

Deep Learning with Random Neural Networks

Erol Gelenbe *IEEE* and Yongha Yin
Intelligent Systems and Networks Group
Electrical & Electronic Engineering Department
Imperial College, London SW7 2AZ, UK

Abstract—The Random Neural Network (RNN) is a recurrent spiking neuronal model that has been used for learning and dynamic optimisation of large scale network systems. Here we use the RNN to construct dense block of spiking neuronal cells in conjunction with Deep Learning to mimic the stochastic behaviour of biological neurons in mammalian brains. Together with prior work on extreme learning machines (ELM), we construct multilayer architectures (MLA) that exploit dense clusters of RNNs for Deep Learning and evaluate their performance on large visual recognition datasets. The results obtained indicate that this approach can reach and exceed the levels of performance that have been previously reported. Finally, we develop an incremental learning algorithm to train such RNN-ELM multilayer architectures for purpose of handling big data.

Index Terms—Deep Learning; Extreme Learning Machines; Random Neural Network; Spiking Neurons; Dense Clusters; Natural Neuronal Ensembles

I. INTRODUCTION

In recent years, deep learning with regular and tightly-coupled arrays of sigmoidal neurons has come to the forefront as a possible way to overcome the limitations of neural networks when they are applied to real-world challenges [1], [2], while many engineering applications require significant advances in learning from big data [3]–[6].

Tightly coupled clusters of *natural neuronal cells* communicate with each other in multiple ways, both through spiking [7], via soma-type interactions with other cells [8], through neuromodulators [9], and with the help of important structures such as glial cells [10] which are known to exercise complex functions in the hippocampus and cerebellum that contribute to synaptic transmission and modulate synaptic function. This complexity of natural neural information processing and learning [11] goes well beyond the models traditionally exploited in machine learning [12], and goes significantly beyond the capabilities of sigmoid-based neural models.

The Random Neural Network (RNN) [13]–[19] is a spiking “integrate and fire” model where an arbitrarily large set of cells interact with each other via excitatory and inhibitory spikes which modify each cell’s action potential in continuous time, and mathematically described by a system of differential equations known as the Chapman-Kolmogorov equations [20]. It was originally developed to mimic the behaviour of biological neurons [21]. However subsequently it was exploited for numerous applications that exploit the *recurrent structure* of the network and of its learning algorithm [22], including combinatorial optimisation [23], several examples of image and video processing [24]–[28], and routing [29]–[31]

in cyber-physical systems and computer networks. All these applications exploit the *recurrent structure* of the network.

The computational power of the RNN originates in the fact that, in steady-state, the network is characterised by the joint probability distribution of the activation state of each cell, which is equal to the product of the marginal probabilities of the activation states. This property, known as “product form” in the probability literature [20], makes the RNN particularly amenable to exact yet simple, fast (and easily parallelisable) computational algorithms.

It is worth pointing out the main contributions of this paper as the followings.

- We derive a mathematical model for soma to soma interactions and investigate a special type of cell clusters, based on which one type of biology-motivated flexible activation function is developed. It bridges the gap between the area of spiking random neural network that mimics the mammalian brains and is supported by a rigorous theoretical framework [13]–[19] and the area of deep learning.
- Utilizing the well-developed ELM techniques [32]–[35], an effective deep-learning architecture has been built up on basis of the RNN clusters with the performance tested on different types of datasets.

II. THE MATHEMATICAL MODEL

We consider the Random Neural Network Model developed in [17], [18], composed of M neurons or cells, each of which receives excitatory (positive) and inhibitory (negative) spike trains from external sources which may be sensory sources or cells. These arrivals occur according to independent Poisson processes of rates λ_m^+ for the excitatory spike train, and λ_m^- for the inhibitory spike train, respectively, to cell $m \in \{1, \dots, M\}$. For readers’ convenience, we summarize the notations in this section in Table I.

In this model, each neuron is represented at time $t \geq 0$ by its internal state $k_m(t)$ which is a non-negative integer. If $k_m(t) > 0$, then the arrival of a negative spike to neuron m at time t results in the reduction of the internal state by one unit: $k_m(t^+) = k_m(t) - 1$. The arrival of a negative spike to a cell has no effect if $k_m(t) = 0$. On the other hand, the arrival of an excitatory spike always increases the neuron’s internal state by +1.

If $k_m(t) > 0$, then the neuron m is said to be “excited”, and it may “fire” a spike with probability $r_m \Delta t$ in the interval $[t, t + \Delta t]$, where $r_m > 0$ is its “firing rate”, so that r_m^{-1} may be viewed as the average firing delay of the excited m -th neuron.

TABLE I
NOTATIONS FOR RNN MATHEMATICAL MODEL IN SECTION II

λ_m^+	External excitatory spike arrival rate of neuron m
λ_m^-	External inhibitory spike arrival rate of neuron m
r_m	Firing rate of neuron m
$k_m(t)$	Internal state of neuron m at time t
$p^+(i, j)$	Probability that neuron i fires a excitatory spike to neuron j
$p^-(i, j)$	Probability that neuron i fires an inhibitory spike to neuron j
$p(i, j)$	Probability that neuron i triggers neuron j
d_i	Probability that the spike or trigger is lost or leaves the network when i fires
$Q(j, m)$	Probability that neurons i, j increment neuron m when i triggers j
$\pi(j, m)$	Probability that a trigger moves on from j to m when i triggers j
q_m	Stationary probability that neuron m is excited
*Here i, j and m mean neurons i, j and m	

Neurons in this model can interact in the following manner at time $t \geq 0$. If neuron i is excited, i.e. $k_i(t) > 0$, then, when i fires, its internal state drops by 1 and we have $k_i(t^+) = k_i(t) - 1$, and the followings.

- It can either send a positive or excitatory spike to neuron j with probability $p^+(i, j)$ resulting in $k_i(t^+) = k_i(t) - 1$ and $k_j(t^+) = k_j(t) + 1$.
- Or it may send a negative or inhibitory spike to neuron j with probability $p^-(i, j)$ so that $k_i(t^+) = k_i(t) + 1$ and $k_j(t^+) = k_j(t) - 1$ if $k_j(t) > 0$, else $k_j(t^+) = 0$ if $k_j(t) = 0$.
- Or neuron i can “trigger” neuron j with probability $p(i, j)$ so that $k_i(t^+) = k_i(t) - 1$ and $k_j(t^+) = k_j(t) - 1$ if $k_j(t) > 0$.
- When neuron i triggers neuron j , both $k_i(t^+) = k_i(t) - 1$ and $k_j(t^+) = k_j(t) - 1$, and one of two things may happen. Either:
 - (A) with probability $Q(j, m)$ we have $k_m(t^+) = k_m(t) + 1$ so that neurons i and j together have incremented the state of neuron m . Thus we see that a trigger allows two neurons i and j to increase the excitation level of a third neuron m by +1, while neurons i and j are both depleted by -1.
 - (B) Or with probability $\pi(j, m)$ the trigger moves on to the neuron m and then the sequence (A) or (B) is repeated.
- Note that $\sum_{j=1}^M [p(i, j) + p^-(i, j) + p^+(i, j)] = 1 - d_i$ where d_i is the probability that, when the neuron i fires, the corresponding spike or trigger is lost or it leaves the network. Also, $1 = \sum_{m=1}^M [Q(j, m) + \pi(j, m)]$.

Since cells in different layers of mammalian brain also communicate through simultaneous firing patterns of densely packet somas, the RNN was extended in [18], [36] using a branch of the theory of stochastic networks called G-Networks [37]. In the sequel, we will exploit this structure for deep learning.

A. Modelling Soma to Soma Interactions

Now let $z(m) = (i_1, \dots, i_l)$ be any ordered sequence of *distinct* numbers where $i_j \in \{1, \dots, M\}$ and $i_j \neq m$;

obviously $2 \leq l \leq M - 1$. Let us denote by $q_m = \lim_{t \rightarrow \infty} \text{Prob}[k_m(t) > 0]$ the probability that neuron m is excited. It is given by the following expression [17], [37]:

$$q_m = \frac{\Lambda_m^+}{r_m + \Lambda_m^-}, \quad (1)$$

where the variables in (1) are of the form:

$$\begin{aligned} \Lambda_m^+ &= \lambda_m^+ + \sum_{j=1, j \neq m}^M r_j q_j p^+(j, m) \\ &+ \sum_{\text{all } z(m)} r_{i_1} \prod_{j=1, \dots, l-1} q_{i_j} p(i_j, i_{j+1}) q_{i_l} Q(i_l, m), \\ \Lambda_m^- &= \lambda_m^- + \sum_{j=1, j \neq m}^M r_j q_j p^-(j, m) \\ &+ \sum_{\text{all } z(m)} r_{i_1} \prod_{j=1, \dots, l-1} q_{i_j} p(i_j, i_{j+1}) q_{i_l} p(i_l, m). \end{aligned}$$

Here we set $p(i, j) = \pi(i, j)$ in order to simplify the expression. In the sequel, to simplify the notations we will write $w_{ji}^+ = r_j p^+(j, i)$ and $w_{ji}^- = r_j p^-(j, i)$.

