A Developmental Approach to Structural Self-Organization in Reservoir Computing

Jun Yin, Yan Meng and Yaochu Jin

Abstract—Reservoir computing (RC) is a computational framework for neural network based information processing. Little work, however, has been conducted on adapting the structure of the neural reservoir. In this paper, we propose a developmental approach to structural self-organization in reservoir computing. More specifically, a recurrent spiking neural network is adopted for building up the reservoir, whose synaptic and structural plasticity are regulated by a gene regulatory network (GRN). Meanwhile, the expression dynamics of the GRN is directly influenced by the activity of the neurons in the reservoir. We term this proposed model as GRN-regulated self-organizing RC (GRN-SO-RC). Contrary to a randomly initialized and fixed structure used in most existing RC models, the structure of the reservoir in the GRN-SO-RC model is self-organized to adapt to the specific task using the GRN-based mechanism. To evaluate the proposed model, experiments have been conducted on several benchmark problems widely used in RC models, such as memory capacity and nonlinear auto-regressive moving average. In addition, we apply the GRN-SO-RC model to solving complex real-world problems, including speech recognition and human action recognition. Our experimental results on both the benchmark and real-world problems demonstrate that the GRN-SO-RC model is effective and robust in solving different types of problems.


I. INTRODUCTION

Many complex real-world problems, such as object tracking, speech recognition, human action recognition, contains inherent temporal information: not only the value of the inputs is important, but also their specific sequence and precise occurrence in time are critical for problem-solving. However, most machine learning models used for temporal pattern recognition either do not explicitly consider the temporal aspect of the input (i.e., the temporal correlation between different time windows is not preserved), or simply compress the time-dependent inputs to static input vectors (some temporal information may be lost).

One option to deal with temporal information in the input patterns is to add recurrent connections to the learning model. Theoretically, recurrent neural networks (RNNs) are powerful tools for solving complex engineering tasks involving dynamic temporal information. Major application areas of RNNs include learning of context free and context sensitive languages [1], speech recognition [2], [3], object tracking and motion prediction [4], [5], and event detection [6]. Nonetheless, several issues remain unresolved that prevent us from applying RNNs to a wider class of real-world problems. For example, most training methods for RNNs suffer from high computational cost and slow convergence rate [7]. Another major issue of the learning algorithms for RNNs is the so-called fading gradient [8], which means that the error gradient vanishes or gets distorted when it is propagated many time steps backward.

In recent years, various training approaches to RNNs have been proposed to reduce the computational cost and address the fading gradient issue, such as liquid state machine (LSM) [9] and echo state network (ESN) [10]. The back propagation decorrelation (BPDC) learning algorithm was proposed in [11], showing that a reservoir-like learning technique can be justified directly from an error minimization approach. The adopted RNNs usually have sparse, fixed random connections and are commonly termed the reservoirs. In [12], the term "reservoir computing" (RC) is suggested to refer to the above approaches that have a common computational framework. In RC, typically, an input signal is fed into a dynamic system (called reservoir) and the dynamics of the reservoir map the input to a higher dimension. Then a simple readout mechanism is trained to read the state of the reservoir and map it to the desired output. Since the training is performed on the readout weights only, the training of the RC models is computationally more efficient compared to that if other RNN models.

RC is of particular interest as it has often been used as a platform for studying how brain works [13], [14], [15], [16]. Note, however, that synaptic connections in the brain have a very high degree of plasticity. In addition to synaptic plasticity, it has also been revealed that the structure and connectivity of the neurons in the brain can change dramatically depending on the neuronal activities [17].

Recent findings suggest that activity-dependent neural plasticity is resulted from changes in expression of relevant genes and thus their protein products [18]. Inspired from these biological studies, the computational neuro-genetic modeling (CNGM) was proposed in [19], [20], which integrated dynamic gene networks with an artificial neural network.
model. Recently, Weng et al. [21], [22] proposed a new lobe component analysis (LCA) based neural plasticity model, which is an in-place learning method with dual (spatiotemporal, “best” and “fast”) optimality. This LCA-based plasticity model is also motivated by the developmental process in multi-cellular mechanisms where an emergent property of cell-centered development regulated by the genes.

Inspired by the biological findings that activity dependent neural plasticity is controlled by changes in protein concentrations under the governance of by gene regulations [18], we developed a gene regulatory network (GRN) mechanism to implement plasticity in both structure and connection strength of neurons in the reservoir. This kind of structural and synaptic plasticity allows the reservoir to adapt their internal dynamics to the given task. One important feature of activity-dependent plasticity is that the dynamics of GRN is also influenced by the activity of the neurons it resides in, resulting in a coupled dynamic system. Compared with the random and fixed reservoir in most existing RC models, the RC model proposed in this work adopts a developmental approach to self-organizing the structure of the reservoir. We call this proposed model as GRN-regulated self-organizing RC (GRN-SO-RC).

This GRN-SO-RC model is extended from our previous work [23], where the parameters for plasticity and meta-plasticity of a feed-forward three-layer spiking neural network are tuned using a GRN. The main weakness of feed-forward spiking neural networks is that it has a relatively limited memory capacity due to limited computer memory, although the spiking neurons regulated by a GRN can encode a certain amount of temporal information in the input space. This work attempts to address this weakness by embedding a recurrent spiking neural network into the reservoir framework. RC models are fundamentally different from feed-forward architectures in the sense that they not only operate on an input space but also on an internal state space – a trace of what already has been processed by the network. Furthermore, reservoir computing models like any recurrent networks can use their internal memory to process arbitrary sequences of inputs. An additional step forward is that the structure of the recurrent spiking network of the reservoir is also regulated online by a GRN to further improve the adaptation capability of the RC model.

In this paper, we will first evaluate the performance of the GRN-SO-RC model on two popular benchmark problems: the memory capacity task [24] and the nonlinear auto-regressive moving average (NARMA) task [25], [26]. Additionally, we apply the GRN-SO-RC model to two real-world problems, i.e., speech recognition and human action recognition to evaluate its effectiveness in temporal pattern recognition.

This rest of the paper is organized as follows. Reservoir computing and other related work are introduced in Section II. Section III introduces a GRN-based realization of structural and synaptic plasticity of the recurrent spiking neural network. Section IV presents the learning results on two benchmark problems. Applying the GRN-SO-RC model to speech recognition has been discussed in Section V, followed by a description of the comparative studies on human action recognition in Section VI. A conclusion and future work is provided in Section VII.

II. RESERVOIR COMPUTING

The reservoir is the internal network of a reservoir computing network, which consists of a set of \( N \) neurons, interfacing with a layer of \( K \) input units and \( L \) output units, one for each class, as shown in Fig. 1.

![Fig.1. The architecture of the reservoir computing network. The reservoir is the internal network.](image)

The fundamental idea of the reservoir computing network is that the inputs to the reservoir excite its nonlinear dynamics, which leads to a complex nonlinear transformation of the input signals into a high-dimensional vector of reservoir neuron states. These state vectors can be regarded functionally as the nonlinear temporal feature vectors with encoded temporal information of the inputs. Input-output mapping can be learned by training the output connections only through readout functions, resulting in a significant speed-up in learning. Meanwhile, the output can be fed back into the internal states (see dashed arrows in Fig. 1). A fixed and randomly created structure is often considered to be a major advantage of traditional RC models. Consequently, various ways of constructing reservoir structures and weight matrices for different applications can be described by a uniform framework [27]. For different RC models, the reservoir can be composed of different types of neurons, e.g. linear units [10], sigmoid neurons [11], threshold gates [28], or spiking neurons [9].

Liquid state machine (LSM) [9] and echo state network (ESN) [10] are two major types of RCs. In practice, most LSMs use a reservoir built from a relatively simple spiking neuron model called the leaky integrate-and-fire (LIF) neuron [29] with a dynamic synapse model [30]. The readout layer of the LSM is also arbitrarily specified as long as it can approximate any continuous function. Sigmoidal neurons are usually used in the ESN model. The echo state property in the ESN indicates that the influence of inputs on the reservoir states fades gradually over time. Some guidelines of searching for a good ESN model for a given problem are offered in [15], yet a systematic method remains to be investigated.

Steil [11] proposed a new RC model called back propagation decorrelation (BPDC). The BPDC rule is an extension of the Aitiya-Parlos recurrent learning [31] for...
RNNs. Sigmoidal neurons are used in BPDC models. The major difference between the BPDC and ESN models is that in the BPDC model, there are feedback connections from the readout layer connected into the reservoir and the readout layer as well, whereas there is no such feedback connection in ESNs. This kind of feedback connections has also been adopted in other models, e.g., in [32], [33].

