Transactions on Machine Learning and Artificial Intelligence - Vol. 10, No. 4
Publication Date: August, 25, 2022
DOI:10.14738/tmlai.104.13100.
De Groff, D., & Neelakanta, P. S. (2022). A Novel Deep Learning ANN Supported on Langevin-Neelakanta Machine. Transactions on Machine Learning and Artificial Intelligence, 10(4). 27-45.

A Novel Deep Learning ANN Supported on Langevin-Neelakanta Machine

Dolores De Groff
Department of Electrical Engineering & Computer Science
Florida Atlantic University, Boca Raton, Fl 33431, United States

Perambur S. Neelakanta
Department of Electrical Engineering & Computer Science
Florida Atlantic University, Boca Raton, Fl 33431, United States

ABSTRACT

In the contexts of deep learning (DL) considered in artificial intelligence (AI) efforts, relevant machine learning (ML) algorithms adopted refer to using a class of deep artificial neural network (ANN) that supports a learning process exercised with an enormous set of input data (labeled and/or unlabeled) so to predict at the output details on accurate features of labeled data present in the input data set. In the present study, a deep ANN is proposed thereof conceived with certain novel considerations: The proposed deep architecture consists of a large number of consequently placed structures of paired-layers. Each layer hosts identical number of neuronal units for computation and the neuronal units are massively interconnected across the entire network. Further, each paired-layer is independently subjected to unsupervised learning (USL). Hence, commencing from the input layer-pair, the excitatory (input) data supplied flows across the interconnected neurons of paired layers, terminating eventually at the final pair of layers, where the output is recovered. That is, the converged neuronal states at any given pair is iteratively passed on to the next pair and so on. The USL suite involves collectively gathering the details of neural information across a pair of the layers constituting the network. This summed data is then limited with a specific choice of a squashing (sigmoidal) function; and, the resulting scaled value is used to adjust the coefficients of interconnection weights seeking a convergence criterion. The associated learning rate on weight adjustment is uniquely designed to facilitate fast learning towards convergence. The unique aspects of deep learning proposed here refer to: (i) Deducing the learning coefficient with a compatible algorithm so as to realize a fast convergence; and, (ii) the adopted sigmoidal function in the USL loop conforms to the heuristics of the so-called Langevin-Neelakanta machine. The paper describes the proposed deep ANN architecture with necessary details on structural considerations, sigmoidal selection, prescribing required learning rate and operational (training and predictive phase) routines. Results are furnished to demonstrate the performance efficacy of the test ANN.

Keywords: Deep learning, Machine learning, Artificial Intelligence, Artificial neural network, Logistic learning, Langevin-Neelakanta machine**.**

INTRODUCTION

In the contexts of artificial intelligence (AI), the scope of the present study is focused to consider deep learning (DL) schemes exercised using deep neural learning (or deep neural networking) algorithms adopted with deep ANN architectures. As well known, the art of DL is a subset of machine learning (ML), which refers to the process wherein the computers are rendered to learn with a labeled and/or unlabeled set of training input data and hence make adjusted predictions on the details of the supplied data. A relevant approach of DL involves using massively interconnected switches (computational neural units) constituting the general notions of artificial neural network (ANN), which can be designed in the image of real neural complex to learn and recognize patterns akin to the functions of a human brain and nervous system [1]-[3]. Relevantly, conceiving the deep ANN is inspired by the neural cognitive structure of enormous complexity framed in the image of human brain and associated massive neural connectivity.

Present study refers to using a novel ANN developed exclusively to simulate neural computations consistent with the paradigm of DL exercised towards framing an AI domain wherein, the human-like intelligent behavior is conceivably exhibited by a machine or a system in extracting pattern details pertinent to a variety of complex entities such as pictures, texts, sounds, etc. In order to produce accurate and adjusted insights in such predictions, the focused approach of DL is to simulate the AI ambient to learn and process the information *via* deep ANN.

The paper is organized as follows: Commensurate with the background considerations on AI and DL, the motivational considerations on proposed deep ANN architecture are presented in the present introduction section (Section 1). Next section (Section 2) describes how the test deep ANN is conceived *vis-a-vis* the notions of ANN with certain novel changes and *ad hoc* additions necessarily introduced in conformance with unsupervised learning (USL) suite towards compatible feature extraction and classification exercises involved. Section 3 is an annotative note narrating the operational details of the proposed deep ANN and pertinent heuristics are indicated. The architecture of the test deep ANN used in simulation studies is illustratively explained in Section 4. Following the results and discussion outlined in Section 5, concluding remarks are presented in Section 6.

In all, the proposed study refers to developing a deep ANN architecture for use in DL contexts with certain novel details on its structural considerations, unsupervised training schedule, deciding learning rate, adjusting the weights of neural interconnections with a coefficient scaled by an appropriate sigmoidal squashing function and complete operational routines towards predicting and recovering the required pattern in conformance with the input data.

DEEP ANN: WHAT IS IT?

Presented in this section are basic considerations defining a deep ANN and how it can be conceived *vis-a-vis* the notions of traditional ANN. As an implied objective of this study, certain novel changes and *ad hoc* additions necessarily introduced on the deep ANN are narrated in conformance with unsupervised learning (USL) suite adopted towards compatible feature extraction and classification exercises in the contexts of deep learning (DL) considered in artificial intelligence (AI) efforts, exercised with an enormous set of input data (labeled and/or unlabeled) so as to predict accurate features of labeled data present in the input data set.

Essentially, as well known an ANN refers to a mathematical model evolved as a neuromorphic computational tool based on the image of biological neural complex as elaborated by Neelakanta and De Groff in [1][2]. There are several versions of ANN in vogue and a compatible structure can be chosen and modified on *ad hoc* basis to form the deep learning architecture useful in DL contexts. The architecture pursued thereof in this study is similar to neocognitron heuristics proposed by Fukushima in 1980 [4]. The "neocognitron" network is a refined scheme of a self-organizing multilayered neural network called "cognitron" developed by Fukushima in 1975 [5]. Relevant network is self-organized by "learning without a teacher", and acquires an ability to recognize the stimulus patterns based on any labeled (earmarked) similarity. Its underlying Hebbian learning of a real neural complex conforms to "the synapse from neuron *x* to neuron *y* is reinforced when *x* fires provided that no neuron in the vicinity of *y* is firing stronger than *y*". Hence, in cognitron (as well as, in neocognitron), the self-organization favorably progresses layer-by-layer without having a "teacher" to instruct in all particulars how the individual neural unit in each layer responds. In big-data contexts, the network becomes self-organized such that, the receptive fields of the neural computational units/cells become relatively larger in the deeper sets of layers; and, after repetitive presentations of several stimulus/excitatory patterns, each neural unit in any layer integrates the information from whole parts of the previous layer and selectively responds to a specific stimulus pattern or a feature in the output.

Proposed deep ANN architecture

In the present study a novel deep ANN is conceived as illustrated in Figure 1. It is a hierarchical network, which comprises even number of several layers: (L_1, L_2, …, L_k; k is an even number).