B. Clusters of Identical and Densely Connected Cells

Let us now consider the construction of special clusters of densely interconnected cells. We first consider a special network, call it $\mathbf{M}(n)$, that contains n identically connected cells, each of which has firing rate r and external inhibitory and excitatory arrivals of spikes denoted by λ^- and λ^+ , respectively. The state of each cell is denoted by q , and it receives an *inhibitory input* from the state of some cell u which does not belong to $\mathbf{M}(n)$. Thus for any cell $i \in \mathbf{M}(n)$ we have an inhibitory weight $w_u^- \equiv w_{u,i}^- > 0$ from u to i .

For any $i, j \in \mathbf{M}(n)$ we have $w_{i,j}^+ = w_{i,j}^- = 0$, but all whenever one of the cells fires, it triggers the firing of the other cells with $p(i, j) = \frac{r}{n}$ and $Q(i, j) = \frac{1-p}{n}$. As a result, we have:

$$q = \frac{\lambda^+ + r q (n-1) \sum_{l=0}^{\infty} \left[\frac{q p (n-1)}{n} \right]^l \frac{1-p}{n}}{r + \lambda^- + q_u w_u^- + r q (n-1) \sum_{l=0}^{\infty} \left[\frac{q p (n-1)}{n} \right]^l \frac{p}{n}} \quad (2)$$

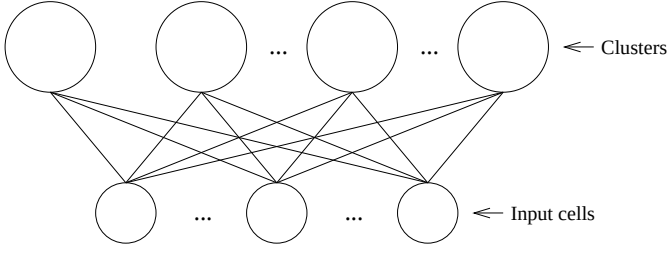


Fig. 1. Schematic representation of the Deep Learning network architecture

which reduces to:

$$q = \frac{\lambda^+ + \frac{rq(n-1)(1-p)}{n-qp(n-1)}}{r + \lambda^- + q_u w_u^- + \frac{rqp(n-1)}{n-qp(n-1)}}, \quad (3)$$

where (3) is a second degree polynomial in q :

$$q^2 p(n-1)[\lambda^- + q_u w_u^-] + q(n-1)[r(1-p) - \lambda^+ p] - qn(r + \lambda^- + q_u w_u^-) + \lambda^+ n = 0. \quad (4)$$

Under the conditions that $n \geq 2$, $p \leq 1$, $\lambda^+ > 0$, $\lambda^- > 0$, $r > 0$ and $\lambda^- \geq \lambda^+$, this equation can be easily solved for its *positive roots*, where one of them is less than 1 which is the only one of interest because q is a probability. Note that the analysis for the conditions is given in Appendix.

C. A RNN with Multiple Clusters of a $M(n)$ Architectures

In this subsection, we build a Deep Learning Architecture (DLA) based on multiple clusters, each of which is made up of a $M(n)$ cluster. This DLA is shown schematically in Figure 1. The DLA is composed of C clusters $M(n)$ each with n hidden cells. For the c -th such cluster, $c = 1, \dots, C$, the state of each of its identical cells is denoted by q_c . In addition, as shown in Figure 1, there are U input cells which do not belong to these C clusters, and the state of the u -th cell $u = 1, \dots, U$ is denoted by $\bar{q}_u$.

Each hidden cell in the clusters c , with $c \in \{1, \dots, C\}$, receives an inhibitory input from each of the U input cells. Thus, for each cell in the c -th cluster, we have inhibitory weights $w_{u,c}^- > 0$ from the u -th input cell to each cell in the c -th cluster. Thus the u -th input cell will have a total inhibitory “exit” weight, or total inhibitory firing rate r_u^- to all of the clusters which is of value:

$$r_u^- = n \sum_{c=1}^C w_{u,c}^-. \quad (5)$$

Then, from (3) and (4), we have

$$q_c = \frac{\lambda_c^+ + \frac{r_c q_c(n-1)(1-p_c)}{n-q_c p_c(n-1)}}{r_c + \lambda_c^- + \sum_{u=1}^U \bar{q}_u w_{u,c}^- + \frac{r_c q_c p_c(n-1)}{n-q_c p_c(n-1)}} \quad (6)$$

yielding a second degree polynomial for each of the q_c :

$$a_c q_c^2 + b_c q_c + d_c = 0, \quad (7)$$

where $a_c = p_c(n-1)(\lambda_c^- + \sum_{u=1}^U \bar{q}_u w_{u,c}^-)$, $b_c = e_c - n \sum_{u=1}^U \bar{q}_u w_{u,c}^-$, $d_c = n \lambda_c^+$ and $e_c = \lambda_c^+ p + r_c p - \lambda_c^- n$

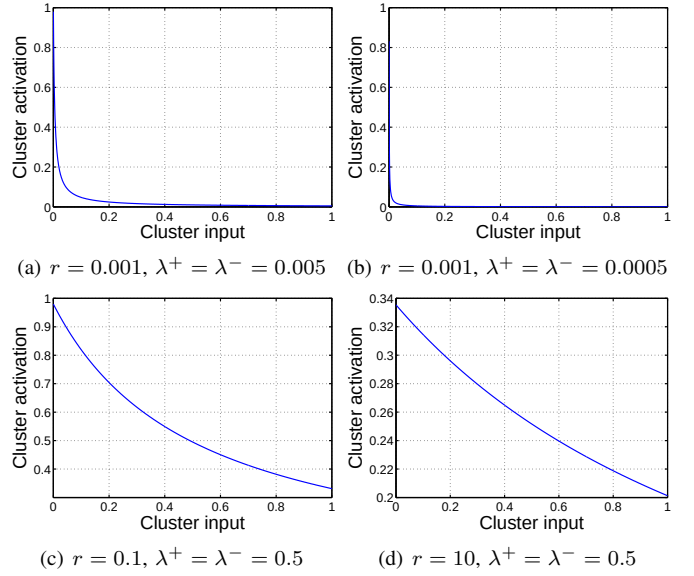


Fig. 2. Cluster activation $\zeta(x)$ versus input x under different parameters for a twenty-cell cluster ($n = 20$) with $p = 0.05$.

$r_c - \lambda_c^+ p_c n - n p_c r_c$. Its positive root is then:

$$\zeta_c(x) = \frac{-(e_c - nx) - \sqrt{(e_c - nx)^2 - 4p_c(n-1)(\lambda_c^- + x)d_c}}{2p_c(n-1)(\lambda_c^- + x)}$$

where:

$$x = \sum_{u=1}^U \bar{q}_u w_{u,c}^-.$$

When all the parameters n , $p_c = p$, $\lambda_c^+ = \lambda^+$, $\lambda_c^- = \lambda^-$, $r_c = r$, $e_c = e$, $d_c = d$ with $c = 1, \dots, C$ are the same for all of the clusters, we will have:

$$\zeta(x) = \frac{-(e - nx) - \sqrt{(e - nx)^2 - 4p(n-1)(\lambda^- + x)d}}{2p(n-1)(\lambda^- + x)}. \quad (8)$$

The resulting activation function (8) is highly nonlinear and flexible, which goes beyond the capabilities of conventional activation functions in machine learning, e.g., the sigmoid, tansig and sine functions [12], [34], [35]. The activation of the cluster (8) depends on parameters n , p , λ^+ , λ^- and r . For better illustration, Figure 2 shows the curves of $\zeta(x)$ versus x under different parameters for a cluster with twenty cells, where the parameters satisfy the conditions derived for (4) ($n \geq 2$, $p \leq 1$, $\lambda^+ > 0$, $\lambda^- > 0$, $r > 0$ and $\lambda^- \geq \lambda^+$).

III. APPLICATION TO THE DESIGN OF AN AUTO-ENCODER

In this section, we will construct an auto-encoder based on two instances of the network shown in Figure 1, which we call Network-1 and Network-2.

