IJCNN 2019. International Joint Conference on Neural Networks. Budapest, Hungary. 14-19 July 2019

Non-negative Autoencoder with Simplified Random Neural Network

Yonghua Yin
Intelligent Systems and Networks Group
Electrical & Electronic Engineering Department
Imperial College, London SW7 2AZ, UK

Erol Gelenbe, *Fellow, IEEE*
Institute of Theoretical & Applied Informatics
Polish Academy of Sciences, 44100 PL

***Abstract*—A new shallow multi-layer auto-encoder that combines the spiking Random Neural Network (RNN) with the network architecture typically used in deep-learning, is proposed with a learning algorithm inspired by non-negative matrix factorization which satisfies the non-negative probability constraints of the RNN. Auto-encoders equipped with this learning algorithm are tested on typical images including the MNIST, Yale face and CIFAR-10 datasets, and also using 16 real-world datasets from different areas, exhibiting the desired high learning and recognition accuracy. Montecarlo simulations of the stochastic spiking behaviour of this RNN auto-encoder have also been carried out, showing that it can be implemented in a highly parallel manner to achieve substantial speed improvements.**

I. Introduction

A mathematical tool that has existed since 1989 [1], [2], but is less well known in the machine-learning community, is the Random Neural Network (RNN), which is a stochastic, recurrent, integer-state "integrate and fire" system which was developed to mimic the behaviour of certain biological neurons. The power of the RNN lies in the fact that, in steady state, the stochastic spiking behavior of the network has been proved [3] to have a remarkable property called "product form", whereby the joint state probability distribution of the network is the product of the marginal probability distributions of the individual neurons. Moreover, the local state of each neuron can be computed from the flows of inhibitory or excitatory spikes reaching each neuron. As a consequence, the recurrent RNN model is easily solvable by iterations on a system of non-linear equations.

The RNN is used in many applications [4]–[6] that exploit its product form solution and recurrent structure, and for deep learning [7]–[10]. The usage of the resultant deep learning tools includes predicting the toxicity of compounds [11], [12] and detecting network attacks [13].

Deep learning has achieved great success in machine learning [14]–[16]. It utilizes multi-processing layers to extract high-level representations from raw data. Pre-training a multi-layer network, layer by layer [17] has been widely used. Furthermore, the stochastic gradient descent (SGD) procedure provides a practical choice for handling large datasets [18]. Another popular topic in machine learning is the non-negative matrix factorization (NMF) [19]–[22], which learns partial representations of raw data, and in [19] it was suggested that the perception of the whole in the brain may be based on these part-based representations, based on the physiological evidence [23], leading to simple yet effective network update rules. A comprehensive review on the NMF can be found in [20].

This paper first exploits the structure of the RNN equations as a quasi-linear structure. Using it in the feed-forward case, an RNN-based shallow non-negative auto-encoder is constructed. Then, this shallow auto-encoder is stacked into a multi-layer feed-forward auto-encoder following the network architecture in the deep learning area [14]–[17]. Since connecting weights in the RNN are products of firing rates and transition probabilities, they are subject to the constraints of nonnegativity and that the sum of probabilities is no larger than 1, which are called the RNN constraints in this paper. In view of that, the conventional gradient descent is not applicable for training such an auto-encoder. By adapting the update rules from non-negative graph embedding [24] that is closely related to NMF, applicable update rules are developed for the auto-encoder that satisfy the first RNN constraint of nonnegativity. For the second RNN constraint, we impose a check-and-adjust procedure into the iterative learning process of the learning algorithms. The training procedure of SGD is also adapted into the algorithms. The efficacy of the non-negative auto-encoders equipped with the learning algorithms is well verified via numerical experiments on both typical image datasets including the MNIST [25], Yale face [26] and CIFAR-10 [27] datasets and 16 real-world datasets in different areas from the UCI machine learning repository [28]. Then, we simulate the spiking behaviors of the RNN-based auto-encoder, where simulation results conform well with the corresponding numerical results, therefore demonstrating that this non-negative auto-encoder can be implemented in a highly-distributed and parallel manner.