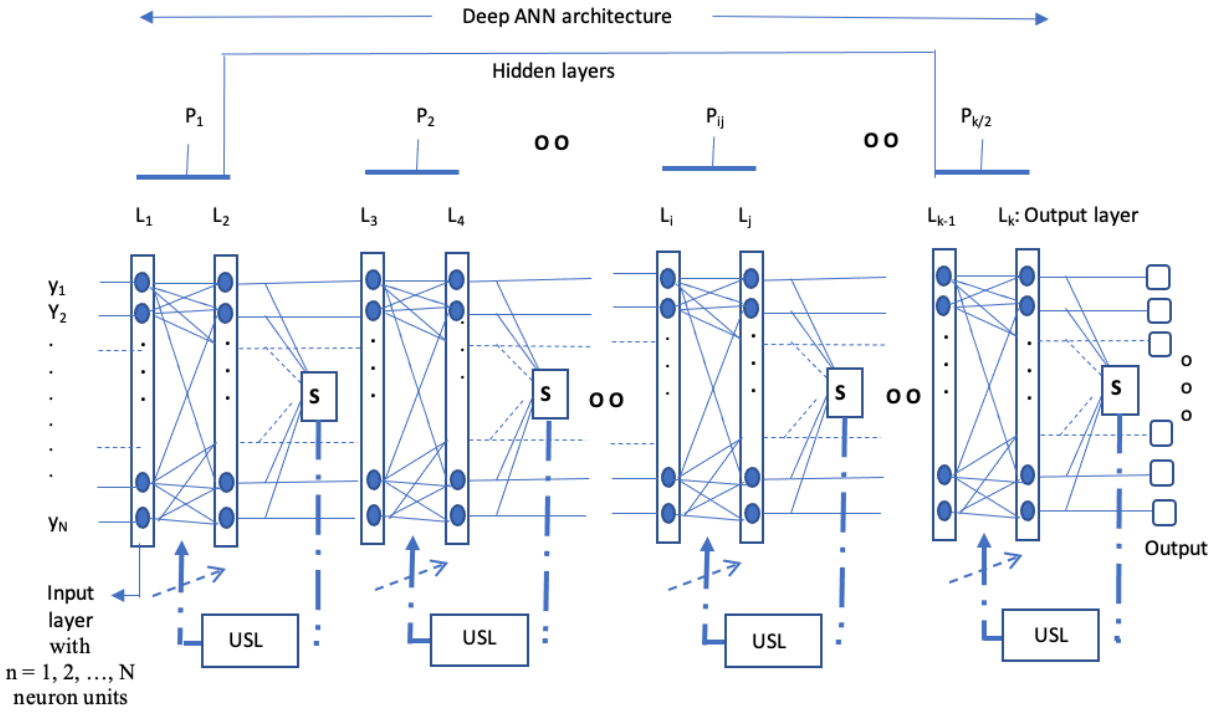


Figure 1: Proposed deep ANN architecture constructed with k layers with each layer having N neuronal units: One input layer, one output layer and (k – 2) hidden layers. The set of k layers are paired into k/2 pairs, each pair denoted as P_1, P_2, …, P_{ij}, …, $P_{k/2}$)

URL: http://dx.doi.org/10.14738/tmlai.104.13100

Each of the layers is housed with identical number of a specified set of neuronal (computational) units. Further, the layers are paired into forming the set, $\{P_1, P_2, \ldots, P_{k/2}\}$ as shown; and, there is a pattern of connectivity locally implied in each pair of layers. In all, the network (as shown in Figure 1,) is divided into connected set of paired-layers. Considering any given $(ij)^{th}$ pair, when the neuronal units in its first layer (i^{th} layer) is addressed with a particular pattern, the neurons of the succeeding interconnected layer (j^{th} layer) sums up the contents and prepares the summed-up details towards a converged self-organization. (The layers L_i and L_j correspond to S- and C-cells of neocognitron [4]). This self-organization is accomplished *via* unsupervised learning (USL) briefly outlined below with reference to the illustrative details of Figure 1 and 2.

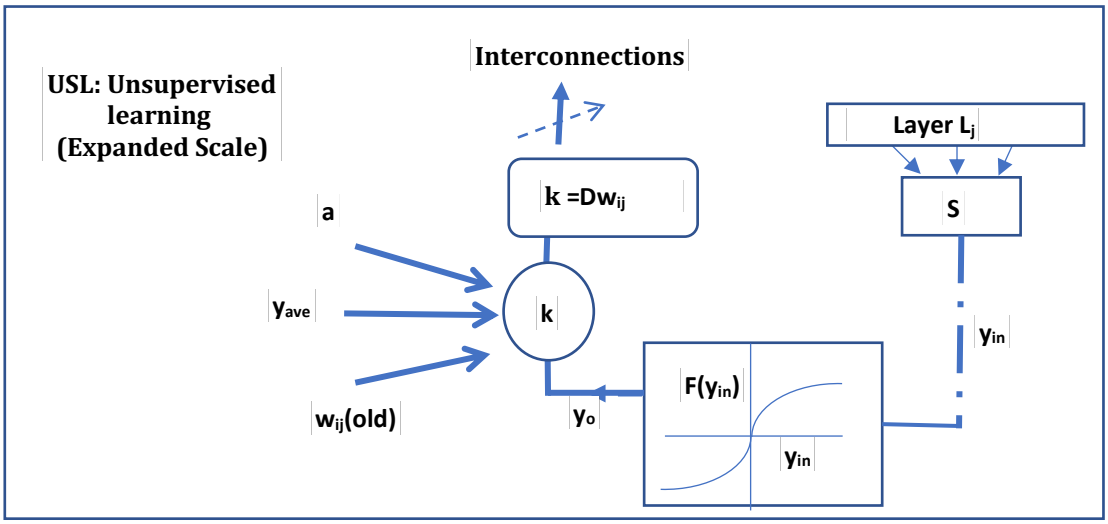


Figure 2 An expanded illustration of USL

Focusing on any $(ij)^{th}$ pair of layers with interconnected neurons identified as P_{ij}: $L_i \leftrightarrow L_j$ in Figure 1, the state vectors of the associated set of neurons, $\{n_1, n_2, \ldots, N\}$ in each layer can be represented as, $\{(z_i)_1, (z_i)_2, \ldots, (z_i)_N\}$ for the layer L_i and $\{(z_j)_1, (z_j)_2, \ldots, (z_j)_N\}$ for the layer L_j; further, the N-components of neuronal states are interconnected by the set of weight states $\{w_{ij}\}$. Hence, the following summation, y_{in} can be specified:

$$y_{in} = \sum_{n=1}^{N} [w_{ij}]_n \times z_n \qquad (1)$$

The summed-up details of equation (1) is then adopted in USL towards realizing the converged state of self-organization in the pair, P_{ij}: $L_i \leftrightarrow L_j$ as shown in Figure 2. Summary of the tasks involved are as follows (and more details will be furnished in later sections: The summed entity y_{in} is next subjected to a nonlinear activation with bipolar limits of (−1 and +1). That is, y_{in} is squashed with a sigmoidal function, $F(y_{in})$ to yield an output $y_o = F(y_{in})$. The next step involved in the USL refers to building a prorated value $(k)_{ij}$ equal to: $\alpha \times [y_{ave} - y_o \times w_{ij}(old)]$.
Here, y_{ave} is the mean value of the excitatory inputs given by:

$$y_{ave} = \left(\frac{1}{N}\right) \sum_{n=1}^{N} y_n \qquad (2)$$

The scaling constant $(k)_{ij}$ refers to the change in the value of w_{ij}, (expressed as $\pm \Delta w_{ij}$) applied iteratively to the interconnections, so that the existing weight, w_{ij} (old) modifies to a new value, w_{ij} (new) until y_{in} converges to a minimum value.