Network-1 has U input cells and C clusters (as shown in the figure). On the other hand, Network-2 has C input cells and U clusters.

We first construct the Network-1 such that the U -vector of input cells is: $\bar{q}^{(1)}$, and we construct the $U \times C$ matrix of

weights from the input cells to the cells in each of the C clusters as

$$W^{(1)} = [w_{u,c}^-]. \quad (9)$$

Denoting by $q^{(1)}$ the C -vector of cells whose state is q_c for cluster c , and for an n -vector y denoting by:

$$\zeta(y) = (\zeta(y_1), \dots, \zeta(y_n)), \quad (10)$$

we have:

$$q^{(1)} = \zeta(\bar{q}^{(1)} W^{(1)}). \quad (11)$$

On the other hand, Network-2 is a pseudo-inverse of Network-1, with C input cells and U clusters, and the $C \times U$ weight matrix between its input cells and the cells of each of the clusters will be denoted by $W^{(2)}$. We will then have:

$$\bar{q}^{(2)} = \zeta(\zeta(\bar{q}^{(1)} W^{(1)}) W^{(2)}). \quad (12)$$

Problem 1 The learning problem is then to adjust $W^{(1)}$ and $W^{(2)}$ so that $\bar{q}^{(2)}$ becomes as close to $\bar{q}^{(1)}$ as possible. When we have a set of data X (a $D \times U$ matrix) which has the form of D rows of U -vectors $x \in [0, 1]^U$, the problem we address can be described as:

$$\begin{aligned} \min_{W^{(1)}, W^{(2)}} & \|X - \zeta(\zeta(XW^{(1)})W^{(2)})\|^2, \\ \text{s.t. } & W^{(1)}, W^{(2)} \geq 0, \end{aligned}$$

where the function $\zeta(\cdot)$ is understood to be extended to the matrix case.

A. A First Approach

We may generalise the approach developed by Liu [38] to solve Problem 1. To this effect, let us define a cost function

$$L(W^{(1)}, W^{(2)}) = \|X - \zeta(\zeta(XW^{(1)})W^{(2)})\|^2. \quad (13)$$

First calculate the partial derivative $\eta(x) = \partial \zeta(x) / \partial x$. We also define another element-wise operation $\eta(H) \in R^{D \times C}$ with $H \in R^{D \times C}$, where the element in the i th row and j th column of $\eta(H)$ is calculated by $\eta(H_{i,j})$ with $i = 1, \dots, D$ and $j = 1, \dots, C$. Then we can derive

$$\begin{aligned} \frac{\partial L}{\partial W^{(1)}} &= -X^T(\eta(XW^{(1)})) \\ &\quad *(((X - \zeta(\zeta(XW^{(1)})W^{(2)})) \\ &\quad * \eta(\zeta(XW^{(1)})W^{(2)}))(W^{(2)})^T)). \end{aligned}$$

Note that, the operation $*$ is defined as an element-wise multiplication operation. For example, if $H = H^{(1)} * H^{(2)}$, then $H \in R^{D \times C}$, $H_1 \in R^{D \times C}$ and $H_2 \in R^{D \times C}$. Besides, the element in the i th row and j th column of H , which is $H_{i,j}$, is calculated by $H_{i,j} = H_{i,j}^{(1)} H_{i,j}^{(2)}$, where $i = 1, \dots, D$ and $j = 1, \dots, C$.

To make it clearer, let $\varphi_1 = \eta(XW^{(1)})$, $\varphi_2 = \zeta(\zeta(XW^{(1)})W^{(2)})$, $\varphi_3 = \eta(\zeta(XW^{(1)})W^{(2)})$ and $\varphi_4 = \zeta(XW^{(1)})$. Then,

$$\begin{aligned} \frac{\partial L}{\partial W^{(1)}} &= -X^T(\varphi_1 * (((X - \varphi_2) * \varphi_3)(W^{(2)})^T)) \\ &= -X^T(\varphi_1 * ((X * \varphi_3)(W^{(2)})^T)) \\ &\quad + X^T(\varphi_1 * ((\varphi_2 * \varphi_3)(W^{(2)})^T)), \end{aligned}$$

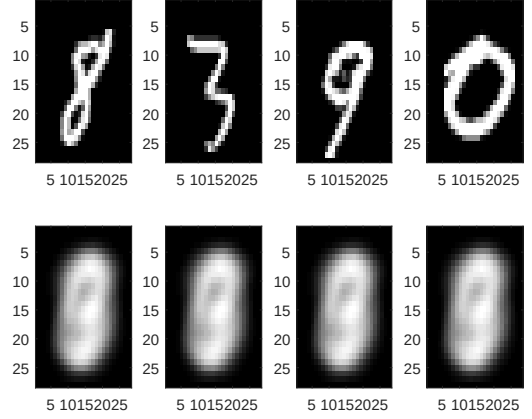


Fig. 3. Results of the autoencoder with only spiking models.

and

$$\frac{\partial L}{\partial W^{(2)}} = -\varphi_4^T((X - \varphi_2) * \varphi_3) \quad (14)$$

$$= -\varphi_4^T(X * \varphi_3 - \varphi_2 * \varphi_3) \quad (15)$$

$$= -\varphi_4^T(X * \varphi_3) + \varphi_4^T(\varphi_2 * \varphi_3).$$

The updates rules for $W^{(1)}$ and $W^{(2)}$ then become

$$W_{i,j}^{(1)} \leftarrow W_{i,j}^{(1)} \frac{(X^T(\varphi_1 * ((X * \varphi_3)(W^{(2)})^T)))_{i,j}}{(X^T(\varphi_1 * ((\varphi_2 * \varphi_3)(W^{(2)})^T)))_{i,j}} \quad (16)$$

and

$$W_{i,j}^{(2)} \leftarrow W_{i,j}^{(2)} \frac{(\varphi_4^T(\varphi_2 * \varphi_3))_{i,j}}{(\varphi_4^T(X * \varphi_3))_{i,j}}, \quad (17)$$

where the symbol $(H)_{i,j}$ denotes the element in the i th row and j th column of H .

To be more specific, in the right-hand side of (16) and (17), we use the original values of $W^{(1)}$ and $W^{(2)}$ in the l th iteration. Then, the left-hand side of (16) and (17) would be the updated values of $W^{(1)}$ and $W^{(2)}$ in the l th iteration.

B. An Auto-encoder Combining RNN and Extreme Learning Machine

In this subsection, to achieve better performance, we modify the learning problem as follows:

Problem 2 Find $W^{(1)}$ such that

$$\min_{W^{(1)}} \|X - \zeta(XW^{(1)})W^{(2)}\|^2, \text{ s.t. } W^{(1)} \geq 0. \quad (18)$$

Thus specifically, we will use Network-1 for encoding, but we use an idea from the “extreme learning machine (ELM)” [33] for the decoding part of the problem.

Generally, when we use an auto-encoder for classification, we connect only the encoding part to the classifier. Thus while $W^{(1)} \geq 0$, we will *remove* the constraint that $W^{(2)} \geq 0$. We will call this autoencoder the Random Neural Network Extreme Learning Machine (RNN-ELM).

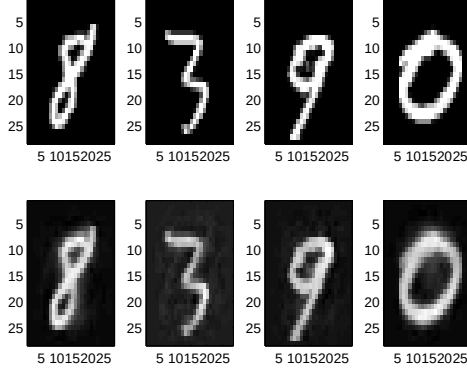


Fig. 4. Results of the autoencoder with structure 784-500 via only (19): up for original figures and down for reconstructions.

Adapting the idea in [33], [39], [40] we use the following operation to determine $W^{(2)}$:

$$W^{(2)} = \text{pinv}(\varphi_4)X, \quad (19)$$

where $\text{pinv}(x)$ denotes the Moore-Penrose pseudo-inverse of a matrix x .

Let us define $\bar{W}^{(2)} = \max(W^{(2)}, 0)$. Let $\varphi_5 = \zeta(XW^{(1)})\bar{W}^{(2)} = \varphi_4\bar{W}^{(2)}$. Then, the update rule for $W^{(1)}$ will be

$$W_{i,j}^{(1)} \leftarrow W_{i,j}^{(1)} \frac{(X^T(\varphi_1 * (X(\bar{W}^{(2)})^T)))_{i,j}}{(X^T(\varphi_1 * (\varphi_5(\bar{W}^{(2)})^T)))_{i,j}}, \quad (20)$$

which guarantees that $W^{(1)} \geq 0$.

