Single-Cell Based Random Neural Network for Deep Learning

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

***Abstract*—Recent work demonstrated the value of multi clusters of spiking Random Neural Networks (RNN) with dense soma-to-soma interactions in deep learning. In this paper we go back to the original simpler structure and we investigate the power of single RNN cells for deep learning. First, we consider three approaches with the single cells, twin cells and multi-cell clusters. This first part shows that RNNs with only positive parameter can conduct convolution operations similar to those of the convolutional neural network. We then develop a multi-layer architecture of single cell RNNs (MLSRNN), and show that this architecture achieves comparable or better classification at lower computation cost than conventional deep-learning methods.**

I. Introduction

The Random Neural Network (RNN) was developed to mimic the behaviour of biological neurons in the brain [1], [2] and its generalisations [3]. Applications of the RNN have focused on its recurrent structure and learning capabilities [4] especially in image processing [5], [6], as well as in recent work [7]–[12].

While early work [13] investigated the links between conventional optimisation and the RNN, much more recently [14] the RNN was connected to deep learning [15]–[18] and the extreme learning machine (ELM) concept [19], [20].

In particular, we proposed a multi-layer architecture composed of multiple clusters of recurrent RNNs with dense soma-to-soma interactions to implement autoencoders with a training procedure [14], [21] that is significantly faster than existing deep-learning tools.

This paper investigates the value of single RNN cells for deep learning. Specifically we show that, with only positive parameters, the RNN implements convolution operations similar to convolutional neural networks [16], [22], [23].

Our work examines single-cell, twin-cell and cluster approaches that model the positive and negative parts of a convolution kernel as the arrival rates of excitatory and inhibitory spikes to receptive cells. We also investigate the relationship between the RNN and ReLU activation [24] and suggest an approximate ReLU-activated convolution operation.

Also, we build a multi-layer architecture composed of single-cell RNNs (MLSRNN), different from the dense-cluster based architecture in [14], [21], in which the final layer is an ELM rather than RNN cells and clusters. The computational complexity of the MLSRNN is much lower than the dense model [14], [21], and it can be generalized to handle multi-channel datasets.

Numerical results regarding multi-channel classification datasets, with an image and two time-series, show that the single-cell based RNN is effective and that it is arguably the most efficient among five different deep-learning approaches.

II. Recurrent Random Neural Network

An arbitrary cell in the Random Neural Networks (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 cells to a given cell, in which case they arrive when the sending cell fires, which happens only if that cell's input state is positive (i.e. the neuron is excited) and inter-firing intervals from the same neuron v are exponentially distributed random variables with rate $r_v \geq 0$. Since the firing times depend on the internal state of the sending cell, the arrival process of cells from other cells is not in general Poisson. From the preceding assumptions it was proved in [1], [2], [25] that for an arbitrary N cell 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:

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

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 λ_h^+, λ_h^- are the external arrival rates of excitatory and inhibitory spikes to cell h. For notation ease, we also use $w_{vh}^+ = r_v p_{vh}^+$ and $w_{vh}^- = r_v p_{vh}^-$ as excitatory and inhibitory connecting weights. Note that $\min(a, b)$ is a element-wise operation whose output is the smaller one between a and b. In [25], it was shown that the system of N non-linear equations (1) have a solution which is unique.

III. Convolution Operation with RNN: Two Approaches

In this and the next sections, we show that, though with only positive parameters, the RNN is capable of conducting convolution operators that produce similar effects to those in conventional convolutional neural networks [16], [23], by three approaches, which is denoted as $O = \text{conv}(I, W)$ with O, I and W being the output, input and convolution kernel.

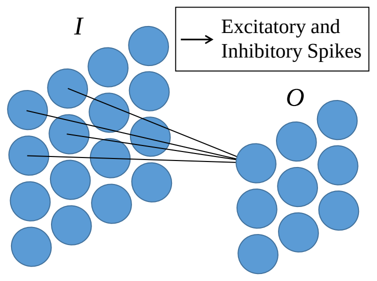

Fig. 1. An RNN convolution operation with single-cell approach.

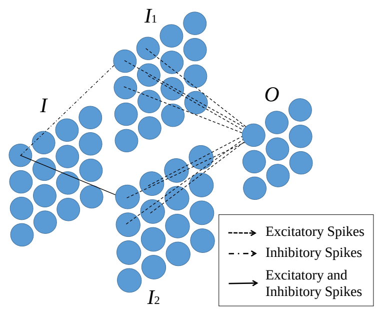

Fig. 2. An RNN convolution operation with twin-cell approach.

A. Single-Cell Approach

We construct a type of RNN cells and present the procedure for conducting convolution operations using these cells.

An RNN cell receives excitatory and inhibitory spikes from external cells denoted by x^+ and x^-, and it also receives excitatory spikes from outside world denoted by λ^+. The firing rate of the cell is r. Then, based on (1), the probability in the steady state that this cell is activated is

$$q = \min(\frac{\lambda^+ + x^+}{r + x^-}, 1), \tag{2}$$

For notation ease, we define $\phi(x^+, x^-) = \min((\lambda + x^+)/(r + x^-), 1)$ and use $\phi(\cdot)$ as a term-by-term function for vectors and matrices.

First, we normalize the convolution kernel to satisfy the RNN probability constraint via $W \leftarrow W/(\text{sum}(|W|)/r)$. Second, we spilt it as $W^+ = \max(W, 0) \geq 0$ and $W^- = \max(-W, 0) \geq 0$ to avoid negativity (another RNN probability constraint). Operation sum(a) produces the summations of all elements in a and $\max(a, b)$ produces the larger element between a and b. Then, the RNN convolution operation can be conducted using RNN cells (2)

$$O = \phi(\text{conv}(I, W^+), \text{conv}(I, W^-)) \tag{3}$$

or

$$O = \phi(\text{conv}(I, W^-), \text{conv}(I, W^+)), \tag{4}$$

where λ^+ and r are usually set as the same value for simplicity. The convolution operation with single cells is shown schematically in Figure 1.