In short, $[w_{ij}\ (new) - w_{ij}\ (old)] = [k = \pm \Delta w_{ij}]$; and, this change is governed by the cybernetics of self-organization pivoted on y_{ave} seeking the goal of convergence or global minimum observed as $[y_{in}]_{min}$. And, the coefficient α defines the learning rate, which can be optimally chosen as will be explained later.

OPERATIONAL DETAILS OF THE PROPOSED DEEP ANN

With reference to the architectural details of deep ANN in Figures 1 and 2, the steps of operation can be stated as a sequential preprocessing of raw, excitatory input data, which is adopted to train progressively (k/2) pairs of layers, $\{P_1, P_2, \ldots, P_{k/2}\}$; and. with each pair is subjected to USL towards a specified goal of convergence. The USL performed independently on each unit of the paired-layer set $\{[P_1], [P_2], \ldots, [P_{k/2}]\}$ refers to a machine learning process of inferring the hidden patterns from a set of some historical data addressed as input to a paired set of layers; hence, the learning model tries to find any similarities, differences, patterns, and structure in the data by itself. No prior human intervention is needed [6] [7].

Commencing from the input pair of layers of the multilayer deep architecture, any (input) excitatory data supplied at the first input layer L_1 progresses forward across interconnected neurons of paired hidden layers; eventually, the output is observed as a converged map on the N neurons at the final layer, L_k pertinent to the input details. Relevant output enables prediction classification on labeled and unlabeled constituents of input stimulus/excitatory input. That is, the required features in the stimulus input data are extracted at the terminal pair of layers of the deep architecture as the resulting converged map of predictive details on labeled and unlabeled contents.

The algorithm on unsupervised training pursued in the proposed deep ANN is based on two analytical considerations:
(i) The first one refers to formulating a matrix of interconnection weights across identical number of N neuronal units present in the adjacent pair of layers, L_i and L_j framing the unit, $P_{i,j.}$ The [N × N] matrix relevant to the layers, L_i and L_j corresponds to the set of weights $\{w_{ij}\}$. Further, inasmuch as this matrix is symmetric, it can be specified in terms of a diagonal set of associated eigen values (λ_{ij}). The learning coefficient (α) is then specified in terms of (λ_{ij}) towards achieving fast convergence. In all, the associated multidimensional Hessian of the interconnection matrix has eigenvalues denoting a measure of steepness of a surface of convergence with iteratively adjusted values of the weights. A large eigenvalue would signify a steep curvature implicitly depicting a small learning rate towards convergence of the said error reaching a minimum value. That is, the learning rate should be inversely proportional to the eigenvalue. The learning rate is indicated before as a part of scaling factor k prescribed as the weight adjustment parameter. In summary, prescribing an appropriate learning rate (α) by formulating a matrix of interconnection weights $\{w_{ij}\}$ across identical number of N neuronal units present in the adjacent pair of layers, L_i and L_j (framing the unit, $P_{i,j}$); and, this interconnection matrix $[N \times N]_{ij}$ relevant to the layers L_i and L_j being symmetric, it is specified in terms of a diagonal set of associated eigen values $\{\lambda_{ij}\}$. The learning rate (α) is then linked to λ_{ij}.

(ii) The second part of USL algorithm refers to specifying an appropriate squashing function, $F(y_{in})$ on the collective details (y_{in}) of neural unit information gathered across the pair, (i, j) as

URL: http://dx.doi.org/10.14738/tmlai.104.13100

shown in Figure 1. The squashing function proposed thereof conforms to the heuristics of the so-called Langevin-Neelakanta machine [1]-[3]. In summary, the summed value y_{in} in Figure 1, corresponds to ensemble of neural information at the layer, L_j; and, the proposed scheme involves prescribing a squashing transfer function, $F(y_{in})$ on the collective neural information gathered at the layer, L_j so that the resulting output (y_o) remains limited (typically between ± 1). Conventionally, a simple hyperbolic tangent (sigmoidal) function [1] is adopted as the squashing function; however, in the present study, the sigmoidal function chosen conforms to the heuristics of the so-called Langevin-Neelakanta machine [1]-[3] as will be detailed later.

In the following subsections, details on aforesaid algorithms are explained.

Learning rate (α) algorithm

As indicated above, the learning/training schedule pursued conforms to unsupervised (Hebbian) learning (USL) applied at each unit of interconnected pair of layers. Considering any such paired-layer, P_{ij}: ($L_i \leftrightarrow L_j$), the weighted sum of neural information gathered from the set of neurons, {n_1, n_2, …, N} in layer L_j is given by equation (1) and denoted as y_{in} in Figure 1. It is then limited or compressed to a value, y_o by a sigmoidal function, F(.); hence, $y_o = F(y_{in})$, which is used to construct a scaling constant, $(k)_{ij}$ equal to: $\alpha \times [y_{ave} - y_o \times w_{ij}(old)]$ where y_{ave} is the mean value of the excitatory inputs given by equation (2). As stated before, this scaling constant $(k)_{ij}$ refers to $\pm \Delta w_{ij}$ denoting the differential change in the value of w_{ij} applied iteratively to the interconnections, until y_{in} converges to a minimum value. And, the coefficient α defines the learning rate, which can be optimally chosen as follows: Proposed thereof is a novel fast-convergence algorithm towards establishing the learning rate (α) based on eigenvalues of the associated Hessian matrix of interconnection weights between N neurons in the pair of layers, ($L_i \leftrightarrow L_j$) [8] [9]. The learning rate so adopted varies dynamically with the input data used for training and enables fast-convergence of learning at the USL.

In any iteration cycle of the USL, considering the neuronal states {z_1,z_2,…, z_N} at the layer L_i, the associated multidimensional interconnection mentioned earlier is a Hessian matrix (corresponding to an N × N square matrix); and this Hessian matrix can be put into a diagonal form, denoted as [**HD**]. Because of the symmetry of the Hessian matrix, it has a unique, single eigenvalue, λ_{HD} in the diagonal form as shown below (with all other eigenvalues being zero):

$$[\mathbf{HD}] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda_{HD} \end{bmatrix} \quad (3)$$

Hence, in each iteration of USL schedule, the differential change in w_{ij} (expressed as $\pm \Delta w_{ij}$) exercised on the interconnections enables a corresponding error function $\varepsilon(w_{ij})$ deciding the value of y_{in}. It is given by:

$$\varepsilon(w_{ij}) = \sum_{n=1}^{N} \left[(z_n)_i - \lambda_{ij} \times F(w_{ij} \times (z_n)_j \right] \quad (4)$$

and its corresponding derivative, $\frac{\partial \varepsilon(w_{ij})}{\partial (w_{ij})}$ can be determined as follows:

$$\frac{\partial \varepsilon(w_{ij})}{\partial (w_{ij})} = \sum_{n=1}^{N}\left[(z_n)_i - \lambda_{ij} \times F(w_{ij} \times (z_n)_j\right] \times [F'(w_{ij} \times (z_n)_j] \times (z_n)_j \qquad (5)$$

The differential change in w_{ij} (namely $\pm \Delta w_{ij}$) equal to: $\alpha \times [y_{ave} - y_o \times w_{ij}(old)]$ implicitly denotes the parameter of delta-rule training schedule imposed on w_{ij}; and, equation (5), explicitly denotes the measurable gradient value of the error and it can be rewritten as follows:

$$\frac{\partial \varepsilon(w_{ij})}{\partial (w_{ij})} = \pm \left[\rho + \left([\mathbf{H}] \times w_{ij}\right)\right] \qquad (6)$$

where $\rho = \frac{1}{N}$, $H = \frac{1}{N}\sum_{n=1}^{N} (z_n)_i (z_n)_j$ and $[\mathbf{H}]$ is the Hessian matrix denoting the average over all inputs of $(z)_i (z)_i^T$, (where $(z)_i^T$ is the transpose of $(z)_i$.

