Highlights

Effect of recurrent infomax on the information processing capability of input-driven recurrent neural networks

Takuma Tanaka, Kohei Nakajima, Toshio Aoyagi

- The central nervous system is modeled as a recurrent network.
- Reservoir computing uses recurrent neural networks as a computational resource.
- We optimized the information processing capability of a recurrent neural network.
- A delay-line structure emerged in the network as a result of optimization.

Effect of recurrent infomax on the information processing capability of input-driven recurrent neural networks

Takuma Tanaka[a,*], Kohei Nakajima[b], Toshio Aoyagi[c]

[a] *Graduate School of Data Science, Shiga University, 1-1-1 Banba, Hikone, Shiga 522-8522, Japan*
[b] *Graduate School of Information Science and Technology, University of Tokyo, Tokyo 113-8656, Japan*
[c] *Graduate School of Informatics, Kyoto University, Yoshida Honmachi, Sakyo-ku, Kyoto 606-8501, Japan*

Abstract

Reservoir computing is a framework for exploiting the inherent transient dynamics of recurrent neural networks (RNNs) as a computational resource. On the basis of this framework, much research has been conducted to evaluate the relationship between the dynamics of RNNs and the RNNs' information processing capability. In this study, we present a detailed analysis of the information processing capability of an RNN optimized by recurrent infomax (RI), an unsupervised learning method that maximizes the mutual information of RNNs by adjusting the connection weights of the network. The results indicate that RI leads to the emergence of a delay-line structure and that the network optimized by the RI possesses a superior short-term memory, which is the ability to store the temporal information of the input stream in its transient dynamics.

*Corresponding author
Email address: `takuma-tanaka@biwako.shiga-u.ac.jp` (Takuma Tanaka)

Keywords: Reservoir computing, Recurrent infomax, Unsupervised learning, Memory

1. Introduction

To elucidate the nature of the central nervous system (CNS), we need to investigate its information processing both experimentally and theoretically. The information processing in the CNS is supported by the dynamics of recurrent networks, whose malfunctioning leads to pathological states such as epilepsy. Because extremely rich dynamics exhibited by recurrent networks make their description complicated, theoretical frameworks capable of giving a clear picture of the information processing in the CNS are needed. A framework called reservoir computing (RC) has been proposed as a brain-inspired information processing model (Maass et al., 2002; Jaeger and Haas, 2004; Lukoševičius and Jaeger, 2009). One of the most notable features of RC is that it exploits the inherent transient dynamics of recurrent neural networks (RNNs) as a computational resource in input-driven RNNs. Owing to this feature, RC has been used as a theoretical framework for examining the information processing mechanism of the central nervous system (Maass et al., 2002; Buonomano and Maass, 2009; Rabinovich et al., 2008). In addition, it has been used to emulate human behavior, such as motor activity (Sussillo and Abbott, 2009; Laje and Buonomano, 2013).

As illustrated in Fig. 1(A), the architecture of RC generally consists of three layers: an input layer, a reservoir layer implemented by an RNN, and an output layer. The framework has three important properties. The first is the ability to embed the previous input information into the transient dynamics of RNNs, which enables real-time information processing of input streams; this is called the short-term memory property. The second property is nonlinearity, which is the ability to emulate nonlinear information processing. The third property is that supervised learning of the reservoir layer is not required if the dynamics of the reservoir layer are sufficient. Only the connection weights from the RNNs to the output, called the readout weights, must be adjusted to learn the required dynamical systems. Although RC is a simple framework, it has been demonstrated to perform well for a large number of machine learning tasks (Antonelo et al., 2008; Jalalvand et al., 2015; Salmen and Ploger, 2005; Skowronski and Harris, 2007; Jaeger, 2003).

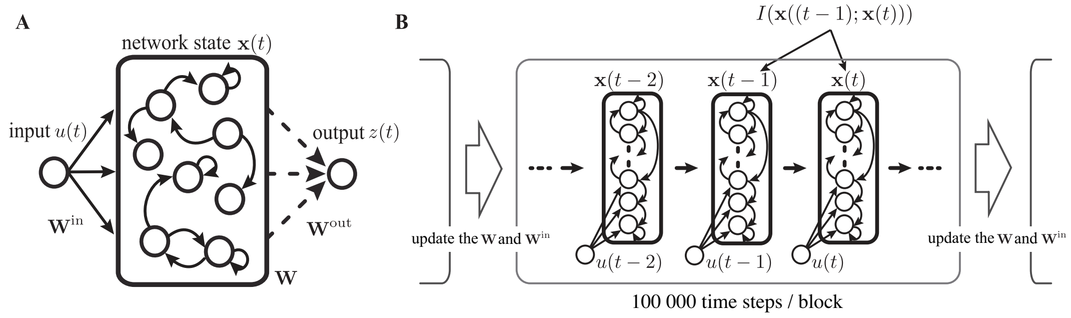
A
network state $\mathbf{x}(t)$
input $u(t)$
output $z(t)$
$\mathbf{W}^{\text{in}}$
$\mathbf{W}^{\text{out}}$
$\mathbf{W}$
B
$I(\mathbf{x}((t-1);\mathbf{x}(t)))$
$\mathbf{x}(t-2)$
$\mathbf{x}(t-1)$
$\mathbf{x}(t)$
update the W and W$^{\text{in}}$
$u(t-2)$
$u(t-1)$
$u(t)$
update the W and W$^{\text{in}}$
100 000 time steps / block

Figure 1: (A) The architecture of RC. The solid and dashed lines represent the connections optimized in this study using RI (task-independent optimization) and the least squares method (task-dependent optimization), respectively. (B) RI method. The connection weights are updated between two consecutive blocks. Each block consists of 100 000 time steps. The first 50 000 time steps are for h_i to converge to a steady state and the final 50 000 time steps are used to calculate mutual information.

The information processing capability of an RC system depends on the inherent dynamics of the reservoir layer. Consequently, much research has been conducted on the information processing capabilities of various dynamical systems using the systems as the reservoir layer. Several studies have investigated the information processing capabilities of physical systems, such as the surface of water (Fernando and Sojakka, 2003), electronic, optoelectronic, and photonic systems (Larger et al., 2012; Appeltant et al., 2011; Woods and Naughton, 2012), ensemble quantum systems (Fujii and Nakajima, 2017), neuromorphic chips and devices (Stieg et al., 2012; Torrejon et al., 2017; Furuta et al., 2018; Tsunegi et al., 2018), and the mechanical bodies of soft robots (Nakajima et al., 2013, 2014, 2015, 2018). These studies exploit the dynamics of physical systems as reservoirs.