B. Twin-Cell Approach

This approach is also based on RNN cells (2). From (2), we have $\phi(0, I)|_{\lambda^+=1,r=1} = 1 - \phi(I, I)|_{\lambda^+=0,r=1}$.

To satisfy the RNN probability constraints, we first normalize the convolution kernel via $W \leftarrow W/\text{sum}(|W|)$ and split it into $W^+ = \max(W, 0) \geq 0$ and $W^- = \max(-W, 0) \geq 0$. As seen from Figure 2, input I passes through a twin-cell array and produces $I_1 = \phi(0, I)|_{\lambda^+=1,r=1}$ and $I_2 = \phi(I, I)|_{\lambda^+=0,r=1}$. The receptive cell are quasi-linear cells [26] (called the LRNN-E cell) that receive excitatory spikes from outside world with rate $\lambda^+ = 1 - \text{sum}(W^-)$. We can also normalize the input to receptive cells via $W^+ \leftarrow W^+/\max(O')$ and $W^- \leftarrow W^-/\max(O')$ to reduce the number of cells that are saturated [26], where $O' = \text{conv}(I_1, W^+) + \text{conv}(I_2, W^-) + 1 - \text{sum}(W^-)$. Then, the RNN convolution operation can be conducted as

$$O = \min(\text{conv}(I_1, W^+) + \text{conv}(I_2, W^-) + 1 - \text{sum}(W^-), 1). \tag{5}$$

IV. Convolution Operation with RNN: a Cluster Approach

This approach is based on a type of multi-cell cluster that approximates the rectified linear unit (ReLU) that has been widely-used in the deep-learning area [16], [24]. The cluster is constituted by the LRNN-E cell [26] and another different quasi-linear RNN cell that is deduced from a first-order approximation of the RNN (1).

A. First-Order Approximation of the RNN

The formula (1) lends itself to various approximations such as the first order approximation:

$$q_h \approx \frac{\lambda_h^+ + \sum_{v=1}^N q_v r_v p_{vh}^+}{r_h}[1 - \frac{\lambda_h^- + \sum_{v=1}^N q_v r_v p_{vh}^-}{r_h}], \tag{6}$$

provided that $r_h >> \lambda_h^- + \sum_{v=1}^N q_v r_v p_{vh}^-$. The proof of (6) is given in the Appendix.

Let $w_{vh}^+ = r_v p_{vh}^+ = 0$, $w_{vh}^- = r_v p_{vh}^-$, $\lambda_h^+ = 1$, $\lambda_h^- = 0$, $r_h = 1$ and $r_v = 1$. Note that these settings do not offend the condition $r_h >> \lambda_h^- + \sum_{v=1}^N q_v r_v p_{vh}^-$, which becomes $1 >> \sum_{v=1}^N w_{vh}^- q_v$. Since $r_v = 1$, then $\sum_{h=1}^N w_{vh}^- = \sum_{h=1}^N p_{vh}^- \leq 1$. The first-order approximation (6) can be rewritten as:

$$q_h = \frac{1}{1 + \sum_{v=1}^N w_{vh}^- q_v} \approx \min(1 - \sum_{v=1}^N w_{vh}^- q_v, 1), \tag{7}$$

subjecting to $\sum_{h=1}^N w_{vh}^- \leq 1$ and $1 >> \sum_{v=1}^N w_{vh}^- q_v$. Since $q_v \leq 1$, we have $\sum_{v=1}^N w_{vh}^- q_v \leq \sum_{v=1}^N w_{vh}^-$, and we then could loose the condition $1 >> \sum_{v=1}^N w_{vh}^- q_v$ to $1 >> \sum_{v=1}^N w_{vh}^-$, which is independent of the cell states. This simplified RNN cell receiving inhibitory spikes is quasi linear, and thus we could call it a LRNN-I cell.

B. Relationship between RNN and ReLU

Based on the LRNN-E cell [26] and LRNN-I cell (7), we investigate the relationship between the RNN and the ReLU activation function [16], [24] and show that a cluster of the LRNN-I and LRNN-E cells produces approximately the ReLU activation.

1) ReLU Activation: We consider a single unit with ReLU activation. Suppose the input to the unit is a nonnegative $1 \times V$ vector $X = [x_{1,v}]$ in the range $[0, 1]$, and the connecting weights is a $V \times 1$ vector $W = [w_{v,1}]$ whose elements can be both positive and negative. Then, the output of this unit is described by $\text{ReLU}(XW) = \max(0, XW)$. Let $W^+ = [w^+_{v,1}] = \max(0, W)$ and $W^- = [w^-_{v,1}] = -\min(0, W)$. It is clearly that $W^+ \geq 0$, $W^- \geq 0$, $W = W^+ - W^-$ and $\text{ReLU}(XW) = \max(0, XW^+ - XW^-)$.

2) A Cluster of the LRNN-I and LRNN-E Cells: First, we import X into a LRNN-E cell with connecting weights W^-:

$$q_1 = \min(XW^-, 1).$$

Let us suppose $XW^- \leq 1$. Then, $q_1 = XW^-$.

In the meantime, we import X into a LRNN-I cell with W^+:

$$q_2 = \frac{1}{1 + XW^+} \approx \min(1 - XW^+, 1).$$

Based on (7), the condition of this approximation is that $\sum_{v=1}^{V} w^+_{v,1} << 1$. Suppose this condition holds, we also have $q_2 \approx 1 - XW^+$.