The derivative of $\varepsilon(.)$ tends towards zero at the global minimum value of $\varepsilon(.)$ indicating the convergence of USL. Further, in terms of ρ and $[\mathbf{H}]$ in equation (6), one can solve for the optimum weight,

$$w_{opt} = \frac{\rho}{[\mathbf{H}]} \qquad (7)$$

That is, the optimal learning rate that would enable the convergence is equal to the inverse Hessian matrix, which signifies the shape of the cost surface involved in the optimization suite; and, the decomposed eigenvalues of $[\mathbf{H}]$ would correspond to a measure of the steepness of the surface along the curvature directions of the cost surface. A large eigenvalue would signify a steep curvature leading to a small learning rate required. In other words, learning rate should be proportional to: [1/(eigenvalues of H)]. Thus, a single learning rate prescribed for all the weights, the learning rate can be chosen, such that it will not cause divergence along the steep directions (namely, the large eigenvalue directions). Hence, a learning rate can be chosen of the order of $1/\lambda_{HD}$ where λ_{HD} depicts the largest eigenvalue of the Hessian matrix.

In summary, the unsupervised training proposed here for the test deep ANN architecture is based on specifying a Hessian matrix of the relevant data applied on the input neurons to the $(L_i)^{th}$ layer interconnected to equal number of neurons of the paired layer, L_j (in Figure 1). Thus, in the proposed model, the learning-rate (α) applied on the test deep ANN correspond to the largest, single eigenvalue, λ_{HD} of the Hessian matrix as above so that, a fast convergence towards a desired extent of output prediction at the pair of layers L_i and L_j is made feasible *via* USL pursued.

PRESCRIBING A SQUASHING FUNCTION

Pertinent details on the choice of the sigmoid towards squashing interposed in the USL loop as indicated earlier is outlined here. The weight modification implied across the neural units $(x_n)_i$ of layer L_i and neural units $(x_n)_j$ of layer L_j corresponds to the differential change in the error on y_{in} as specified by equation (5) where F(.) denotes a squashing function, which should conform to the limiting extents of nonlinearity of neural interaction across interconnected neural units as described in [2]; hence, it is a monotonic differentiable function; and, its prescription in the USL loop (in conjunction with the learning rate chosen) should lead to minimizing the cost-function, ε imposed on y_{in}.

URL: http://dx.doi.org/10.14738/tmlai.104.13100

Typically, the underlying data and the resulting values of the weights in the iterations of USL schedule are mostly random entities. Relevantly, it is indicated in [2] that the associated F(.) is a solution of a stochastic Bernoulli-Riccati differential equation developed in the context of nonlinear input-output response of neural interaction. That is, pertinent dynamics refers to the interacting neurons being regarded as a stochastic system of dipole activity. Hence, Neelakanta et al. [3] extended Langevin's theory of dipole polarization and obtained the following explicit function for, F(.):

$$F(X) \equiv L_Q(X) = (1 + 1/2Q) \times \{\coth[(1 + 1/2Q) \times X]\} - (1/2Q) \times \{\coth[(1/2Q) \times X]\} \quad (8)$$

where the parameter, Q denotes the stochastic state of the interacting entities bounded by the limits, ($\frac{1}{2} \leq Q < \infty$). The lower bound on $Q = 1/2$ depicts a totally anisotropic state of statistically interacting units; and, the other limit $Q \rightarrow \infty$ implies a totally-isotropic state of stochastic interactions involved. Hence, in the contexts of ANN, the configuration of neural interaction of disorder states can be characterized in the bounding limits of ($\frac{1}{2} \leq Q < \infty$) as outlined and summarized in Table 5.3 of [1]. Usage of $L_Q(.)$ as ANN-implied sigmoid leads to Langevin machine concept as detailed in [1]-[3]. (Framing neuromorphic functioning *via* Langevin dynamics and hence, a neural network based on stochastically justifiable sigmoidal function was proposed in [1] and relevant considerations are extended to discrete Langevin machine in [3]). In summary, as regard to deep ANN study pursued currently, it is proposed to adopt the aforesaid $L_Q(y_{in})$ sigmoidal squashing in the USL loop of Figure 1.

TEST DEEP ANN FOR SIMULATION STUDIES

The efficacy of the proposed deep ANN method is cross-verified against details of simulated results obtained with a test architecture illustrated in Figure 3. Relevant architectural details, data adopted towards simulations and for cross-verification towards predictions and operational routines are described in the following sections:

The test architecture of deep ANN shown in Figure 3 is consistent with the general structure depicted in Figure 1 and constructed with 6 layers as shown: One input layer, one output layer and four hidden layers. Each layer in the network of Figure 2 has seven neuronal units and the set of six layers are paired into three pairs represented as: P_1: ($L_1 \leftrightarrow L_2$); P_2: ($L_3 \leftrightarrow L_4$) and P_3: ($L_5 \leftrightarrow L_6$). The implied USL is exercised on each of the three pairs and illustrated in expanded scale in Figure 2.

Simulations performed on the test network and the details of data used in the training and prediction phases, are as follows: As well known, the ANN has the ability to learn or get trained from experience (or from the existing data). In general, the exhaustively known details adopted for ANN training would conform to a big-data universe wherein the information/pattern to be predicted remains obscured as an admixture of random constituents of labeled and unlabeled in the stochastic framework of large data. The learning/training exercise imposed on the test ANN as seen earlier, involves adjustment of weight coefficients pertinent to massively interconnected neuronal units; and, the adjustment is based on either by capturing the randomness (stochastic aspects) of the stimuli (depicting the input training data) and/or by using information present in a set of tagged (labeled and unlabeled) classification flags present in the input data. Eventually, by virtue of trained details, the test ANN is rendered to predict the desired pattern or labeled details in a given set of data. Further, it will be shown (in a later

section) that a fast convergence towards accurate prediction is realized with the learning rate (α) in the USL routine implemented.

