# Extreme Learning Classifier with Deep Concepts

Bernardete Ribeiro[1] and Noel Lopes[1,2]

[1] CISUC - Department of Informatics Engineering, University of Coimbra, Portugal
[2] UDI/IPG - Research Unit, Polytechnic Institute of Guarda, Portugal
bribeiro@dei.uc.pt, noel@ipg.pt

**Abstract.** The text below describes a short introduction to extreme learning machines (ELM) enlightened by new developed applications. It also includes an introduction to deep belief networks (DBN), noticeably tuned into the pattern recognition problems. Essentially, the deep belief networks learn to extract invariant characteristics of an object or, in other words, an DBN shows the ability to simulate how the brain recognizes patterns by the contrastive divergence algorithm. Moreover, it contains a strategy based on both the kernel (and neural) extreme learning of the deep features. Finally, it shows that the DBN-ELM recognition rate is competitive (and often better) than other successful approaches in well-known benchmarks. The results also show that the method is extremely fast when the neural based ELM is used.

**Keywords:** Extreme Learning Machines, Deep learning, Neural Networks.

## 1 Introduction

Since the ability of pattern recognition systems to correctly recognize objects in real time with high accuracy is of primary concern, in this paper, we will consider the performance of machine learning-based systems with respect to classification accuracy. In particular we will focus on neural networks approaches. First, on Extreme Learning Machines (ELM) which are shallow architectures with high potential in regression and classification problems [6]; second, on deep neural networks more precisely on Deep Belief Networks (DBNs) which seek to learn concepts instead of recognizing objects. In fact, motivated by the extreme efficiency of the visual recognition system recent studies in brain science show that this is largely due to the expressive deep architecture employed by human visual cortex systems [14]. Deep architectures transform inputs through multiple layers of nonlinear processing. This nonlinearity is in parametric form such that they can learn deep concepts and be adapted through training. Both methodologies have gained popularity in recent years and many successful applications have been reported [13,11,12,9]. Finally, we empirically show that by designing an extreme learning classifier over deep concepts learned in pattern recognition benchmark problems will enhance the performance as compared to the baseline approaches. More concisely, in both cases of kernel (and neural) based ELM over

J. Ruiz-Shulcloper and G. Sanniti di Baja (Eds.): CIARP 2013, Part I, LNCS 8258, pp. 182–189, 2013.

extracted learned features from deep belief networks, respectively, on Convex, Rectangles and HHreco image datasets the approach is shown to be competitive and extremely fast for the neural based ELM. The paper is organised as follows. Section 2 presents the basic principles of extreme learning machine. Section 3 illustrates the rationale behind deep belief networks. In section 4 we describe the proposal of the extreme learning classifier of deep concepts. In section 5 the experimental set up and the benchmarks are described, and the results discussed. Finally, in section 6 conclusions and future work are presented.

## 2 Extreme Learning Machines

There has been a raising interest in Extreme Learning Machines (ELM) since the original work of Huang et al. [5]. The ELM randomly chooses the hidden-unit weights and analytically determines the output weights of single hidden-layer feedforward network (SLFN). Since then many applications have spread in various fields of pattern recognition. Extreme learning machine is a simple learning algorithm for (SLFN) with attractive properties such as fast learning speed, no need for tuning of parameters, universal function approximation and good generalization [6].

### 2.1 Basic Form of Extreme Learning Network

Suppose we are given $N$ instances of training data. Each instance consists of a $(\mathbf{x}_i, t_i)$ pair where $\mathbf{x}_i \in \mathbb{R}^d$ is a vector containing $d$ attributes of the instance $i$, and $t_i \in \{+1, -1\}$ is the correspondent class label. The method uses input-output training pairs from $\mathcal{D} = \left\{(\mathbf{x}_i, t_i) \in \mathcal{X} \subseteq \mathbb{R}^d \times \mathcal{T} : 1 \leq i \leq N\right\}$ such that the ELM classifies correctly unobserved data $(\mathbf{x}, t)$. In its basic form ELM with $L$ hidden nodes are mathematically modeled as:

$$f_L = \sum_{i=1}^{L} \beta_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{1}$$

where $\boldsymbol{\beta} = [\beta_1, \cdots, \beta_L]^T$ is the output weight vector connecting the hidden nodes and the output node, and $\mathbf{h}(\mathbf{x}) = [h_1, \cdots, h_L]$ is the vector with the outputs of the $L$ hidden nodes with respect to the vector $\mathbf{x}$. The model above $\mathbf{h}(\mathbf{x})$ with $L$ hidden nodes maps the $N$ data samples from the $d$-dimensional input space to the feature hidden space $H$. The ELM minimizes the training errors as well as the norm of the output weights to achieve better generalization [5,6] according to:

$$\text{Minimize:} \sum_{i=1}^{N} ||\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} - \mathbf{t}_i|| \tag{2}$$