Although the inherent dynamics of systems can be exploited as reservoirs, the dynamics of the reservoirs are empirically optimized using the control parameters prior to task-dependent optimization (i.e., optimization of the readout weights). This signifies that RC requires task-independent and task-dependent optimization. One of the best known task-independent optimization approaches involves adjusting a dynamical system to be at the edge of chaos (or the edge of stability), which is the transition point from ordered to chaotic dynamics. Such systems are reported to exhibit superior information processing capabilities (Bertschinger and Natschläger, 2004; Toyoizumi and Abbott, 2011). However, it is not optimal for some tasks (Yildiz et al., 2012; Manjunath and Jaeger, 2013). As a result, various methods should be available for task-independent optimization.

In this study, we examine whether the recurrent infomax (RI) method (Tanaka et al., 2008) can be used for task-independent optimization of a reservoir layer. RI was developed as an extension of feedforward infomax, which was originally proposed by Linsker (Linsker, 1988). It provides an unsupervised learning technique for maximizing information retention in an RNN, which is quantified by mutual information. The RNNs optimized by RI exhibit the dynamical characteristics of neural activities such as cell-assembly-like and synfire-chain-like spontaneous activities as well as critical neuronal avalanches, which are a manifestation of the edge of chaos, in the CNS (Tanaka et al., 2008). Because an RNN optimized by RI replicates the properties of the CNS, it is expected to exhibit improved information processing capabilities. However, whether RI improves the information processing capabilities of a reservoir layer has not yet been investigated, and any improvements should thus be examined and quantified. In this study, we first optimize RNNs using RI and evaluate their information processing capabilities using benchmark tasks.

2. Model

We consider an input-driven network consisting of N neurons where the state of each neuron $x_i(t) \in \{0, 1\}$ $(i = 1, ..., N)$ is updated synchronously and stochastically at discrete time steps. Simulations were performed with $N = 50$ neurons unless otherwise stated. The firing probability of neuron i is determined by its interaction with neuron j $(j = 1, ..., N)$, which is connected with weight W_{ij} and by its interaction with input $u(t) \in \{0, 1\}$, which is connected with weight W_i^{in}, as fololws:

$$p(x_i(t+1) = 1) = \frac{p_{\max}}{1 + \exp(-U_i(t))}, \tag{1}$$

$$U_i(t) = \sum_{j=1}^{N} W_{ij}(x_j(t) - \bar{p}_j) + W_i^{\text{in}}(u(t) - \bar{p}_0) - h_i(t), \tag{2}$$

where $p_{\max}$ is the maximal firing probability, $\bar{p}_0$ is the mean input firing rate, $\bar{p}_j$ is the mean firing rate of neuron j, and $h_i(t)$ is the bias of neuron i. The firing frequency of each neuron is indicated by the mean firing rate. For example, neuron i is activated once in 10 time steps on average for $\bar{p}_i = 0.1$. The maximal firing probability, $p_{\max}$, indicates the firing reliability in response to the input. In this study, the maximal firing probability and the mean firing rate are fixed at $p_{\max} = 0.8$ and $\bar{p}_i = 0.1$, respectively. Input $u(t)$ is randomly sampled from $\{0, 1\}$ with an expected value of $\bar{p}_0 = 0.5$. The bias $h_i(t)$ is used to fix the average firing probability $\bar{p}_i$ and is updated at each time step by

$$h_i(t+1) = h_i(t) + \epsilon(x_i(t+1) - \bar{p}_i), \tag{3}$$

where ϵ is the learning rate, which remains constant at 0.01. The initial conditions of the network (W_{ij} and W_i^{in}) are sampled from the Gaussian distribution with $\mu = 0$ and $\sigma^2 = 0.01$. A greater value of σ^2 deteriorates the performance of the approximated weight update rule.

3. Recurrent Infomax

In this section, we briefly describe the RI algorithm (Tanaka et al., 2008). In this study, RI is applied to the connection weights of the RNN and the connection weights from the input to the RNN. The connection weights are represented by solid lines in Fig. 1(A). As illustrated in Fig. 1 (B), the connection weights are updated at the end of each block, which consists of 100 000 time steps. The first 50 000 time steps are the washout phase for h_i to converge to a steady state. The final 50 000 time steps are the accumulation phase, the firing activity in which is used to update the connection weights (Table 1, training block). The connection weights at the $b{+}1^{\text{th}}$ block $\mathbf{W}(b{+}1)$ are calculated using the steepest gradient method:

$$W_{ij}(b+1) = W_{ij}(b) + \eta \frac{\partial I(b)}{\partial W_{ij}}, \tag{4}$$

where η is the learning rate, which is constant at $\eta = 0.2$ in this study, and $I(b)$ is the approximated mutual information at the b^{th} block. A greater learning rate η tends to destabilize the network dynamics. The input connection weights W_i^{in} and input $u(t)$ are represented by W_{i0} and $x_0(t)$, respectively. The gradient of the mutual information with respect to W_{ij} is formulated using several approximations (Tanaka et al., 2008). The mutual information between two successive states is defined by

$$I = \sum_{\mathbf{u},\mathbf{v}\in\{0,1\}^N} p(\mathbf{u},\ \mathbf{v}) \log \frac{p(\mathbf{v}|\mathbf{u})}{p(\mathbf{v})}, \tag{5}$$

where $p(\mathbf{v})$ is the probability that the firing activity of neurons at a time step is $\mathbf{v}$ and $p(\mathbf{v}|\mathbf{u})$ is the conditional probability of the firing activity $\mathbf{v}$ given that the previous state is $\mathbf{u}$. Equation 5 is the summation of 2^{2N} terms, which is intractable even for the network with several neurons. Thus, we approximate it with the mutual information of the Gaussian distribution with the same mean and covariance matrix as the neurons. The approximation of the mutual information at the b^{th} block is given by

$$I(b) = \log|\mathbf{C}| - \frac{1}{2}\log|\mathbf{D}|, \tag{6}$$

where

$$\mathbf{C} = \begin{pmatrix} E_{00} & \dots & E_{0N} \\ \vdots & \ddots & \vdots \\ E_{N0} & \dots & E_{NN} \end{pmatrix}, \tag{7}$$