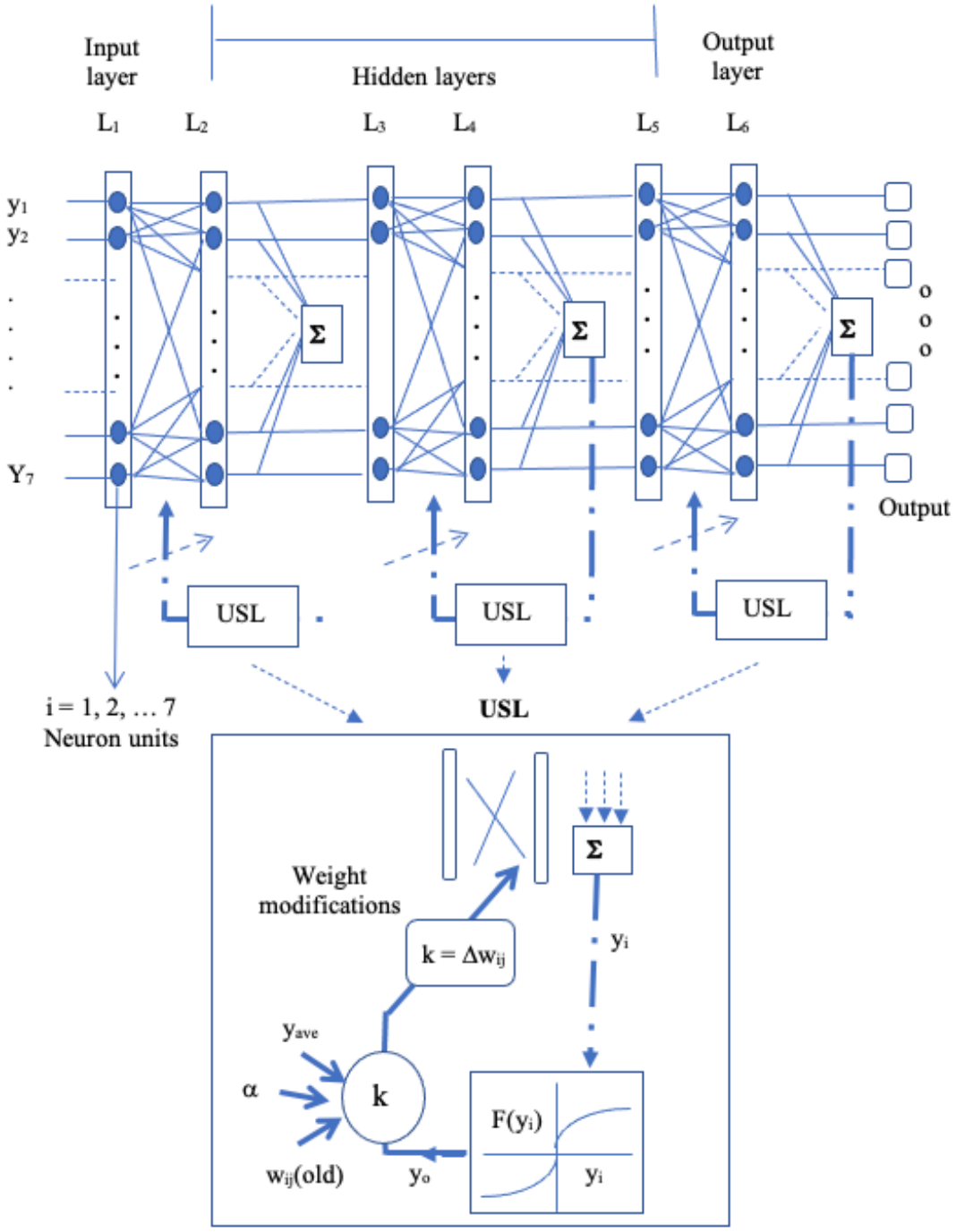


Figure 3: Deep ANN architecture adopted in simulation studies. (Note: $k = \alpha \times [y_{ave} - y_o \times w_{12}(old)]$).

URL: http://dx.doi.org/10.14738/tmlai.104.13100

In all, objectively the tasks in deep neural networking imply processing a set of raw input data so as to extract required features or the desired/labeled data from it. The extracted details are based on the inherent capability of classification in an arbitrary input-output mapping and specify a predictive classification of the desired data at the output. That is, the ANN is trained to classify the labeled and unlabeled data supplied as the input and spot the labeled pattern as predictive output. Relevant training scheme in the proposed deep ANN simulation is based on unsupervised learning (USL) method indicated earlier in Figures 1 and 2.

Consistent with the algorithmic suites described earlier for the test deep ANN, relevant simulation studies indicated here conform to a set of the input data $\{y_1, y_2, \ldots, y_7\}$ addressed at the input layer and constructing a corresponding transpose $[y_{1,} y_{2,\ldots,} y_7]^T$; hence, a [7 × 7] Hessian matrix [**H**] is specified as follows: [**HD**] = $[y_{1,} y_{2,\ldots,} y_7]^T$. $[y_{1,} y_{2,\ldots,} y_7]$. The input data set used in the simulations consists of normalized values attributed to seven observed data pertinent to an arbitrary, random set of epochs. Corresponding [**HD**] refers to a symmetric square [7 × 7] matrix yielding one non-vanishing eigenvalue as a diagonal element. That is, for a given ensemble of input data set there is one non-vanishing eigenvalue (λ_{ij}) for the pair of interconnected layers L_i and L_j. The learning coefficient adopted towards weight adjustments is linked to this eigenvalue as will be indicated later. For training the test ANN of Figure 3, four sets of independent data, (each set consisting of six unlabeled details plus one labeled data) are gathered from an observed universe of random epochs as listed in Table 1. While the unlabeled entries refer to stochastic details of the random epochs, the labeled entity, marked (bold) as X is a deterministic information on the pattern or desired details of focused relevance in the stochastic occurrence of epochs. The discrete value of X can be addressed randomly at any one of the input neurons.

Table 1: Four sets {A, B, C, D} of independent input data sets, {y_1, y_2, ..., y_7} of randomly mixed unlabeled and labeled entities gathered from observed random epochs

Input data sets, {**A, B, C, D**} each of {y_1, y_2, ..., y_7} containing a labeled data **X** addressed randomly at any of the input neurons, n: 1, 2, ..., 7			Computed Hessian eigenvalues, $\lambda_{7\times7}$ for the ensemble data sets: {**A, B, C, D**}
n	**Set A**		
1	y_1	0.149655050	3.1845
2	y_2	0.174597559	
3	y_3	0.644897959	
4	y_4	0.523615779	
5	y_5	0.810366177	
6	**X**	**1.000000000**	
7	y_6	0.860958783	
n	**Set B**		
1	y_1	0.481426601	6.1747
2	y_2	0.528792601	
3	y_3	0.572735169	
4	y_4	0.690153799	
5	y_5	0.732643780	
6	y_6	0.962546625	
7	**X**	**1.000000000**	
n	**Set C**		
1	y_1	0.799630824	4.9099
2	y_2	0.672045659	
3	y_3	0.729379726	
4	y_4	0.780542921	
5	y_5	0.881102066	
6	y_6	0.949358287	
7	**X**	**1.000000000**	
n	**Set D**		
1	y_1	0.755463639	3.7790
2	y_2	0.927014579	
3	y_3	0.993780985	
4	y_4	0.948810467	
5	y_5	0.995805098	
6	**X**	**1.000000000**	
7	y_7	0.930119896	

Simulation studies on the test ANN: A pseudocode

Training phase

%% Reference to the test ANN (Figure 3), the training schedule is applied to each paired set and progresses from the input layer, L_1 to output layer, L_6. Considering any pair of layers L_i and L_j, the training computation involves algorithms pertinent to the adjustment of interconnection

URL: http://dx.doi.org/10.14738/tmlai.104.13100

weights; and, the USL performed thereof frames a converged set of weights across each paired layer that are kept stored.