C. Testing the RNN-ELM

To test the RNN-ELM, we use the MNIST dataset of handwritten digits [41] which contains 60,000 images in the training dataset and 10,000 images in the test dataset. We conduct numerical experiments on the auto-encoder with two different structures: one is a $784 \rightarrow 50$ structure with 50 intermediate or hidden units, while the second one is a $784 \rightarrow 500$ structure with 500 hidden units. In both cases we exploit clusters with $n = 20$. Exhaustive tests were carried out as follows:

- We first randomly generated the elements of $W^{(1)}$ in the range of $[0, 1]$.
- Then, we used (19) to determine $W^{(2)}$.

Examples of the results obtained with this approach are shown in Figure 4.

In a second approach, we use (20) to update $W^{(1)}$, and then use (19) again to update $W^{(2)}$. The results obtained are shown in Figures 5 and 6.

It is evident that the results in the second approach Figures 5 and 6 are far better than those in Figure 4 (for the same structures). This illustrates that both (19) and (20) are important for adjusting the parameters of the auto-encoder.

IV. STACKING THE CLASSIFIERS

Following Tang's work [32], we can also stack multi auto-encoders together and connect them to an ELM to construct

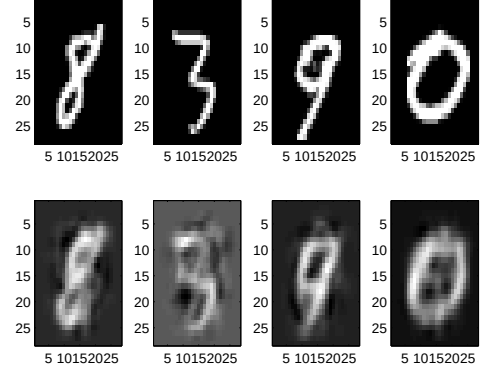


Fig. 5. Results of the autoencoder with structure 784-50 via both (19) and (20): up for original figures and down for reconstructions.

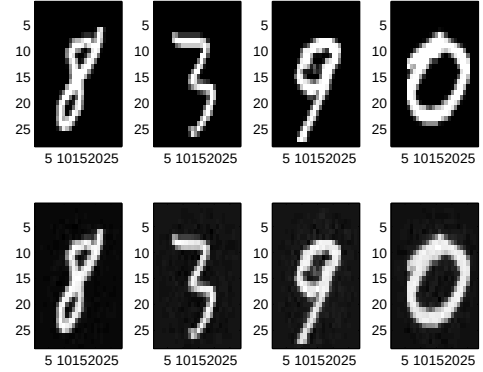


Fig. 6. Results of the autoencoder with structure 784-500 via both (19) and (20): up for original figures and down for reconstructions.

a multi-layer classifier. Based on how we solve the auto-encoders and utilize the weights, we construct two different classifiers (Classifier-1 and Classifier-2) in this section and then test them on visual recognition datasets.

A. Classifier-1

First, let us consider a different approach from the one in Section III-B, using the advice from [32] regarding the use of the $L-1$ norm to generate more sparse and compact features. Then, the problem to be addressed can be described as

$$\begin{aligned} \min_{W^{(2)}} & \|X - \zeta(XW^{(1)})W^{(2)}\|^2 + \|W^{(2)}\|_{\ell_1}, \quad (21) \\ \text{s.t. } & W^{(2)} \geq 0, \end{aligned}$$

showing that we only need to adjust $W^{(2)}$. Indeed, based on [32], [33], a randomly-generated $W^{(1)}$ could be sufficient to obtain effective learning with reduced computational complexity. Note that the constraint of $W^{(2)} \geq 0$ is the characteristic that allows us to use $W^{(2)}$ in the RNN.

Since $W^{(1)}$ is fixed once generated, we can then use the fast iterative shrinkage-thresholding algorithm (FISTA) in [42] to solve problem (21), with the modification that we set the negative elements in the solution to zero in each iteration.

Once (21) is solved, $W^{(2)}$ is obtained and let $\tilde{W}^{(1)} = (W^{(2)})^T$. Then, we import $\tilde{W}^{(1)}$ to the RNN with X as input, and output $X^{(2)} = \zeta(X\tilde{W}^{(1)})$. By using $X^{(2)}$ as the input to the next auto-encoder, we then seek the weights $\tilde{W}^{(2)}$ for the next layer of the multi-layer classifier. Note that the last layer of the multi-layer classifier is an ELM with activation function $\zeta(x)$.

Consider a multi-layer classifier with a structure denoted by $784 - 700 - 700 - 5000 - 10$. This is a multi-level structure where the weights between successive layers are denoted by $\tilde{W}^{(i)}$ with $i = 1, \dots, 4$.

The details of this structure are given by the nature of the layers themselves, and the interconnections of each of the successive feedforward layers. The RNN layers use, instead of single nodes, new spiking clusters that we have defined, where we have used clusters of twenty cells, rather than single cells. The specific experiments we have run deal with architectures with characteristics such as:

- The first two layers $784 \rightarrow 700$ and $700 \rightarrow 700$ are RNNs where the $\tilde{W}^{(1)}$ and $\tilde{W}^{(2)}$ are determined by the auto-encoder.
- The third layer $700 \rightarrow 5000$ is also an RNN, where $\tilde{W}^{(3)}$ is obtained by a random generation of each entry in $[0, 1]$.
- Finally, the last layer $5000 \rightarrow 10$ is an ELM.

B. Classifier-2

The RNN blocks of the multi-layer classifier (called Classifier-1) in subsection IV-A do not incorporate the negative information from the autoencoder. Specifically, when we addressed the problem in (21) for $W^{(2)}$ with the FISTA, we abandoned the negative information and then build the multi-layer Classifier-1. In this subsection, we will preserve the negative information and then build Classifier-2. Let us consider the problem as follows:

$$\min_{W^{(2)}} \|X - \zeta(XW^{(1)})W^{(2)}\|^2 + \|W^{(2)}\|_{\ell_1}, \quad (22)$$

where $W^{(1)}$ are randomly generated. We can solve this problem with the FISTA or with the equation in the ELM literature [32]–[35]:

$$W^{(2)} = \left(\frac{I}{\kappa} + \varphi^T \varphi\right)^{-1} \varphi^T X, \quad (23)$$

where $\varphi = \zeta(XW^{(1)})$, κ is a positive scalar and I is an identity matrix with appropriate dimensions. Using (23) may not achieve as good performance as using the FISTA. However, the advantage is that it is suitable for incremental learning [34], [35], which can be utilized for handling big data (Section VI). Since there are both positive and negative elements in $W^{(2)}$, we separate it into two parts before using it with the RNN blocks.

Let $\bar{W}^{(2)} = \max(W^{(2)}, 0)$ and $\hat{W}^{(2)} = -\min(W^{(2)}, 0)$. It is evident that $\bar{W}^{(2)}, \hat{W}^{(2)} \geq 0$. Let us store $\bar{W}^{(1)} = [(\bar{W}^{(2)})^T (\hat{W}^{(2)})^T]^T$ to build the first layer of Classifier-2. This first layer is an RNN that takes X as input, where the output is $X^{(2)} = \zeta(X\bar{W}^{(1)})$. Then, we use $X^{(2)}$ as the input of the next auto-encoder, in order to seek the weights $\tilde{W}^{(2)}$ for building the next layer of Classifier-2. Note that the last two layers of

Classifier-2 are the RNN-ELM architecture, which is the same as that of Classifier-1 in Subsection IV-A.

C. Test Results

To test Classifier-1 and Classifier-2, we use the MNIST dataset [41] and the small NORB dataset [43].

1) *MNIST Dataset*: The MNIST dataset of handwritten digits [41] is a classical dataset for testing performance of a machine-learning tool. It contains 60,000 images in the training dataset and 10,000 images in the test dataset. The number of input attributes is 784 (28×28 images), whose values are in the range $[0, 1]$. The attributes are preprocessed by using the MATLAB operation “zscore(X)”.

2) *NORB Dataset*: The small NORB dataset [43] is intended for experiments in 3D object recognition from shape. It contains stereo image pairs of 50 toys belonging to 5 generic categories: four-legged animals, human figures, airplanes, trucks and cars. In either the training or testing dataset, the number of instances is 24300. There are two 96×96 images in each instance. To fit the training dataset into the available memory, we first downsample each image from 96×96 into 32×32 . Besides, we whiten all images using the code provided by [32].