II. A Quasi-linear Simplified Random Neural Network

An arbitrary neuron in the RNN can receive excitatory or inhibitory spikes from external sources, in which case they arrive according to independent Poisson processes. Excitatory or inhibitory spikes can also arrive from other neurons to a given neuron, in which case they arrive when the sending neuron fires, which happens only if that neuron's input state is positive (i.e. the neuron is excited) and inter-firing intervals from the same neuron v are exponentially distributed random

978-1-7281-2009-6/\$31.00 ©2019 IEEE

variables with rate $r_v \geq 0$. Since the firing times depend on the internal state of the sensing neuron, the arrival process of neurons from other cells is not in general Poisson. From the preceding assumptions it was proved in [3] that for an arbitrary N neuron RNN, which may or may not be recurrent (i.e. containing feedback loops), the probability in steady-state that any cell h, located anywhere in the network, is excited is given by the expression:

$$q_h = \min\left(\frac{\lambda_h^+ + \sum_{v=1}^{N} q_v r_v p_{vh}^+}{r_h + \lambda_h^- + \sum_{v=1}^{N} q_v r_v p_{vh}^-}, 1\right), \quad (1)$$

for $h = 1, \ \dots \ , N$, where $p_{vh}^+, \ p_{vh}^-$ are the probabilities that cell v may send excitatory or inhibitory spikes to cell h, and $\lambda_h^+, \ \lambda_h^-$ are the external arrival rates of excitatory and inhibitory spikes to neuron h. Note that $\min(a, b)$ is a element-wise operation whose output is the smaller one between a and b. In [3], it was shown that the system of N non-linear equations (1) have a solution which is unique.

Before adapting the RNN as a non-negative auto-encoder (Section III), we will simplify the recurrent RNN model into the feed-forward structure shown in Figure 1. The simplified RNN has an input layer and a hidden layer. The V input neurons receive excitatory spikes from the outside world, and they fire excitatory spikes to the H hidden neurons.

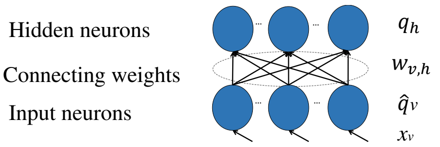


Fig. 1. Brief model structure of the quasi-linear RNN.

Let us denote by $\hat{q}_v$ the probability that the vth input neuron ($v = 1, \cdots, V$) is excited and q_h the probability that the hth hidden neuron ($h = 1, \cdots, H$) is excited. According to [1] and (1), they are given by $\hat{q}_v = \min(\hat{\Lambda}_v^+ / \hat{r}_v, 1)$, and $q_h = \min(\Lambda_h^+ / r_h, 1)$, where the quantities $\hat{\Lambda}_v^+$ and Λ_h^+ represent the total average arrival rates of excitatory spikes, $\hat{r}_v$ and r_h represent the firing rates of the neurons. Neurons in this model interact with each other in the following manner, where $h = 1, \cdots, H$ and $v = 1, \cdots, V$. When the vth input neuron fires, it sends excitatory spikes to the hth hidden neuron with probability $p_{v,h}^+ \geq 0$. Clearly, $\sum_{h=1}^{H} p_{v,h}^+ \leq 1$.

- The vth input neuron receives excitatory spikes from the outside world with rate $x_v \geq 0$.
- When the hth hidden neuron fires, it sends excitatory spikes outside the network.

Let us denote $w_{v,h} = p_{v,h}^+ \hat{r}_v$. For simplicity, let us set the firing rates of all neurons to $\hat{r}_v = r_h = 1$ or that $\sum_{h=1}^{H} w_{v,h} \leq 1$. Then, $\hat{\Lambda}_v^+ = x_v$, $\hat{r}_v = 1$, $\Lambda_h^+ = \sum_{v=1}^{V} w_{v,h} \hat{q}_v$, and using the fact that q_h, q_v are probabilities, we can write: $\hat{q}_v = \min(x_v, 1)$, $q_h = \min(\sum_{v=1}^{V} w_{v,h} \hat{q}_v, 1)$, subject to $\sum_{h=1}^{H} w_{v,h} \leq 1$, from which we can that this simplified RNN is quasi linear. For the network shown in Figure 1, we call it a quasi-linear RNN (LRNN).

III. SHALLOW NON-NEGATIVE LRNN AUTO-ENCODER

We add an output layer with O neurons on top of the hidden layer of the LRNN shown in Figure 1 to construct a shallow non-negative LRNN auto-encoder. Let $\overline{q}_o$ denote the probability that the oth output neuron is excited, and the oth output neurons interact with the LRNN in the following manner, where $o = 1, \cdots, O$.

- When the hth hidden neuron fires, it sends excitatory spikes to the oth output neuron with probability $\overline{p}_{h,o}^+ \geq 0$. Also, $\sum_{o=1}^{O} \overline{p}_{h,o}^+ \leq 1$.
- The firing rate of the oth output neuron $\overline{r}_o = 1$.

Let $\overline{w}_{h,o} = \overline{p}_{h,o}^+ r_h = \overline{p}_{h,o}^+$. Then, $\sum_{o=1}^{O} \overline{w}_{h,o} \leq 1$. The shallow LRNN auto-encoder is described by

$$\hat{q}_v = \min(x_v, 1), q_h = \min(\textstyle\sum_{v=1}^{V} w_{v,h} \hat{q}_v, 1),$$
$$\overline{q}_o = \min(\textstyle\sum_{h=1}^{H} \overline{w}_{h,o} q_h, 1),$$

where $O = V$ and the input, hidden and output layers are the visual, encoding and decoding layers.