$$\mathbf{D} = \begin{pmatrix} E_{00} & \dots & E_{0N} & E_{0\widehat{0}} & \dots & E_{0\widehat{N}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ E_{00} & \dots & E_{0N} & E_{N\widehat{0}} & \dots & E_{N\widehat{N}} \\ E_{\widehat{0}0} & \dots & E_{\widehat{0}N} & E_{\widehat{0}\widehat{0}} & \dots & E_{\widehat{0}\widehat{N}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ E_{\widehat{N}0} & \dots & E_{\widehat{N}N} & E_{\widehat{N}\widehat{0}} & \dots & E_{\widehat{N}\widehat{N}} \end{pmatrix}, \tag{8}$$

$$E_{ij} = \frac{1}{T}\sum [x_i(t) - \bar{p}_i][x_j(t) - \bar{p}_j], \tag{9a}$$
$$E_{\widehat{\imath}j} = \frac{1}{T}\sum [x_i(t+1) - \bar{p}_i][x_j(t) - \bar{p}_j], \tag{9b}$$
$$E_{i\widehat{\jmath}} = \frac{1}{T}\sum [x_i(t) - \bar{p}_i][x_j(t+1) - \bar{p}_j], \tag{9c}$$
$$E_{\widehat{\imath}\widehat{\jmath}} = \frac{1}{T}\sum [x_i(t+1) - \bar{p}_i][x_j(t+1) - \bar{p}_j]. \tag{9d}$$

The summation is taken over all time steps of the accumulation phase of the b^{th} block. The gradient of mutual information with respect to the connection weights is approximated by

$$\begin{aligned} \frac{\partial I(b)}{\partial W_{ij}} \approx & \frac{1}{2}\sum (1-\delta_{ij})(E_{\widehat{\imath}\widehat{k}}E_{\widehat{\jmath}l} + E_{\widehat{\imath}l}E_{\widehat{\jmath}\widehat{k}})[2(\mathbf{C}^{-1})_{ji} - (\mathbf{D}^{-1})_{ji} - (\mathbf{D}^{-1})_{j+Ni+N}] \\ & - \frac{1}{2}[(1-2\bar{p}_k)(1-2\bar{p}_l)E_{\widehat{k}l} + \bar{p}_k\bar{p}_l(1-\bar{p}_k)(1-\bar{p}_l)][(\mathbf{D}^{-1})_{lk+N} + (\mathbf{D}^{-1})_{k+Nl}]. \end{aligned} \tag{10}$$

Typical instances of the firing activity of a network optimized by RI at the 0th and 1500th block are depicted in Figs. 2(A) and (B). The neurons in the network optimized by RI tend to fire simultaneously. The dynamics of the network are stochastic and the reliability of the activation in response to the input is low. If the neurons in the network fired independently, it would be difficult to retain past information by their firing activity. Therefore, an appropriate strategy is to retain the least amount of past information by firing simultaneously. Similar dynamics has been reported in (Tanaka et al., 2008). As depicted in Fig. 2(C), mutual information is increased by RI and is almost saturated at the 1500th block. Therefore, RI is terminated at the 1500th block in the following simulations.

Table 1: Number of time steps in each block.

	Training block	Benchmark task block
Washout phase	50 000 time steps	50 000 time steps
Accumulation phase	50 000 time steps	–
Learning phase	–	1500 time steps
Testing phase	–	1500 time steps

4. Information processing capability

Two benchmark tasks are used to evaluate the information processing capability of a network optimized by using RI. Each benchmark task comprises a block consisting of a washout phase (50 000 time steps), learning phase (1500 time steps), and testing phase (1500 time steps) (Table 1, benchmark task block). The washout phase is used to eliminate the influence of the initial state of the network and to converge the bias $h_i(t)$ to a steady-state value. The learning phase is used to train readout weights $\mathbf{W}_\tau^{\text{out}}$, and the testing phase is used to evaluate the performance of each task.

The first task is the short-term memory task, which has been used extensively as a benchmark task in the RC framework (Bertschinger and Natschläger, 2004; Jaeger, 2002). In general, the expressive power of recurrent neural networks can be characterized by the length of the previous input sequence that the network can store and by the type of function that the network can express by taking these stored inputs. Accordingly, it is important to analyze these two properties if we want to understand the computational capability of a recurrent neural network, and in this case, the reservoir. This first task is devoted to evaluate the former property, the so-called short-term memory, which requires the system to store recent input sequence into the current state. This task evaluates the extent of decoding the previous input information from the network state with memoryless readout. The performance of this task is measured by the memory capacity (MC), which is the summation of the τ-delay memory functions [MF_τ ($\tau = 1, ..., 50$)] defined by

$$\text{MC} = \sum_{\tau=1}^{50} \text{MF}_\tau. \tag{11}$$

The memory function MF_τ is the determination coefficient between $u(t-\tau)$ and $z_\tau(t) = \mathbf{W}_\tau^{\text{out}}\mathbf{x}(t)$, defined by

$$\text{MF}_\tau = \max_{\mathbf{W}_\tau^{\text{out}}} \frac{\text{cov}^2(z_\tau(t), u(t-\tau))}{\sigma^2(z_\tau(t))\sigma^2(u(t-\tau))}. \tag{12}$$

The readout weights, $\mathbf{W}_\tau^{\text{out}}$, are trained in accordance with $\mathbf{W}_\tau^{\text{out}} = (\mathbf{X}^{\text{T}}\mathbf{X})^{-1}\mathbf{X}^{\text{T}}\mathbf{y}$, where $\mathbf{y}$ is the vector of τ-time-step past inputs and $\mathbf{X}$ is the matrix of network states during the learning phase.

The second benchmark task is the n-bit (n = 2, 3) Boolean emulation task, which evaluates whether the network can process a Boolean operation using the past n-bit inputs. An example is depicted in Table 2. Each rule of the Boolean emulation tasks requires different types of information processing. Therefore, the execution of each rule can provide useful insights into the properties of information processing of RNNs optimized by RI. Furthermore, the target Boolean operators contain not only linear functions but also nonlinear functions (such as XOR), which suggests that by using this task, we can evaluate both the system's capability to store a recent input stream and its expressive power to approximate nonlinear functions. The total number of rules is 2^{n^2}, and all rules, except for two, are applied. The two exceptions are rules for which the output f_τ is 0 or 1 regardless of the input. Thus, the number of rules R for each n-bit Boolean emulation task is $2^{n^2} - 2$.