With reference to the test ANN (Figure 3), a training data set is applied to the input layer L_1 interconnected with L_2. Training involves adjustment of interconnection weights between L_1 and L_2 *via* USL schedule until convergence is observed as a minimum value of summed entity y_{in}; and, the converged set of weights $\{w_{12}\}$ are kept frozen. Further, the converged details/states of the layer, L_2 is conveyed as inputs to the neuronal units of the next succeeding layer L_3 of the hidden pair $[L_3, L_4]$. That is, the converged states of neural unit from the second layer (L_2) are passed to the first layer (L_3) of the next pair to be trained, and so on. Likewise, with the convergence of the pair, P_{34}:(L_3, L_4) *via* USL, the set of weights $\{w_{34}\}$ are kept stored and the converged neural states in the layer L_4 is passed on as inputs to L_5. Hence, the pair P_{56}:(L_5, L_6) is trained *via* USL leading to the converged set of weights $\{w_{56}\}$ that are kept stored; and, corresponding converged neural states in the layer L_6 are availed as the out being sought. Exercising the training algorithm as above at a paired set of layers is outlined below:

Initialize

Input

→ Considering the input layer of the test ANN, the training data of **SET A**: $\{y_n\}$ in Table 1 is applied at the input neurons of the Layer L_1

Weight coefficients

→ A set of uniformly-distributed random values (−1 to +1) is prescribed as the initial weight coefficients $\{w_{12}\}$ on the interconnections between L_1 and L_2

Compute

→ Weighted sum y_{in} at L_2: (Figure 3)

$$y_{in} = \sum_{n=1}^{7} w_{12} y_n$$

← $\{y_n\}$ is the excitatory input to the neuronal units of L_1

← $\{w_{12}\}$ is the weight adjusted from $L_1$and L_2

Compute: Sigmoidal squashing on y_{in}

→ Perform computation to assess the squashed value of y_{in} namely, y_o (Figure 3)

$$y_o = F(y_{in}) = L_Q(y_{in})$$

Choice of Q

%% In exercising sigmoidal squashing as above, the required value of Q in the Langevin-Bernoulli function (equation 8) is prescribed as follows: The value of Q is limited between (0.5 to ∞). As stated earlier [1][2], with Q = 0.5, the statistical aspects of squashed value would conform to a totally isotropic random state of participating neurons; and, with Q → ∞, the neuronal states assume a totally anisotropic framework.

→ To begin with, Q = 0.5 can be assumed; and, for any refinements on convergence required in the USL, Q can be modified between: $[0.5 \leq Q < \infty]$.

Convergence *via* weight modification

%% The summed output, $y_{in} = \sum_{n=1}^{7} w_{12} y_n$ would change with the adjustments done on weight coefficients of the interconnections between the layers L_1 and L_2. That is, the value of y_{in}

will modify (increase or decrease) with iteratively imposed adjustments on w_{12} until a desired minimum of y_{in} is obtained, meaning that the convergence of USL is attained.

→ A limit on the growth of synaptic weights should, however, be imposed so as to be consistent with modified Hebbian learning; thus, relevant weight modification is specified for any pair of layers, L_i and L_j as follows:

← w_{ij} (new) = w_{ij} (old) + learning-rate ($\alpha = \frac{1}{\text{Max } \lambda_{7\times7}}$])

× [$y_{ave} - y_o \times w_{ij}$(old)]

With reference to L_1 and L_2:

w_{12} (new) = w_{12} (old) + learning-rate ($\alpha = \frac{1}{\text{Max } \lambda_{7\times7}}$])

× [$y_{ave} - y_o \times w_{12}$(old)]

→ α refers to the learning coefficient equal to: (1/(Max $\lambda_{7\times7}$). It decides the rate of convergence

→ (Max $\lambda_{7\times7}$): It is the maximum of the values computed Hessian eigenvalues, $\lambda_{7\times7}$ for ensemble data of the training sets {A, B, C, D} listed in Table 1

→ $y_{ave} = (\frac{1}{N}) \sum_{n=1}^{N} y_n$ is the mean value of the set {y_n = 7} listed in Table 1

← The entity [$y_{ave} - y_o \times w_{12}$(old)] indicated above denotes the differential change ($\pm w_{12}$) to be applied to alter the value of y_{in} towards convergence. This differential change on existing (old) w_{12} with reference to y_{ave} conforms to a self-organizing task of the USL (Figure 2) consistent with Hebbian learning

Compute: Scaling coefficient for weight modification in USL loop

→ Required scaling coefficient for weight adjustment on any pair, (L_i, L_j)

→ $(k)_{ij} = \Delta w_{ij} = [w_{ij}$ (new) − w_{ij} (old)]

← $(k)_{12} = \alpha \times [y_{ave} - y_0 \times w_{12}(\text{old})]$ (Figure 3)

→ w_{12} (new) = w_{12} (old) + α [y_{ave} - (y_o) w_{12} (old)]

where y_{ave} is the average of the input data set

→ α denotes the learning coefficient decided as follows:

%% As indicated earlier, the desired learning-rate is set inversely proportional to maximum of computed Hessian eigenvalues, $\lambda_{7\times7}$ listed in Table 1 for the ensemble data of the sets {A, B, C, D}.

From Table 1, this maximum eigenvalue occurs for the ensemble: **Set B** and has a value, λ_{max} = 6.1747. Hence, corresponding learning-rate α is: $(1/\lambda_{max})$ = 0.1620.

When USL is exercised with the learning coefficient as above, it results in a change in the value of y_{in}; and, a relevant error can be specified by equation (4):

$$\varepsilon(w_{12}) = \sum_{n=1}^{7} [(z_n)_1 - \alpha \times F(w_{12} \times (z_n)_2]$$

$$= [y_{in} - \alpha \times L_Q(y_{in})]$$

$$= y_{in} - [\frac{1}{\text{Max } \lambda_{7\times7}}] \times L_Q(y_{in})$$

→ Learning coefficient: $\alpha = (1/(\text{Max } \lambda_{7\times7})$

← (Max $\lambda_{7\times7}$): Maximum of the values computed

URL: http://dx.doi.org/10.14738/tmlai.104.13100

Hessian eigenvalues, $\lambda_{7\times7}$ listed in Table 1 for
ensemble data of the sets {A, B, C, D}

Output: Paired-layers L_1 and L_2

→ With reference to USL exercised on L_1 and L_2 as above, the resulting output upon reaching convergence at the completion of training schedule as above refers to the following:

1. Set of neuronal states $\{y_1, y_2, \ldots, y_7\}$ at layer L_2 achieved and kept stored
2. Set of weight coefficients $\{w_{12}\}$ kept frozen upon convergence

Next

→ The set of neuronal states $\{y_1, y_2, \ldots, y_7\}$ stored at layer L_2 is then transferred to the next layer, L_3 of the hidden pair: L_3 and L_4

Perform

→ With the transferred details of neuronal states $\{y_1, y_2, \ldots, y_7\}$ as input data to layer L_3, the cycle of USL is implemented on the hidden pair: L_3 and L_4 *via* the same procedure pursued for the pair L_1 and L_2.

Output: Paired-layers L_3 and L_4

→ Hence, the corresponding converged details, namely, set of neuronal states at layer L_4 is stored and the set of adjusted weight coefficients $\{w_{34}\}$ is frozen

Next

→ The set of neuronal states $\{y_1, y_2, \ldots, y_7\}$ stored at layer L_4 is transferred to the subsequent layer, L_5 of the last/output pair: L_5 and L_6

Perform

→ With the transferred details of neuronal states $\{y_1, y_2, \ldots, y_7\}$as input data to layer L_5, the cycle of USL is implemented on the last/output pair: L_5 and L_6 *via* the same procedure pursued for the pair L_1 and L_2.