Second, we connect q_1 and q_2 to a LRNN-E cell with connecting weight being 1:

$$\begin{aligned} q_3 &= \min(q_2 + q_1, 1) = \min(\frac{1}{1 + XW^+} + XW^-, 1) \\ &\approx \min(1 - XW^+ + XW^-, 1). \end{aligned}$$

In q_3, the information where $1 - XW^+ + XW^- = 1 - XW > 1$ (i.e., $XW < 0$) is removed by the LRNN-E cell. We have

$$q_3 = \varphi(XW^+, XW^-) \approx 1 - \text{ReLU}(XW), \tag{8}$$

where, for notation ease, we define the activation of this cluster as $\varphi(x^+, x^-) = \min(1/(1 + x^+) + x^-, 1)$ and use $\varphi(\cdot)$ as a term-by-term function for vectors and matrices. The conditions for the approximation in (8) are $XW^- \leq 1$ (that can be loosed as $\sum_{v=1}^{V} w^-_{v,1} = \text{sum}(W^-) \leq 1$ if $X \leq 1$) and $\sum_{v=1}^{V} w^+_{v,1} = \text{sum}(W^+) << 1$.

C. Convolution with RNN Cluster

First, we normalized the convolution kernel via $W \leftarrow W/\text{sum}(|W|)/10$. Second, let us split the kernel as $W^+ = \max(W, 0) \geq 0$ and $W^- = \max(-W, 0) \geq 0$. It is evident that $\text{sum}(W^+) \leq 0.1 << 1$ and $\text{sum}(W^-) \leq 0.1 << 1$. (Here we assume that $0.1 << 1$.) Then, the convolution operation with the RNN cluster (8) can be conducted as

$$O = \varphi(\text{conv}(I, W^+), \text{conv}(I, W^-)), \tag{9}$$

or

$$O = \varphi(\text{conv}(I, W^-), \text{conv}(I, W^+)). \tag{10}$$

The convolution operation with clusters is shown schematically in Figure 3.

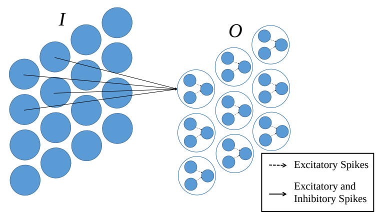

Fig. 3. An RNN convolution operation with cluster approach.

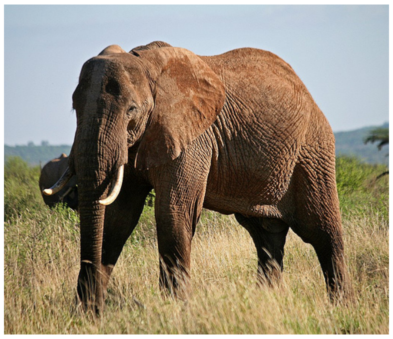

Fig. 4. The original image.

V. Numerical Verification of RNN Convolution

This section conducts numerical experiments on images (implemented in Theano [27]) to verify the three approaches in Sections III and IV for adapting the convolution structure into the RNN. The convolution kernels used in the experiments are obtained from the first convolution layer of the pre-trained GoogLeNet [28] provided by [29], while the image is obtained from [30], shown in Figure 4. The gray-inverted output of a standard convolution operation with the ReLU activation is given in Figure 5.

Single-cell approach: The output of the convolution is given in Figure 6, from which we can see that it is similar to that in Figure 5.

Twin-cell approach: The output of the convolution is given in Figure 7. This approach also produces similar edge-detection effect to the standard convolution.

Cluster approach: The output of the convolution is given in Figure 8, which is more similar to those in Figure 5 than that in Figure 6.

These results demonstrate well the feasibility of the RNN for convolution operations via these approaches.

VI. Single-Cell Based Multi-Layer RNN

Recent work in [14], [21] presented a mathematical model of multi RNN clusters with dense soma-to-soma interactions, based on which a multi-layer architecture of the RNN is built up that was shown to be more efficient than convolutional deep-learning tools on both image and time-series datasets. Striving into efficiency, this section presents a multi-layer architecture of single RNN cells described in (2) (called MLSRNN). Based on [14], [21], the MLSRNN is then adapted to handle multi-channel datasets (MCSRNN).

A. The Mathematical Model of MLSRNN

The MLSRNN, shown schematically in Figure 9, has $L+2$ layers, where the lth layer has N_l cells. The first layer is the external-source layer, where s_{n_1} denotes the n_1th source. The successive L layers are hidden layers composed of single

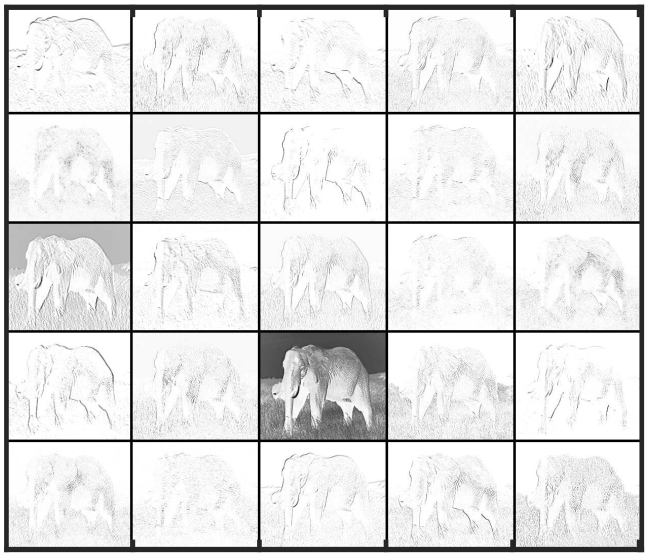

Fig. 5. Gray-inverted output of a standard convolution operation with ReLU activation.