3) *Numerical results*: The results that use Classifier-1 and Classifier-2 for classifying the MNIST dataset are summarized in Table II under different structures and clusters. For the RNN clusters, parameters $n = 20$, $p = 0.05$ are fixed, while the values of $\lambda = \lambda^- = \lambda^+$ are varied. For Classifier-2, we also compare performances using the FISTA and (23) (denoted by different values of κ). From the results, we can see that using the FISTA generally gets better results than using (23). The highest testing accuracy obtained is 98.95%, which is a very high accuracy for a fully-connected multi-layer neural network [32], [41], [44]. We could also see that $\kappa = 10^{-5}$ is a generally good parameter choice when using (23) (whose results are sometimes better than those using the FISTA), while parameter $\lambda = \lambda^- = \lambda^+$ for the cluster could be chosen as 0.01, 0.005 or 0.001. In addition, we reverted to the pure RNN-ELM architecture without the auto-encoder, where $\lambda = 0.005$ is used for the cluster. The structure is of the form $784 - 10000 - 10$, and we observed 97.26% accuracy at testing.

Then, Classifier-1 and Classifier-2 are used for classifying the NORB dataset. The results of testing accuracies are shown in Table III, where parameter κ is fixed as 10^{-5} when using (23) and parameter λ for the cluster is chosen as 0.005. As shown in the table, the highest testing accuracy is 90.89%, which is only slightly lower than the reported 91.28% in [32].

V. CLASSIFIERS FOR MULTI-SOURCE DATASETS AND THEIR EVALUATION

Based on Classifier-1 and Classifier-2, three more classifiers, Classifier-3, Classifier-4 and Classifier-5, are constructed to handle datasets where the attributes of each instance in the dataset come from multi sources, e.g., each instance in the small NORB dataset has two individual images, which will be used as an example to illustrate the construction of the classifiers.

TABLE II
TESTING ACCURACIES (%) OF CLASSIFIER-1 AND CLASSIFIER-2 UNDER DIFFERENT STRUCTURES AND CLUSTERS FOR MNIST DATASET.

Network structure	Classifier-1	Classifier-2					
	FISTA	FISTA	$\kappa = 10$	$\kappa = 1$	$\kappa = 10^{-3}$	$\kappa = 10^{-5}$	$\kappa = 10^{-7}$
784-500-5000-10/ $\lambda = 0.1$	98.31	98.50	97.11	97.18	97.21	98.02	98.36
784-500-5000-10/ $\lambda = 0.05$	98.42	98.46	97.35	97.46	97.24	98.11	98.43
784-500-5000-10/ $\lambda = 0.01$	98.61	98.56	97.20	97.20	97.30	98.31	98.45
784-500-5000-10/ $\lambda = 0.005$	98.51	98.55	97.19	97.22	97.43	98.26	98.54
784-500-5000-10/ $\lambda = 0.001$	98.43	98.58	97.19	97.31	97.37	98.29	98.55
784-500-5000-10/ $\lambda = 0.0005$	98.65	98.55	97.25	97.22	97.38	98.38	98.52
784-500-500-5000-10/ $\lambda = 0.1$	94.89	95.24	94.51	94.26	94.60	96.68	92.44
784-500-500-5000-10/ $\lambda = 0.05$	96.28	96.29	94.81	94.80	94.68	97.19	95.28
784-500-500-5000-10/ $\lambda = 0.01$	98.23	98.45	97.06	97.18	97.39	98.27	97.88
784-500-500-5000-10/ $\lambda = 0.005$	98.50	98.52	97.32	97.31	97.35	98.32	98.42
784-500-500-5000-10/ $\lambda = 0.001$	98.61	98.47	97.19	97.42	97.30	98.41	98.47
784-500-500-5000-10/ $\lambda = 0.0005$	98.34	98.56	97.37	97.16	97.36	98.43	98.48
784-700-5000-10/ $\lambda = 0.1$	98.38	98.42	97.36	97.50	97.16	97.91	98.30
784-700-5000-10/ $\lambda = 0.05$	98.43	98.65	97.44	97.61	97.23	97.90	98.54
784-700-5000-10/ $\lambda = 0.01$	98.57	98.64	97.72	97.66	97.34	98.29	98.63
784-700-5000-10/ $\lambda = 0.005$	98.58	98.62	97.60	97.68	97.33	98.14	98.56
784-700-5000-10/ $\lambda = 0.001$	98.55	98.61	97.55	97.75	97.22	98.11	98.61
784-700-5000-10/ $\lambda = 0.0005$	98.46	98.55	97.83	97.83	97.27	98.13	98.54
784-700-700-5000-10/ $\lambda = 0.1$	94.69	95.59	96.00	95.96	95.47	96.91	93.60
784-700-700-5000-10/ $\lambda = 0.05$	96.44	97.16	95.97	95.92	95.47	97.09	96.44
784-700-700-5000-10/ $\lambda = 0.01$	98.00	98.37	97.70	97.61	97.23	98.20	98.12
784-700-700-5000-10/ $\lambda = 0.005$	98.51	98.59	97.79	97.64	97.25	98.27	98.55
784-700-700-5000-10/ $\lambda = 0.001$	98.63	98.57	97.78	97.65	97.37	98.20	98.45
784-700-700-5000-10/ $\lambda = 0.0005$	98.55	98.72	97.83	97.82	97.15	98.29	98.56
784-500-8000-10/ $\lambda = 0.1$	98.58	98.63	97.50	97.42	97.39	98.38	98.66
784-500-8000-10/ $\lambda = 0.05$	98.59	98.75	97.56	97.42	97.59	98.43	98.76
784-500-8000-10/ $\lambda = 0.01$	98.76	98.87	97.48	97.53	97.61	98.40	98.70
784-500-8000-10/ $\lambda = 0.005$	98.65	98.85	97.64	97.46	97.68	98.52	98.95
784-500-8000-10/ $\lambda = 0.001$	98.80	98.74	97.49	97.51	97.76	98.65	98.77
784-500-8000-10/ $\lambda = 0.0005$	98.73	98.67	97.46	97.55	97.76	98.48	98.82
784-700-8000-10/ $\lambda = 0.1$	98.63	98.73	97.59	97.62	97.38	98.09	98.66
784-700-8000-10/ $\lambda = 0.05$	98.80	98.82	97.70	97.64	97.23	98.25	98.71
784-700-8000-10/ $\lambda = 0.01$	98.77	98.77	97.88	97.97	97.43	98.33	98.75
784-700-8000-10/ $\lambda = 0.005$	98.69	98.77	97.97	97.97	97.56	98.50	98.74
784-700-8000-10/ $\lambda = 0.001$	98.70	98.79	98.13	97.95	97.60	98.40	98.84
784-700-8000-10/ $\lambda = 0.0005$	98.69	98.77	97.95	98.03	97.66	98.57	98.77
784-500-10000-10/ $\lambda = 0.1$	98.76	98.70	97.49	97.44	97.67	98.47	98.65
784-500-10000-10/ $\lambda = 0.05$	98.82	98.90	97.46	97.53	97.45	98.45	98.78
784-500-10000-10/ $\lambda = 0.01$	98.83	98.86	97.58	97.54	97.83	98.66	98.76
784-500-10000-10/ $\lambda = 0.005$	98.84	98.92	97.69	97.69	97.83	98.63	98.79
784-500-10000-10/ $\lambda = 0.001$	98.91	98.84	97.71	97.65	97.76	98.64	98.85
784-500-10000-10/ $\lambda = 0.0005$	98.79	98.77	97.76	97.65	97.78	98.67	98.81

TABLE III
TESTING ACCURACIES (%) OF CLASSIFIER-1 AND CLASSIFIER-2 UNDER DIFFERENT STRUCTURES AND CLUSTERS FOR NORB DATASET.