Algorithm 1 Procedure for training a shallow non-negative LRNN auto-encoder (2)

```
Randomly initialize $W$ and $\overline{W}$ that satisfy RNN constraints
while terminal condition is not satisfied do
    for each minibatch $\bar{X}$ do
        update $W$ with (3)
        for $v = 1, \cdots, V$ do
            if $\sum_{h=1}^{H} w_{v,h} > 1$
                $w_{v,h} \leftarrow w_{v,h} / \sum_{h=1}^{H} w_{v,h}$,
                for $h = 1, \cdots, H$
        $W \leftarrow W / \max(\bar{X} W)$
        update $\overline{W}$ with (4)
        for $h = 1, \cdots, H$ do
            if $\sum_{o=1}^{O} \overline{w}_{h,o} > 1$
                $\overline{w}_{h,o} \leftarrow \overline{w}_{h,o} / \sum_{o=1}^{O} \overline{w}_{h,o}$,
                for $o = 1, \cdots, O$
        $H = \min(\bar{X} W, 1)$
        $\overline{W} \leftarrow \overline{W} / \max(H \overline{W})$
```

Suppose there is a dataset represented by a non-negative $D \times V$ matrix $X = [x_{d,v}]$, where D is the number of instances, each instance has V attributes and $x_{d,v}$ is the vth attribute of the dth instance. We import X into the input layer of the LRNN auto-encoder. Let $\hat{q}_{d,v}$, $q_{d,h}$ and $\overline{q}_{d,o}$ respectively denote the values of $\hat{q}_v$, q_d and $\overline{q}_o$ for the dth instance.

Let a $D \times V$-matrix $\hat{Q} = [\hat{q}_{d,v}]$, a $D \times H$-matrix $Q = [q_{d,h}]$, a $D \times O$-matrix $\overline{Q} = [\overline{q}_{d,o}]$, a $V \times U$-matrix $W = [w_{v,h}]$ and a $H \times O$-matrix $\overline{W} = [\overline{w}_{h,o}]$. Then, the shallow LRNN auto-encoder can be rewritten as the following matrix manner:

$$\hat{Q} = \min(X, 1), \ Q = \min(\hat{Q} W, 1), \ \overline{Q} = \min(Q \overline{W}, 1), \quad (2)$$

subject to the RNN constraints $W \geq 0$, $\overline{W} \geq 0$, $\sum_{h=1}^{H} w_{v,h} \leq 1$ and $\sum_{o=1}^{O} \overline{w}_{h,o} \leq 1$. The problem for the autoecoder to learn the dataset X can be described as

$$\arg\min_{W, \overline{W}} ||X - \overline{Q}||^2,$$
$$\text{s.t. } W \geq 0, \overline{W} \geq 0, \textstyle\sum_{h=1}^{H} w_{v,h} \leq 1, \sum_{o=1}^{O} \overline{w}_{h,o} \leq 1.$$

We use the following update rules to solve this problem,

Algorithm 2 Proceduce for training a multi-layer LRNN-based non-negatvie autoencder (5)

```
$X_1 = X$
for $m = 1, \cdots, M$ do
    Train $W_m$ and $\overline{W}_{M-m+1}$ with Algorithm 1 that takes $X_m$ as input dataset
    if $m \neq M$ do
        $X_{m+1} = \min(X_m W_m, 1)$
```

which are simplified from Liu's work [24]:

$$w_{v,h} \leftarrow w_{v,h} \frac{(X^{\mathrm{T}} X \overline{W}^{\mathrm{T}})_{v,h}}{(X^{\mathrm{T}} X W \overline{W}\, \overline{W}^{\mathrm{T}})_{v,h}}, \tag{3}$$

$$\overline{w}_{h,o} \leftarrow \overline{w}_{h,o} \frac{(W^{\mathrm{T}} X^{\mathrm{T}} X)_{h,o}}{(W^{\mathrm{T}} X^{\mathrm{T}} X W \overline{W})_{h,o}}, \tag{4}$$

where the symbol $(\cdot)_{v,h}$ denotes the element in the vth row and hth column of a matrix. Note that, to avoid the division-by-zero problem, zero elements in the denominators of (3) and (4) are replaced with tiny positive values, (e.g., "eps" in MATLAB). After each update, adjustments need to be made such that W and $\overline{W}$ satisfy the RNN constraints. The procedure to train the auto-encoder (2) is given in Algorithm 1, where the operation $\max(W)$ produces the maximal element in W, the operations of $w_{v,h} \leftarrow w_{v,h}/\sum_{h=1}^{H} w_{v,h}$ and $\overline{w}_{h,o} \leftarrow \overline{w}_{h,o}/\sum_{o=1}^{O} \overline{w}_{h,o}$ guarantee that the weights satisfy the RNN constraints, and the operations $W \leftarrow W/\max(\bar{X}W)$ and $\overline{W} \leftarrow \overline{W}/\max(H\overline{W})$ normalize the weights to reduce the number of neurons that are saturated.

