frontiers in Physiology

ORIGINAL RESEARCH
published: 18 June 2021
doi: 10.3389/fphys.2021.685121

Optimizing Reservoir Computers for Signal Classification

*Thomas L. Carroll**

US Naval Research Lab, Washington, DC, United States

Reservoir computers are a type of recurrent neural network for which the network connections are not changed. To train the reservoir computer, a set of output signals from the network are fit to a training signal by a linear fit. As a result, training of a reservoir computer is fast, and reservoir computers may be built from analog hardware, resulting in high speed and low power consumption. To get the best performance from a reservoir computer, the hyperparameters of the reservoir computer must be optimized. In signal classification problems, parameter optimization may be computationally difficult; it is necessary to compare many realizations of the test signals to get good statistics on the classification probability. In this work, it is shown in both a spiking reservoir computer and a reservoir computer using continuous variables that the optimum classification performance occurs for the hyperparameters that maximize the entropy of the reservoir computer. Optimizing for entropy only requires a single realization of each signal to be classified, making the process much faster to compute.

Keywords: reservoir computer, machine learning, neuromorphic, nonlinear dynamics, neuron

OPEN ACCESS

Edited by:
Plamen Ch. Ivanov,
Boston University, United States

Reviewed by:
Bolun Chen,
Boston University, United States
Ruedi Stoop,
ETH Zürich, Switzerland

***Correspondence:**
Thomas L. Carroll
thomas.carroll@nrl.navy.mil

Specialty section:
This article was submitted to Fractal Physiology, a section of the journal Frontiers in Physiology

Received: *24 March 2021*
Accepted: *24 May 2021*
Published: *18 June 2021*

Citation:
Carroll TL (2021) Optimizing Reservoir Computers for Signal Classification. Front. Physiol. 12:685121. doi: 10.3389/fphys.2021.685121

1. INTRODUCTION

A reservoir computer is a type of recurrent neural network that is particularly easy to train. A typical reservoir computer is created by connecting a set of nonlinear nodes in a network that includes feedback connections. The first reservoir computers used nodes that were modeled on a hyperbolic tangent function (Jaeger, 2001) or excitable neurons that responded to an input by spiking (Natschlaeger et al., 2002). Unlike most neural networks, the network connections in a reservoir computer never change. Instead, training to a reservoir computer takes place by fitting the signals from the individual nodes to a training signal, usually by a linear fit. Because the network never changes, reservoir computers can be constructed from analog systems in which it is not possible to alter the connections between nodes.

Because the connections between nodes do not change, only the output parameters, training a reservoir computer can be faster than training a conventional neural network. Training a reservoir computer that has M nodes requires finding M linear fit coefficients. Train by adjusting network connections requires adjusting at least M^2 parameters, and most implementations of neural networks have far more parameters than this. In addition, stability is a concern in recurrent neural network training, so the number useful node activation functions is limited.

Examples of reservoir computers so far include photonic systems (Appeltant et al., 2011; Larger et al., 2012; der Sande et al., 2017), analog circuits (Schurmann et al., 2004), mechanical systems (Dion et al., 2018) and field programmable gate arrays (Canaday et al., 2018). Many other examples are included in the review paper (Tanaka et al., 2019), which describes hardware implementations of reservoir computers that are very fast, and yet consume little power, while being small and light. Reservoir computers have been shown to be useful for solving a number of problems, including

reconstruction and prediction of chaotic attractors (Jaeger and Haas, 2004; Lu et al., 2017, 2018; Antonik et al., 2018; Zimmermann and Parlitz, 2018), recognizing speech, handwriting or other images (Jalalvand et al., 2018) or controlling robotic systems (Lukoševičius et al., 2012). Reservoir computers have also been used to better understand the function of neurons in the brain (Stoop et al., 2013). Several groups have been using theory to better understand reservoir computers; in Hart et al. (2020), the authors show that there is a positive probability that a reservoir computer can be an embedding of the driving system, and therefore can predict the future of the driving system within an arbitrary tolerance. Lymburn et al. (2019) study the relation between generalized synchronization and reconstruction accuracy, while Herteux and Räth examine how the symmetry of the activation function affects reservoir computer performance (Herteux and Rath, 2020).

As with any neural network, there are hyperparameters of the nodes that must be optimized to get the best performance from the reservoir computer. Usually some form of nonlinear optimization routine is used. These routines require evaluation of the reservoir computer error for many different hyperparameter combinations. This computation can be slow for reservoir computers used for signal fitting or prediction, but it is even slower for reservoir computers used for classification. To evaluate the performance of a reservoir computer for classification, many realizations of a set of test signals must be generated in order to get good statistics on the probability of making an error in signal classification. In this work, for example, there is a set of four signals to be classified, and 100 test examples for each signal are used, so that for each combination of hyperparameters, the reservoir computer output must be computed 400 times. This large number of computations makes optimization impractical.

In this work, a set of hyperparameters is scanned one parameter at a time, and for each scan, the optimum parameter value is found. Some parameter combinations that might generate better performance may be missed, but it is seen that the best results for classification come when the entropy of the reservoir computer is maximized. Minimizing the error in fitting an input signal would seem to be another target for optimization, but it is found that the best classification performance sometimes comes when the fitting error is large.