Network structure	Classifier-1 FISTA	Classifier-2 FISTA	$\kappa = 10^{-5}$
2048-1000-5/ $\lambda = 0.01$	78.12	77.73	78.00
2048-1000-5/ $\lambda = 0.005$	78.53	78.92	77.67
2048-150-3000-5/ $\lambda = 0.01$	89.38	89.26	89.86
2048-150-3000-5/ $\lambda = 0.005$	89.23	87.93	89.28
2048-50-1000-5/ $\lambda = 0.01$	88.36	89.14	87.21
2048-50-1000-5/ $\lambda = 0.005$	89.68	86.47	87.84
2048-500-1000-5/ $\lambda = 0.01$	89.86	90.28	89.21
2048-500-1000-5/ $\lambda = 0.005$	90.89	90.06	88.36
2048-5000-1000-5/ $\lambda = 0.01$	90.29	88.18	87.25
2048-5000-1000-5/ $\lambda = 0.005$	89.40	87.54	87.63
2048-500-500-1000-5/ $\lambda = 0.01$	80.62	69.42	89.51
2048-500-500-1000-5/ $\lambda = 0.005$	79.83	69.61	89.14

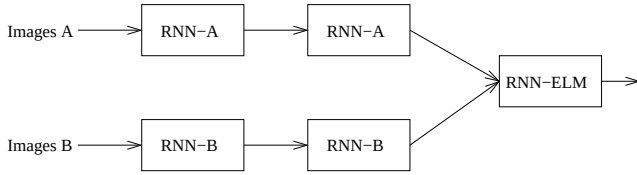


Fig. 7. Framework of how images in NORB dataset flow inside Classifier-3.

A. Classifier-3

In the construction of Classifier-1 and Classifier-2, multi images in each instance are wired together into a single auto-encoder. In Classifier-3, the encoding of each image in each instance are first handled separately, but, in the final two layer (i.e., the RNN-ELM architecture), all encoding information of all images are wired together.

Consider a multi-layer Classifier-3 with structure $2 \times 32 \times 32 - 2 \times 50 - 2000 - 5$ for the NORB dataset. Since each instance in the NORB dataset has two images (Images A and B), we can split the training dataset into two subsets, one containing only Images A denoted by X_A and the other containing only Images B denoted by X_B .

- The first layer consists of two separate RNNs (RNN-A and RNN-B) both with structure $32 \times 32 \rightarrow 50$. The weights $\tilde{W}_A^{(1)}$ in RNN-A are determined by the auto-encoder in Subsection IV-B with X_A , while the weights $\tilde{W}_B^{(1)}$ in RNN-B are determined with X_B . The outputs of RNN-A and RNN-B are respectively $X_A^{(2)} = \zeta(X_A \tilde{W}_A^{(1)})$ and $X_B^{(2)} = \zeta(X_B \tilde{W}_B^{(1)})$.
- The last two layers $2 \times 50 - 2000 - 5$ are the RNN-ELM architecture that takes the whole $X^{(2)} = [X_A^{(2)} X_B^{(2)}]$ as input.

For better illustration, Figure 7 illustrates the framework of how images in the NORB dataset flow inside Classifier-3.

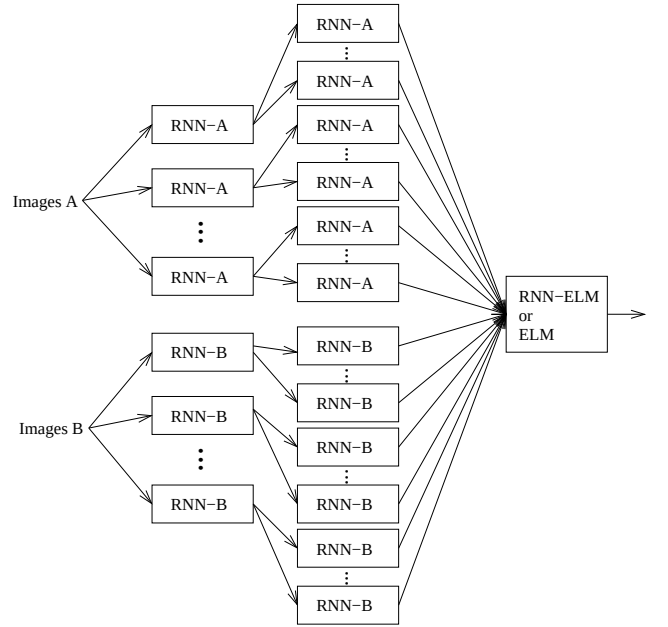


Fig. 8. Framework of how images in NORB dataset flow inside Classifier-4 and Classifier-5.

B. Classifier-4 or Classifier-5

For comparison, we make a small modification to the encoding layers of Classifier-3 (all layers except the last two layers) and develops Classifier-4. In addition, Classifier-5 is modified from Classifier-4 by removing a RNN layer with randomized weights. The inputs for Classifier-4 and 5 are also two subsets X_A and X_B . In the first encoding layer of Classifier-4 and 5, X_A and X_B are respectively encoded by multi separate RNNs. In the successive encoding layers, the outputs of the former RNNs are also encoded by multi separate RNNs. The outputs of the former RNNs are not wired together until reaching the last two layers for Classifier-4 or the last layer for Classifier-5.

Consider a multi-layer Classifier-4 with structure $2 \times 32 \times$

$32 - 2 \times 5 \times 50 - 2000 - 5$ or Classifier-5 with $2 \times 32 \times 32 - 2 \times 5 \times 50 - 5$ for the NORB dataset.

- The first layer consists of 10 separate RNNs (RNN-A i and RNN-B i , $i = 1, \dots, 5$) with structure $32 \times 32 \rightarrow 50$. RNN-A i takes X_A as input and outputs $X_{Ai}^{(2)}$ ($i = 1, \dots, 5$), while RNN-B i takes X_B as input and outputs $X_{Bi}^{(2)}$ ($i = 1, \dots, 5$). The weights of these RNNs are determined by the autoencoder in Subsection IV-B with their corresponding inputs.
- Classifier-4 differs from Classifier-5 in the following.
 - For Classifier-4, the last two layers $2 \times 5 \times 50 - 2000 - 5$ are the RNN-ELM architecture that takes the whole $X^{(2)} = [X_{A1}^{(2)} \dots X_{A5}^{(2)} X_{B1}^{(2)} \dots X_{B5}^{(2)}]$ as input.
 - For Classifier-5, the last layer $2 \times 5 \times 50 - 5$ is an ELM that takes the whole $X^{(2)}$ as input.

For better illustration, Figure 8 illustrates the framework of how images in the NORB dataset flow inside Classifier-4 and Classifier-5.

C. Numerical Results

Under the same computer settings and dataset preprocessing as in Section IV, we conduct numerical experiments using Classifier-3, Classifier-4 and Classifier-5 for classifying the NORB dataset. The results are given in Table IV, where the network structures and clusters are varied in the numerical experiments. It can be seen that the developed classifiers in this section perform better than Classifier-1 and Classifier-2 that wire multi images together in the beginning. The highest testing accuracy 92.21% is achieved by Classifier-3, while, for Classifier-4 and Classifier-5, the highest results are respectively 91.21% and 91.72%. All these results are higher than those by using Classifier-1 and 2, and the ones using Classifier-4 and Classifier-5 are higher than the reported 91.28% in [32], which demonstrates the advantages of Classifier-3, 4 and 5 versus Classifier-1 and 2 for handling the NORB dataset whose instance has multi images.

VI. TEST RESULTS ON LARGE DATASETS

This section investigates the feasibility of utilizing the RNN-cluster-based classifiers for handling big data. We first illustrate how to utilize the technique in the ELM literature [32]–[35] to train the proposed Classifier-2 in a incremental-learning manner. On this basis, we can train Classifier-2 using large datasets with millions of instances chunk by chunk. The numerical results demonstrate the feasibility and efficacy of using the proposed classifier for handling big data.

A. Incremental learning with ELM

The ELM literature [32]–[35] suggests that the ELM output weights denoted by W can be calculated by

$$W = \left(\frac{I}{\kappa} + H^T H \right)^{-1} H^T Y, \quad (24)$$

where H is the hidden-layer output of the ELM, κ is a positive scalar and I is an identity matrix with appropriate dimensions.

This formula can be further developed for incremental learning [34], [35], which is more desirable for handling large datasets. Let $\Phi = H^T H$ and $\Psi = H^T Y$. Suppose $\{H^+, Y^+\}$ is the ELM hidden-layer output of a newly arrival dataset for incremental learning. Then, we have $\Phi^+ = (H^+)^T H^+$ and $\Psi^+ = (H^+)^T Y^+$. And Φ and Ψ are updated via

$$\Phi \leftarrow \Phi + \Phi^+, \Psi \leftarrow \Psi + \Psi^+. \quad (25)$$

Then, the ELM output weights are updated by

$$W = \left(\frac{I}{\kappa} + \Phi \right)^{-1} \Psi. \quad (26)$$

In this algorithm, matrices Φ and Ψ need to be stored. This idea is also applicable to updating (23). Therefore, we can train Classifier-2 with large datasets chunk by chunk using (26) and test its performance for handling big data.