Table 2: Example of a rule table for the 2-bit Boolean emulation task.

$u(t-\tau)$	$u(t-\tau-1)$	$f_\tau(\mathrm{t})$
0	0	1
0	1	0
1	0	1
1	1	0

The performance of this task is measured by n-bit Boolean capacity (n-bit BC), which is the summation of the n-bit Boolean function (n-bit $\mathrm{BF}_\tau^\mathrm{r}$) at each rule r ($r = 1, \ldots, R$) and time delay τ ($\tau = 1, ..., 50$). The n-bit

Boolean capacity is defined as

$$n\text{-bit BC} = \frac{1}{R}\sum_{r=1}^{R}\sum_{\tau=1}^{50}\mathrm{BF}_{\tau}^{r}, \tag{13}$$

where

$$n\text{-bit BF}_{\tau}^{r} = \max_{W_{\tau}^{r,\mathrm{out}}}\frac{\mathrm{cov}^2(z_r(n), f_{\tau}^{r}(n))}{\sigma^2(z_r(n))\sigma^2(f_{\tau}^{r}(n))}. \tag{14}$$

The results of the benchmark tasks are presented in Fig. 2(D). We suspended the RI update and performed the tasks while evaluating the networks' information processing capabilities with RI. The process of RI update was resumed after evaluating the performance of the benchmark tasks. Although both MC and n-bit BC ($n = 2$, 3) improved by RI, they exhibited a peak at the 1000th block; this is in sharp contrast to the mutual information, which monotonically increased and saturated at approximately the 1500th block.

To investigate why neither MC nor n-bit BC ($n = 2$, 3) monotonically increased with respect to block number, we visualized the network structure. Typical examples of network structures are depicted in Figs. 3(A)–(F); here, connections with the 50 largest absolute values are presented. Figures 3(A) and (B) depict a randomly sampled initial RNN. Most of the visualized connections are from within the RNN, and several are from the input to the RNN at 1000th block [Figs. 3(C) and (D)]. At the 1500th block, all visualized connections are from within the RNN, and the RNN is almost disconnected from the input [Figs. 3(E) and (F)]. In addition, we calculated the mean of the absolute value of the connection weights ($\overline{W}$) within the RNN and from the input to the RNN, as illustrated in Fig. 3(G). The former was calculated using the 50 largest absolute values in $\mathbf{W}$, and the latter was calculated using

$\mathbf{W}^{\text{in}}$. At later blocks, the value of $\overline{W}$ in the RNN becomes much larger than the value of $\overline{W}$ from the input to the RNN. This result suggests that the input information is not stored in the network, but rather, only information about the activity of neurons in the network is stored when the network is optimized by RI for a long period of time. Consequently, the performances of both MC and n-bit BC ($n = 2, 3$) deteriorates.

5. Introduction of input multiplicity

5.1. Recurrent infomax and information processing capability

In Section 4, we demonstrate that the information processing capability of a network increases at the beginning of RI but peaks at the 1000th block because the connection weights in the RNN become stronger than the input connection weights. Thus, the input information may not be preserved in the network. To address this problem, we attempted to increase the number of input neurons with common input information. However, using K input neurons is virtually the same as using one input neuron and replacing the connection weights W_{i0} with KW_{i0}. Because the initial values of the connection weights are small, this is equivalent to introducing a weight coefficient (K), also refered to as input multiplicity, into the update rule of the connection weights from the input to the RNN as follows:

$$W_{i0}(b+1) = W_{i0}(b) + K\eta \frac{\partial I(b)}{\partial W_{i0}}, \tag{15}$$

where $K \in \mathbb{Z}$ and $2 \leq K \leq 35$. This is also equivalent to accelerating the growth of the connection weights from the input neuron. We optimized the input-driven RNN by RI using the above formula to update the connection weights from the input to RNN. Subsequently, we executed the three tasks described in Section 4. With the exception of the weight coefficient K, the parameter values and benchmark tasks of RI are identical to those described in Section 4.

Here, we provide an overview of the results of the mutual information and benchmark tasks. As depicted in Fig. 4(A), a large K value decreases the mutual information at the learning phase; however, RI increases the mutual information for all values of K. Furthermore, for all values of K, with the exception of $K = 1$, MC also increases throughout the optimization with RI and is maximized at approximately $K = 7$ [Fig. 4(B)]. The n-bit BC ($n = 2$, 3) exhibits a similar tendency to that of MC [Fig. 4(C)].

Figures 5(A)–(D) present typical network structures at $K = 5$ and 30. At $K = 5$, both strong connection weights from the input to the RNN and strong connection weights in the RNN coexist. Because the postsynaptic input neurons become the presynaptic neurons of other neurons, the input generates a delay-line structure. However, at $K = 30$ the strongest connection weights are from the input to the RNN. The mean absolute value of the connection weights from the input as a function of the block increases more rapidly than that of the RNN, as depicted in Figs. 5(E) and (F); this result is the opposite of that observed in Section 4. These network structures suggest that, for a large K value, the input information at the final time step may be dominantly stored in the network, and the majority of the previous input information may be lost.

We confirmed this speculation quantitatively by investigating the mutual information between the input at the previous time step and each neuron in the RNN, $I(x_i(t); u(t-1))$ $(i = 1, ..., N)$, as plotted in Fig. 6(A). The results indicate that, for large values of K, a network optimized by RI contains information on the input provided at the previous time step. The performances of the short-term memory task and the n-bit Boolean emulation task ($n = 2$, 3) as a function of the time delay are plotted in Figs. 6(B) and (C). The performance of the network at $K = 30$ at $\tau \leq 2$ is higher than the performance at $K = 5$. As the time delay τ increases, the performance of the network at $K = 30$ is inferior to the performance at $K = 5$. These results are consistent with the observation that a network optimized by RI for a large K value can only store recent input information. As discussed in the previous sections, learning with RI using a small K value optimizes the network such that the previous input information is not dominantly stored, but rather, information about the past state of the network itself is stored.

In addition, we investigated information processing capabilities for various network sizes ($N = 25$, 75, and 100); the results are presented in Fig. 7. The results indicate that a larger network size leads to a larger optimal K. In addition, a large input multiplicity value results in the increase in an number of neurons that have the previous input information. For networks with a large number of neurons, even if the number of neurons containing the previous input information increases, other neurons can store this information.