Two different types of reservoir computer node are used in this work. The first type of node is a two dimensional ordinary differential equation that resembles a spiking neuron. This model was not created with any particular biological system in mind, but biological systems do consist of spiking neurons, so there may be some biological relevance. It is possible that the results seen for one type of node are only true for that type of node, so the second reservoir computer uses nodes that follow a third order ordinary differential equation.

Signals from the Sprott chaotic systems (Sprott, 1994) were used to test the ability of both node types to classify signals. It was found in a previous work (Carroll, 2021a) that the first four Sprott systems (A,B,C and D) were the hardest to distinguish, so these four systems are used here.

After mentioning the four Sprott systems, the statistic used to calculate the entropy of the reservoir computer is introduced. Next the spiking nodes are defined and their classification performance is measured as several parameters are changed. Following the sections on spiking nodes, the polynomial ordinary differential equation nodes are described and their classification performance is evaluated.

TABLE 1 | Lyapunov exponents of the Sprott systems A, B, C, and D, from Sprott (1994).

System			
A	0.014	0	–0.014
B	0.210	0	–1.210
C	0.163	0	–1.163
D	0.103	0	–1.320

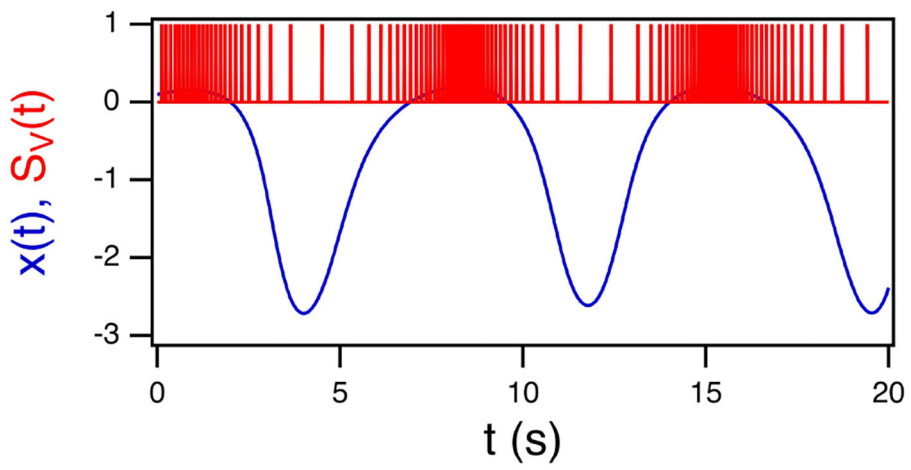

FIGURE 1 | The *x* signal from the Sprott A system in blue and its conversion to a spike time series $S_V(t)$ in red.

2. RESERVOIR COMPUTERS

A reservoir computer may be described by

$$\chi_i(n+1) = f(\chi_i(n)) + \sum_{j=1}^{M} A_{ij}\chi_j(n) + w_i s(t) \quad (1)$$

where the reservoir computer variables are the $\chi_i(n), i = 1 \ldots M$ with M the number of nodes, A is an adjacency matrix that described how the different nodes in the network are connected to each other, $\mathbf{W} = [w_1, w_2, \ldots w_M]$ describes how the input signal $s(t)$ is coupled into the different nodes, and f is a nonlinear function.

When the reservoir computer was driven with $s(t)$, the first 1,000 time steps were discarded as a transient. The next N time steps from each node were combined in a $N \times (M+1)$ matrix

$$\Omega = \begin{bmatrix} \chi_1(1) & \ldots & \chi_M(1) & 1 \\ \chi_1(2) & & \chi_M(2) & 1 \\ \vdots & & \vdots & \vdots \\ \chi_1(N) & \ldots & \chi_M(N) & 1 \end{bmatrix} \quad (2)$$

The last column of Ω was set to 1 to account for any constant offset in the fit. The training signal is fit by

$$h(t) = \Omega \mathbf{C} \quad (3)$$

where $h(t) = [h(1), h(2) \ldots h(N)]$ is the fit to the training signal $g(t) = [g(1), g(2) \ldots g(N)]$ and $\mathbf{C} = [c_1, c_2 \ldots c_N]$ is the coefficient vector.