Termination of the training schedule paired-layers L_5 and L_6

→ Upon convergence seen at L_5 and L_6, the set of neuronal states $\{y_1, y_2, \ldots, y_7\}$ at layer L_6 is stored and the set of adjusted weight coefficients $\{w_{56}\}$ is frozen

← This flags the completion of training phase

End Completion of training phase

Next

→ The test ANN is ready for prediction phase

Prediction phase

Initialize

← Use the frozen data sets: $\{w_{12}\}$, $\{w_{34}\}$, $\{w_{56}\}$ on the network

← Address fresh data set**:** $\{y_1, y_2, \ldots, y_7\}$at the input layer L_1

→ This fresh data set is an exemplar intended for prediction of its labeled content

→ **Example**: Data set (**SET P**) is furnished below (Table 2). It is compatible with the random details of the sets {A, B, C, D} of Table 1 adopted in the training phase simulations outlined above

Table 2: Example data set, {y_1, y_2, …, y_7}: SET P containing unlabeled and labeled entities considered for prediction phase simulations

Data **SET P**			Parameters used in USL schedule
n	**Values of {y_n} and C**		
1	y_1	0.138655049	Learning coefficient: α = 0.1620 (As specified by the maximum eigen-value used in the training phase)
2	y_2	0.164597563	
3	y_3	0.688897953	
4	y_4	0.623615765	
5	**C**	**1.000000000**	
6	y_6	0.730366133	
7	y_7	0.860958783	
Data **C** is the labeled entity to be classified and identified in the output set of the prediction phase. (It can be present randomly at any of the input neurons, n: 1, 2, …, 7)			

Perform

→ With the initializations as above and prediction phase data set **SET P**, of Table 2, do USL operations on the three paired layers as done in training phase

Output

→ With the completion of USL implemented on the three paired layers of Figure 3, the observed set of {y_1, y_2, …, y_7} on layer L_6 of the last/output pair: L_5 and L_6 is noted. It depicts the required prediction details in the output

Result

← That is, the predicted results on the input data of Table 2 refer to the converged values on neural units {y_1, y_2, …, y_7} of layer L_6

→ Relevant output data set: **SET R** is presented in Table 3

URL: http://dx.doi.org/10.14738/tmlai.104.13100

Table 3: Result: SET R containing the mix of unlabeled and labeled entities observed as output data in layer L_6 with SET P details addressed as the input

RESULT Data **SET R**				Identified output corresponding to the labeled entity **C** in the input data of Table 2	
	Values of $\{y_n\}$				
n	Output	Q = 0.5	Q =1.0		
1	y_1	0.9867	0.9129	Labeled value classified as output	Percentage error: [(1.0000−0.9867)/1] ×100
2	y_2	0.9867	0.9129	0.9867	1.33 %
3	y_3	0.9867	0.9129		
4	y_4	0.9867	0.9129		
5	y_5	0.9867	0.9129		
6	y_6	0.9867	0.9129		
7	y_7	0.9867	0.9129		

1. Learning coefficient: α = 0.1620 (As specified by the maximum eigen-value used in the training phase)
2. Learning curves obtained are shown in Figures 4 and 5
3. The output values 0.9867) of $\{y_n\}$ shown are close to the data **C = 1** of the input set of details in **SET P** used in prediction phase. It conforms the required classification as the labeled entity by the test deep ANN

Next

Finding the data to be classified as the pattern details sought

← List the output data set $\{y_n\}$ of the prediction phase at the output layer L_6 as **SET R** shown in Table 3

→ One or more values of $\{y_n\}$ in **SET R** conform to the required data **C** corresponding to the input set of details in **SET P** used in prediction phase classified as the labeled entity.

End

RESULTS AND DISCUSSION

The present study refers to a multilayer deep ANN network, which uses unsupervised learning (USL) scheme across a hierarchical set of paired-layers with a pattern of connectivity locally in those layers. In the following subsections, summary of the operational details is outlined: In the test multilayer deep ANN network, the USL or training scheme progresses along the paired stack of layers. The inputs are addressed at the first layer and USL is performed on the first pair of layers. The resulting interconnection weights of this input pair upon convergence is stored frozen. The converged values on neuronal states in the second layer are then transferred to the third layer. Hence, the pair of third and fourth layers are trained and converged details on weights are stored; and, the converged neuronal states at fourth layer is passed on to the next (fifth) layer and so on. This procedure is carried out across entire set of paired layers of the

designed architecture and adopted in both training and prediction phases. The output being sought refers to the final converged details of the last (output) layer corresponding the prediction phase. The internal (local) calculations between the adjacent layers in any paired set depend upon the neuronal state details coming from the previous layers.

The proposed deep ANN conceived for ML in big data environments of AI has the following uniquely specified novel features: (i) The algorithm prescribes at each pair of layers a learning coefficient inversely proportional to maximum of computed Hessian eigenvalues evaluated for the ensemble data sets processed across the paired set of layers in training and prediction phases. This prescription enables a desirable fast convergence of learning involved; and, (ii) necessary sigmoidal squashing is adopted while performing the USL, which conforms to Langevin-Neelakanta machine concepts. The choices as above rely on the motivational considerations towards fast convergence of the test deep ANN even under big data ambient, yielding the result on correct classification of labeled entity present in the input data analyzed.

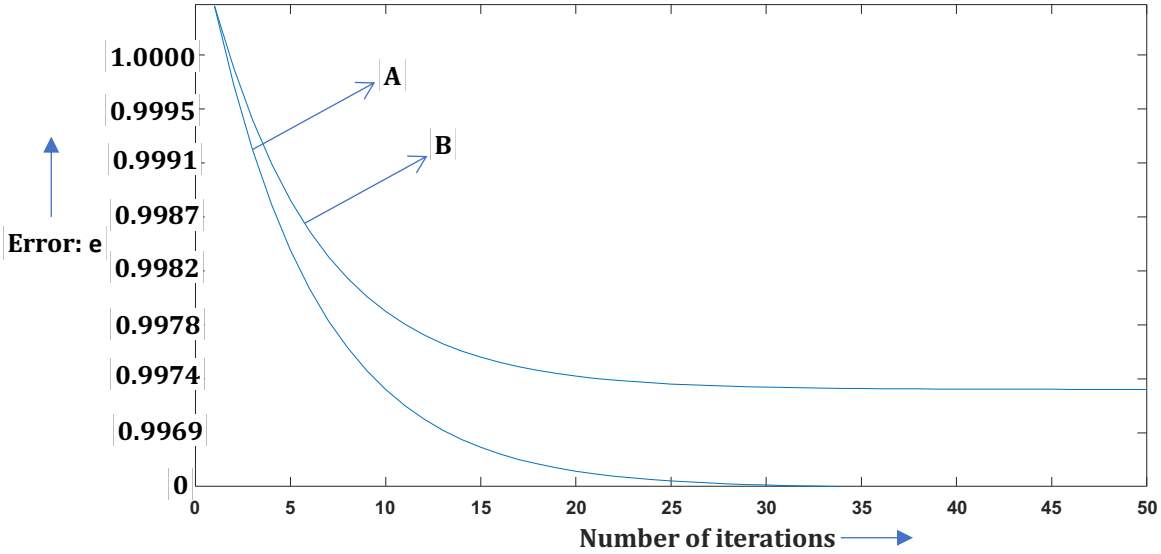


Figure 4: Learning curves depicting the observed error ε in y_{in} *versus* number of iterations in a USL loop of Figure 3 obtained (normalized) with the ensemble of data SET P in Table 2 for two cases using an optimum value of α = 0.1620 (as deduced by the proposed method): (A) with Q = 0.5 and (B) with Q = 1.0

Presented in Figures 4 and 5 are examples of learning-curves obtained for the exemplar ensemble of input data (of **SET P**) of the prediction phase. The learning curves presented are pertinent to two cases: (i) With learning coefficient, α = 0.1620, inversely proportional to the maximum value of computed Hessian eigenvalues evaluated for the ensemble data sets; and, (ii) with α equal to an arbitrary value: 0.001. As seen in Figures 4 and 5, the convergence is faster with the learning rate based on $\alpha = (1/\lambda_{max}) = 0.1620$. Also, the convergence is influenced by the value of Q adopted in the USL loop. Further, the chosen value of Q decides the accuracy of predicted output with respect to the labeled value implied at the input as in Table 3.