IV. MULTI-LAYER NON-NEGATIVE LRNN AUTO-ENCODER

We stack multi LRNNs to build a multi-layer non-negative LRNN auto-encoder. Suppose the multi-layer auto-encoder has a visual layer, M encoding layers and M decoding layer ($M \geq 2$), and they are connected in series with excitatory weights W_m and $\overline{W}$ with $m = 1, \cdots, M$. We import a dataset X into the visual layer of the auto-encoder. Let H_m and O_m denote the numbers of neurons in the mth encoding layer and decoding layer, respectively. For the auto-encoder, $V = O_M$, $H_m = O_{M-m}$ with $m = 1, \cdots, M-1$.

Let $\hat{Q}$ denote the state of the visual layer, Q_m denote the state of the mth encoding layer and $\overline{Q}_m$ denote the state of the mth decoding layer. Then, the multi-layer LRNN auto-encoder is described by

$$\begin{aligned} &\hat{Q} = \min(X, 1), \\ &Q_1 = \min(\hat{Q} W_1, 1),\ Q_m = \min(Q_{m-1} W_m, 1), \\ &\overline{Q}_1 = \min(Q_M \overline{W}_1, 1),\ \overline{Q}_m = \min(\overline{Q}_{m-1} \overline{W}_m, 1), \end{aligned} \tag{5}$$

with $m = 2, \cdots, M$. The RNN constraints for (5) are $W_m \geq 0$, $\overline{W}_m \geq 0$ and the summation of each row in W_m and $\overline{W}_m$ is not larger than 1, where $m = 1, \cdots, M$. The problem for the multi-layer LRNN auto-encoder (5) to learn dataset X can be described as

$$\arg\min_{W_m, \overline{W}_m} ||X - \overline{Q}_M||^2,$$

subject to the RNN constraints, where $m = 1, \cdots, M$. The procedure to train the multi-layer non-negative LRNN auto-encoder (5) is given in Algorithm 2.

Algorithm 3 Proceduce for training a multi-layer LRNN-based non-negative auto-encoder (5) (minibatch manner)

```
Randomly initialize $W_m$ and $\overline{W}_m$ that satisfy RNN constraints (with $m = 1, \cdots, M$)
while terminal condition is not satisfied do
    for each minibatch $\bar{X}$ do
        for $m = 1, \cdots, M$ do
            update $W_m$ with (6)
            adjust $W_m$ to satisfy RNN constraints
            normalize $W_m$ subject to $\bar{X}$
            update $\overline{W}_m$ with (7)
            adjust $\overline{W}_m$ to satisfy RNN constraints
            normalize $\overline{W}_m$ subject to $\bar{X}$
```

To avoid loading the whole dataset into the computer memory, we could also use Algorithm 3 to train the auto-encoder, where the update rules could be

$$\begin{aligned} &W_1 \leftarrow W_1 \odot \frac{\hat{Q}^{\mathrm{T}} \hat{Q} \overline{W}_M^{\mathrm{T}}}{\hat{Q}^{\mathrm{T}} \hat{Q} W_1 \overline{W}_M \overline{W}_M^{\mathrm{T}}}, \\ &W_m \leftarrow W_m \odot \frac{Q_{m-1}^{\mathrm{T}} Q_{m-1} \overline{W}_{M-m+1}^{\mathrm{T}}}{Q_{m-1}^{\mathrm{T}} Q_{m-1} W_m \overline{W}_{M-m+1} \overline{W}_{M-m+1}^{\mathrm{T}}}, \end{aligned} \tag{6}$$

$$\begin{aligned} &\overline{W}_M \leftarrow \overline{W}_M \odot \frac{W_1^{\mathrm{T}} \hat{Q}^{\mathrm{T}} \hat{Q}}{W_1^{\mathrm{T}} \hat{Q}^{\mathrm{T}} \hat{Q} W_1 \overline{W}_M}, \\ &\overline{W}_{M-m+1} \leftarrow \overline{W}_{M-m+1} \odot \frac{W_m^{\mathrm{T}} Q_{m-1}^{\mathrm{T}} Q_{m-1}}{W_m^{\mathrm{T}} Q_{m-1}^{\mathrm{T}} Q_{m-1} W_m \overline{W}_{M-m+1}}, \end{aligned} \tag{7}$$

with $m = 2, \cdots, M$ and the operation $\odot$ denoting element-wise product of two matrices. To avoid the division-by-zero problem, zero elements in denominators of (6) and (7) are replaced with tiny positive values. The operations of adjusting the weights to satisfy the RNN constraints and normalizing the weights are the same as those in Algorithm 1.