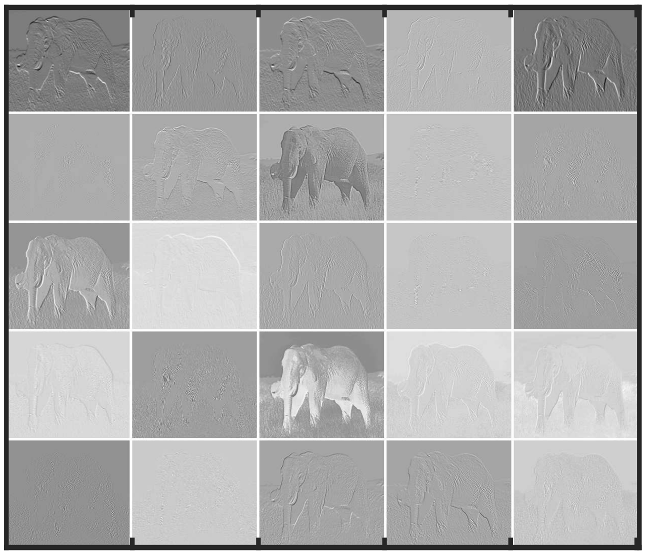

Fig. 7. Output of an RNN convolution operation with the twin-cell approach.

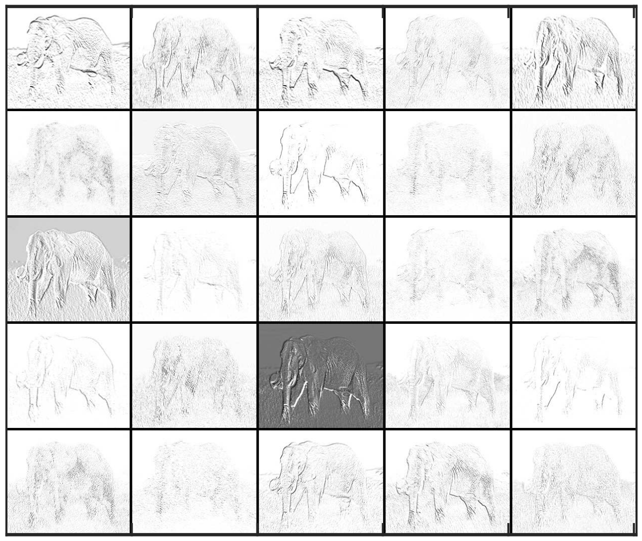

Fig. 6. Output of an RNN convolution operation with the single-cell approach.

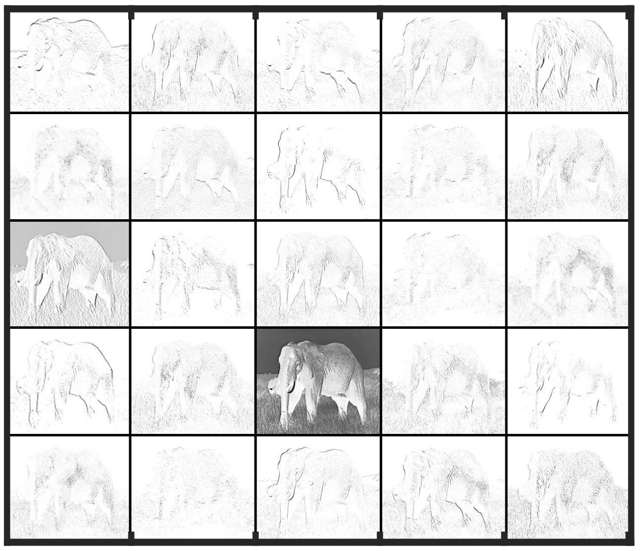

Fig. 8. Output of an RNN convolution operation with the cluster approach.

RNN cells (2) that receive both excitatory and inhibitory spike trains (x^+ and x^-) from cells in the previous layer, with a resultant activation function $q(x^+, x^-) = \phi(x^+, x^-)$. The last layer is made up of cells that receives excitatory spike trains from the previous layer and outside world, resulting in a linear cell activation $q(x) = x$. For these layers, let q_{n_l} ($n_l \leq N_l$) denotes the excitation probability for the l-layer ($2 \leq l \leq L$) cell n_l. In addition, let two $N_l \times N_{l+1}$ matrices $W_l^+ = [w^+_{n_l n_{l+1}}]$ and $W_l^- = [w^-_{n_l n_{l+1}}]$ denote excitatory and inhibitory connecting weight matrices between the lth and $(l+1)$th layers for $l = 1, \cdots, L+1$. Let a $1 \times N_{L+2}$ vector $\Lambda^+ = [\lambda^+_{n_{L+2}}]$ denote the external arrival rates of excitatory spikes for the $(L+2)$th layer (i.e., the output layer).

Suppose there is a dataset represented by a nonnegative $D \times N_1$ matrix $X = [x_{dn_1}]$, where D is the number of instances, each instance has N_1 attributes and x_{dn_1} is the n_1th attribute of the dth instance. Let s_{dn_1} and q_{dn_l} denote the values of s_{n_1} and q_{n_l} for the dth instance. Let a $D \times N_1$ matrix $Q_1 = [s_{dn_1}]$ and a $D \times N_l$ matrix $Q_l = [q_{dn_l}]$ ($2 \leq l \leq L+2$). Then, the MLSRNN can be described as:

$$\begin{cases} Q_1 = X, \\ Q_l = \phi(Q_{l-1}W^+_{l-1}, Q_{l-1}W^-_{l-1}), \\ \text{for } l = 2, \cdots, L+1, \\ Q_{L+2} = \min(Q_{L+1}W^+_{L+1} + \Lambda, 1), \end{cases}$$

where Λ is a $D \times N_{L+2}$ matrix denoting the external arrival rates of excitatory spikes and each row of Λ is Λ^+, subject to the RNN probability constraints $W_l^+ \geq 0$, $W_l^- \geq 0$,

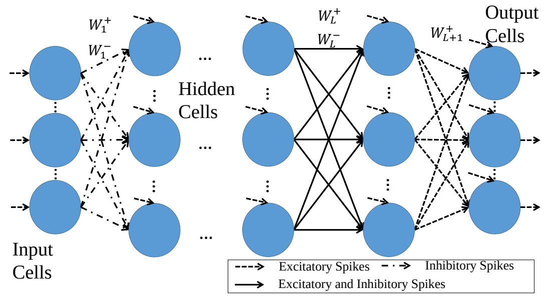

Fig. 9. Schematic representation of the MLSRNN.