URL: http://dx.doi.org/10.14738/tmlai.104.13100

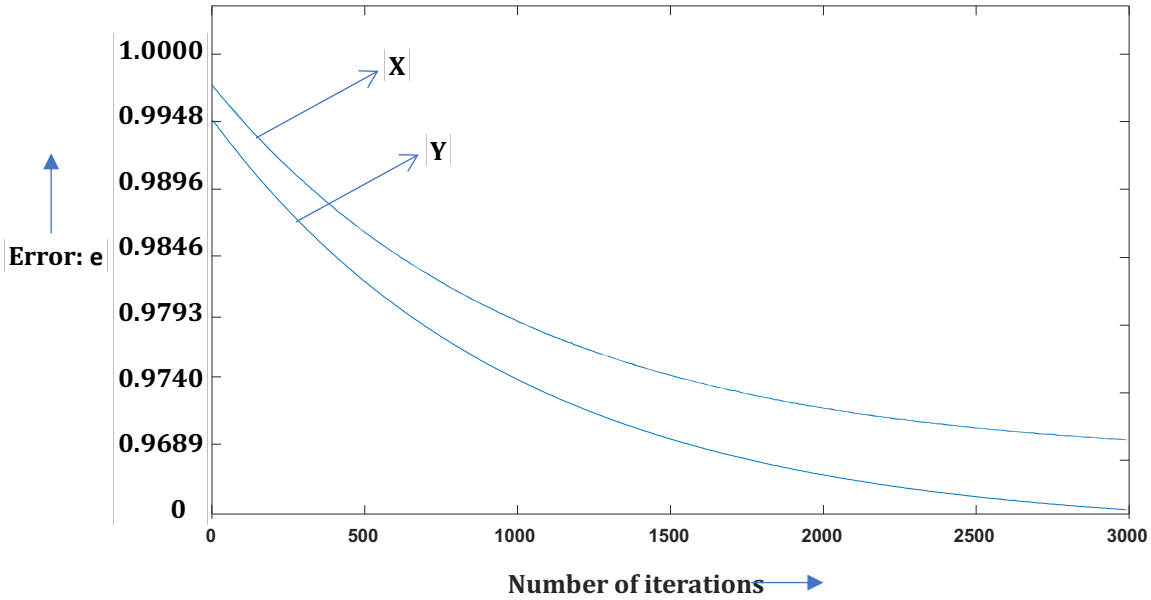


Figure 5: Learning curves depicting the observed (normalized) error ε in y_{in} *versus* number of iterations in a USL loop of Figure 3 obtained with the ensemble of data SET P, in Table 2 for two cases using an arbitrary value of α = 0.001: (X) with Q = 0.5 and (Y) with Q = 1.0

CONCLUDING REMARKS

The present study demonstrates a promising application of deep ANN for classifying and predicting a labeled pattern present as an admixture of details in a big data environment. A multilayered deep ANN is described thereof with an architecture of successively placed set of paired layers, each containing N neuronal units that are massively interconnected. Each pair of layers is independently trained *via* unsupervised learning schedule. The underlying training and prediction phases are implemented with specifically defined algorithms on sigmoidal squashing involved in defining the weight adjustment parameter and learning rate pursued independently in the USL loop of each pair of layers.

To illustrate the efficacy of the proposal, simulated details on training a test architecture and evaluating the prediction phase data are furnished. Results on fast prediction of labeled data (that could be obscurely present in the stochastic mix of unlabeled and labeled patterns in the input) concur with the novel heuristics and suggested algorithms proposed. In essence, the following can be indicated as salient features of the novel deep learning ANN supported on Langevin-Neelakanta machine:

Machine learning in big data environment objectively requires a deep learning ANN architecture to handle enormous data and achieving fast convergence. The computational power of traditional recurrent neural networks depends ultimately on the complexity of massive interconnections of the network. Relevant implications of information processing are decided by Kolmogorov complexity associated with the size of the network architecture and extents of weights present. Any neural net learning algorithms aim at finding simple nets to assimilate the training (test) data robustly and lead to better the generalization on test data in the prediction phase [10][11]. This complexity issue grows significantly with increased number of hidden layers manifesting as computational bottleneck in handling large data and involving excessive processing time towards convergence. This issue is obviously serious in big data environment; as such, novelty in conceiving better deep ANN architecture is a motivated goal

towards ML efforts in AI contexts. Hence, the present study on deep ANN has been indicated with the feasibility to accommodate a large number of cascaded sets of independent paired layers; and, each pair is trained *via* USL towards self-organization. In the relevant effort, the algorithmic suites pursued enable fast convergence with a uniquely specified learning rate as well as, achieving better (more accurate) results on prediction by resorting to Langevin-Neelakanta machine concept with the associated squashing function in the USL loop. The architecture of using a cascaded set of individualized paired set of layers is akin to that of Fukushima's cognitron [4][5], but modified to match the motivated context of the present study. Simulation results presented establish the desirable efficacy of the proposed concept.

References

1. Neelakanta, P. S., De Groff, D., Neural Network Modeling: Statistical Mechanics and Cybernetic Perspectives, 1994, CRC Press, Boca Raton, Fl: USA.
2. Neelakanta, P. S., (Editor), Information-Theoretic Aspects of Neural Networks, 1999, CRC Press, Boca Raton, Fl: USA.
3. Neelakanta, P. S., Sudhakar, R., and DeGroff, D., Langevin machine: A neural network based on stochastically justifiable sigmoidal function, Biological Cybernetics, 1999, 65, p. 331-338.
4. Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 1980, 36, p.193-202.
5. Fukushima, K. Cognitron: A self-organizing multilayered neural network. Biological Cybernetics, 1975, 20, p.121-136.
6. Becker, S., Plumbley, M. Unsupervised neural network learning unsupervised neural network learning procedures for feature extraction and classification. Applied Intelligence, 1996, 6, p.185-203.
7. Wittek, P., Unsupervised learning in quantum machine learning, Elsevier Inc/Morgan Kaufmann, 2014, Cambridge, MA: USA.
8. De Groff, D., Neelakanta, P. S., Faster convergent neural networks, International Journal of Computers and Technology, 2018, 17(1), p.7126-7132.
9. Margoulas, G. D., Vrahatis, M. N., Androulakis, G.S. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods, Neural Computation, 1999, 11(7), p. 1769-1796.
10. Schmidhuber, J. Discovering neural nets with low Kolmogorov complexity and high generalization capability. Neural Network, 1997, 10(5), p.857-873
11. Balcázar, J. L. Computational power of neural networks: A characterization in terms of Kolmogorov complexity. IEEE Transactions on Information Theory, 1997, 43(4), p.1175-1183.

URL: http://dx.doi.org/10.14738/tmlai.104.13100