V. NUMERICAL EXPERIMENTS

A. Datasets

MNIST: The MNIST dataset of handwritten digits [25] contains 60,000 and 10,000 images in the training and test dataset. The number of input attributes is 784 (28×28 images), which are in $[0, 1]$.

Yale face: This database (http://vision.ucsd.edu/content/yale-face-database) contains 165 gray scale images of 15 individuals. Here we use the pre-processed dataset from [26], where each image is resized as 32×32 (1024 pixels).

CIFAR-10: The dataset consists of 60,000 32×32 colour images [27]. Each image has 3072 attributes. It contains 50,000 and 10,000 images in the training and test dataset.

UCI real-world datasets: In addition to image datasets, we also conduct numerical experiments on different real-world datasets in different areas from the UCI machine learning repository [28]. The names, attribute numbers and instance numbers of these datasets are listed in Table I.

TABLE I
FEATURES OF DIFFERENT UCI REAL-WORLD DATASETS FROM DIFFERENT AREAS

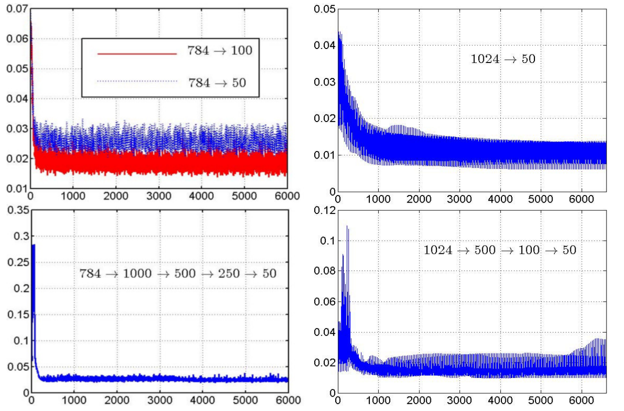
Dataset	Inputs	Size
Iris	4	150
Teaching Assistant Evaluation (TAE)	5	151
Liver Disorders (LD)	5	345
Seeds	7	210
Pima Indians Diabetes (PID)	8	768
Breast Cancer Wisconsin (BC) [29]	9	699
Glass	9	214
Wine	13	178
Zoo	16	100
Parkinsons [30]	22	195
Wall-Following Robot Navigation (WFRN) [31]	24	5456
Ionosphere [32]	34	351
Soybean Large (SL)	35	186
First-Order Theorem Proving (FOTP) [33]	51	6118
Sonar [34]	60	208
Cardiac Arrhythmia (CA) [35]	279	452

B. *Convergence and Reconstruction Performance*

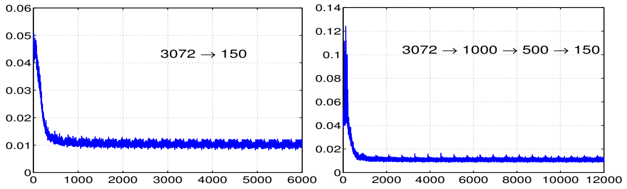
Results of MNIST: Let us first test the convergence and reconstruction performance of the shallow non-negative LRNN auto-encoder. We use structures of $784 \rightarrow 100$ (for simplicity, we use the encoding part to represent an auto-encoder) and $784 \rightarrow 50$ and the MNIST dataset for experiments. The whole training dataset of 60,000 images is used for training. Figure 2(a) shows the curves of training error (mean square error) versus the number of iterations, where, in each iteration, a minibatch of size 100 is handled. Then, we use a multi-layer non-negative LRNN auto-encoder with structure $784 \rightarrow 1000 \rightarrow 500 \rightarrow 250 \rightarrow 50$, and the corresponding curve of training error versus iterations is also given in Figure 2(a). It can be seen from Figure 2(a) that reconstruction errors using the LRNN auto-encoders equipped with the developed algorithms converge well for different structures. In addition, the lowest errors using the shallow and multi-layer auto-encoders are respectively 0.0204 and 0.0190. The results show that, for the same encoding dimension, the performances of the shallow and multi-layer structures are similar for this dataset.

Results of Yale face: Attribute values are normalized into $[0, 1]$ (by dividing by 255). The structures for the shallow and multi-layer LRNN auto-encoders are respectively $1024 \rightarrow 50$ and $1024 \rightarrow 500 \rightarrow 100 \rightarrow 50$. The size of a minibatch is 5. Curves of reconstruction errors versus iterations are given in Figure 2(b). For this dataset, the shallow auto-encoder seems more stable than the multi-layer one.