Typically, the structure of a reservoir is created with a certain degree of sparseness and the weights are assigned with a random number drawn from a normal or uniform distribution. Then, the weight matrix is globally scaled to set the spectral radius to a certain value. As a result, the variance in the performance across different reservoirs can be substantial. Some research efforts for optimizing the weights of a reservoir for specific applications have been reported recently. For example, an unsupervised learning algorithm such as spike-time-dependent plasticity [34] or intrinsic plasticity [35], [36] can automatically tune internal dynamics of reservoir. However, it is difficult for these learning algorithms to achieve the desired reservoir if sigmoid neuron models are used, as they exhibit a serious fading memory of time-varying inputs [7], [37].

To apply the RC models to real-world applications, such as temporal pattern recognition, a reservoir can be any type of the neural network that has sufficient internal dynamics. In principle, spiking neural networks with Hebbian plasticity have been considered well suited for learning temporal patterns [38]. Pitti et al. [39] proposed an algorithm to detect contingency in sensorimotor networks inspired by the biological mechanism of spike timing-dependent plasticity. Luciw and Weng [33] used top-down connections in a self-organizing Hebbian learning network, which has been applied to the topographic class grouping for static visual recognition problems. Evidence found in neuroscience studies suggests that the precise timing of the spike firing is critical for cognitive processing [40]. From this point of view, spiking RNNs can provide an efficient cognitive tool for modeling the synaptic interactions between the neurons. Therefore, in this paper, we employ a spiking RNN to construct the reservoir in a RC model, which will be discussed in the Section III.

III. A SELF-ORGANIZING RESERVOIR COMPUTING MODEL

A. RC Using a Recurrent Spiking Neural Network

In this paper, a recurrent spiking neural network (RSNN) is applied to construct the reservoir in a GRN-SO-RC model, which is shown in Fig. 2. This GRN-SO-RC model consists of two different dynamics, i.e., an RSNN and a gene regulatory network (GRN). The main challenge here is how to couple the neural dynamics with that of the GRN so that the structure of the reservoir can be optimized for specific tasks at hand.

Fig. 2. The diagram of the GRN-SO-RC model

As shown in Fig. 2, the RSNN consists of $K$ input units with an activation vector $u(t) = (u_1(t), \ldots, u_K(t))^T$, $N$ internal units with an activation vector $x(t) = (x_1(t), \ldots, x_N(t))^T$, and $L$ output units with an activation vector $y(t) = (y_1(t), \ldots, y_L(t))^T$. The input, internal, output, and feedback weight matrices are defined as $W^{in} = (w^{in}_{ij})$ of a size of $N \times K$, $W = (w_{ij})$ of a size of $N \times N$, $W^{out} = (w^{out}_{ij})$ of a size of $L \times N$, and $W^{back} = (w^{back}_{ij})$ of a size of $N \times L$, respectively. Specifically, $w^{in}_{ij}$ denotes the synaptic weight from input neuron $i$ to internal neuron $j$. $w_{ij}$ denotes the synaptic weight from internal neuron $i$ to internal neuron $j$. $w^{out}_{ij}$ denotes the synaptic weight from internal neuron $i$ to output neuron $j$. $w^{back}_{ij}$ defines the synaptic weight from output neuron $i$ projecting back to internal neuron $j$. The activation vector of the internal units is updated according to the following equation:

$$x(t + 1) = W^{in} \cdot f(u(t + 1)) + W \cdot x(t) + W^{back} \cdot f(y(t)),$$

(1)

where $x(t + 1)$ denotes the concatenated vector built from the internal vectors in the reservoir. $u(t + 1)$ is the input, and $f$ denotes the transfer function of the individual neurons, which is defined as $f(t) = \frac{L}{\tau} \cdot \exp(1 - \frac{t}{\tau})H(t)$, thus implementing a leaky-integrate-and-fire (LIF) spiking neuron. $H(t)$ is the Heavy-side step function: $H(t) = 0$ for $t \leq 0$ and $H(t) = 1$ for $t > 0$. The time-constant $\tau$ is set to 2.7. The output $y(t + 1)$ can be computed as follows:

$$y(t + 1) = f^{out}(W^{out} \cdot x(t + 1)),$$

(2)

where $f^{out}(\cdot)$ is defined as the output neuron transfer function (i.e. readout function) that maps the internal state $x(t)$ into the target output $y(t)$ at any time $t$. Possible readout functions include linear projection, Fisher discriminant, a perceptron or a feed forward MLP. Actually, it usually suffices to use linear readout functions [41]. In this paper, an identity linear projection function, i.e., $y(t + 1) = \sum_{i} w^{out}_{i} \cdot x(t + 1)$, is used as
The readout function which can linearly map the internal states onto the outputs. However, we cannot simply use the binary output of the spiking reservoir for the linear readout functions. Instead, we apply a low-pass filter on the output of the reservoir neurons [9]. Though rather simplistic, a low-pass filter can be applied to transform the spike-trains into continuous output that can be weighted and fed to the readout functions. In other words, the outputs are of continuous values.

The formulation of a synaptic modification rule, known as Bienenstock-Cooper-Munro (BCM) theory [23], is used to train the synaptic plasticity for the RSNN. The BCM theory of synaptic plasticity has successfully reproduced the development of orientation selectivity and ocular dominance in kitten visual, and is also arguably the most accurate model of synaptic plasticity [42]. Basically, the BCM rule combines Hebbian and anti-Hebbian synaptic modulation [43]. Additionally, the BCM learning rule follows directly from spike-timing-dependent plasticity (STDP) when pre- and postsynaptic neurons fire uncorrelated or weakly correlated Poisson spike trains, and only nearest-neighbor spike interactions are taken into account [44]. Statistical analysis shows that the STDP rule is closely related to the widely used BCM rule [45]. We use a simplified, discrete-time version of the BCM-based RSNN.

The equations governing the behavior of synaptic weights of the RSNN can be described as follows:

\[ w_i(t + 1) = \eta x_i(t) \phi(y(t), \theta(t)) + (1 - \varepsilon) w_i(t) \]  
\[ \phi(y(t), \theta(t)) = y(t)(y(t) - \theta(t)) \]  
\[ \theta(t) = \frac{\sum_{t' = t-h}^{t} y^{2}(t') \lambda^{t-t'}}{\sum_{t' = t-h}^{t} \lambda^{t-t'}} \]

where \( w_i \) denotes the synaptic weight of the \( i \)-th synapse (note: we use a weight term here for the BCM model in order to be consistent with the training of other weights in the RC model). \( \eta \) is a constant learning rate. \( x_i \) is the pre-synaptic input of the \( i \)-th synapse. \( y \) is the post-synaptic output activation level. \( \theta \) is a sliding modification threshold. \( \phi(.) \) is a non-linear activation function that swings with the sliding threshold \( \theta \), and can be defined by Eqn. (4). \( \varepsilon \) is a time-decay constant that is the same for all synapses. The interpretation of \( \theta \) in Eqn. (5) clearly shows that the sliding threshold is a time-weighted average of the squared post-synaptic signals \( y \) within the time interval of \( h \), where \( \lambda \) is a forgetting factor.

**B. Synthetic and Structural Plasticity of the Reservoir Regulated by a GRN Model**

Contrary to the existing methods with a fixed and randomly wired structure in the reservoir, we propose in this work a developmental approach to self-organizing the structure of the reservoir. It has been observed that both the structure and synaptic weights of the neurons in the brain can be changed based on activity-dependent functions [17]. Such activity dependency over time can be taken advantage for learning temporal patterns. Furthermore, it has been found that the expression of relevant genes can influence the activity-dependent neural plasticity [18]. To model the activity dependent plasticity of neural networks, a gene regulatory model is needed.

Among others [39], [46], [47], [48], [49], [50], ordinary or partial differential equations have been widely used to model regulatory networks. For example, Mjolsness et al. [49] proposed a GRN model for describing the gene expression data of developmental processes as follows:

\[ \frac{dg_{ij}}{dt} = -\xi_j g_{ij} + \psi \left( \sum_{\ell=1}^{n} W^{j\ell} g_{i\ell} + \nu_j \right) + D_j \lambda^2 g_{ij} \]

where \( g_{ij} \) denotes the concentration of \( j \)-th gene product (protein) in the \( i \)-th cell. The first term on the right-hand side of Eqn. (6) represents the degradation of the protein at a rate of \( \xi_j \), the second term specifies the production of protein \( g_{ij} \), and the last term describes protein diffusion at a rate of \( D_j \). \( \psi \) is an activation function for the protein production, which is usually defined as a sigmoid function \( \psi(z) = 1/(1 + \exp(-\mu z)) \).

