless case. From the obtained results in this case (Table 7), it is possible to notice that the proposed networks have achieved significantly better results.
than their counterparts: ELM and RKELM. This observation is corroborated when we compare the AMSE values associated with each version of the proposed network. Taking RKELM and PVRKELM as examples, consider the case where \( l = 40 \) for the former network and \( M = 10 \) for the proposed one, the testing AMSE value achieved by the proposed architecture (5.585e-07) are 5 orders of magnitude smaller than the RKELM (5.223e-02). This is an important evidence of the advantages obtained with the use of a Volterra filter structure in the output layer. On the other hand, there is not much difference between the AMSE values obtained with the PVELM and PVRKELM, which can reveal the flexibility and the dominance of the proposed architecture compared to ELM versions with linear output layer. Besides, it can also be observed when \( M \) increase the performance of the proposed ELM versions is improved in most cases.

Now, we consider the case where the received samples are corrupted with a Gaussian additive noise. This means that the inputs of the ELMs are given by \( u(k) = x(k) + \sigma N(0, 1) \), where \( \sigma \) denotes the standard deviation of the noise and \( N(0, 1) \) corresponds to a zero-mean and unit-variance Gaussian random variable. In this case, \( l = 40 \) considered for each network mentioned, whereas \( M = 10 \) for the proposed networks, we performed 20 independent experiments for each network considering three different SNR values. Compared with Table 7, Table 8 reveals that there is a significant increase in the reachable error levels as the noise appears and becomes more pronounced. Nonetheless, we can observed from Table 8 for all the values of SNR, the proposed PVELM and PVRKELM led to a better performance with respect to ELM and RKELM, although the difference between the AMSE values associated with each network is relatively small.

### 5.3.2 The Second Group of Experiment

In this part, we shall consider three particular channels related to phase: in the first case, it is a common minimum phase channel characterized by the transfer function: 
\[
H_2(z) = 1 + 0.5 z^{-1} + 0.25 z^{-2} + 0.125 z^{-3};
\]
and in the second case, the channel is a non-minimum phase channel with the transfer function: 
\[
H_3(z) = 0.3482 + 0.8704 z^{-1} + 0.6057 z^{-2} + 0.3482 z^{-3};
\]
the last we shall deal with is a mixed-phase channel which is described by the transfer function: 
\[
H_4(z) = 1 - 2 z^{-1} + z^{-2} + 0.68 z^{-3}.
\]
And the obtained results are presented in Table 9, 10, and 11 respectively.

The AMSE values displayed in these three tables clearly reveal that the application of a nonlinear output layer, which forms the basic idea of the proposal, led to a good performance improvement with respect to the classical ELM and RKELM, although the benefits were not as significant as occurred in the typical telephone channel. Besides, we can find the proposed networks can obtain a better performance with a small \( M \) value and the performance maintains at a certain order of magnitude as \( M \) value increase. For example, take nonminimum phase channel \( H_3(z) \): AMSE values of the novel structure remain at 4 orders of magnitude when \( M \) value increases from 6 to 10, which indicates its unsupervised equalization performance can only reach this level.

### 5.3.3 The Third Group of Experiment

In this part, we shall consider two kinds of channels: linear channel and nonlinear channel. The first kind of channel involved in this work is defined by the transfer function: 
\[
H_5(z) = -0.3038 - 0.1426 z^{-1} - 0.6057 z^{-2} - 0.7937 z^{-3},
\]
although the linear channel seems to be relatively simple, it can introduces the most severe distortion into the transmitted signal due to four coefficients. And we present in Table 12 the AMSE values and the standard deviations related