Results of CIFAR-10: Attribute values of the dataset are also divided by 255 for normalization in range $[0, 1]$. The structures used are $3072 \rightarrow 150$ and $3072 \rightarrow 1000 \rightarrow 500 \rightarrow 150$. Both the training and testing dataset (total 60,000 images) are used for training the auto-encoders. The size of minibatch is chosen as 100. The results are given in Figure 2(c). We can see that reconstruction errors for both structures converge as the number of iterations increases. In addition, the lowest reconstruction errors in using the shallow and multi-layer auto-encoders are the same (0.0082). These results together with those with the MNIST and Yale face datasets (Figures 2(a) to 2(c)) verify the good convergence and reconstruction performance of both the shallow and multi-layer LRNN auto-encoders for handling image datsets.



(a) MNIST (b) Yale face



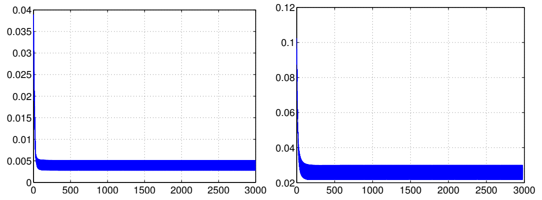
(c) CIFAR-10

Fig. 2. Reconstruction error (Y-axis) versus iteration number (X-axis) of shallow and multi-layer LRNN auto-encoders for the MNIST, Yale face and CIFAR-10 datasets.

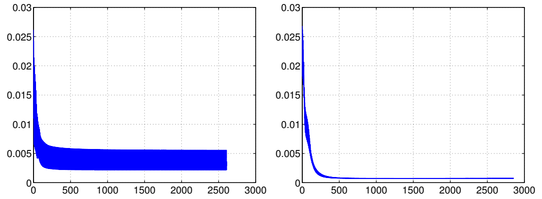
Results of UCI real-world datasets: Let N denote the attribute number in a dataset. The structures of the LRNN auto-encoders used are $N \rightarrow \text{round}(N/2)$, where the operation $\text{round}(\cdot)$ produces the nearest integer number of the element. The attribute values are linear normalized in range $[0, 1]$. The size of mini-batches is set as 50. Curves of reconstruction errors versus iterations are given in Figure 4, which use the entire datasets. We see that the reconstruction errors generally decrease as the number of iterations increases. These results also demonstrate the efficacy of the non-negative LRNN auto-encoders equipped with the training algorithms.

VI. SIMULATING THE SPIKING RNN

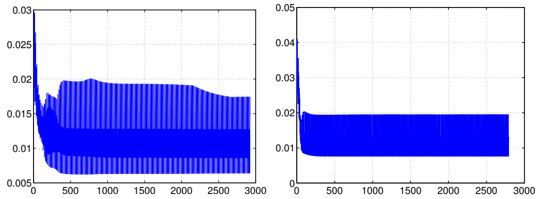
The advantage of a spiking model, such as the LRNN auto-encoder, lays on its highly-distributed nature. In this section, rather than numerical calculation, we simulate the stochastic spiking behaviors of the LRNN auto-encoder. The simulation in this section is based on the numerical experiment of Subsection V-B. Specifically, in Subsection V-B, we construct



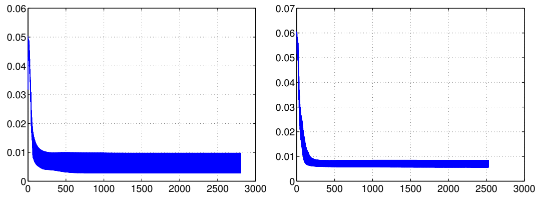
(a) IRIS (b) TAE



(c) LD (d) Seeds

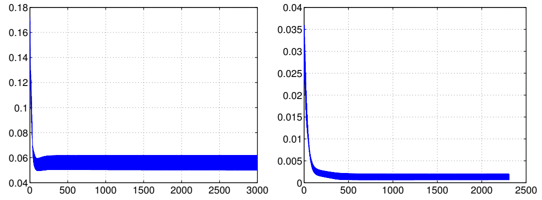


(e) PID (f) BC

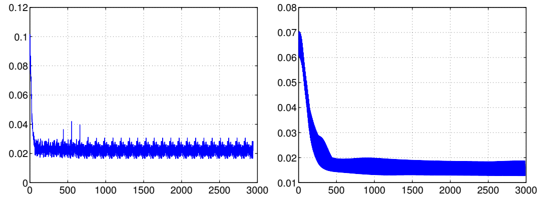


(g) Glass (h) Wine

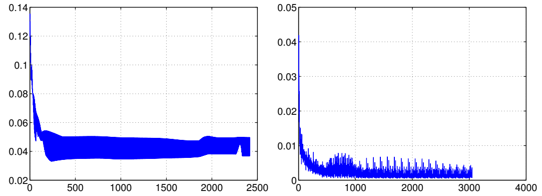
Fig. 3. Reconstruction error (Y-axis) versus iteration number (X-axis) of non-negative LRNN auto-encoders for UCI real-world datasets.