5.2. Comparison of information processing capability for linear and nonlinear rules

To investigate the information processing properties of a network optimized by RI, we classified the rules of the Boolean emulation tasks into two classes: linear and nonlinear. The classification was based on whether the rule was linearly separable (Chua et al., 2002). Figure 8 illustrates the performance of each rule sorted by score, BF^{r}_{τ}. The results demonstrate that, for the majority of rules, RI improved performance. In addition, the networks exhibit higher performance in executing linear rules; this suggests that the networks optimized by RI performed linear information processing better than nonlinear information processing. Rules 1 and 2 were executed more efficiently than the other linear rules because the difficulty of executing a rule is dependent on both linear separability and short-term memory. These two rules are identical to the short-term memory task which only requires the information of $u(t-\tau)$, not $u(t-\tau-1)$. Therefore, these rules are simpler to execute than the other linear rules.

The results of the 3-bit Boolean emulation task presented in Fig. 8(B) indicate that, in general, linear rules are executed more efficiently than nonlinear rules. However, some nonlinear rules are executed more efficiently than linear rules. As a result, the difference between the two types of rules observed in the 2-bit Boolean emulation task is unclear as some rules require information on the input $u(t-\tau-2)$. In addition, the performance of each rule may be influenced not only by nonlinearity but also by short-term memory in the 3-bit Boolean emulation task.

6. Discussions

In this study, we used RI to optimize an input-driven RNN and evaluated the information processing capability of the network using short-term memory and Boolean emulation tasks. Although naive RI did not lead to improvements in information processing capability, RI with input multiplicity improved MC and n-bit BC ($n = 2, 3$) because the input connection weights increase preferentially, and the input information is stored in the network. An appropriate input multiplicity optimizes a network for storing the information about a past input, and the network partially acquires a delay-line structure, which is one of the preferred structures for achieving optimal short-term memory (Jaeger, 2002; Ganguli et al., 2008; Rodan and Tino, 2010). However, our model is stochastic, and a complete delay-line network may not be suitable for short-term memory. In a delay-line network, the optimal short-term memory is based on the assumption that the information of presynaptic neurons is conveyed to postsynaptic neurons without any loss. In addition, the retention of the input information in the delay-line network is vulnerable to the unreliable firing of neurons.

The information processing property of the network optimized by RI are demonstrated by the performance for each rule in the 2-bit Boolean emulation task. Although RI improves the performance for the majority of the rules, the degree of improvement greatly depends on the linearity of the rules. Linear rules are processed more efficiently than nonlinear rules, which include exclusive OR operations (i.e., $u(t - \tau) \otimes u(t - \tau - 1)$). The tradeoff between short-term memory and nonlinearity can be observed in several dynamical systems such as logistic maps, diffusion equations and echo state network (Dambre et al., 2012). Although no theoretical proof regarding this tradeoff exists, it is possible that RI optimizes a network under this tradeoff constraint. Thus, in a network that is optimized by RI, the short-term memory may be superior to the nonlinearity. However, the dependency of the information processing capability on input multiplicity and the number of neurons should be examined more thoroughly in the future. In addition, the properties of the critical dynamics underlying the information processing of the present model should be investigated.

As mentioned in Section 1, one of the most widely used task-independent optimizations involves adjusting a dynamical system to be at the edge of chaos, that is, setting the Lyapunov exponent of the reservoir close to 1. However, recent studies (Yildiz et al., 2012; Manjunath and Jaeger, 2013) have demonstrated that input intensity affects the spectral radius of the reservoir. This implies that a reservoir optimized without input intensity information may not be optimal for a specific input intensity. RI is a promising method for task-independent optimization because it can adapt to input intensity and the temporal correlation of the input. In our study, the network structure changed dramatically depending on input multiplicity, which can be regarded as input intensity. Optimization by RI is thus task independent but input dependent.

The main result of our study is that input multiplicity is required to prevent the reservoir layer from losing the connection from the input neuron when using RI. Introducing input multiplicity is scaling the learning rate of the connection weights from input neurons by K and leaving other learning rates unchanged. The behavior of networks with different parameter setting should be examined in the future. The present result can also be applied to RC, which exploits an *in vitro* cortical network. The information processing capabilities of RC using *in vitro* cortical networks have been reported in (Dockendorf et al., 2009; Dranias et al., 2013; Ju et al., 2015; Johnson et al., 2010; Goel and Buonomano, 2016). Given the findings in Tanaka et al. (2008)

that *in vivo* cortical networks are optimized by RI, it is possible that the *in vitro* cortical networks used in RC may also be optimized by RI. In this case, the input intensity must sufficiently be high to encode input information into the network.

7. Acknowledgments

We thank Mr. Hisashi Iwade for his assistance in the numerical simulations of the model. This work was supported by MEXT KAKENHI Grant Number 15H05877 and JSPS KAKENHI Grant Numbers 16KT0019, 18H05472, and 19K12184. This work is partially based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

Antonelo, E.A., Schrauwen, B., Stroobandt, D., 2008. Event detection and localization for small mobile robots using reservoir computing. Neural Networks 21, 862–871. doi:10.1016/j.neunet.2008.06.010.

Appeltant, L., Soriano, M.C., Van der Sande, G., Danckaert, J., Massar, S., Dambre, J., Schrauwen, B., Mirasso, C.R., Fischer, I., 2011. Information processing using a single dynamical node as complex system. Nature Communications 2, 468.

Bertschinger, N., Natschläger, T., 2004. Real-time computation at the edge of chaos in recurrent neural networks. Neural Computation 16, 1413–1436.

Buonomano, D.V., Maass, W., 2009. State-dependent computations: spatiotemporal processing in cortical networks. Nature Reviews Neuroscience 10, 113–125. doi:10.1038/nrn2558.

Chua, L.O., Yoon, S., Dogaru, R., 2002. A Nonlinear Dynamics Perspective of Wolfram's New Kind of Science Part I: Threshold of Complexity. International Journal of Bifurcation and Chaos 12, 2655–2766. doi:10.1142/S0218127402006333.

Dambre, J., Verstraeten, D., Schrauwen, B., Massar, S., 2012. Information Processing Capacity of Dynamical Systems. Scientific Reports 2, 514. URL: `http://www.nature.com/doifinder/10.1038/srep00514`, doi:10.1038/srep00514.

Dockendorf, K.P., Park, I., He, P., Príncipe, J.C., DeMarse, T.B., 2009. Liquid state machines and cultured cortical networks: The separation property. Biosystems 95, 90–97.