B. Higgs Dataset

The Higgs dataset is from the field of high energy physics [45], obtained from UCI machine learning repository [46]. The problem is to distinguish between a signal process which produces Higgs bosons and a background process which does not based on 28 features (21 low-level features and 7 high-level features). The features, also called attributes, can be both positive and negative. The number of instance in the dataset is 11,000,000, where the first 5,000,000 instances are used for training while the last 500,000 instances are used as a test dataset.

C. Coverttype Dataset

The Coverttype dataset [47] is also downloaded from UCI machine learning repository [46], which is donoted by Jock A. Blackard, Dr. Denis J. Dean and Dr. Charles W. Anderson. This is a classification problem to predict the forest cover type from cartographic variables only (no remotely sensed data). There are 54 input attributes, where 44 of them are binary. The number of forest cover types are 7. In total, there are 581,012 instances in the dataset, where 50,000 of them are used as a test dastate (via random selection) while the rest 531,012 instances are used for training. In addition, the input attributes are normalized into range [0 1].

D. Results on Higgs Dataset

All the following experiments both on the Higgs and Coverttype datasets are conducted in MATLAB R2016a, where we can utilize its datastore function that is able to read from a large dataset chunk by chunk. In addition, parameters for RNN clusters are set as $n = 20$, $p = 0.05$ and $\lambda = \lambda^- = \lambda^+ = 0.005$.

1) *Experiment 1:* In this experiment, we use Classifier-2 with structure $2 \times 28 - 1000 - 10000 - 2$. To fit the data into the RNN clusters, we simply use $[\max(X, 0) - \min(X, 0)]$ as the input attributes. Parameter $\kappa = 0.1$ for the auto-encoder part, while $\kappa = 1/2^{-20}$ for the ELM part. Incremental learning are used based on the procedure discussed in Subsection VI-A. The testing accuracy obtained is 64.86%.

TABLE IV
TESTING ACCURACIES (%) OF CLASSIFIER-3, 4 AND 5 UNDER DIFFERENT STRUCTURES AND CLUSTERS FOR NORB DATASET.

Classifier	Network structure	Accuracy
Classifier-3	$2 \times 32 \times 32 - 2 \times 1000 - 10000 - 5/\lambda = 0.01$	92.12
Classifier-3	$2 \times 32 \times 32 - 2 \times 1000 - 10000 - 5/\lambda = 0.005$	91.93
Classifier-4	$2 \times 32 \times 32 - 80 \times 500 - 10 \times 10 - 10000 - 5/\lambda = 0.01$	90.70
Classifier-4	$2 \times 32 \times 32 - 80 \times 100 - 10 \times 10 - 8000 - 5/\lambda = 0.01$	91.21
Classifier-4	$2 \times 32 \times 32 - 80 \times 100 - 10 \times 10 - 8000 - 5/\lambda = 0.005$	90.71
Classifier-5	$2 \times 32 \times 32 - 10 \times 100 - 80 \times 10 - 5/\lambda = 0.01$	91.35
Classifier-5	$2 \times 32 \times 32 - 50 \times 100 - 10 \times 10 - 5/\lambda = 0.005$	91.40
Classifier-5	$2 \times 32 \times 32 - 50 \times 500 - 10 \times 10 - 5/\lambda = 0.01$	91.11
Classifier-5	$2 \times 32 \times 32 - 80 \times 100 - 10 \times 5 - 5/\lambda = 0.005$	91.72
Classifier-5	$2 \times 32 \times 32 - 80 \times 100 - 10 \times 10 - 5/\lambda = 0.01$	91.46
Classifier-5	$2 \times 32 \times 32 - 80 \times 100 - 10 \times 10 - 5/\lambda = 0.005$	91.26

2) *Experiment 2*: The network structures used are $2 \times 28 - 5000 - 2$ and $28 - 5000 - 2$, which is the pure RNN-ELM architecture. For the first structure, we use $[\max(X, 0) - \min(X, 0)]$ as the input, while for the second one we simply use $\max(X, 0)$ to remove the negative part of the input attributes. Parameter $\kappa = 1/2^{-30}$ is used for updating the ELM output weights. The testing accuracy obtained by using $2 \times 28 - 5000 - 2$ and $28 - 5000 - 2$ are respectively 69.79% and 70.41%.

3) *Experiment 3*: The network structure is $2 \times 7 - 10000 - 2$. We use only the last 7 high-level feature for input and $[\max(X, 0) - \min(X, 0)]$ to avoid negativity. Parameter $\kappa = 1/2^{-25}$. The testing accuracy obtained is 69.57%.

Comparing the results of experiments 1 to 3, we could see that the pure RNN-ELM architecture performs better than Classifier-2 with an auto-encoder for this dataset.

E. Results on Coverttype Dataset

1) *Experiment 1*: Classifier-2 with structures $54 - 1000 - 5000 - 7$ and $54 - 2000 - 8000 - 7$ is used in the experiment. For both structures, $\kappa = 10^{-5}$ for the auto-encoder part, while $\kappa = 1/2^{-30}$ for the ELM part. The testing accuracies obtained by structures $54 - 1000 - 5000 - 7$ and $54 - 2000 - 8000 - 7$ are respectively 72.21% and 71.90%.

2) *Experiment 2*: We revert to the pure RNN-ELM architectures $54 - 5000 - 7$ ($\kappa = 1/2^{-30}$) and $54 - 10000 - 7$ ($\kappa = 1/2^{-25}$). The testing accuracies obtained are respectively 83.79% and 84.21%.

Comparing the results of experiments 1 to 2, the pure RNN-ELM architecture performs better than Classifier-2 for the Coverttype dataset. Together with the results on the MNIST, NORB, Higgs datasets, we may conclude that, for datasets with relatively high attribute dimensions (e.g., image datasets), the RNN-ELM classifiers with auto-encoders perform better than the pure RNN-ELM architecture, while, for datasets with low attribute dimension, the pure RNN-ELM architecture could be a better option.

VII. CONCLUSIONS

This paper has developed a RNN-ELM deep learning architecture that combines spiking random neural networks and extreme learning machines. We have considered very large networks with hundreds of cells in each layer, and have exploited clustered neurons in the RNN layers.

The main contributions of this paper are as follows.

- A mathematical model for soma to soma interactions has been derived, based on which a special type of clusters has been investigated and one type of biology-motivated flexible activation function has been developed. It bridges the gap between the area of spiking random neural network that is supported by a rigorous theoretical framework [13]–[19] and the area of deep learning.
- Utilizing the well-developed ELM techniques [32]–[35], an effective deep-learning architecture has been built up on basis of the RNN clusters.

Our main experimental results show that:

- On a standard and significant problem of visual character recognition, the RNN-ELM classifiers provides better recognition performance than extreme learning machines on their own, reaching recognition ratios that exceed 98.90% that is a very high accuracy for a fully-connected multi-layer neural network.
- On the NORB dataset for 3D object recognition, the RNN-ELM classifiers produce 92.21% testing accuracy which is better than the reported 91.28% in [32].
- Results on the Higgs (5 million instances used) and Coverttype (0.5 million instances used) datasets demonstrate that the RNN-ELM classifiers are feasible for handling big data.

In future work, we plan to examine the value of recurrent networks, and we will address more particularly the types of recurrent networks that may be used and also we will exploit the asymptotic properties of RNN clusters to render the learning process more efficient.

APPENDIX: ANALYSIS FOR CONDITIONS OF (4)

From (4), we have

$$q^2(pn\lambda^- + pnq_uw_u^- - p\lambda^- - pq_uw_u^-) \quad (27)$$

$$+q(\lambda^+p + rp - \lambda^-n - r - nq_uw_u^- - \lambda^+pn - npr) \quad (28)$$

$$+\lambda^+n = 0. \quad (29)$$

For illustration convenience, let

$$y(q) = aq^2 + bq + c, \quad (30)$$

where

$$a = pn\lambda^- + pnq_uw_u^- - p\lambda^- - pq_uw_u^-, \quad (31)$$

$$b = \lambda^+p + rp - \lambda^-n - r - nq_uw_u^- - \lambda^+pn - npr, \quad (32)$$

$$c = \lambda^+n. \quad (33)$$

Let us deduce the conditions needed for $y(q)$ to two positive roots with one of them being less than 1 as follows.