$\sum_{n_{l+1}=1}^{N_{l+1}}(w^+_{n_l n_{l+1}} + w^-_{n_l n_{l+1}}) \leq r_l$. Note that the number of hidden cell (N_{L+1}) in the $(L+1)$th layer needs to be a multiple of 2, which will be explained in the training-procedure part (Subsection VI-B).

B. Training Procedures of MLSRNN

Here we move to the training procedures of the MLSRNN.

1) Determine W_l^+, W_l^- with $l = 1, \cdots, L-1$: First, $W_l^+ \leftarrow 0$. Then, we solve an reconstruction problem for W_l^- using the modified FISTA [31] with the modification of setting negative elements in the solution to zero in each iteration [21]:

$$\min_{W_l^+} ||X - \text{adj}(\phi(0, X\bar{W}^-))W_l^-||^2 + ||W_l^-||_{\ell_1}, \quad \text{s.t. } W_l^- \geq 0, \tag{11}$$

where X is either the data or its layer encodings, $\bar{W}^- \geq 0$ is randomly generated that satisfies the RNN constraints and operation adj$(\cdot)$ first maps its input into [0 1] linearly, then uses the "zcore" MATLAB operation and finally adds a positive constant to remove negativity.

After solving (11), we normalize W_l^- to satisfy the RNN constraints. Then, we adjust the external arrival rate λ^+ and firing rate r of hidden cells in the lth hidden layer via $\lambda^+_{l+1} \leftarrow \max(XW_l^-)/5$ and $r_{l+1} \leftarrow \lambda^+_{l+1}$.

2) Determine W_L^+, W_L^- and W_{L+1}^+: We first construct a one-hidden-layer RNN-based ELM [19], [20] (with $N_{L+1}/2$ hidden units and N_{L+2} output units) and then map its weights to W_L^+, W_L^- and W_{L+1}^+. For this ELM whose activation function is $\rho(x) = a/(a+x)$ with parameter $a > 0$ to be determined, the input and output weights are $\bar{W}_1$ and $\bar{W}_2$.

Suppose X is the Lth-layer output of the MLSRNN and Y is the desired output (or say, labels corresponding to the training dataset). We randomly generate $\bar{W}_1$ in range [0 1] and then normalize $2 \odot \bar{W}_1$ to satisfy the RNN constraint, where $\odot$ denotes element-wise multiplication. Then, $a \leftarrow \max(X\bar{W}_1)/5$. Then, we determine $\bar{W}_2$ using the Moore-Penrose pseudo-inverse [14], [19], [20], [32]–[34] (denoted by "pinv") as:

$$\bar{W}_2 \leftarrow \text{pinv}(\rho(X\bar{W}_1))Y. \tag{12}$$

Then, $\bar{W}_2 \leftarrow \bar{W}_2/\text{sum}(|\bar{W}_2|)$ to guarantee the summation of all elements in $|\bar{W}_2|$ is no larger than 1. Let us define $\varrho(x) = x/(a+x)$. It is evident that $\rho(x) = 1 - \varrho(x)$. Let $\bar{W}_2^+ = \max(\bar{W}_2, 0) \geq 0$ and $\bar{W}_2^- = \max(-\bar{W}_2, 0) \geq 0$.

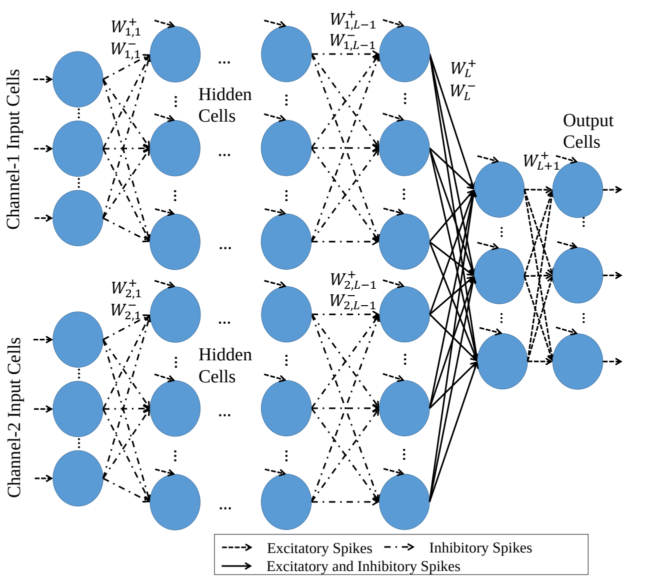

Fig. 10. Schematic representation of the MCSRNN.

The ith ($i = 1, \cdots, N_{L+2}$) output of this ELM for the dth instance is

$$\begin{aligned} o_{di} &= \sum_{j=1}^{N_{L+1}/2} (\rho(X\bar{W}_1))_{dj}(\bar{W}_2)_{ji} \\ &= \sum_{j=1}^{N_{L+1}/2} (\rho(X\bar{W}_1))_{dj}(\bar{W}_2^+)_{ji} - \sum_{j=1}^{N_{L+1}/2} (\rho(X\bar{W}_1))_{dj}(\bar{W}_2^-)_{ji} \\ &= \sum_{j=1}^{N_{L+1}/2} (\rho(X\bar{W}_1))_{dj}(\bar{W}_2^+)_{ji} + \sum_{j=1}^{N_{L+1}/2} (\varrho(X\bar{W}_1))_{dj}(\bar{W}_2^-)_{ji} \\ &\quad - \sum_{j=1}^{N_{L+1}/2} (\bar{W}_2^-)_{ji} \end{aligned}$$