Dranias, M.R., Ju, H., Rajaram, E., VanDongen, A.M., 2013. Short-term memory in networks of dissociated cortical neurons. Journal of Neuroscience 33, 1940–1953.

Fernando, C., Sojakka, S., 2003. Pattern recognition in a bucket, in: European Conference on Artificial Life, Springer. pp. 588–597.

Fujii, K., Nakajima, K., 2017. Harnessing disordered-ensemble quantum dynamics for machine learning. Physical Review Applied 8, 024030.

Furuta, T., Fujii, K., Nakajima, K., Tsunegi, S., Kubota, H., Suzuki, Y., Miwa, S., 2018. Macromagnetic simulation for reservoir computing utilizing spin dynamics in magnetic tunnel junctions. Phys. Rev. Applied 10, 034063.

Ganguli, S., Huh, D., Sompolinsky, H., 2008. Memory traces in dynamical systems. Proceedings of the National Academy of Sciences USA 105, 18970–18975. doi:10.1073/pnas.0804451105.

Goel, A., Buonomano, D.V., 2016. Temporal interval learning in cortical cultures is encoded in intrinsic network dynamics. Neuron 91, 320–327.

Jaeger, H., 2002. Short term memory in echo state networks. GMD Report 152 , 60.

Jaeger, H., 2003. Adaptive nonlinear system identification with echo state networks, in: Advances in neural information processing systems, pp. 609–616.

Jaeger, H., Haas, H., 2004. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. Science 304, 78–80. doi:10.1126/science.1091277.

Jalalvand, A., Wallendael, G.V., Walle, R.V.D., 2015. Real-time reservoir computing network-based systems for detection tasks on visual contents, in: 2015 7th International Conference on Computational Intelligence, Communication Systems and Networks, pp. 146–151. doi:10.1109/CICSyN.2015.35.

Johnson, H.A., Goel, A., Buonomano, D.V., 2010. Neural dynamics of in vitro cortical networks reflects experienced temporal patterns. Nature Neuroscience 13, 917.

Ju, H., Dranias, M.R., Banumurthy, G., VanDongen, A.M., 2015. Spatiotemporal memory is an intrinsic property of networks of dissociated cortical neurons. Journal of Neuroscience 35, 4040–4051.

Laje, R., Buonomano, D.V., 2013. Robust timing and motor patterns by taming chaos in recurrent neural networks. Nature Neuroscience 16, 925.

Larger, L., Soriano, M.C., Brunner, D., Appeltant, L., Gutierrez, J.M., Pesquera, L., Mirasso, C.R., Fischer, I., 2012. Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing. Optics express 20, 3241–9. URL: `http://www.osapublishing.org/viewmedia.cfm?uri=oe-20-3-3241&seq=0&html=true`, doi:10.1364/OE.20.003241.

Linsker, R., 1988. Self-Organization in a Perceptual Network. Computer 21, 105–117. doi:10.1109/2.36.

Lukoševičius, M., Jaeger, H., 2009. Reservoir computing approaches to recurrent neural network training. Computer Science Review 3, 127–149.

Maass, W., Natschläger, T., Markram, H., 2002. Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Computation 14, 2531–2560. doi:10.1162/089976602760407955.

Manjunath, G., Jaeger, H., 2013. Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks. Neural Computation 25, 671–696.

Nakajima, K., Hauser, H., Kang, R., Guglielmino, E., Caldwell, D.G., Pfeifer, R., 2013. A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm. Frontiers in Computational Neuroscience 7, 91.

Nakajima, K., Hauser, H., Li, T., Pfeifer, R., 2015. Information processing via physical soft body. Scientific Reports 5, 10487. URL: `http://www.nature.com/srep/2015/150527/srep10487/full/srep10487.html`, doi:10.1038/srep10487.

Nakajima, K., Hauser, H., Li, T., Pfeifer, R., 2018. Exploiting the dynamics of soft materials for machine learning. Soft Robotics 5, 339–347.

Nakajima, K., Li, T., Hauser, H., Pfeifer, R., 2014. Exploiting short-term memory in soft body dynamics as a computational resource. Journal of The Royal Society Interface 11, 20140437.

Rabinovich, M., Huerta, R., Laurent, G., 2008. Transient dynamics for neural processing. Science 321, 48–50.

Rodan, A., Tino, P., 2010. Minimum Complexity Echo State Network. IEEE Transactions on Neural Networks 22, 1–14. doi:10.1109/TNN.2010.2089641.

Salmen, M., Ploger, P.G., 2005. Echo state networks used for motor control, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 1953–1958. doi:10.1109/ROBOT.2005.1570399.

Skowronski, M., Harris, J., 2007. Automatic speech recognition using a predictive echo state network classifier. Neural Networks 20, 414423.

Stieg, A.Z., Avizienis, A.V., Sillin, H.O., Martin-Olmos, C., Aono, M., Gimzewski, J.K., 2012. Emergent criticality in complex turing b-type atomic switch networks. Advanced Materials 24, 286–293.

Sussillo, D., Abbott, L.F., 2009. Generating coherent patterns of activity from chaotic neural networks. Neuron 63, 544–557.

Tanaka, T., Kaneko, T., Aoyagi, T., 2008. Recurrent Infomax Generates Cell Assemblies, Neuronal Avalanches, and Simple Cell-Like Selectivity. Neural Computation 21, 1038–1067. doi:10.1162/neco.2008.03-08-727.

Torrejon, J., Riou, M., Araujo, F.A., Tsunegi, S., Khalsa, G., Querlioz, D., Bortolotti, P., Cros, V., Yakushiji, K., Fukushima, A., et al., 2017. Neuromorphic computing with nanoscale spintronic oscillators. Nature 547, 428.

Toyoizumi, T., Abbott, L.F., 2011. Beyond the edge of chaos: Amplification and temporal integration by recurrent networks in the chaotic regime. Physical Review E 84, 051908. doi:10.1103/PhysRevE.84.051908.

Tsunegi, S., Taniguchi, T., Miwa, S., Nakajima, K., Yakushiji, K., Fukushima, A., Yuasa, S., Kubota, H., 2018. Evaluation of memory capacity of spin torque oscillator for recurrent neural networks. Japanese Journal of Applied Physics 57, 120307.

Woods, D., Naughton, T.J., 2012. Optical computing: Photonic neural networks. Nature Physics 8, 257.