The interaction between genes is described with an interaction matrix \( W^{j\ell} \), the element of which can be either active (a positive value) or repressive (a negative value). \( \nu_j \) is a threshold for the activation of gene expression of gene \( j \). \( n_g \) is the number of proteins.

Inspired by these studies and the GRN model developed in [49], we propose a GRN-based model to regulate synaptic and structural plasticity of the reservoir to adapt to different tasks at hand. In order to apply the BCM theory to the modifications of synaptic weights of the RSNN, three plasticity parameters (\( \eta, \varepsilon, \lambda \)) in the BCM model need to be defined. First, the gene expression levels that correspond to three plasticity parameters (\( \eta, \varepsilon, \lambda \)) in the BCM model described in Eqns. (3)-(5) can be defined by a set of discrete-time ordinary differential equations of GRN as follows:

\[ \eta(t+1) = (1 - \gamma \eta) \eta(t) + \alpha_{\eta} g(c_{Na^+}, c_{Ca^{2+}}(t)) \]  
\[ \varepsilon(t+1) = (1 - \gamma \varepsilon) \varepsilon(t) + \alpha_{\varepsilon} g(c_{Na^+}, c_{Ca^{2+}}(t)) \]  
\[ \lambda(t+1) = (1 - \gamma \lambda) \lambda(t) + \alpha_{\lambda} g(c_{Na^+}, c_{Ca^{2+}}(t)) \]

where protein expression levels for sodium ion and calcium ion concentration (\( c_{Na^+}, c_{Ca^{2+}} \)) are defined as:

\[ c_{Na^+}(t+1) = (1 - \gamma_{Na^+}) c_{Na^+}(t) + \alpha_{Na^+} \sum_{j} x_j(t) w_j(t) \]  
\[ c_{Ca^{2+}}(t+1) = (1 - \gamma_{Ca^{2+}}) c_{Ca^{2+}}(t) + \alpha_{Ca^{2+}} \theta(t) \]

where \( \gamma_{Na^+}, \gamma_{Ca^{2+}}, \alpha_{Na^+}, \alpha_{Ca^{2+}} \) are coefficients. \( w_j \) and \( x_j \) are the synaptic weight and pre-synaptic input of the \( i \)-th synapse defined in Eqn. (3), which affect the protein expression levels of sodium ion \( c_{Na^+} \). \( \theta \) is the sliding threshold defined in Eqn. (5), which influences the protein expression level of calcium ion \( c_{Ca^{2+}} \). \( g(.) \) is the activation function of the gene
expression of three plasticity parameters ($\eta$, $\epsilon$, $\lambda$), which is defined as the following sigmoid function:

$$g(c_{Na}, c_{Ca}) = \left(1 + e^{-k_1c_{Na} + k_2c_{Ca}} \right)^{-1}$$  \hspace{1cm} (12)

where $k_1$ and $k_2$ are coefficients.

Compared with the GRN model described in Eqn. (6), the protein diffusion term is ignored in our proposed GRN model. The sliding threshold $\theta$ of each spiking neuron in the reservoir is adjusted by the GRN model simulating homeostatic plasticity. A spiking neuron that is being activated very frequently tends to raise its threshold while an inactive neuron tends to reduce its threshold gradually. This rule will help to encourage every neuron to participate in the network dynamics on a regular basis, which aims at ensuring the population sparseness. Therefore, the sliding threshold of a neuron becomes correlated with the strengths of the afferent connections to the neuron. It is noted that only the synaptic weights of internal neurons $w_{ij}$ in the reservoir are regulated by the GRN, while the synaptic weights of output neurons are trained separately (will be discussed in Section III.C).

The reservoir is initialized with a full connectivity, but all synaptic weights are allowed to grow or shrink subsequently according to the GRN regulation model. A predefined threshold 0.005 is set to prune the synaptic weights. When the network converges, those synaptic connections whose weights are less than the threshold will be pruned from the reservoir consequently. Only those synapse connections with a weight higher than the threshold will be kept in the reservoir. In this way, both structure and synaptic weight strength of the reservoir can be regulated by the GRN. As a result, an optimal structure of the reservoir can be generated for different tasks in a self-organizing way.

Pruning connections having a very weak weight simulates an important phenomenon observed in biological neural development in which synaptic connections that are very rarely used will die out. The pruning of the weak connections is an important part of activity-dependent neural self-organization. No significant performance degradation is observed if these weak connections are kept, although the computational cost will be much higher.

In this GRN-SO-RC model, 12 parameters in total, namely, $\gamma_y, \gamma_e, \gamma_L, \gamma_{Na}, \gamma_{Ca}, \alpha_y, \alpha_e, \alpha_L, \alpha_{Na}, \alpha_{Ca}, k_1$, and $k_2$, need to be defined for the GRN model. Fine tuning these parameters for each specific task manually to achieve optimal performance is tedious and time-consuming. Therefore, an evolutionary algorithm, called Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [51], [52] is adopted to optimize these parameters. We will define the fitness function of the CMA-ES method in Section III.C.

Please be noted that the GRN-SO-RC model is not a uniform plasticity model. Actually, Eqs. (3)-(5) describe the dynamics of the synaptic weights of the spiking neurons based on the BCM model. Since different neurons have different input signals, different local connections, and different activities, the resulting plasticity parameters of the BCM model are different for different neurons.

### C. Training of the GRN-SO-RC Model

The training data for the GRN-SO-RC model consists of $T$ input-output (vector-valued) data pairs: $u(t) = (u_1(t), \ldots, u_K(t))$, and $d(t) = (d_1(t), \ldots, d_L(t))$, where $t$ denotes the number of training instances. The objective of the training is to find a set of synaptic weights of the reservoir such that the summed squared error

$$E = \sum_{t=1}^{T} \|d(t) - y(t)\|^2 = \sum_{t=1}^{T} E(t)$$  \hspace{1cm} (13)

is minimized. At each time step $t$, $x_i(t)$ is $i$-th internal neuron activation, $d_j(t)$ is the target value of $j$-th output neuron and $y_j(t)$ is the actual value of $j$-th output neuron. Furthermore, $\delta_j^{\text{out}}(t)$ and $\delta_i^{\text{back}}(t)$ are the back propagated errors for the synaptic weights from the reservoir to the output neuron and from the output neuron to the reservoir (since this is a recurrent neural network), respectively, which are defined as follows:

$$\delta_j^{\text{out}}(t) = \left[ (d_j(t) - y_j(t)) + \sum_{t=1}^{T} \delta_j^{\text{back}}(t+1) \right] \frac{\partial f(u)}{\partial u}$$  \hspace{1cm} (14)

$$\delta_i^{\text{back}}(t) = \sum_{j=1}^{N} \delta_j^{\text{out}}(t+1) w_{ji} + \sum_{j=1}^{N} \delta_i^{\text{out}}(t) w_{ji}^{\text{out}}$$  \hspace{1cm} (15)

Then the synaptic weights of input and output neurons of the RC model can be adjusted according to the following equations:

$$w_{ij}^{\text{in(new)}} = w_{ij}^{\text{in(old)}} + \gamma \sum_{t=1}^{T} \delta_i^{\text{back}}(t+1) u_j(t)$$  \hspace{1cm} (16)

$$w_{ij}^{\text{out(new)}} = w_{ij}^{\text{out(old)}} + \gamma \sum_{t=1}^{T} \delta_i^{\text{back}}(t) y_j(t-1)$$  \hspace{1cm} (17)

$$w_{ij}^{\text{back(new)}} = w_{ij}^{\text{back(old)}} + \gamma \sum_{t=1}^{T} \delta_i^{\text{back}}(t+1) y_j(t-1)$$  \hspace{1cm} (18)

where $w_{ij}^{\text{in}}, w_{ij}^{\text{out}},$ and $w_{ij}^{\text{back}}$ represent the connection weights from the input neuron to the reservoir, from the reservoir to the output neuron, and from the output neuron to the reservoir, respectively. $\gamma$ is the learning rate.

The logic flow of the training procedure for the GRN-SO-RC model is summarized in Fig. 3. Please be noted that only internal neurons in the reservoir are spiking neurons, while input and output neurons are analog neurons. Therefore, the back propagation algorithm can be used to train the synaptic weights of input and output neurons directly using Eqsns. (16)-(18), but cannot be used to train the internal neurons in the reservoir directly.