The solution is given by:

$$\boldsymbol{\beta} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{3}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$. The singular value decomposition (SVD) method can be used to calculate the generalized Moore-Penrose generalized inverse of matrix $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix}.$$

### 2.2 Kernel Based Extreme Learning Machine

The equality constrained optimization method is proposed in [4] to solve the optimization problem in eq. (2). With the universal approximation capability as shown in [6] this classification problem can be formulated as:

$$\begin{aligned} \text{Minimize: } L_{PELM} &= \frac{1}{2}||\boldsymbol{\beta}||^2 + \lambda\frac{1}{2}\sum_{i=1}^{N}\xi^2 \\ \text{Subject to: } &\mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = t_i - \xi_i \end{aligned} \tag{4}$$

where $\lambda$ is the regularization constant and $\xi$ are the slack variables. By solving the above equations the output of the ELM classifier is (5); if the feature mapping $\mathbf{h}(\mathbf{x})$ is unknown, Mercer's conditions apply [4] and the kernel matrix $\boldsymbol{\Omega}_{ELM}$ can be constructed; the final form is shown in (6):

$$f_L(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{\lambda} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \tag{5}$$

$$= \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}\left(\frac{I}{\lambda} + \boldsymbol{\Omega}_{ELM}\right)^{-1}\mathbf{T} \tag{6}$$

## 3 Deep Belief Networks

DBNs were proposed by Hinton who showed how to carry out unsupervised greedy learning with Contrastive Divergence (CD) [3]. This algorithm learns a generative model from the data distribution. With the proviso that by combining Restricted Boltzmann Machines (RBMs) learning in DBNs is sequentially achieved [3], the approach represents an efficient way of accomplishing tasks that would otherwise be out of reach. Figure 1 illustrates this process.

Each RBM has a layer of visible units $\mathbf{v}$ that represent the data and a layer of hidden units $\mathbf{h}$ that learn to represent features that capture higher-order correlations in the data. Given an energy function $E(\mathbf{v}, \mathbf{h})$ on the whole set of visible and hidden units, the joint probability is $p(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z}$ where $Z$ is a normalizing partition function.

The two layers are connected by a matrix of symmetrically weighted connections, $\mathbf{W}$, and there are no connections within a layer. Given conditionally

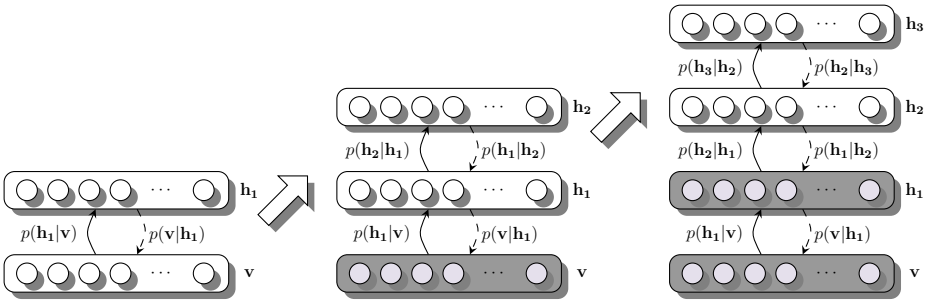


**Fig. 1.** Training process of a Deep Belief Network (DBN) with one input layer, $\mathbf{v}$, and three hidden layers $\mathbf{h_1}$, $\mathbf{h_2}$, $\mathbf{h_3}$. From left to right, trained layers are drawn with lighter color, otherwise layers are with darker color.

independence it is easy to sample from the factorial posterior distribution over hidden vectors $p(\mathbf{h}|\mathbf{v}, \mathbf{W})$ and from the factorial posterior distribution over visible units $p(\mathbf{v}|\mathbf{h}, \mathbf{W})$. By starting with the data vector on the visible units and alternating several times between sampling from $p(\mathbf{h}|\mathbf{v}, \mathbf{W})$ and $p(\mathbf{v}|\mathbf{h}, \mathbf{W})$, it is easy to get the learning weights $\mathbf{W}$.