<table>
<thead>
<tr>
<th>Network</th>
<th>Parameter</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELM</td>
<td>( l = 5 )</td>
<td>3.918(0.1279)e-02</td>
<td>3.917(0.1252)e-02</td>
</tr>
<tr>
<td></td>
<td>( l = 20 )</td>
<td>2.643(0.3459)e-02</td>
<td>2.643(0.4662)e-02</td>
</tr>
<tr>
<td></td>
<td>( l = 40 )</td>
<td>1.355(0.4599)e-02</td>
<td>1.543(0.5467)e-02</td>
</tr>
<tr>
<td>PVELM</td>
<td>( M = 3 )</td>
<td>6.503(0.0389)e-03</td>
<td>6.542(0.0449)e-03</td>
</tr>
<tr>
<td></td>
<td>( M = 6 )</td>
<td>4.466(0.0056)e-03</td>
<td>4.492(0.0011)e-03</td>
</tr>
<tr>
<td></td>
<td>( M = 10 )</td>
<td>4.456(0.0122)e-03</td>
<td>4.498(0.0172)e-03</td>
</tr>
<tr>
<td>RKELM</td>
<td>( l = 5 )</td>
<td>3.980(0.0091)e-03</td>
<td>3.980(0.0078)e-03</td>
</tr>
<tr>
<td></td>
<td>( l = 20 )</td>
<td>3.980(0.0093)e-03</td>
<td>3.980(0.0094)e-03</td>
</tr>
<tr>
<td></td>
<td>( l = 40 )</td>
<td>3.976(0.0098)e-03</td>
<td>3.979(0.0075)e-03</td>
</tr>
<tr>
<td>PVRKELM</td>
<td>( M = 3 )</td>
<td>6.074(0.0115)e-03</td>
<td>6.095(0.0199)e-03</td>
</tr>
<tr>
<td></td>
<td>( M = 6 )</td>
<td>4.532(0.0219)e-03</td>
<td>4.480(0.0157)e-03</td>
</tr>
<tr>
<td></td>
<td>( M = 10 )</td>
<td>4.461(0.0110)e-03</td>
<td>4.489(0.0212)e-03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>Parameter</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELM</td>
<td>( l = 5 )</td>
<td>4.118(0.0464)e-02</td>
<td>4.115(0.0439)e-02</td>
</tr>
<tr>
<td></td>
<td>( l = 20 )</td>
<td>4.991(0.0654)e-02</td>
<td>4.001(0.0651)e-02</td>
</tr>
<tr>
<td></td>
<td>( l = 40 )</td>
<td>9.400(0.2458)e-04</td>
<td>9.428(0.2623)e-04</td>
</tr>
<tr>
<td>PVELM</td>
<td>( M = 3 )</td>
<td>6.530(0.0017)e-04</td>
<td>6.551(0.0024)e-04</td>
</tr>
<tr>
<td></td>
<td>( M = 6 )</td>
<td>6.540(0.0112)e-04</td>
<td>6.564(0.0108)e-04</td>
</tr>
<tr>
<td></td>
<td>( M = 10 )</td>
<td>6.410(0.0971)e-02</td>
<td>6.448(0.0966)e-02</td>
</tr>
<tr>
<td>RKELM</td>
<td>( l = 5 )</td>
<td>4.120(0.0011)e-02</td>
<td>4.012(0.0039)e-02</td>
</tr>
<tr>
<td></td>
<td>( l = 10 )</td>
<td>4.009(0.0035)e-02</td>
<td>4.010(0.0041)e-02</td>
</tr>
<tr>
<td>PVRKELM</td>
<td>( M = 3 )</td>
<td>4.474(0.1087)e-03</td>
<td>4.478(0.1065)e-03</td>
</tr>
<tr>
<td></td>
<td>( M = 6 )</td>
<td>6.552(0.0121)e-04</td>
<td>6.566(0.0117)e-04</td>
</tr>
<tr>
<td></td>
<td>( M = 10 )</td>
<td>6.551(0.0197)e-04</td>
<td>6.565(0.0268)e-04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network</th>
<th>Parameter</th>
<th>Training</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELM</td>
<td>( l = 5 )</td>
<td>4.182(0.2192)e-02</td>
<td>4.188(0.2103)e-02</td>
</tr>
<tr>
<td></td>
<td>( l = 20 )</td>
<td>3.486(0.1965)e-02</td>
<td>3.492(0.1982)e-02</td>
</tr>
<tr>
<td></td>
<td>( l = 40 )</td>
<td>2.338(0.6315)e-02</td>
<td>2.347(0.6323)e-02</td>
</tr>
<tr>
<td>PVELM</td>
<td>( M = 3 )</td>
<td>8.173(0.0365)e-03</td>
<td>8.228(0.0303)e-03</td>
</tr>
<tr>
<td></td>
<td>( M = 6 )</td>
<td>5.983(0.0099)e-03</td>
<td>6.000(0.0223)e-03</td>
</tr>
<tr>
<td></td>
<td>( M = 10 )</td>
<td>5.971(0.0176)e-03</td>
<td>6.002(0.0324)e-03</td>
</tr>
<tr>
<td>RKELM</td>
<td>( l = 5 )</td>
<td>4.036(0.0594)e-02</td>
<td>4.043(0.0559)e-02</td>
</tr>
<tr>
<td></td>
<td>( l = 10 )</td>
<td>3.797(0.0380)e-02</td>
<td>3.801(0.0356)e-02</td>
</tr>
<tr>
<td></td>
<td>( l = 40 )</td>
<td>3.778(0.0047)e-02</td>
<td>3.783(0.0050)e-02</td>
</tr>
<tr>
<td>PVRKELM</td>
<td>( M = 3 )</td>
<td>1.019(0.2008)e-02</td>
<td>1.017(0.1911)e-02</td>
</tr>
<tr>
<td></td>
<td>( M = 6 )</td>
<td>5.977(0.0105)e-03</td>
<td>6.007(0.0118)e-03</td>
</tr>
<tr>
<td></td>
<td>( M = 10 )</td>
<td>5.971(0.0183)e-03</td>
<td>6.014(0.0178)e-03</td>
</tr>
</tbody>
</table>
to the ELMs studied in this channel.