Evolutionary algorithms have been shown effective in training spiking neural networks [53]. To adapt the weights connecting the spiking neurons in the reservoir, CMA-ES has been adopted to minimize the following classification error:

$$\frac{1}{2M} \sum_{m=1}^{M} \left \|d(m) - x(m) \cdot \left( w_{ij}^{\text{out}} + \gamma \sum_{t=1}^{T} \delta_i^{\text{back}}(t) y_j(t-1) \right) \right \|^2,$$  \hspace{1cm} (19)

which is the fitness function of the CMA-ES method. Based on this fitness function, the CMA-ES method is employed to
optimize the parameters of the GRN accordingly. Then the GRN model will regulate the plasticity parameters of the BCM model which further dynamically change the synaptic weights/structure of the reservoir through the plasticity mechanism during the training. It is important to tune the parameters of the GRN because different tasks may require different reservoir synaptic structure/weights. For each task, the CMA-ES method needs to optimize 12 parameters of the GRN model using all the training samples. In this way, the generated GRN parameters are optimized for all the training samples, and we can apply this set of optimized GRN parameters to different testing samples.

Fig. 3. The training logic flow of the GRN-SO-RC model.

Since the back propagation method cannot be directly applied to the proposed recurrent neural network (RNN) model GRN-SO-RC (the internal neurons in the reservoir are LIF-based spiking neurons), we adopted the back propagation through time (BPTT) method [54] as the training approach for the GRN-SO-RC model. BPTT method is a gradient-based method for training certain types of RNNs. Basically, BPTT aims to transform a RNN to a feed-forward neural network (FWDD) by "unfolding" the RNN through time. When the network starts processing a data sequence, identical copies of the RNN are created and the connections within the RNN are redirected to generate new feed-forward connections over time from one network to the subsequent network. Eventually the RNN is converted to a large FDNN. Then the training proceeds in a similar manner to training a FDNN with back propagation.

To make it easier to understand, the pseudo code of the training process of the GRN-SO-RC model is listed as follows:

Generate $K(=20)$ individuals for the initial populations, where each individual has $P(=12)$ parameters of the GRN model, i.e.,

\[ \gamma_0, \gamma_1, \gamma_{\text{in}}, \gamma_{\text{out}}, \gamma_{\text{back}}, \alpha_0, \alpha_e, \alpha_w, \alpha_{\text{in}}, \alpha_{\text{out}}, \alpha_{\text{back}}, \text{and } k_1, k_2 \]

Initialize the synaptic weights $w_{\text{in}}^0, w_{\text{out}}^0$, and $w_{\text{back}}^0$

Unfold the network to contain $T$ instances of reservoir;

While (stopping criteria (either classification errors are less than a predefined threshold or maximum iteration is reached) is met) do

For $i = 1$ to $M$ ($M$ is number of training samples) do

For time step $t = 1$ to $T$ do

Compute the propagation errors according to Eqns. (14) and (15);

End For

Update the weights $w_{ij}^{\text{in}}, w_{ij}^{\text{out}},$ and $w_{ij}^{\text{back}}$ according to Eqns. (16)-(18);

End For

For generation $l = 1$ to $L$ do

For individual $k = 1$ to $K$ do

For $i = 1$ to $M$ do

For $t = 1$ to $T$ do

Update GRN model including sodium and calcium ion concentration ($c_{Na}$, $c_{Ca^{2+}}$)

According to Eqns. (10), (11), and three plasticity parameters ($\eta$, $\epsilon$, $\lambda$) of BCM according to Eqns. (7)-(9);

Update $w$, $\phi(\cdot)$ and $\theta$ of BCM according to Eqns. (3)-(5);

End For

End For

End For

End For

End While

There are two developmental components in the GRN-SO-RC model: neural dynamics (determined by BCM parameters) and gene regulation dynamics (determined by GRN model). From the neuroscience point of view, the timescale of neural dynamics should be faster than that of the gene regulation dynamics. Although one set of GRN parameters can be used to regulate the BCM plasticity parameters multiple times, interestingly, we have conducted some experiments in our previous work [23] to show that the recognition performance is not sensitive to the change in the time scales of the two dynamic systems. The major performance improvement comes from the GRN regulation on the BCM (we will demonstrate this point in Sec. IV, V, and VI through experiments). Therefore, to reduce the overall computation cost, we only tune BCM parameters once for each GRN dynamics (i.e., the same timescale for both dynamics).

Since the back propagated errors need to be computed only once for each new time step, the complexity is $O(MT^2)$ per time step. In the reservoir, $P = 12$ parameters need to be evolved and $K = 20$ individuals per generation in the CMA-ES. Assuming the number of training sample is $M$, the average number of time steps for each training sample is $T$, and the number of internal neurons in the reservoir is $N$, then the average computational complexity of the learning algorithm within $L$ generations is $O(KLMNT + P^2)$, where $L$ is dependent on the stop criterion. Therefore, the overall
computational complexity for the training of the GRN-SO-RC model is $O(MT^2 + KL(MNT + P^2))$.

IV. BENCHMARK EXPERIMENTS OF THE GRN-SO-RC MODEL

To evaluate the performance of the GRN-SO-RC model, comparative studies on two benchmark problems are conducted, including the memory capacity task [24] and the nonlinear auto-regressive moving average (NARMA) task [25], [26]. Spiking neural networks use sets of spike-trains as their inputs, but usually the signals for computation in these two benchmarks are analog data. We need to encode the analog data in terms of spike trains in order to feed them to the proposed network. A filter encoding scheme, called Bens Spiker Algorithm (BSA) [55] is applied for converting analog values to spike trains.

The common way to optimize a reservoir is to set the spectral radius of the inter connection matrix to a given value. This method, however, has a few limitations. For example, the spectral radius can be used to measure the stability/dynamics of the reservoir only if the reservoir is not driven by the inputs [56]. However, there are many possible weight matrices with the same spectral radius, and unfortunately they do not all present the same level of performance for functional approximation. In other words, a quite large variance on the performance may occur when creating several random reservoirs with the same spectral radius. We will show that the GRN-SO-RC model can resolve these two issues using the NARMA benchmark.

First, from a system theory perspective, the eigenvalues (poles) of a linear system implemented by a RC model correspond to the band-pass filters with center frequencies according to their angles in the complex plane [56]. Larger eigenvalues will lead to longer time constants for the filters, which means longer information preservation in the network. According to [57], the spectral radius is a key parameter for a RC model and the optimum can be achieved when a spectral radius is close to one. Here, the spectral radius only depends on the connection weights between the neurons in the reservoir (i.e., the structure of the networks). For example, Fig. 4 shows that the spectral radius of the GRN-SO-RC model for the NARMA benchmark task is $\rho = 0.95$, which is close to one. This spectral radius can guarantee asymptotic stability, which means that the dynamics stimulated by an input will eventually die out. In other words, the proposed GRN-SO-RC model can automatically generate an optimal spectral radius for the given task, which indicates that the proposed model can resolve the first issue in common reservoir computing models.

On the other hand, since the synaptic and structural plasticity are regulated by the GRN, Fig. 5 shows the weight histogram of a reservoir with 100 nodes after the regulation, which is also generated from the NARMA benchmark task. It can be seen from Fig. 5 that most of synaptic weights (>50%) are closed to zero and only those synaptic connections with a weight larger than a predefined threshold will be kept in the reservoir (as we mentioned in Section III.B). We performed ten independent runs on this benchmark task using both GRN-SO-RC model and the standard RC model. The variances of the results using our model and the standard RC model are 0.0012 and 0.0241, respectively. It can be seen that the performance variance from the GRN-SO-RC model is much smaller. This example demonstrates that both the synaptic and structural plasticity can be automatically regulated for a given task, which resolves the second issue in the reservoir computing.

A. Benchmark 1: Memory Capacity Task

First we will evaluate the short-term memory performance of the GRN-SO-RC model on the memory capacity (MC) benchmark problem introduced by Jaeger [24], which is a measure of how much information contained in the input can
be extracted from the reservoir after a certain time. The input of the reservoir consists of a temporal signal \( u(t) \), which is drawn from a uniform distribution in the interval \([-0.8, 0.8]\). The desired outputs consist of a series of delayed versions of the input signal. In this case, a series of 100 repressors was trained to reproduce the input signal delayed up to 100 time steps. The stopping criterion for this task is set up as follows: mean error = 0.005 or maximum iterations = 600. The performance measure is the memory capacity (MC) [24], which is defined as:

\[
MC = \sum_{k=1}^{100} MC_k ,
\]

where the \( k \)-delay memory capacity is defined as:

\[
MC_k = \max w_k \max \left[ \mathbb{E} \left( u_{t-k} \right) \mathbb{E} \left( y_{t} \right) \right] = \max w_k \frac{\text{cov}(u_{t-k}, y_t)}{\sigma^2(u_t) \sigma^2(y_t)} ,
\]

which is essentially a squared correlation coefficient between the desired signal delayed by \( k \) steps and the reconstruction by the \( k \)-th output node of the reservoir network. \( d(W_k) \) is the determination coefficient for the \( k \)-th readout weight matrix, which is a measure of the variance of one signal caused by another.