## 4 Proposed Approach

Most of the problems in pattern recognition fall in the category of classification where objects are represented by a set of features (or attributes) usually extracted manually. Very often the challenging nature of many problems lie on the difficulty of extracting features such as behavioral characteristics like mood, fatigue, energy, etc.. This is a very hard task for manual extraction of features. The unsupervised training of the DBNs allows to learn complex functions by mapping the input to the output directly from data, without depending on human-crafted features [1]. The process works as follows. The first layers are expected to extract low-level features from the input data while the upper layers are expected to gradually refine previously learnt concepts, therefore producing more abstract ones [7]. Now the output of the higher DBN layer can easily be functioning as the input to a supervised classifier [2,10]. The idea is to use an extreme learning machine (ELM) as the classifier of the deep concepts. Notwithstanding the training cost of DBNs, however, our recent work with an adaptive learning rate technique and Graphics Processing Units (GPU)[1] implementation of DBNs [8] has highlighted the way to circumvent these pitfalls which appear to favor (deep) architectures. The inputs to the (shallow) ELM are thus the extracted features from the top DBN layer and its output are the classes of the target pattern problem.

[1] `http://gpumlib.sourceforge.net/`

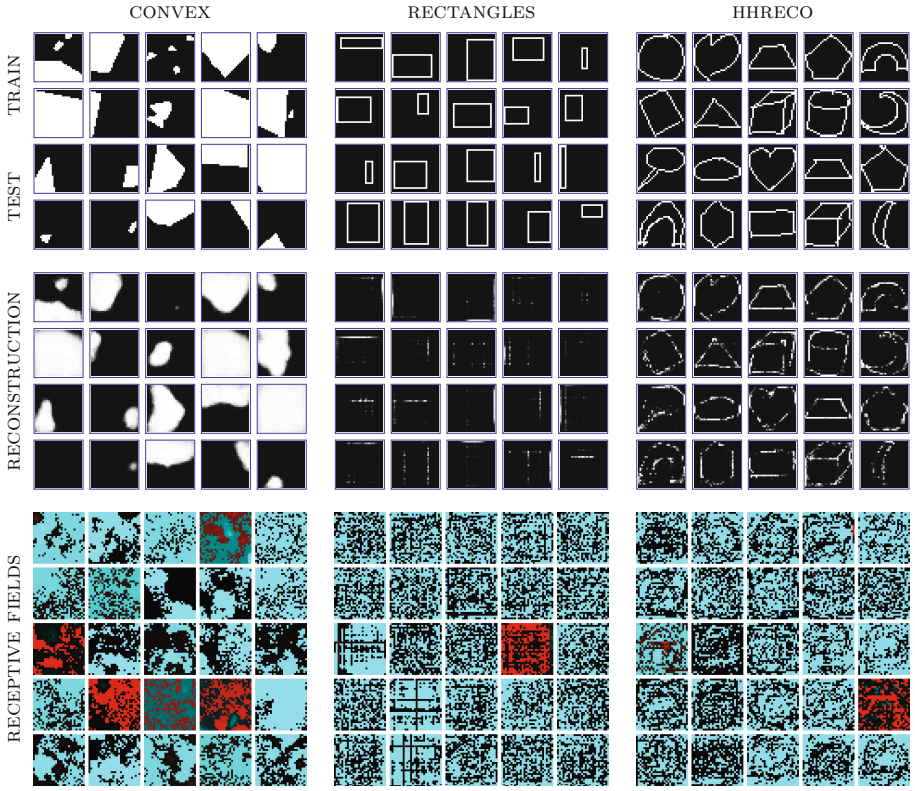


**Fig. 2.** From left to right examples of the Convex, Rectangles and HHreco multi-stroke images datasets. Each square figure contains a symbol while each row contains images of each data set. The first two top rows correspond to training samples of the three data sets w.r.t. the order above (e.g. the upper right first two rows correspond to Convex images); the second two rows correspond to the test samples. The corresponding reconstruction for train and test data is in the middle row-range. The DBNs were trained with two and three layers and the best configuration chosen. The local receptive fields (weights of the hidden neurons) which play an important role in visual tasks are illustrated in the last row-range.

**Table 1.** DBN-ELM and DBN-SVM F1-measure versus the baseline (DBN-MLP)

| **Datasets** | **Sampling** | | | **Methods** | | |
|---|---|---|---|---|---|---|
| **Images** | **Training** | **Testing** | **Classes** | **DBN-SVM** | **DBN-ELM** | **DBN-MLP** |
| **Data** | samples | samples | Nr. | $C = 1$, $\gamma$ | $\lambda = 1$ $\gamma = 0.1$ | F1 $n_H = 100$ |
| Convex | 8000 | 50000 | 2 | **98.32** ($\gamma = 10$) | 77.58 | 73.85 $(10 - 1)$ |
| Rectangles | 1200 | 50000 | 2 | 89.90 ($\gamma = 0.5$) | **92.55** | 91.01 $(10 - 1)$ |
| HHreco | 650 | 7141 | 13 | 88.93 ($\gamma = 0.05$) | **91.33** | 80.37 $(10 - 13)$ |