- If the conditions $n \geq 2$ and $\lambda^- > 0$ holds, then $pn\lambda^- - p\lambda^- > 0$ and $pnq_uw_u^- - pq_uw_u^- > 0$. Then, $a > 0$.
- In addition, $\lambda^+p - \lambda^+pn < 0$. Since p is a probability, $p \leq 1$ holds. Then, $rp - r \leq 0$. Then, $b < 0$. Then, $-b/2a > 0$.
- If the condition $\lambda^+ > 0$ holds, then we have $y(q = 0) = c = \lambda^+n > 0$.
- Let us analyze $y(q = 1) = a + b + c = (np - p - n)(\lambda^- + q_uw_u^- - \lambda^+) + (p - np - 1)r$. It is clear that $np - p - n < 0$ and $p - np - 1 < 0$. Then, if the condition $r > 0$ holds, $(p - np - 1)r < 0$. Then, if $\lambda^- + q_uw_u^- \geq \lambda^+$ holds, then $y(q = 1) < 0$. Since $q_uw_u^-$ is the input from outside the cluster that can be out of control of the cluster, we loose $\lambda^- + q_uw_u^- \geq \lambda^+$ as the condition $\lambda^- \geq \lambda^+$.

Until now, the second degree parabola $y(q)$ has two positive roots with one of them being less than 1. It can be concluded that the conditions are $n \geq 2$, $p \leq 1$, $\lambda^+ > 0$, $\lambda^- > 0$, $r > 0$ and $\lambda^- \geq \lambda^+$. The analysis is thus completed.

ACKNOWLEDGEMENT

We gratefully acknowledge the support of the EC 7th Framework Program PANACEA Project, Grant Agreement No. 610764, to Imperial College London.

REFERENCES

- [1] R. Raina, A. Madhavan, and A. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proc. 26th Int. Conf. on Machine Learning*. ACM, 2009, pp. 873–880.
- [2] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 22, pp. 3207–3220, 2010.
- [3] J. Doncel, U. Ayesta, O. Brun, and B. Prabhu, "A resource-sharing game with relative priorities," *Performance Evaluation*, vol. 79, p. 287305, September 2014.
- [4] G. Gorbil, D. Perez, and E. H. Cuesta, "Principles of pervasive monitoring," in *ISCIS 2014*, T. Czachorski and E. Gelenbe, Eds. Springer Verlag, 2014, pp. 117–124.
- [5] P. D. Sanzo, A. Pellegrini, and D. R. Avresky, "Machine learning for achieving self-* properties and seamless execution of applications in the cloud," in *NCCA 2015, Munich, Germany*. IEEEExplore, 2015, pp. 51–58.
- [6] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *Cloud Computing, IEEE Transactions on*, vol. PP, 2015.

- [7] W. Gerstner, "Spiking neurons," in *Pulsed Neural Networks*, W. Maass and C. M. Bishop, Eds. MIT Press, 2001, p. 377.
- [8] Y. Gu, Y. Chen, X. Zhang, G. W. Li, C. Y. Wang, and L.-Y. M. Huang, "Neuronal soma-satellite glial cell interactions in sensory ganglia and the participation of purinergic receptors," *Neuron Glia Biology*, vol. 6, no. 1, p. 5362, February 2010.
- [9] E. S. Krames, P. H. Peckham, and A. R. Rezai, Eds., *Neuromodulation*. Academic Press, 2009, vol. 1-2.
- [10] E. A. Newman, "New roles for astrocytes: Regulation of synaptic transmission," *TRENDS in Neurosciences*, vol. 10, no. 26, p. 536542, 2001.
- [11] M. Arbib, Ed., *The Handbook of Brain Theory and Neural Networks*. MIT Press, 2003.
- [12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.
- [13] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural computation*, vol. 1, no. 4, pp. 502–510, 1989.
- [14] —, "Stability of the random neural network model," *Neural computation*, vol. 2, no. 2, pp. 239–247, 1990.
- [15] —, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, 1993.
- [16] E. Gelenbe and J.-M. Fournau, "Random neural networks with multiple classes of signals," *Neural computation*, vol. 11, no. 4, pp. 953–963, 1999.
- [17] E. Gelenbe, "G-networks with triggered customer movement," *Journal of Applied Probability*, pp. 742–748, 1993.
- [18] E. Gelenbe and S. Timotheou, "Random neural networks with synchronized interactions," *Neural Computation*, vol. 20, no. 9, pp. 2308–2324, 2008.
- [19] S. Timotheou, "The random neural network: a survey," *The computer journal*, vol. 53, no. 3, pp. 251–267, 2010.
- [20] J. Medhi, *Stochastic Processes*. New Age International, 1994.
- [21] E. Gelenbe and C. Cramer, "Oscillatory corticothalamic response to somatosensory input," *Biosystems*, vol. 48, no. 1, pp. 67–75, 1998.
- [22] E. Gelenbe, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, pp. 154–164, 1993.
- [23] E. Gelenbe, V. Koubi, and F. Pekergin, "Dynamical random neural network approach to the traveling salesman problem," in *Proceedings IEEE Symp. Systems, Man and Cybernetics*. IEEE, 1993, p. 630635.
- [24] E. Gelenbe and T. Koçak, "Area-based results for mine detection," *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 38, no. 1, pp. 12–24, 2000.
- [25] C. Cramer, E. Gelenbe, and H. Bakirciloglu, "Low bit-rate video compression with neural networks and temporal subsampling," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1529–1543, 1996.
- [26] C. E. Cramer and E. Gelenbe, "Video quality and traffic qos in learning-based subsampled and receiver-interpolated video sequences," *Selected Areas in Communications, IEEE Journal on*, vol. 18, no. 2, pp. 150–167, 2000.
- [27] H. M. Abdelbaki, K. Hussain, and E. Gelenbe, "A laser intensity image based automatic vehicle classification system," in *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*. IEEE, 2001, pp. 460–465.
- [28] E. Gelenbe and K. F. Hussain, "Learning in the multiple class random neural network," *Neural Networks, IEEE Transactions on*, vol. 13, no. 6, pp. 1257–1267, 2002.
- [29] A. Filippopolitis, L. Hey, G. Loukas, E. Gelenbe, and S. Timotheou, "Emergency response simulation using wireless sensor networks," in *Proceedings of the 1st international conference on Ambient media and systems*, 2008, p. 21.
- [30] E. Gelenbe, "Steps toward self-aware networks," *Communications of the ACM*, vol. 52, no. 7, pp. 66–75, 2009.
- [31] E. Gelenbe and F.-J. Wu, "Large scale simulation for human evacuation and rescue," *Computers & Mathematics with Applications*, vol. 64, no. 12, pp. 3869–3880, 2012.
- [32] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," to appear in *IEEE Transactions on Neural Networks and Learning Systems*, May 2015.
- [33] L. L. C. Kasun, H. Zhou, and G.-B. Huang, "Representational learning with extreme learning machine for big data," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 31–34, 2013.
- [34] J. Xin, Z. Wang, L. Qu, and G. Wang, "Elastic extreme learning machine for big data classification," *Neurocomputing*, vol. 149, pp. 464–471, 2015.

- [35] A. Akusok, K.-M. Björk, Y. Miche, and A. Lendasse, “High-performance extreme learning machines: a complete toolbox for big data applications,” *IEEE Access*, vol. 3, pp. 1011–1025, 2015.
- [36] E. Gelenbe and J.-M. Fourneau, “Random neural networks with multiple classes of signals,” *Neural computation*, vol. 11, no. 4, pp. 953–963, 1999.
- [37] E. Gelenbe, “The first decade of g-networks,” *European Journal of Operational Research*, vol. 126, no. 2, pp. 231–232, 2000.
- [38] X. Liu, S. Yan, and H. Jin, “Projective nonnegative graph embedding,” *Image Processing, IEEE Transactions on*, vol. 19, no. 5, pp. 1126–1137, 2010.
- [39] Y. Zhang, Y. Yin, D. Guo, X. Yu, and L. Xiao, “Cross-validation based weights and structure determination of chebyshev-polynomial neural networks for pattern classification,” *Pattern Recognition*, vol. 47, no. 10, pp. 3414–3428, 2014.
- [40] Y. Zhang, Y. Yin, X. Yu, D. Guo, and L. Xiao, “Pruning-included weights and structure determination of 2-input neuronet using chebyshev polynomials of class 1,” in *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*. IEEE, 2012, pp. 700–705.
- [41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [42] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.
- [43] Y. LeCun, F. J. Huang, and L. Bottou, “Learning methods for generic object recognition with invariance to pose and lighting,” in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. II–97–104.
- [44] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [45] P. Baldi, P. Sadowski, and D. Whiteson, “Searching for exotic particles in high-energy physics with deep learning,” *Nature communications*, vol. 5, 2014.
- [46] M. Lichman, “Uci machine learning repository,” 2013. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [47] J. A. Blackard and D. J. Dean, “Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables,” *Computers and electronics in agriculture*, vol. 24, no. 3, pp. 131–151, 1999.