The second kind of channel we considered is nonlinear, and the output is given by: $y_{\text{channel}}(k) = y_{(1)}(k) + D_2 \cdot y_{(2)}(k) + D_3 \cdot y_{(3)}(k) + D_4 \cdot y_{(4)}(k)$, which is very common in digital satellite communications, where $y_{(1)}(k) = 1 + 0.7e^{-k}$ and the gain coefficient $D_2$, $D_3$, and $D_4$ determine the extent of nonlinear distortion caused by the channel to the transmission signal. And the nonlinear nature of the second scenario brings additional difficulties to the equalization task, so that the ability to perform nonlinear mappings represents a decisive element for an adequate recovery of the information. Table 13 and 14 are the performance obtained with $D_2 = 1$, $D_3 = 0.7$, $D_4 = 0.5(H_5(z))$ and $D_2 = 0.6$, $D_3 = 0.5$, $D_4 = 0.4(H_5(z))$ respectively.

As we can observed from Table 12, compared with the classical ELM and RKELM, the proposed networks: PVELM and PVRKELM achieved a better performance in this linear channel, which, again, indicates the advantages of the proposed architecture.

These encouraging observations are clearly corroborated by the results obtained by the networks in the nonlinear scenario, as shown in Table 13 and 14, the proposed architecture reached superior performances for each value of $M$ when compared to ELM and RKELM, specifically, the proposed architectures can achieve the testing AMSE value which is two orders of magnitude smaller than that of their counterparts.

### 5.3.4 The Fourth Group of Experiment

To further demonstrate the practical effectiveness of the proposed architecture, we compared it with a state-of-the-art algorithm: SVM in terms of the training time and testing AMSE on unsupervised equalization. In this part, we only choose PVELM to compare with SVM, because from the previous three groups of experimental results we noticed that there is not much difference among the AMSE values obtained with the PVELM and PVRKELM. Besides, considering the validity of the proposed algorithm, the number of weight of Volterra filter $M$ was set to 6.

From the results in Table 15, it can be observed that with respect to the SVM, PVELM exhibited significantly better equalization performances and incurred a lower training time on all seven channels considered. For example on the typical telephone channel ($H_1(z)$), PVELM took 1.4461 s to achieve the 5.57e-07 testing AMSE while SVM took 4.0383 s to achieve the AMSE of 6.75e-07. This is another evidence of the advantages obtained by employing the nonlinear output layer: Volterra filter structure.

Based on all of these results we obtained, we can conclude that it is effective for ELM to use a nonlinear architecture like the Volterra filter as the output layer which will contribute to a great improvement in performance through further exploration of the information of the hidden layer of ELM. Moreover, with the PCA technique, the improvement was reached without a soaring complexity of the training process, which is another attractive feature associated with the proposal. And more importantly, the results obtained in
unsupervised equalization clearly reveal that the novel ELM architecture is a promising tool to deal with signal processing tasks.