In addition, we have $\rho(X\bar{W}_1) = \phi(0, X\bar{W}_1)|_{\lambda^+=a, r=a}$ and $\varrho(X\bar{W}_1) = \phi(X\bar{W}_1, X\bar{W}_1)|_{\lambda^+=0, r=a}$. Let $m = \max(\text{sum}(\bar{W}_2^-, 1))$, where operation sum$(X, 1)$ produces the summation vector of each column in matrix X. Let $\lambda_i^+ \leftarrow m - \sum_{j=1}^{N_{L+1}/2}(\bar{W}_2^-)_{ji}$ Then, it is evident that

$$\begin{aligned} o_{di} + m = &\sum_{j=1}^{N_{L+1}/2} (\phi(0, X\bar{W}_1)|_{\lambda^+=a, r=a})_{dj}(\bar{W}_2^+)_{ji} \\ &+ \sum_{j=1}^{N_{L+1}/2} (\phi(X\bar{W}_1, X\bar{W}_1)|_{\lambda^+=0, r=a})_{dj}(\bar{W}_2^-)_{ji} \\ &+ \lambda_i^+. \end{aligned} \tag{13}$$

Now let us map weights in the RNN-based ELM illustrated in (13) to W_L^+, W_L^- and W_{L+1}^+ in the MLSRNN. From the 1st to $(N_{L+1}/2)$th hidden cells in the $(L+1)$th layer, we use $\lambda = a$ and $r = a$; while for the $(N_{L+1}/2 + 1)$th to N_{L+1}th hidden cells, we use $\lambda = 0$ and $r = a$. Then, $W_L^+ \leftarrow [0; \bar{W}_1]$, $W_L^- \leftarrow [\bar{W}_1; \bar{W}_1]$ and $W_{L+1}^+ \leftarrow [\bar{W}_2^+; \bar{W}_2^-]$.

The procedure to train the MLSRNN is also briefly sum-

Algorithm 1 Training procedure for the MLSRNN

```
Get data matrix $X$ and label matrix $Y$
$r_1 \leftarrow 1$
**for** $l = 1, \cdots, L-1$ **do**
    $W_l^+ \leftarrow 0$
    solve Problem (11) for $W_l^-$ with input $X$
    $e \leftarrow \max(\text{sum}(W_l^-, 2))$
    **if** $e > r_l$
        $W_l^- \leftarrow W_l^- / (e/r_l)$
    $\lambda_{l+1}^+ \leftarrow \max(XW_l^-)/5$
    $r_{l+1} \leftarrow \lambda_{l+1}^+$
    $X \leftarrow \phi(XW_l^+, XW_l^-)|_{\lambda_{l+1}^+, r_{l+1}}$
determine $\bar{W}_1, \bar{W}_2$ of an ELM with input $X$, label $Y$
map $\bar{W}_1, \bar{W}_2$ to $W_L^+, W_L^-, W_{L+1}^+, \Lambda^+$
```

Algorithm 2 Training procedure for the MCSRNN

```
Get data matrices $X_c$ $(c = 1, \cdots, C)$ and label matrix $Y$
**for** $l = 1, \cdots, L-1$ **do**
    **for** $c = 1, \cdots, C$ **do**
        $W_l^+ \leftarrow 0$
        solve Problem (11) for $W_{c,l}^-$ with input $X_c$
        $e \leftarrow \max(\text{sum}(W_{c,l}^-, 2))$
        **if** $e > r_{c,l}$
            $W_{c,l}^- \leftarrow W_{c,l}^- / (e/r_l)$
        $\lambda_{c,l+1}^+ \leftarrow \max(X_c W_{c,l}^-)/5$
        $r_{c,l+1} \leftarrow \lambda_{c,l+1}^+$
        $X_c \leftarrow \phi(XW_{c,l}^+, XW_{c,l}^-)|_{\lambda_{c,l+1}^+, r_{c,l+1}}$
$X \leftarrow [X_1 \ \cdots \ X_C]$
determine $\bar{W}_1, \bar{W}_2$ of an ELM with input $X$, label $Y$
map $\bar{W}_1, \bar{W}_2$ to $W_L^+, W_L^-, W_{L+1}^+, \Lambda^+$
```

marized in Algorithm 1, where operation $\text{sum}(X, 2)$ produces the summation vector of each row in matrix X.

C. MCSRNN: Mathematical Model and Training Procedure

Based on the work of dense cluster [21], the MLSRNN is adapted to handle multi-channel datasets (MCSRNN), shown in Figure 10, where the connecting weights between layers for only Channel-c $(c = 1, \cdots, C)$ are $W_{c,l}^+, W_{c,l}^- \geq 0$ $(l = 1, \cdots, L-1)$, those between the $(L-1)$th and Lth hidden layers are $W_L^+, W_L^- \geq 0$ and output weights are $W_{L+1}^+ \geq 0$. Besides, a vector Λ^+ denotes the external arrival rates of excitatory spikes for the cells in the $(L+2)$th layer (output).

We can generalize the training procedure of the MLSRNN for the MCSRNN, which is given in Algorithm 2.

VII. Numerical Results

In this section, we conduct numerical tests for the MLSRNN and MCSRNN that use three multi-channel classification datasets: an image dataset and two real-world time-series datasets.

NORB Dataset: The small NORB dataset [36] is intended for 3D object recognition from shape. The sizes of training and testing sets are both 24300. Each instance contains two 96×96 images, which are downsampled into 32×32. All images are whitened using the code provided by [20]. The number of classes and channels are 5 and 2.

Daily and Sports Activities (DSA) Dataset: The DSA dataset [37]–[39] comprises time-series data of 19 daily and sports activities performed by 8 subjects recorded by 45 motion sensors (25 Hz sampling frequency). The attribute number is 5,625 (45x5x25) since 5-second segments are used. Two thirds of 9120 instances are for training while the rest for testing. The dataset has 19 classes and 45 channels.