Yildiz, I.B., Jaeger, H., Kiebel, S.J., 2012. Re-visiting the echo state property. Neural Networks 35, 1–9. URL: `http://dx.doi.org/10.1016/j.neunet.2012.07.005`, doi:10.1016/j.neunet.2012.07.005.

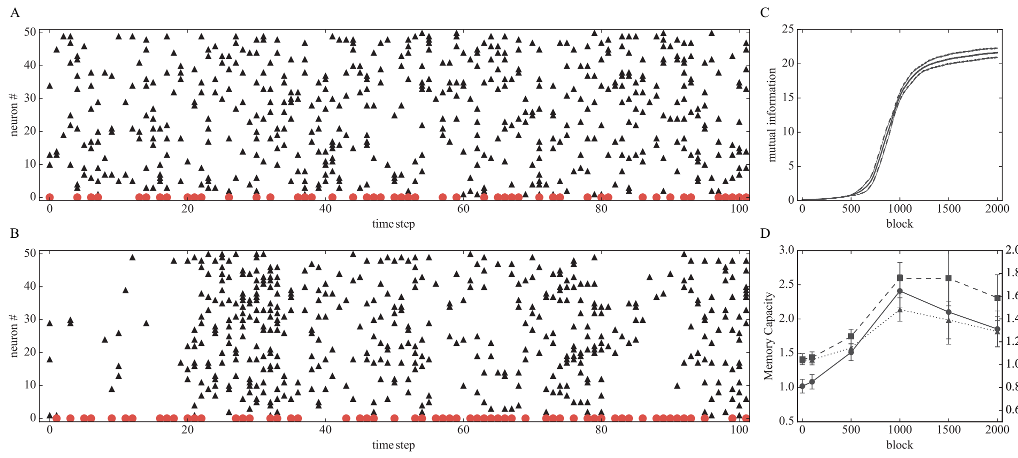

Figure 2: (Color online) (A, B) Raster plots of the firing activity of the network at the 0th and the 1000th blocks from the 50 000 to 50 100 time steps. The circular and triangular points represent the firing of input and neurons in the RNN, respectively. (C) Mutual information as a function of the block. The solid line represents the mean of the initial connection weights, and the dotted line represents the standard deviation for networks with different initial conditions. The number of trials is 10, which is constant throughout this study. (D) Results of the short-term memory and n-bit Boolean emulation tasks ($n = 2$, 3) as a function of the block. The solid line represents the results of short-term memory task, and the dashed and dotted lines represent the results of 2-bit and 3-bit Boolean emulation tasks, respectively. The error bar represents the standard deviation for networks with different initial conditions.

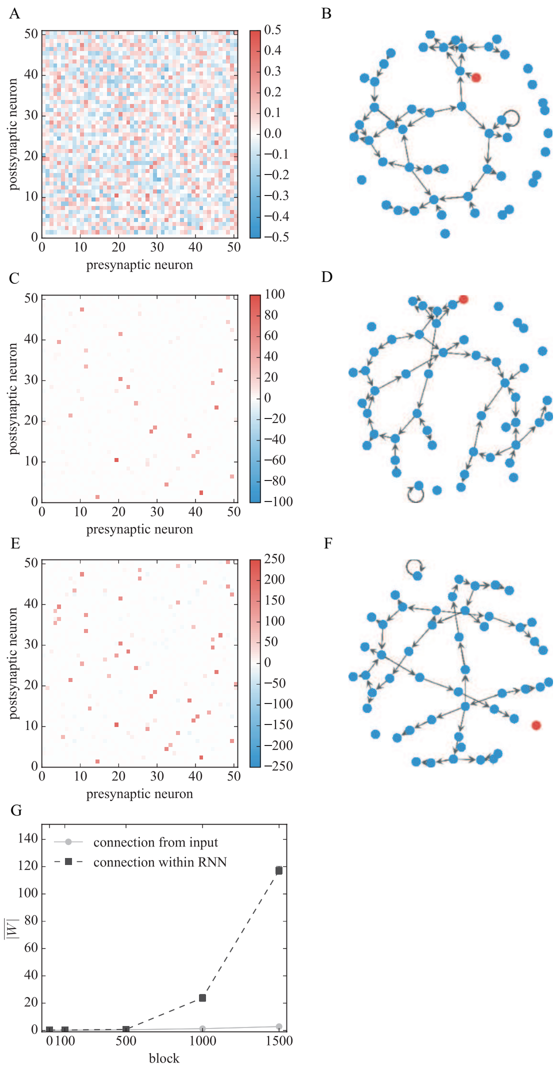

Figure 3: (Color online) (A, C, E) Heatmap of the connection weights at the (A) 0$^{\text{th}}$, (C) 1000$^{\text{th}}$ and (E) 1500$^{\text{th}}$ blocks. Neuron #0 represents the input. (B, D, F) Network structures corresponding to the heatmaps. A total of 50 connections with the largest absolute values are visualized. The blue and red nodes represent the neurons in the RNN and the input, respectively. (G) Mean absolute value of the connection weights from the input to the RNN and of the RNN as a function of the block. The mean and standard deviations for networks with random initial condition are given; however, the error bar is not substantial because of the small standard deviation.

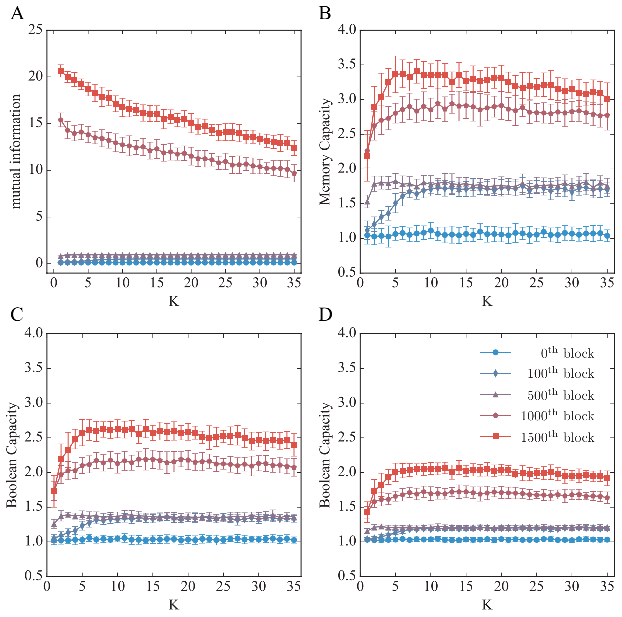

Figure 4: (Color online) (A) Mutual information as a function of input multiplicity, K. Color indicates block number. (B) Performance of the short-term memory task. (C) Performance of the 2-bit Boolean emulation task. (D) Performance of the 3-bit Boolean emulation task.