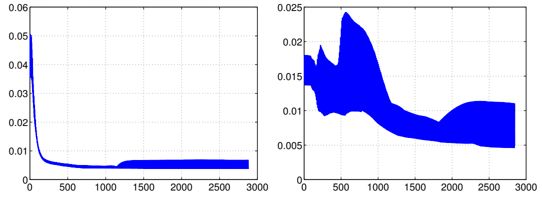
(a) Zoo (b) Parkinsons



(c) WFRN (d) Ionosphere



(e) SL (f) FOTP



(g) Sonar (h) CA

Fig. 4. Reconstruction error (Y-axis) versus iteration number (X-axis) of non-negative LRNN auto-encoders for UCI real-world datasets.

a LRNN auto-encoder of structure 784 → 100 (with appropriate weights found), which has three layers: the visual layer (784 neurons), hidden layer (100 neurons) and output layer (784 neurons). First, an image with $28 \times 28 = 784$ attributes is taken from the MNIST dataset. Each visual neuron receives excitatory spikes from outside the network in a Poisson stream with the rate being the corresponding attribute value in the image. When activated, the visual neurons fire excitatory spikes to the hidden neurons according to the Poisson process with rate 1 (meaning $w_{v,h} = p^+_{v,h}$). When the vth visual neuron fires to the hidden layer, the spike goes to the hth hidden neuron with probability $p^+_{v,h}$ or it goes outside the network with probability $1 - \sum_{h=1}^{H} p^+_{v,h}$. The hidden neurons fire excitatory spikes to the output layer in a similar manner subjecting to $\overline{w}_{h,o}$. The firing rate of output neurons is 1 and their spikes leave the network with probability 1.

In the simulation, an event occurs whenever a spike arrives from outside the network or a neuron fires. During the simulation, we observe the potential or level of activation of each neuron every 1,000 events. If $k_{i,b}$ represents the bth observation of the ith neuron, the average potential of the ith neuron is estimated by its average $\bar{k}_i = (\sum_{b=1}^{B} k_{i,b})/B$ over B observations. The relation between q_i and $\bar{k}_i$ is $\bar{k}_i \approx q_i/(1-q_i)$, and q_i is estimated as: $q_i = \bar{k}_i/(1+\bar{k}_i) \approx \sum_{b=1}^{B} k_{i,b}/(B + \sum_{b=1}^{B} k_{i,b})$. In Figure 5, we visualise the estimated values of q_i for all neurons in different layers after 10,000, 100,000 and 1,000,000 events. For comparison, numerical results from Subsection V-B are also given. At the beginning of the simulation the numerical and simulated results only agree for the input or visual layer. As time evolves, the simulation results of the hidden and output layers and their corresponding numerical results tend to the same values.

VII. CONCLUSIONS

In this paper, new Shallow and Multi-layer LRNN auto-encoders are proposed based on the spiking RNN model, by using a feed-forward multi-layer network architecture with deep-learning. Weight updates based on non-negative matrix factorisation are used to comply with the RNN's constraint of non-negative excitatory and inhibitory weights. Experiments with typical images from the MNIST, "Yale face" and CIFAR-10 datasets, as well as 16 real-world datasets

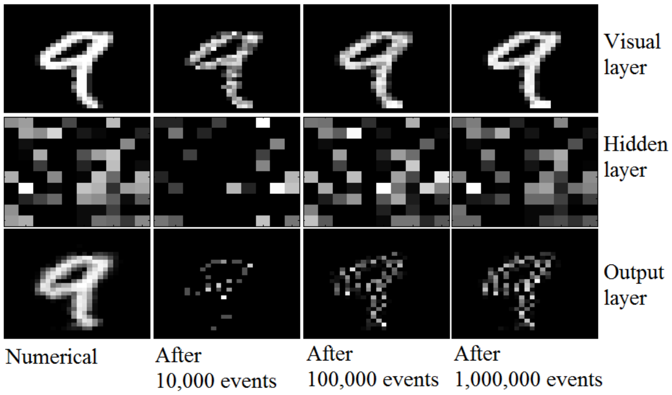


Fig. 5. Comparisons between numerical and spiking-behavior simulation results of difference layers in a LRNN auto-encoder.

from different areas, demonstrate the robust convergence and reconstruction performance of LRNN auto-encoders. We have also conducted simulations of the auto-encoder's stochastic spiking behaviour, showing good agreement with the corresponding numerical results. This also indicates that the LRNN auto-encoder may be implemented in a highly parallel manner for greater computational speed.