Twin Gas Sensor Arrays (TGSA) Dataset: The TGSA dataset includes 640 recordings of 5 twin 8-sensor detection units exposing to 4 different gases [40]. The duration of each recording is 600 seconds (100Hz sampling frequency) producing 480,000 (8x600x100) features. We use 30-second segments, and then each instance has 24,000 (8x3000) attributes. The objective is to classify gas types using recording features. The number of classes and channels are 4 and 8. Two tasks are conducted, in both of which two thirds of instances are used for training while the rest for testing:

- Task 1: (3,029 instances): build a specific classifier for Unit 1 to fulfill the objective.
- Task 2: (12,089 instances): build one classifier for all units to fulfill the objective.

In the numerical experiments, we compare the proposed MCSRNN in this paper with the dense-cluster based multi-layer RNNs presented in [14], [21], the multi-layer perception (MLP) [35], the convolutional neural network (CNN) [35], [41] and hierarchical ELM (H-ELM) [20]. All numerical experiments are conducted in the same personal computer.

The results based on the above three datasets are giving Tables I and II, including the testing accuracy, training time and testing time. The proposed MCSRNN achieves the highest testing accuracies in all four classification tasks. Compared with the conventional CNN and MLP, the MCSRNN is more than one hundred times faster in training and achieves better accuracies. For the DAS dataset, the accuracies of the MCSRNN in this paper and MCRNN-MLA in the previous work [21] are the same, however, the MCSRNN is more than two times faster in training and more than five times faster in testing, where they are in the same structures of 45x125-45x200-45x100-2000-19. The MCSRNN costs the least testing time among all tools in three of the four tasks. For the DAS dataset, where the MCSRNN achieved the highest accuracy, its testing time is also less than 1 second, the same as that the lowest one (the CNN). These numerical results well demonstrate that the MCSRNN is effective and the most efficient among the compared deep-learning tools, as well as the value of single RNN cells for deep learning.

VIII. Conclusion

This paper has demonstrated the power of single RNN cells in multi-layer architectures for deep learning. First, this paper has presented three approaches, including a single-RNN-cell approach, a twin-RNN-cell approach and a RNN-cluster approach, that allow the RNN to conduct approximate ReLU-activated convolution operations, producing similar edge-detection effects. Then, a multi-layer architecture based

TABLE I
TESTING ACCURACIES (%), TRAINING AND TESTING TIME (S) OF DIFFERENT METHODS FOR NORB AND DAS DATASETS.

Method	Testing accuracy		Training time		Testing time	
	NORB	DAS	NORB	DAS	NORB	DAS
MCSRNN	**92.44**	**99.21**	13.38	12.38	1.74	0.96
MCRNN-MLA [21]	92.10	**99.21**	28.80	26.81	11.07	4.97
MCRNN-MLA1 [21]	91.21	98.98	1750.85	89.16	229.13	12.86
MCRNN-MLA2 [21]	91.72	94.67	1169.61	177.03	83.49	10.88
Improved RNN-MLA [21]	90.96	92.17	20.63	13.11	7.63	1.09
Original RNN-MLA [14]	88.51	92.83	18.80	6.02	7.43	0.74
MLP+dropout [35]	67.12	91.94	2563.27	3291.47	1.88	0.50
CNN [35]	90.80	98.52	1223.93	1289.76	17.39	0.47
CNN+dropout [35]	90.76	99.05	1282.99	1338.35	16.83	0.53
H-ELM [20]	87.56	96.58	125.86	9.60	23.86	0.82
H-ELM * [20]	91.28	–	–	–	–	–

*This data is obtained directly from [20].

TABLE II
TESTING ACCURACIES (%), TRAINING AND TESTING TIME (S) OF DIFFERENT METHODS FOR TGSA DATASET.

Method	Testing accuracy		Training time		Testing time	
	Task 1	Task 2	Task 1	Task 2	Task 1	Task 2
MCSRNN	**99.31**	**96.08**	16.52	40.06	0.27	0.43
MCRNN-MLA [21]	98.32	95.98	29.56	41.34	1.60	1.84
MCRNN-MLA1 [21]	98.61	92.48	55.08	68.03	3.00	3.39
MCRNN-MLA2 [21]	94.75	79.66	51.96	42.89	1.72	2.38
Improved RNN-MLA [21]	97.03	90.23	16.78	49.38	0.76	1.78
Original RNN-MLA [14]	85.64	83.60	29.94	48.23	0.89	1.67
MLP+dropout [35]	25.05	24.83	3327.52	4504.54	0.84	1.37
CNN [35]	61.78	25.03	1842.38	16296.69	0.83	5.43
CNN+dropout [35]	69.11	76.33	2484.18	16294.74	0.83	5.58
H-ELM [20]	61.98	60.03	14.21	36.57	0.71	2.72

on single RNN cells has been proposed for deep learning, i.e., the MLSRNN and its generalized multi-channel version (the MCSRNN). Numerical results based on multi-channel classification datasets, an image and two time-series datasets, have well demonstrated that the single-cell based RNN is effective and that it is arguably the most efficient among the compared deep-learning tools.

ACKNOWLEDGE

We would like to thank the support of the President's PhD Scholarship from Imperial College London.