# 5 Results and Discussion

## 5.1 Experimental Setup and Dataset Benchmarks

In our testbed experiments we have used HHreco images, Convex and Rectangles datasets. Figure 2 presents examples of the three datasets. The first two benchmarks are purely synthetic data sets[2]. The task in Convex is to classify a single white region that appears in each image as convex or non-convex. The task in Rectangles is to classify a single rectangle that appears in each image as tall or wide. Finally, the HHreco database[3], contains a total of $7,791$ samples generated by 19 different persons, and contains a total of 13 different symbol classes. Each user created at least 30 multi-stroke images per class, which means that for each symbol there are at least $19 \times 30 = 570$ samples. We converted the original HHreco vector strokes into a $28 \times 28 = 784$ raster pixel image, maintaining the aspect ratio of the images. Moreover, the resulting images have been binarized and no further pre-processing was done. Since we are interested in evaluating the capacity of the DBNs for extracting information from the original (images) raw data, we discarded both the number of strokes and time span information.

## 5.2 Experiments and Results

The performance of kernel (and neural) based ELM, SVM and MLP classifiers were tested on the output of the DBN which learned well the features for image representation as demonstrated through the reconstruction obtained in the previous step (see Figure 2). For all the datasets the input data have been normalized into $\{-1, +1\}$. Note that the patterns in Rectangles and Convex datasets involve abstract, shape-based features that cannot be computed directly from raw pixel inputs, but rather seem to require many layers of processing. The positive and negative examples also exhibit tremendous variability, making these problems difficult for template-based approaches (e.g., SVMs with RBF kernels). For the kernel based ELM, SVM and MLP classifiers the generalization performance depends on the setting of parameters. Specifically, for SVM classifiers the combination of $C$ cost parameter and $\gamma$ kernel parameter have to be carefully chosen to get the best results. In our simulations, we have carried the parameter selection based on the F1 measure. Its computation for binary class is given by $F\text{-}measure = 2 \times \frac{precision \times recall}{precision + recall}$ where precision and recall are determined from the confusion matrix as the rate of true positives (tp) from all retrieved positives given by the algorithm while the recall gives the rate of tp from all the positives in the dataset. For multi-class, let $tp_c$ be the number of samples belonging to class $c$ that were correctly classified. Let $fp_c$ be the number of samples that were incorrectly classified as being of class $c$ when in fact they belong to a different class ($i \neq c$). Let $fn_c$ be the number of samples that were

[2] `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/PublicDatasets`

[3] `http://embedded.eecs.berkeley.edu/research/hhreco/`

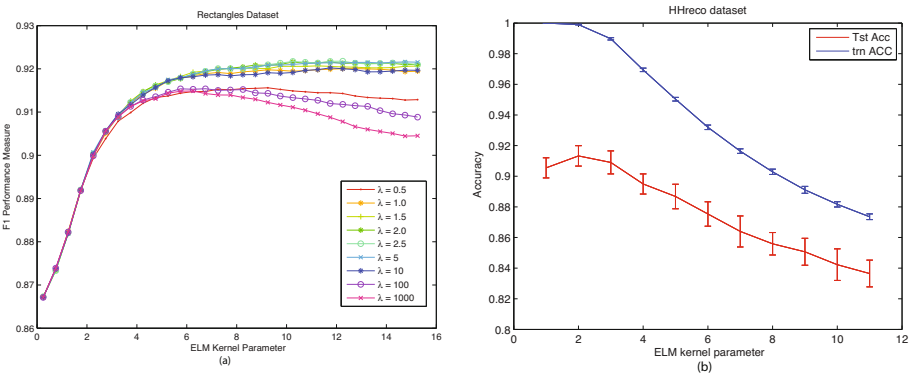


**Fig. 3.** DBN-ELM performance for datasets (a) Rectangles and (b) HHreco. In (a) the various plots for several values of the kernel parameter $\gamma$ and regularization constant $\lambda$ are shown. Notice in (b) the standard deviation for train and test data.

incorrectly classified as belonging to class $i \neq c$ when in fact they belong to class $c$. Assuming that there are $C$ different classes the macro-average precision and macro-average recall can be computed as follows: $precision = \frac{1}{C}\sum_{c=1}^{C}\frac{tp_c}{tp_c+fp_c}$ and $recall = \frac{1}{C}\sum_{c=1}^{C}\frac{tp_c}{tp_c+fn_c}$. The macro-average F-measure can be now computed as indicated above.