Acknowledgements: This research was supported by the GHOST project funded by the European Unions Horizon 2020 Framework Programme for Research and Innovation under Grant Agreement No. 740923.

REFERENCES

[1] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural computation*, vol. 1, no. 4, pp. 502–510, 1989.

[2] ——, "Stability of the random neural network model," *Neural computation*, vol. 2, no. 2, pp. 239–247, 1990.

[3] ——, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, pp. 154–164, 1993.

[4] 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.

[5] E. Gelenbe, "Dealing with software viruses: a biological paradigm," *Information Security Technical Report*, vol. 12, no. 4, pp. 242–250, 2007.

[6] 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.

[7] E. Gelenbe and Y. Yin, "Deep learning with random neural networks," *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1633–1638, 2016.

[8] Y. Yin and E. Gelenbe, "Single-cell based random neural network for deep learning," in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 86–93.

[9] Y. Yin, "Random neural networks for deep learning," *Imperial College London, PhD Thesis, available in http://hdl.handle.net/10044/1/64917 and https://san.ee.ic.ac.uk/publications.shtml*, 2018.

[10] ——, "Random neural network methods and deep learning," *Probability in the Engineering and Informational Sciences*, accepted.

[11] I. Grenet, Y. Yin, J.-P. Comet, and E. Gelenbe, "Machine learning to predict toxicity of compounds," in *International Conference on Artificial Neural Networks*. Springer, Cham, 2018, pp. 335–345.

[12] I. Grenet, Y. Yin, and J.-P. Comet, "G-networks to predict the outcome of sensing of toxicity," *Sensors*, vol. 18, no. 10, 2018. [Online]. Available: http://www.mdpi.com/1424-8220/18/10/3483

[13] O. Brun, Y. Yin, and E. Gelenbe, "Deep learning with dense random neural network for detecting attacks against iot-connected home environments," *Procedia Computer Science*, vol. 134, pp. 458–463, 2018.

[14] E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui, "Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 12, pp. 2486–2498, 2016.

[15] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun, "Stacked what-where auto-encoders," *arXiv preprint arXiv:1506.02351*, 2015.

[16] V. Turchenko, E. Chalmers, and A. Luczak, "A deep convolutional auto-encoder with pooling-unpooling layers in caffe," *arXiv preprint arXiv:1701.04949*, 2017.

[17] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[18] P. Goyal, P. Dollár, R. B. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch SGD: training imagenet in 1 hour," *CoRR*, vol. abs/1706.02677, 2017.

[19] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[20] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2013.

[21] A. El Khatib, S. Huang, A. Ghodsi, and F. Karray, "Nonnegative matrix factorization using autoencoders and exponentiated gradient descent," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[22] K. Zen, M. Suzuki, H. Sato, S. Oyama, and M. Kurihara, "Monophonic sound source separation by non-negative sparse autoencoders," in *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2014, pp. 3623–3626.

[23] E. Wachsmuth, M. Oram, and D. Perrett, "Recognition of objects and their component parts: responses of single units in the temporal cortex of the macaque," *Cerebral Cortex*, vol. 4, no. 5, pp. 509–522, 1994.

[24] X. Liu, S. Yan, and H. Jin, "Projective nonnegative graph embedding," *Image Processing, IEEE Transactions on*, vol. 19, no. 5, pp. 1126–1137, 2010.

[25] 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.

[26] D. Cai, X. He, Y. Hu, J. Han, and T. Huang, "Learning a spatially smooth subspace for face recognition," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–7.

[27] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[28] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[29] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology." *PNAS*, vol. 87, no. 23, pp. 9193–9196, 1990.

[30] M. A. Little, P. E. McSharry, S. J. Roberts, D. A. Costello, and I. M. Moroz, "Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection," *BioMedical Engineering OnLine*, vol. 6, no. 1, p. 1, 2007.

[31] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela, "Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study," in *Robotics Symposium (LARS), 2009 6th Latin American*. IEEE, 2009, pp. 1–6.

[32] V. G. Sigillito, S. P. Wing, L. V. Hutton, and K. B. Baker, "Classification of radar returns from the ionosphere using neural networks," *Johns Hopkins APL Technical Digest*, vol. 10, no. 3, pp. 262–266, 1989.

[33] J. P. Bridge, S. B. Holden, and L. C. Paulson, "Machine learning for first-order theorem proving," *Journal of automated reasoning*, vol. 53, no. 2, pp. 141–172, 2014.

[34] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural networks*, vol. 1, no. 1, pp. 75–89, 1988.

[35] H. A. Guvenir, B. Acar, G. Demiroz, and A. Cekin, "A supervised machine learning algorithm for arrhythmia analysis," in *Computers in Cardiology 1997*. IEEE, 1997, pp. 433–436.