6. Conclusion

In this paper, the core contributions of our work are outlined: (1) we proposed an ELM architecture based on Volterra filtering and PCA. A key characteristic of the architecture is that the linear output layer of extreme learning machine is substituted by the nonlinear filtering structure: Volterra filter and the common data analysis method: PCA, in which the former enables a better exploitation of the higher-order statistics of the hidden layer signals compared with the classical ELM and the latter is employed to avoid the occurrence of an excessive growth in the number of weights to be adapted. (2) We proved that this novel architecture can lead to an excellent generalization performance and efficiency, both on binary classification and multi-classification problems. (3) We extended its application to a more representative scenario of signal processing problems: unsupervised equalization, and characterized it as a promising tool to deal with signal processing tasks. Last but not least, it is worth noting that there still remain many different issues to be considered in future works, like, we can embed this novel architecture into an extended ELM with different hidden layer design methods. Besides, it should be interesting to extend the application of the proposed networks to other signal processing tasks, like separation of convolution mixtures.

Acknowledgements

This work was supported by the Fundamental Research Funds for the Central Universities [lzujbky-2015-109].

References

classification,” Innovations in Information Technology (IIT), 2013
neural predictor for blind equalization in digital communications,”
IEEE Adaptive Systems for Signal Processing, Communications,
de Castro, F.J. Von Zuben, and J.M.T. Romano, “Unsupervised
channel equalization using fuzzy prediction-error filters,” NNSP’03,
[32] G.-B. Huang, “An insight into extreme learning machines: Ran-
dom neurons, random features and kernels,” Cognitive Computation,
[33] Z. Bai, G.-B. Huang, D. Wang, H. Wang, and M.B. Westover,
“Sparse extreme learning machine for classification,” IEEE Trans-
[34] G.H. Golub and C.F. Van Loan, Matrix Computations, 3rd ed., Johns
Hopkins University Press, Baltimore, MD, USA, 1996.
Verlag, New York, NY, USA, 1995.
tor machine classifiers,” Neural Processing Letters, vol.9, no.3,
[37] Y. Zhai, Y.-S. Ong, and I.W. Tsang, “The emerging “big dimen-
sionallity”,” IEEE Computational Intelligence Magazine, vol.9, no.3,
echo state network using Volterra filtering and principal component
[40] N. Godard, “Self-recovering equalization and carrier tracking in
two-dimensional data communication system,” IEEE Transactions
[41] A. Benveniste and M. Goursat, “Blind Equalizers,” IEEE Transac-

**Li Chen** was born in China, in May, 1992. In June, 2015, she received her B.S. degree from the Department of Measurement and Control Technology at Lanzhou University of Technology, Lanzhou, China. Now she is working towards her M.S. degree in the School of Information Science and Engineering at Lanzhou University, Lanzhou, China. Her research interests include extreme learning machine and blind equalization.

**Ling Yang** was born in China in 1966, received the B.S. degree in Radio physics and the M.Sc. degrees in communication and information systems from Lanzhou University, Lanzhou, China, in 1989 and 1999, respectively. She is currently an associate professor of School of information science and Engineering, Lanzhou University, China. Her research interests include digital communication, software radio, and Artificial Intelligence.

**Juan Du** received her Ph.D degree from Beijing University of Posts and Telecommunications, China, in 2006. She is a lecturer at Lanzhou University. Her research interests include wireless communication technology and secret transmission.

**Chao Sun** was born in China, in March, 1992. In June, 2015, he received his B.S. degree from the School of Information Science and Engineering at Shandong Agricultural University, Tai’an, China. He is now working towards his M.S. degree in the School of Information Science and Engineering at Lanzhou University, Lanzhou, China. He researches the blind equalization of MIMO systems currently.

**Shenglei Du** was born in China, in September, 1992. In June, 2015, he received his B.S. degree from the School of Electrical Engineering at Wuhan University, Wuhan, China. He is now working towards his M.S. degree in the School of Information science and Engineering at Lanzhou University, Lanzhou, China. He is dedicated in ANN and machine learning field. His current research interests include channel estimation for low earth satellites.

**Haipeng Xi** was born in China, in November, 1985. In 2008, he achieved his B.S. degree of Communication Engineering at Northwest University, Xi’an, China. He has been engaged in the field of telecommunication since 2008. He is now working towards his M.S. degree in the School of Information Science and Engineering at Lanzhou University, Lanzhou, China. His current research interests is AD hoc networks.