When solving an engineering task using an RC model, one needs first to determine the neuron type in the reservoir. To handle analog signals, reservoirs can be built from linear nodes without an activation function [10], which can provide very fast computation but less computational power. Another option is to use the standard sigmoidal neurons, such as a tanh function [11], or others that have a form of internal memory, which are also called as analog integrator neurons [57]. To analyze how different neuron types affect the performance of the RC models, RC models with a fixed, randomly determined structure using four different neuron types, i.e., linear node [10], tanh node [11], analog integrator node [57], and leaky integrate-and-fire (LIF) spiking node are implemented and compared. The comparison results on the MC task are shown in Fig. 6. The result of the GRN-SO-RC model with LIF spiking neurons is also shown in Fig. 6. Here, the spectral radius of the proposed GRN-SO-RC model for the MC benchmark task is \( \rho = 0.93 \).

The following observations can be made from the results in Fig. 6. First, the proposed model has a much larger memory capacity than other models with a small number of neurons in the reservoir. As the number of neurons increases, the memory capacity of the proposed model increases. However, the increase in memory capacity becomes very small when the number of neurons is larger than 100. The memory capacity of all other compared RC models also increases as the number of neurons becomes larger, but remains smaller than that of the proposed model with one exception: The RC model having linear nodes has the highest memory capacity when the number of neurons is larger than 200, which is an experimental confirmation of the theoretical result proved in Jaeger [24]. Second, with plasticity mechanisms being introduced in the RC model with spiking neurons, the memory capacity has been substantially increased.

To show the exact contribution of each component of the proposed learning method for the GRN-SO-RC model, three alternative RC models with the same type of neurons (i.e., linear, tanh, analog integrator, and spiking neurons) are applied to this benchmark problem as well. In all of these models, the reservoir consists of 150 internal neurons. First one is the classical LSM model, where the weights of the reservoir are randomly generated and fixed. More specifically, the parameterization of the classical LSM model is adopted from [41]. The probability of an existing connection between two neurons \( a \) and \( b \) in the reservoir is defined by \( P(a, b) = C \cdot \exp(-D(a, b)/\lambda^2) \), where \( D(a, b) \) is the Euclidean distance between neurons \( a \) and \( b \), \( C = 0.3 \) and \( \lambda = 2 \) control the reservoir connectivity and dynamics, respectively. In the second alternative RC model, the reservoir also has fixed weights, instead of randomly generated, the weights are obtained from the final distribution of the GRN-SO-RC model. The third one is the RC with adapted weights using the BCM rule only, in which three plasticity parameters \((\eta, e, \alpha)\) of BCM are fixed and obtained from the final loop of the training process of the GRN-SO-RC model. The comparison result in memory capacity for the MC benchmark is listed in Table I. For the MC task, the higher of the value of the memory capacity, the better the performance is. From Table I, we can see that the GRN-SO-RC model (with adapted weight using the BCM-GRN) has the highest value, 33.86, outperforms other three methods. The achievable MC in the classical LSM and the RC with fixed weights are very close as the latter can be considered as the special case of the LSM model. The MC value is increased about 10% when the weights of the reservoir are adapted by the BCM model only. Then, when the GRN model is applied to dynamically regulate the plasticity parameters of the BCM model (i.e., GRN-SO-RC model), the system performance is improved significantly (about 29%). From these experimental results, we can say that the reservoir with the self-organizing structure regulated by GRN model can improve the overall system performance dramatically in this benchmark problem.
TABLE I: COMPARATIVE RESULTS IN MEMORY CAPACITY FOR THE MC BENCHMARK

<table>
<thead>
<tr>
<th>Models</th>
<th>MC value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical LSM with fixed weights in reservoir</td>
<td>23.16</td>
</tr>
<tr>
<td>RC with fixed weights in reservoir</td>
<td>23.82</td>
</tr>
<tr>
<td>RC with adapted weights in reservoir (BCM only)</td>
<td>26.22</td>
</tr>
<tr>
<td>GRN-SO-RC</td>
<td>33.86</td>
</tr>
</tbody>
</table>

B. Benchmark 2: NARMA Task

To further evaluate the GRN-SO-RC model, we also conduct the nonlinear auto-regressive moving average (NARMA) task described in [25], [26], which consists of modeling the output of the following tenth-order system:

\[
y(t+1) = 0.3y(t) + 0.05y(t)[\sum_{i=0}^{9}y(t-i)] + 1.5u(t-9)u(t) + 0.1
\]

(22)

where \( u(t) \) is a uniform random signal in [0, 1], which serves as the sole input to the reservoir. The readout neuron is trained to reproduce the signal \( y(t+1) \). The performance is measured using the normalized root mean square error (NRMSE) defined by:

\[
NRMSE = \sqrt{\frac{\sum (\overline{y}_i - \overline{y})(\overline{y}_i - \overline{y})^2}{\sum (\overline{y}_i - \overline{y})(\overline{y}_i - \overline{y})^2}}
\]

(23)

where \( y(t) \) is the desired output and \( \overline{y}(t) \) is the actual output.

Then, to analyze the influence of the neuron types on the performance of the RC models, RC models (with fixed, randomly determined structure) using four different neuron types, i.e., linear node [10], tanh node [11], analog integrator node [57], LIF spiking node, have been applied to the NARMA task and the comparison results in NRMSE are shown in Fig. 7. We also show the result of the GRN-SO-RC model using spiking neurons in NRMSE task in Fig. 7. It can be seen that the GRN-SO-RC has much better performance than the one using the same spiking neuron type. Among other models, only those using the tanh and integrator neurons have slight increases in performance with a larger reservoir. The RC model with spiking neuron performs better than the tanh and int-tanh neurons but worse than linear neurons in NRMSE task. With the same spiking neurons, the performance can be improved dramatically using the GRN-SO-RC model. In addition, the GRN-SO-RC model can achieve impressive performance with a reservoir of a relatively small size.

Fig.7. Performance comparisons of the same RC model (fixed and random structure) with different node types (i.e., linear, tanh, analog integrator, spiking neuron) and the GRN-SO-RC model with spiking neurons in the NARMA task. The smaller the NRMSE, the better performance of the corresponding algorithm is.

Similar to the MC benchmark problem in Section IV.A, a comparative study on the GRN-SO-RC model with other three alternative RC models (the classical LSM model, the RC with fixed weights in reservoir, the RC with adapted weights in reservoir using BCM only) is conducted on the NARMA benchmark. All the parameter settings are the same as we described in Section IV.A except that the stopping criteria for this task is NRMSE = 0.2 and maximum iterations = 300. Table II lists the comparative results in both training and testing errors for the NARMA benchmark. From Table II, similar conclusions can be drawn with the ones in Section IV.A. Similar performance can be achieved by using the classical LSM model and the fixed-weighted RC model which is derived from our trained self-organizing RC model. By using the BCM model with fixed plasticity parameters, the RC model with adapted weights can reduce errors in a reasonable manner (about 7% in both training and testing errors). The system performance can be further improved significantly (about 23% in training error and 29% in testing error) when the BCM parameters are dynamically regulated by the GRN model (i.e., GRN-SO-RC model). Therefore, we can conclude that the GRN regulation on the GRN-SO-RC model plays a critical role in performance improvement.

TABLE II: COMPARATIVE RESULTS IN NRMSE FOR NARMA BENCHMARK

<table>
<thead>
<tr>
<th>Models</th>
<th>Training Error</th>
<th>Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical LSM with fixed weights in reservoir</td>
<td>0.29</td>
<td>0.56</td>
</tr>
<tr>
<td>RC with fixed weights in reservoir</td>
<td>0.28</td>
<td>0.55</td>
</tr>
<tr>
<td>RC with adapted weights in reservoir (BCM only)</td>
<td>0.26</td>
<td>0.51</td>
</tr>
<tr>
<td>GRN-SO-RC</td>
<td>0.20</td>
<td>0.36</td>
</tr>
</tbody>
</table>

To empirically evaluate the learning efficiency of the GRN-SO-RC model, we apply the proposed model to solving the
NARMA benchmark task. As described in the previous sections, the synaptic plasticity of spiking neurons in the BCM model is regulated by the GRN, and CMA-ES is used to evolve the parameters of the GRN model. Fig. 8 illustrates the normalized root mean square error vs. the number of training samples on the NARMA problem. From Fig. 8 we can see that the performance converges gracefully within approximately 20000 samples, which means that the proposed method is computationally efficient during the learning process.

\[
\text{WER} = 100 \times \frac{N_{wr}}{N},
\]

where \(N_{wr}\) denotes the number of incorrectly recognized samples, and \(N\) is the total number of samples presented.