For each specific training set we search the optimal cost parameter from the following settings: Parameters $\gamma = \{0.01, 0.05, 0.1, 0.5, 1, 10, 100\}$ e $\lambda, C = \{0.001, 1, 10, 100, 1000, 10000\}$. The kernels chosen for both SVM and kernel-based ELM were RBF and Linear. The results are presented in Table 1[4]. The kernel based ELM can achieve better results than both the SVM and the baseline MLP as highlighted in the Table 1 for the Rectangles and HHreco datasets. However, for the Convex data set the SVM is better while ELM still outperforms MLP. This might be due to an DBN not properly tuned into this difficult dataset and as such it deserves further study. Figure 3 plots the parameters sensitivity for the rectangles and HHreco datasets. For the sake of comparison we also tested neural-based ELM with neurons in the range $\{10, 100, 1000, 5000\}$ and activation functions *sigmoid*, *hardlim* and *radbas* and *sin*. We observed that F1 obtains the best value for *sigmoid* activation function on the convex dataset and decreases, respectively, by 12.6%, 5.19% and 1.98% for *hardlim*, *radbas* and *sin*. Regarding the training and testing times the neural-based ELM network is extremely fast compared to kernel-based ELM (with similar parameters settings) by a factor ca. 270. This is expected since in the former an $L \times L$ matrix must be inverted while in the second the matrix is $N \times N$ where $L$ (number of hidden units) $< N$ (number of data points).

[4] Due to paper space restrictions, larger tables with more results are not presented.

## 6 Conclusions and Future Work

We explored extreme learning machines (ELM) in the classification stage of features constructed by nonlinear processing in deep architectures (DBN) on leading benchmarks. Comparison of the DBN-ELM with previous approaches show that they uphold competitive accuracies. When it comes to training times they are extremely fast when the shallow neural ELM is used. This might be due to the fact that there is no need to tune weights and bias in the final classification step which allows us to harness the advantages of extreme learning of deep concepts in visual tasks. While the paper focus on extreme learning classifier with deep concepts for model construction, future work envisaging hybrid methods will be pursued to deal with unbalanced data and their geometric distribution in the feature space.

## References

1. Bengio, Y.: Learning deep architectures for AI. Foundations and Trends in Machine Learning 2(1), 1–127 (2009)
2. Hinton, G.: A practical guide to training Restricted Boltzmann Machines. Tech. rep., Dep. of Computer Science, University of Toronto (2010)
3. Hinton, G.E., Osindero, S., Teh, Y.-W.: A fast learning algorithm for deep belief nets. Neural Comput. 18, 1527–1554 (2006)
4. Huang, G.-B., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B 42(2), 513–529 (2012)
5. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: A new learning scheme of feedforward neural networks. In: IEEE International Joint Conference on Neural Networks, pp. 985–990 (2004)
6. Huang, G.-B., Zhu, Q.-Y., Siew, C.-K.: Extreme learning machine: Theory and applications. Neurocomputing 70, 489–501 (2006)
7. Le Roux, N., Bengio, Y.: Representational power of restricted boltzmann machines and deep belief networks. Neural Comput. 20, 1631–1649 (2008)
8. Lopes, N., Ribeiro, B.: GPUMLib: An efficient open-source GPU machine learning library. International Journal of Computer Information Systems and Industrial Management Applications 3, 355–362 (2011)
9. Lu, B., Wang, G., Yuan, Y., Han, D.: Semantic concept detection for video based on extreme learning machine. Neurocomputing 102, 176–183 (2013)
10. Ranzato, M., Boureau, Y., LeCun, Y.: Sparse feature learning for deep belief networks. In: Advances in Neural Information Processing Systems, vol. 20 (2007)
11. Shi, L.C., Lu, B.L.: EEG-based vigilance estimation using extreme learning machines. Neurocomputing 102, 135–143 (2013)
12. Wu, S., Wang, Y., Cheng, S.: Extreme learning machine based wind speed estimation and sensorless control for wind turbine power generation system. Neurocomputing 102, 163–175 (2013)
13. Yeu, C.W., Lim, M.H., Huang, G.B., Agarwal, A., Ong, Y.S.: A new machine learning paradigm for terrain reconstruction. IEEE Geoscience and Remote Sensing Letters 3(3), 382–386 (2006)
14. Yu, K., Xu, W., Gong, Y.: Deep learning with kernel regularization for visual recognition. In: Neural Information Processing Systems, NIPS (2009)