APPENDIX: PROOF OF FIRST-ORDER APPROXIMATION (6)

According to (1),

$$q_h = \frac{\lambda_h^+ + \sum_{v=1}^N q_v r_v p_{vh}^+}{r_h + \lambda_h^- + \sum_{v=1}^N q_v r_v p_{vh}^-}.$$

Let $\phi = \lambda_h^- + \sum_{v=1}^N q_v r_v p_{vh}^-$. Then,

$$q_h = \frac{\lambda_h^+ + \sum_{v=1}^N q_v r_v p_{vh}^+}{r_h + \phi}.$$

Then,

$$\frac{\partial q_h}{\partial \phi} = -\frac{\lambda_h^+ + \sum_{v=1}^N q_v r_v p_{vh}^+}{(r_h + \phi)^2}.$$

Suppose a is a real number satisfying $r_h >> a$. Then, we can approximate q_h in a neighbourhood $\phi = r$ as:

$$\begin{aligned}
q_h &\approx q_h\,|_{\phi=a} + \frac{\partial q_h}{\partial \phi}\,|_{\phi=a}\,(\phi - a) \\
&= \frac{\lambda_h^+ + \sum_{v=1}^N q_v r_v p_{vh}^+}{r_h + a} - \frac{\lambda_h^+ + \sum_{v=1}^N q_v r_v p_{vh}^+}{(r_h + \phi)^2}(\phi - a) \\
&\approx \frac{\lambda_h^+ + \sum_{v=1}^N q_v r_v p_{vh}^+}{r_h} - \frac{\lambda_h^+ + \sum_{v=1}^N q_v r_v p_{vh}^+}{r_h^2}(\phi - a) \\
&\approx \frac{\lambda_h^+ + \sum_{v=1}^N q_v r_v p_{vh}^+}{r_h}(1 - \frac{\phi}{r_h} + \frac{a}{r_h})
\end{aligned}$$

$$\approx \frac{\lambda_h^+ + \sum_{v=1}^N q_v r_v p_{vh}^+}{r_h}(1 - \frac{\phi}{r_h}).$$

Then,

$$q_h \approx \frac{\lambda_h^+ + \sum_{v=1}^{N} q_v r_v p_{vh}^+}{r_h}(1 - \frac{\lambda_h^- + \sum_{v=1}^{N} q_v r_v p_{vh}^-}{r_h}).$$

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] ——, "G-networks: a unifying model for neural and queueing networks," *Annals of Operations Research*, vol. 48, no. 5, pp. 433–461, 1994.

[4] ——, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, 1993.

[5] C. Cramer, E. Gelenbe, and H. Bakircloglu, "Low bit-rate video compression with neural networks and temporal subsampling," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1529–1543, 1996.

[6] E. Gelenbe and C. Cramer, "Oscillatory corticothalamic response to somatosensory input," *Biosystems*, vol. 48, no. 1, pp. 67–75, 1998.

[7] S. Mohamed and G. Rubino, "A study of real-time packet video quality using random neural networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 12, pp. 1071–1083, 2002.

[8] G. Rubino and M. Varela, "A new approach for the prediction of end-to-end performance of multimedia streams," in *Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings. First International Conference on the*. IEEE, 2004, pp. 110–119.

[9] M. Martínez, A. Morón, F. Robledo, P. Rodríguez-Bocca, H. Cancela, and G. Rubino, "A grasp algorithm using rnn for solving dynamics in a p2p live video streaming network," in *Hybrid Intelligent Systems, 2008. HIS'08. Eighth International Conference on*. IEEE, 2008, pp. 447–452.

[10] H. Larijani and K. Radhakrishnan, "Voice quality in voip networks based on random neural networks," in *Networks (ICN), 2010 Ninth International Conference on*. IEEE, 2010, pp. 89–92.

[11] K. Radhakrishnan and H. Larijani, "Evaluating perceived voice quality on packet networks using different random neural network architectures," *Performance Evaluation*, vol. 68, no. 4, pp. 347–360, 2011.

[12] T. Ghalut and H. Larijani, "Non-intrusive method for video quality prediction over lte using random neural networks (rnn)," in *Communication Systems, Networks & Digital Signal Processing (CSNDSP), 2014 9th International Symposium on*. IEEE, 2014, pp. 519–524.

[13] E. Gelenbe and S. Timotheou, "Random neural networks with synchronized interactions," *Neural Computation*, vol. 20, no. 9, pp. 2308–2324, 2008.

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

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

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[17] 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.

[18] 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.

[19] 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.

[20] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *to appear in IEEE Transactions on Neurla Networks and Learning Systems*, May 2015.

[21] Y. Yin and E. Gelenbe, "Deep Learning in Multi-Layer Architectures of Dense Nuclei," *ArXiv e-prints*, Sep. 2016.

[22] 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.

[23] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[24] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks." in *Aistats*, vol. 15, no. 106, 2011, p. 275.

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

[26] Y. Yin and E. Gelenbe, "Nonnegative autoencoder with simplified random neural network," *ArXiv e-prints*, Sep. 2016.

[27] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[28] I. MODULE, "Googlenet: Going deeper with convolutions."

[29] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for matlab," in *Proceeding of the ACM Int. Conf. on Multimedia*, 2015.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[31] 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.

[32] 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.

[33] Y. Yin and Y. Zhang, "Weights and structure determination of chebyshev-polynomial neural networks for pattern classification," *Software*, vol. 11, p. 048, 2012.

[34] 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.

[35] F. Chollet, "Keras," https://github.com/fchollet/keras, 2015.

[36] 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.

[37] K. Altun, B. Barshan, and O. Tunçel, "Comparative study on classifying human activities with miniature inertial and magnetic sensors," *Pattern Recognition*, vol. 43, no. 10, pp. 3605–3620, 2010.

[38] B. Barshan and M. C. Yüksek, "Recognizing daily and sports activities in two open source machine learning environments using body-worn sensor units," *The Computer Journal*, vol. 57, no. 11, pp. 1649–1667, 2014.

[39] K. Altun and B. Barshan, "Human activity recognition using inertial/magnetic sensor units," in *International Workshop on Human Behavior Understanding*. Springer, 2010, pp. 38–51.

[40] J. Fonollosa, L. Fernández, A. Gutiérrez-Gálvez, R. Huerta, and S. Marco, "Calibration transfer and drift counteraction in chemical sensor arrays using direct standardization," *Sensors and Actuators B: Chemical*, 2016.

[41] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.