Fig. 8. The performance convergence of the GRN-SO-RC model vs. the number of training samples on the NARMA task.

V. GRN-SO-RC MODEL FOR SPEECH RECOGNITION

To further evaluate the performance of the GRN-SO-RC model, we apply the model to a complex real-world problem: speech recognition. Speech recognition has attracted extensive research interest in the signal processing community over the past several decades [58] due to its wide application background, such as booking systems over the telephone, various forms of aids for the disabled, dictation systems, toys, games, just to name a few.

Feature extraction in speech recognition is a process where the acoustic signal is converted into a sequence of feature vectors. Several auditory perceptual models have been proposed to extract speech-specific features, such as mel-frequency cepstral coefficients (MFCC) [59], [60] and perceptual linear prediction (PLP) coefficients [61] features. In this work, we use a relatively simple and effective feature extraction method, i.e., Lyon’s cochlear model [62], due to its lower computation cost compared to the MFCC and PLP methods. The Lyon’s model is a biologically plausible model of the human inner ear or cochlea that describes the propagation of sound in the inner ear and the conversion of the acoustical energy into neural representations. An important feature of this model is to combine a series of filters to separate the acoustic wave by the frequency and resemble the frequency selectivity of the ear at certain frequencies. Specifically, the Lyon’s model can transform the one-dimensional sound signal into a series of channels, each representing the instantaneous firing probability of a hair cell.

The transformation of speech into feature vectors is followed by a classification process [2], [63], [64]. Several classification approaches have been proposed for speech recognition, where the hidden Markov model (HMM) [64] has widely been used over the past thirty years. The formalism of the HMM is a probability measure that uses a Markov chain to represent the linguistic structure by compressing the time-dependent speech signal into static inputs, resulting in a loss of some temporal information. Since the speech recognition problem requires to handle the temporal variation, a more compact and useful classification model is desirable. The GRN-SO-RC model is a good candidate for this task, which has an inherent capacity to handle temporal information. We apply the GRN-SO-RC model to a subset of TI46 corpus, where the goal is to recognize ten isolated digits (‘zero’ to ‘nine’) spoken by five different female speakers. The isolated digits dataset contains 500 samples and is available from the Linguistic Data Consortium (http://www.ldc.upenn.edu). System performance can be measured by the word error rate (WER), which is defined as the percentage of the fraction of the incorrectly classified words over the total number of words as follows.

\[
\text{WER} = \frac{N_{wr}}{N} \times 100\%.
\]
temporal information processing compared to other neuron types. Furthermore, the use of a GRN regulatory mechanism on the RC model (i.e., GRN-SO-RC model) with the spiking neurons offers more significant improvement. Our GRN-SO-RC model outperforms other RC models with the lowest average word error rate of 2.3%. Moreover, the reservoir size affects the recognition performance of the fixed-structure RC model, while it has little influence on the performance of the GRN-SO-RC model, indicating that the GRN-SO-RC model with is more robust to the reservoir size due to its self-organization capabilities provided by the GRN model. Here, the spectral radius of the GRN-SO-RC model for the speech recognition task is $\rho = 0.95$.

![Fig. 10. Performance comparisons of the same RC model (with a fixed, randomly determined structure) having different node types (i.e., linear, tanh, analog integrator, spiking neuron) and the GRN-SO-RC model with spiking neurons in the speech recognition task.](image-url)

Again, to show the contribution of each component in the proposed self-organizing RC model, we have conducted the comparative study of the three other state-of-the-art RC models described in Section IV.A on the speech recognition task. All RC models use the same RSNNs. Table III lists the results from these models in terms of both training and test errors for the speech recognition task. Since the HMM has been successfully applied to speech recognition, we also list the results from the HMM model in Table III. We can see that the HMM performs better than other three RC models with a testing error of 3.8%, but worse than GRN-SO-RC model. The performance of the classical LSM and the RC with fixed-weight reservoir are similar to each other. The training and test errors have been reduced to 14% and 16%, respectively with the reservoir with its weights adapted using the BCM-based plasticity mechanism. The training and test errors can be further reduced to 25% and 48%, respectively when the parameters and structure of the reservoir are adapted using the BCM that is regulated by the GRN model. These observations are consistent with our observations from the results on the two benchmark problems. From these observations, we can conclude that the application of the GRN model to the tuning the plasticity parameters of BCM model, which further adapt the weights of the reservoir can offer significant performance improvement in the speech recognition task.

**Table III: Comparative Results for Speech Recognition**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Training Error</th>
<th>Testing Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical LSM</td>
<td>2.2%</td>
<td>3.8%</td>
</tr>
<tr>
<td>Classical LSM with fixed weights in reservoir</td>
<td>1.75%</td>
<td>4.8%</td>
</tr>
<tr>
<td>RC with fixed weights in reservoir</td>
<td>1.75%</td>
<td>5.2%</td>
</tr>
<tr>
<td>RC with adapted weights in reservoir (BCM only)</td>
<td>2.0%</td>
<td>4.4%</td>
</tr>
<tr>
<td>GRN-SO-RC</td>
<td>1.5%</td>
<td>2.3%</td>
</tr>
</tbody>
</table>

VI. GRN-SO-RC MODEL FOR HUMAN ACTION RECOGNITION

A. Human Action Recognition

Enabling computers to understand human actions has a potential to revolutionize many areas such as clinical diagnosis, human computer interaction, social robotics, and surveillance systems. Earlier work has identified two main promising strategies for action recognition. The first one uses the tracking methods to represent the actions, such as the tracking of motion trajectories [65], [66], or the tracking of specific shapes [67], [68], like silhouettes and hand shapes. These tracking-based methods process the video frame-by-frame and segment the interested object from the background clutter. The articulated objects (e.g. human body) may undergo various changes in appearance and geometry due to rotations, deformations, and partial occlusions, which gives rise to the difficulty in acquiring reliable trajectories for human action recognition.

The other approach focuses on local interest points that have shown to be an effective method for action classification. A few methods have been proposed recently using visual code books to detect and recognize objects [69], [70], [71], [72], where the local features are quantized to form a “visual codebook”. However, a serious limitation of the “bag of words” representations is that they can be too local, failing to capture adequate spatial or temporal relationships. In the extreme case, the orderless “words” lack cues about motion trajectories, relationships, or the relative layout of objects and actions.

To address this issue, several methods have been proposed to capture the temporal and spatial relationships among the features. These spatial-temporal features are the locations in space and time where sudden changes of movement occur in the video. Dollar et al. [73] proposed space-time interest point detector based on a set of linear filters, and uses these local features with a neural network based classifier for action recognition. A support vector machine classifier has been successfully integrated with spatiotemporal features to recognize the human actions in [74]. In a similar fashion, Laptev and Lindeberg [75] extended the Harris corner detector [76] to 3D, where space-time interest points are extracted based on a significant variation in both spatial and temporal domain. Ke et al. [77] trained a cascade of boosted classifiers to process the vertical and horizontal components of flow in a video sequence using spatiotemporal volumetric features. The above mentioned approaches have been successfully applied to action recognition, where local spatiotemporal features are...
useful for extracting semantic meaning from the video by providing a compact and abstract representation of patterns.

In this work, the GRN-SO-RC model is able to encode the dynamic temporal information into the model, therefore, only the spatial features need to be extracted. Next, we introduce the spatial descriptors to represent a video sequence. Generally, 2D spatial interest points are the locations where the direction of motion changes abruptly. More specifically, these features correspond to the local maxima based on a response function (e.g., Harris corner detector) in the image. These locations are most informative for the recognition of human actions, which are also robust against illumination, clutter, and viewpoint changes.