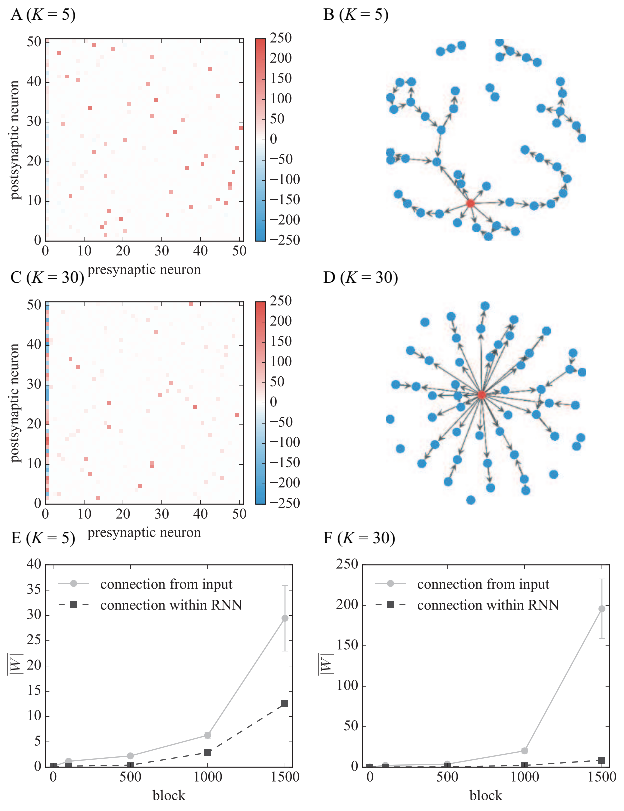

Figure 5: (Color online) (A, C) Heatmaps of connection weights for $K = 5$ and 30 at the 1500$^{\text{th}}$ block. Neuron #0 presents the input. (B, D) Network structures corresponding to the heatmaps on the left. A total of 50 connections with the largest absolute values are visualized. The blue nodes represent the neurons in the RNN, and the red nodes represent the input. (E, F) Mean absolute value of the connection weights from the input to the RNN and of the RNN as a function of the block number.

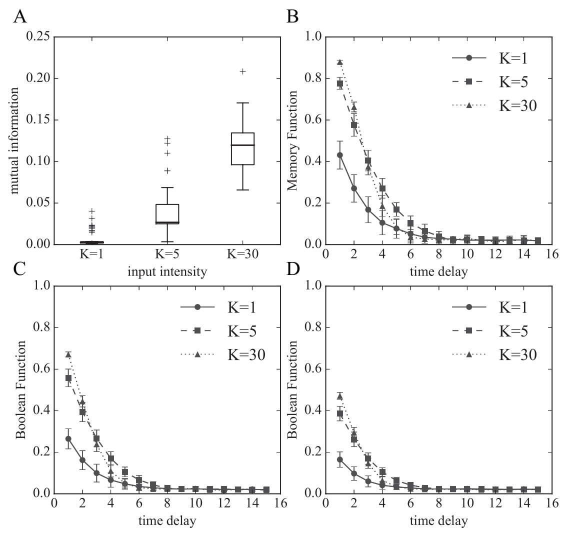

Figure 6: (A) Boxplot of mutual information between the previous input and the RNN estimated by sampling 50 000 time steps after washout time steps (50 000 time steps). (B)–(D) Performance of the (B) short-term memory task, (C) 2-bit Boolean emulation task, and (D) 3-bit Boolean emulation task as a function of the time delay for $K = 1$, 5 and 30. The error bar indicates the standard deviation for the networks for different initial conditions.

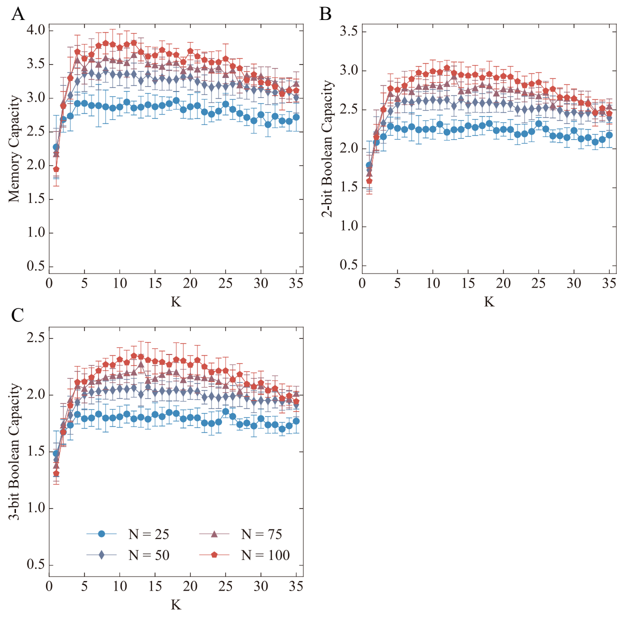

Figure 7: (Color online) Performance of the (A) short-term memory task, (B) 2-bit Boolean emulation task, and (C) 3-bit Boolean emulation task for various network sizes ($N = 25$, 50, 75, and 100).

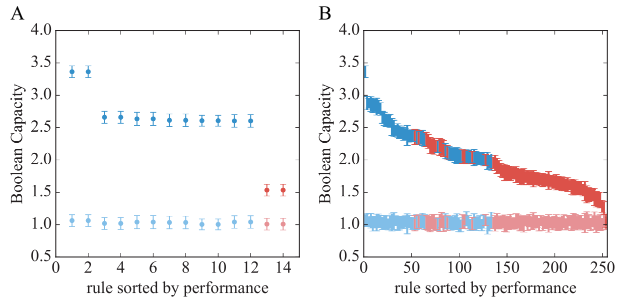

Figure 8: (Color online) Performance of each rule of the Boolean emulation tasks for $K = 5$ at the 0^{th} block (light colored points) and 1500^{th} block (dark colored points). The blue points represent the linear rules, and the red points represent the nonlinear rules. The error bar indicates the standard deviation for different initial networks. (A) 2-bit and (B) 3-bit Boolean emulation tasks.