A variety of methods exist to detect interest points in the spatial domain. One of the most popular approaches is based on the detection of corners. Corners are defined as spatial locations where the local gradient vectors are in orthogonal directions. The gradient vectors are defined as:

\[
L(i,j,\sigma) = I(i,j) * g(i,j,\sigma),
\]

which is the first order derivative of a smoothed image. \(I\) is an input image, where \(I(i,j)\) denotes the pixel value at location \((i,j)\) of the image, and \(g\) is the Gaussian smoothing kernel. \(*\) is the convolution operation in \(x\) and \(y\), and \(\sigma\) controls the spatial scale where corners are detected. The response strength at each point is denoted by \(R(x,y,\sigma) = (L_{\text{cur}}(x,y,\sigma) - L_{\text{pre}}(x,y,\sigma))^2\), where \(L_{\text{cur}}(x,y,\sigma)\) and \(L_{\text{pre}}(x,y,\sigma)\) denote the gradient values of the current frame and the previous frame, respectively. Since human behaviors are a result of a sequence of poses, a single pattern can be represented to be the semantic relationships between these poses. We construct video representations in terms of local spatial features and integrate such representations with the GRN-SO-RC model for human action recognition. Specifically, the original input frames are converted into binary images, where all the pixel values of the extracted spatial features are set as 1 and the rest as 0 in the binary images for each frame. In this manner, the extracted spatial features can be considered as spikes to feed into the proposed RC model frame by frame. Fig. 11 shows one example of extracting the spatial features of “hand-waving” behavior from a video sequence, where the red shaded areas represent the extracted spatial features.

![Fig. 11. An example of extracting the spatial features of a “hand-waving” behavior. The bottom row shows the detected spatial features (represented by red blocks) from original frames in the video.](image)

### B. Dataset Configuration for Experiments

In our experiments we use two different datasets of human actions, the KTH dataset [74] and Weizmann human motion dataset [78]. All results presented in this paper are test results after performing four-fold cross-validation on the total dataset. Therefore, the original video sequences are randomly partitioned into four subsets. To classify a test example, we train the model using all available data instances except for the test samples. The process is then repeated four times until every video is used for validation exactly once.

### C. Human Action Recognition on KTH Dataset

First, we evaluate our model on the KTH human action dataset containing six actions: walking, jogging, running, boxing, hand waving and hand clapping, each of which is performed several times by 25 different participants, for a total of 600 video sequences. Each video sequence exhibits one individual action. There are 4 different scenarios where the video sequences are captured: outdoors, outdoors with scale variation, outdoors with different clothes and indoors, as shown in Fig. 12. There are considerable variations in the performance and duration, and a minor variation in the viewpoint. The backgrounds are relatively static.

![Fig.12. Example images from video sequences in KTH dataset corresponding to different types of actions and scenarios.](image)

The recognition rate is defined as the percentage of correctly recognized actions from the number of all samples, which is \(R = \frac{N_\text{r}}{N} \times 100\%\), where \(R\) is the recognition rate, \(N\) is the number of input actions, and \(N_\text{r}\) is the number of correctly recognized actions.

<table>
<thead>
<tr>
<th>Actions</th>
<th>(N)</th>
<th>(N_\text{r})</th>
<th>(R(%))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>100</td>
<td>89</td>
<td>89</td>
</tr>
<tr>
<td>Jogging</td>
<td>99</td>
<td>84</td>
<td>84.8</td>
</tr>
<tr>
<td>Running</td>
<td>100</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>Hand-waving</td>
<td>100</td>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>Boxing</td>
<td>100</td>
<td>88</td>
<td>88</td>
</tr>
<tr>
<td>Hand-clapping</td>
<td>100</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td><strong>599</strong></td>
<td><strong>528</strong></td>
<td><strong>88.15</strong></td>
</tr>
</tbody>
</table>

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
Among the 599 actions, 528 actions are recognized correctly, resulting in an average recognition accuracy of 88.15%, as listed in Table IV. To figure out why some action sequences are missclassified, a confusion matrix with respect to visual inputs is calculated and listed in Table V. The elements of each row in the confusion matrix represent the misclassification rate. From Table V, most classes are almost perfectly predicted, except for “running” and “jogging”. This is reasonable as many of these actions can even be misjudged by a human being since running sometimes appears very similar to jogging and vice versa.

To demonstrate the advantage of the proposed GRN-SO-RC model, we first performed a set of experiments using the four RC models to be compared on the KTH dataset. The reservoirs are all arranged in 3-D column, containing 32x24x3 spiking neurons. We set classification error = 0.06 or maximum iterations = 500 as the stopping criteria in the training process. Compared to the three benchmarks discussed previously, video behavior recognition is much more complex. The comparative results from the four RC models are listed in Table VI, where all these models use the same spatial features extracted from the video sequences. From Table VI, it can be seen that the proposed GRN-SO-RC model has much better recognition rate than other three compared methods with the same spatial features. In other words, the reservoir adapted by the BCM that is further regulated by the GRN can significantly increase the classification capability of the proposed RC model.

![Image](image-url)

**TABLE V: CONFUSION MATRIX FOR THE KTH DATASET**

<table>
<thead>
<tr>
<th></th>
<th>Walk</th>
<th>Jog</th>
<th>Run</th>
<th>Hand-wave</th>
<th>Box</th>
<th>Hand-clap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walk</td>
<td>89%</td>
<td>7%</td>
<td>4%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Jog</td>
<td>5%</td>
<td>85%</td>
<td>10%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Run</td>
<td>2%</td>
<td>13%</td>
<td>85%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hand-wave</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>92%</td>
<td>0</td>
<td>8%</td>
</tr>
<tr>
<td>Box</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3%</td>
<td>88%</td>
<td>9%</td>
</tr>
<tr>
<td>Hand-clap</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7%</td>
<td>3%</td>
<td>90%</td>
</tr>
</tbody>
</table>

Then, we compare the results of the proposed RC model on the KTH dataset with other state-of-the-art methods proposed in [73], [74], [77], [79], [80], [81], including our previous work [23]. The results are listed in Table VII. Some of these compared methods use more complex and advanced spatiotemporal features, and some [73] used spatial features only with a different feature extraction method. The models proposed in this work and in our previous work [23] only use spatial features. From Table VII, it can be seen that despite its simplicity, the recognition accuracy of our RC model outperforms all others compared models except for the models reported in Le [80] and Jhuang [81]. These two models, however, have more advanced spatiotemporal feature extraction methods. Specifically, Le et al. [80] employed independent subspace analysis (ISA) to extract spatiotemporal features. Similarly, all extracted spatiotemporal features in Jhuang et al. [81] are carefully hand-crafted. In addition, the GRN-SO-RC model is much more powerful compared with the feed forward neural network model in our previous work [23].

**TABLE VII: COMPARISON OF DIFFERENT METHODS ON KTH DATASET**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Features</th>
<th>Classifier</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRN-SO-RC</td>
<td>Spatial features</td>
<td>RC+BCM+GRN</td>
<td>88.15</td>
</tr>
<tr>
<td>Meng et al. [23]</td>
<td>Spatial features</td>
<td>BCM+GRN</td>
<td>84.81</td>
</tr>
<tr>
<td>Dollar et al. [73]</td>
<td>Spatiotemporal features</td>
<td>1-nearest neighbor</td>
<td>81.17</td>
</tr>
<tr>
<td>Schuldt et al. [74]</td>
<td>Spatiotemporal features</td>
<td>SVM</td>
<td>71.83</td>
</tr>
<tr>
<td>Ke et al. [77]</td>
<td>Optical flow</td>
<td>Boosting</td>
<td>62.97</td>
</tr>
<tr>
<td>Antonios et al. [79]</td>
<td>Spatiotemporal features</td>
<td>Gentle boost + RVM</td>
<td>80.80</td>
</tr>
<tr>
<td>Le et al. [80]</td>
<td>Spatiotemporal features</td>
<td>Independent subspace analysis</td>
<td>93.90</td>
</tr>
<tr>
<td>Jhuang et al. [81]</td>
<td>Spatiotemporal features</td>
<td>Feedforward hierarchical NN</td>
<td>91.7</td>
</tr>
</tbody>
</table>

### D. Human Action Recognition on Weizmann Dataset

Next, we evaluate the performance of the GRN-SO-RC model on another widely used video dataset, Weizmann human action dataset. This dataset recorded at the Weizmann institute contains 10 actions: walking, running, walking, skipping, jumping-jack (or shortly jack), jump-forward-on-two-legs (or jumping), jump-in-place-on-two-legs (or pjumping), galloping-sideways (or siding), one-hand waving (or waving1), two-hands waving (or waving2), and bending, each performed by 10 participants. The background and viewpoint are static. Fig. 13 shows some examples of these actions. Due to their low resolution and the smaller number of samples in this dataset, the stopping criterion is defined by a classification error = 0.08 or the maximum number of iterations equals 1000 in the training.

The average recognition rate for this dataset is 77.78%, as listed in Table VIII, which is worse compared with the results on the KTH dataset (88.15% on average). One possible reason is that there are more classes of actions (i.e. 9) and less video samples (i.e. 90) in the Weizmann dataset. Furthermore, more confusion between similar classes may occur, like jump and side, wave1 and wave2, as shown in the confusion matrix in Table IX. However, the confusion degrees are relatively small compared to the correct recognition rate, which means that the proposed approach is effective in distinguishing similar actions.

![Image](image-url)

**Fig. 13.** Action video sequences in Weizmann dataset. (From left to right and top to bottom) walking, running, waving2, pjumping, siding, jack, skipping, waving1, bending, and jumping.
model outperforms the compared state-of-the-art methods using more complex and advanced spatiotemporal features. In addition, since only spatial features need to be extracted from the video, the computational complexity for video processing for the whole system can be considerably reduced.

### E. Robustness Assessment

In this section, we evaluate the robustness of the proposed GRN-SO-RC model in the presence of various noisy and irregular environments for human action recognition. Besides the actions shown in Fig. 13, Weizmann dataset also provides another 20 video sequences where ten of them contain only walking actions under various difficult scenarios. For example, one person is walking on the street with lots of trees in the background (Fig. 14(a)), walking with a dog (Fig. 14(b)), walking indoors and is occluded by desks/boxes (Fig. 14(d)), and walking and is occluded by a pole (Fig. 14(e)). Ten more video sequences were collected, where each one shows the “walking” behavior captured from a different viewpoint (varying between 0 degree and 45 degrees). For example, one person is walking on the street but with a different viewpoint with 45 degrees (as shown in Fig. 14(f)). It is noted that all the backgrounds in these video sequences are static, and only a single moving person is included in each video sequence.

Based on the Weizmann dataset, the experimental results for action recognition of walking under various backgrounds using the proposed GRN-SO-RC model have been conducted. Our model fails to recognize only one sequence, which is between the similar classes “sleepwalking” and “siding”. The average recognition rate is 95%, which demonstrates that the proposed method is fairly robust to changes in backgrounds, occlusions and viewpoints.

![Fig.14. Example behavior patterns with a high degree of irregularities. (a) “Swinging bag”. (b) “Walking with a dog”. (c) “Knees up”. (d) “Occluded legs”. (e) “Occluded by a pole”. (f) “Diagonal walk” (with 45 degrees point of view)]](image)

---

**TABLE VIII: ACTION RECOGNITION RESULTS**

<table>
<thead>
<tr>
<th>Actions</th>
<th>N</th>
<th>Np</th>
<th>R (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>9</td>
<td>8</td>
<td>88.89</td>
</tr>
<tr>
<td>Running</td>
<td>9</td>
<td>7</td>
<td>77.78</td>
</tr>
<tr>
<td>Waving2</td>
<td>9</td>
<td>7</td>
<td>77.78</td>
</tr>
<tr>
<td>Jumping</td>
<td>9</td>
<td>5</td>
<td>55.56</td>
</tr>
<tr>
<td>Sidling</td>
<td>9</td>
<td>8</td>
<td>88.89</td>
</tr>
<tr>
<td>Jack</td>
<td>9</td>
<td>6</td>
<td>66.67</td>
</tr>
<tr>
<td>Skipping</td>
<td>9</td>
<td>7</td>
<td>77.78</td>
</tr>
<tr>
<td>Waving1</td>
<td>9</td>
<td>6</td>
<td>66.67</td>
</tr>
<tr>
<td>Bending</td>
<td>9</td>
<td>8</td>
<td>88.89</td>
</tr>
<tr>
<td>Jumping</td>
<td>9</td>
<td>8</td>
<td>88.89</td>
</tr>
<tr>
<td>Overall</td>
<td>90</td>
<td>70</td>
<td>77.78</td>
</tr>
</tbody>
</table>

**TABLE IX: CONFUSION MATRIX FOR THE WEIZMANN DATASET**

<table>
<thead>
<tr>
<th></th>
<th>Walking</th>
<th>Running</th>
<th>Waving2</th>
<th>Jumping</th>
<th>Sidling</th>
<th>Jack</th>
<th>Skipping</th>
<th>Waving1</th>
<th>Bending</th>
<th>Jumping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walking</td>
<td>89%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>89%</td>
</tr>
<tr>
<td>Running</td>
<td>0</td>
<td>87%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>11%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Waving2</td>
<td>0</td>
<td>0</td>
<td>88%</td>
<td>0</td>
<td>0</td>
<td>11%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Jumping</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>56%</td>
<td>22%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sidling</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>89%</td>
<td>0</td>
<td>11%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Jack</td>
<td>0</td>
<td>0</td>
<td>22%</td>
<td>0</td>
<td>0</td>
<td>67%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Skipping</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Waving1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>22%</td>
<td>22%</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bending</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Jumping</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100%</td>
</tr>
</tbody>
</table>

Similarly, we compare our proposed RC model with other three RC models (i.e., the classical LSM, the RC model with fixed-weighted reservoir and the RC model with adapted-weighted reservoir using BCM with fixed plasticity parameters) on Weizmann dataset. These models use the same spatial features from video sequences. The comparison results listed in Table X confirm our conclusions made from the KTH dataset. The other three methods exhibited similar performance, while the proposed RC model has much better performance. The reservoir with the GRN regulation on the plasticity parameters of the BCM model can significantly improve the classification performance of the RC model in human behavior recognition tasks.

**TABLE X: COMPARISON OF DIFFERENT METHODS ON WEIZMANN DATASET**

<table>
<thead>
<tr>
<th>Methods</th>
<th>Features</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classical LSM with fixed weights in reservoir</td>
<td>Spatial features</td>
<td>58.89</td>
</tr>
<tr>
<td>RC with fixed weights in reservoir</td>
<td>Spatial features</td>
<td>59.45</td>
</tr>
<tr>
<td>RC with adapted weights in reservoir (BCM only)</td>
<td>Spatial features</td>
<td>58.33</td>
</tr>
<tr>
<td>GRN-SO-RC</td>
<td>Spatial features</td>
<td>77.78</td>
</tr>
</tbody>
</table>

In addition, we compare the proposed model with other state-of-the-art approaches reported in [23], [70], [78] and [82]. The comparison results are provided in Table XI. Note that Niebles et al. [70] and Liu et al. [82] both extracted spatial-temporal features for action recognition, which are much more complex and sophisticated features compared with the spatial features we used in this work and in our previous work [23]. As shown in Table XI, even with the spatial features only, the overall recognition rate of the proposed
VII. CONCLUSION AND FUTURE WORK

In this paper we have proposed a reservoir computing model called a GRN-SO-RC model. Spiking neurons are employed to construct the reservoir for the specific tasks explicitly involving temporal information. Inspired by biological findings, we adopted a GRN model to regulate the plasticity parameters of the BCM-based spiking recurrent neural network, resulting in a synaptic and structural self-organization of the reservoir. The performance of the GRN-SO-RC model was first evaluated on two benchmark problems, i.e., the memory capacity task and the NARMA tasks. Simulation results demonstrated that the proposed GRN-SO-RC model shows much better performance, particularly when the size of the reservoir is relatively small. Then the proposed RC model is applied on two real-world problems, namely speech recognition and human action recognition. Compared to three other RC models (with the same type of neurons) having a fixed-weighted reservoir or adapted-weighted reservoir, the GRN-SO-RC model delivers significant performance improvement on two benchmark problems as well as a few real-world a speech recognition problem. In addition, experimental results on both KTH and Weizmann datasets demonstrated that the proposed GRN-SO-RC model outperforms other state-of-the-art algorithms in human behavior recognition.

Activity-dependent neural plasticity modeled in this work plays an important role in neural development and is the neuronal basis of mental development. The fact that only supervised learning is considered in this paper does not imply that our work is not relevant to mental development. For example, recent findings suggest that supervised learning is used in hippocampus [83], [84]. Although the conclusions are based on experiments on two specific data sets, the reservoir adopted in our work is very generic.

The idea of activity-dependent neural development realized by a GRN model will be further investigated. In biology, activity dependent neural plasticity is closely coupled with activity-independent neural development [85] and morphological development [86]. In the future, we plan to integrate the activity-dependent and activity-independent neural plasticity and morphological development governed by gene regulatory networks into one computational model. Although the work in this paper focuses on supervised learning our on-going work is applying the proposed model to incremental learning.

In addition to the reservoir structure, we will also focus on the output distributions since they can lead to sparse coding, which can increase signal-to-noise ratio and improve the detection of coincidences. Furthermore, we will apply the current model to more complex recognition scenarios, such as multi-object-multi-action tasks for human action recognitions.

ACKNOWLEDGEMENTS

The authors are grateful to the anonymous reviewers and the editors for their insightful comments on an early version of the paper, which have greatly improved the quality of the paper.

REFERENCES