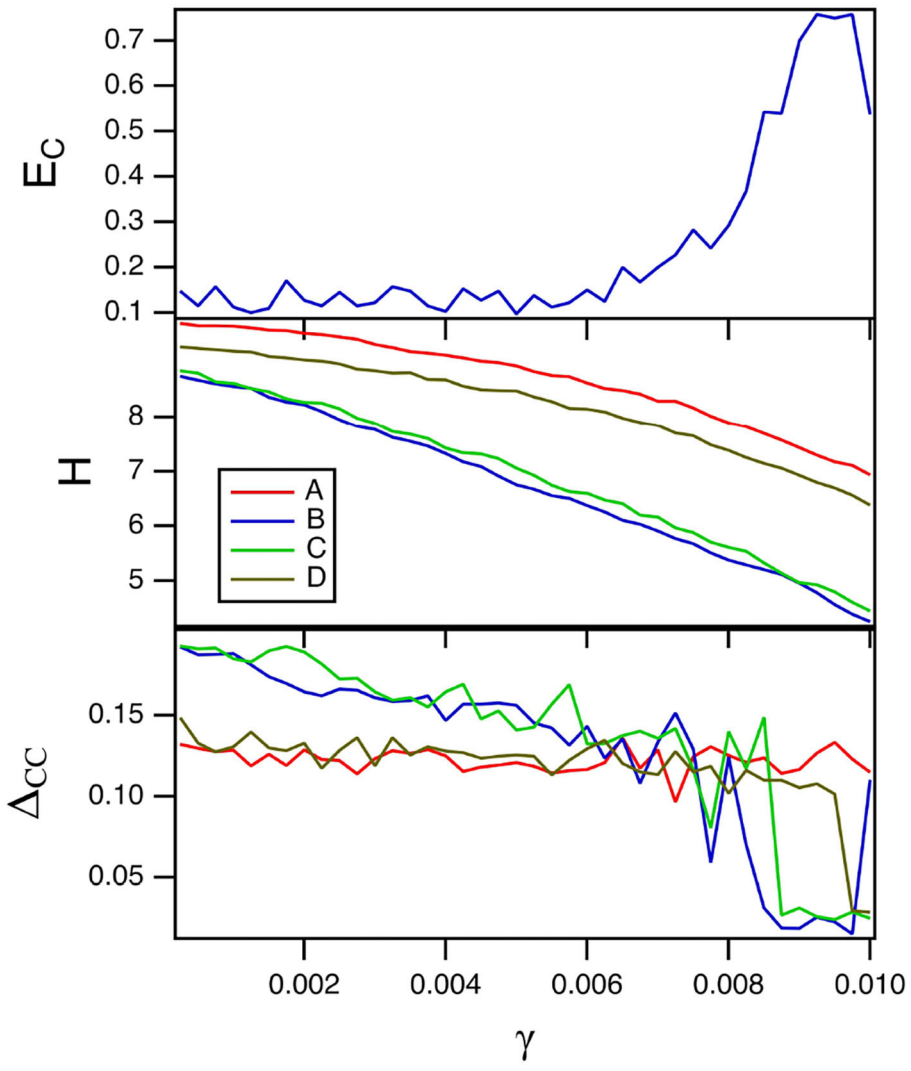

FIGURE 2 | The top plot shows the fraction of errors E_C in identifying which of the four Sprott systems was present, as a function of the damping constant γ for the spiking reservoir computer. The middle plot shows the entropies for the four Sprott systems over the same parameter range. The bottom plot shows the training error Δ_{CC}. For this plot, $T_C = 15$ and $R_\tau = 36$.

The fit coefficient vector is then found by

$$\mathbf{C} = \Omega_{inv} g(t) \tag{4}$$

where Ω_{inv} is the Moore-Penrose pseudo-inverse of Ω (Penrose, 1955) and S' is an $(M+1)\times(M+1)$ diagonal matrix constructed from S, where the diagonal element $S'_{i,i} = S_{i,i}/(S_{i,i}^2 + k^2)$, where $k = 1 \times 10^{-5}$ is a small number used for ridge regression (Tikhonov, 1943) to prevent overfitting.

The training error may be computed from

$$\Delta_{RC} = \frac{\text{std}\left[\Omega\mathbf{C} - g(t)\right]}{\text{std}\left[g(t)\right]} \tag{5}$$

where std[ ] indicates a standard deviation.

3. SPROTT SYSTEMS

The input signals for this work were the x signals from one of the Sprott systems A, B, C, or D (Sprott, 1994). These systems may be described by

$$\begin{array}{cc} \text{A} \begin{bmatrix} \dot{x} = y \\ \dot{y} = -x + yz \\ \dot{z} = 1 - y^2 \end{bmatrix} & \text{B} \begin{bmatrix} \dot{x} = yz \\ \dot{y} = x - y \\ \dot{z} = 1 - xy \end{bmatrix} \\ \text{C} \begin{bmatrix} \dot{x} = yz \\ \dot{y} = x - y \\ \dot{z} = 1 - x^2 \end{bmatrix} & \text{D} \begin{bmatrix} \dot{x} = -y \\ \dot{y} = x + z \\ \dot{z} = xz - 3y^2 \end{bmatrix} \end{array}. \tag{6}$$

In a previous study using a reservoir computer based on continuous variables, these four Sprott systems were more difficult to distinguish than the other Sprott systems. The Sprott systems were numerically integrated by a 4th order Runge-Kutta integration routine that had a variable step size. The time step for the output of the integration routine was 0.01.

Table 1 lists the Lyapunov exponents for the four Sprott systems used in this work.

4. ENTROPY STATISTIC

Besides calculating such statistics as Lyapunov exponents for the reservoir computers, it was useful to characterize the entropy of the reservoir computer. Measuring entropy requires a partitioning of the dynamical system. Xiong et al. (2017) lists a number of ways to do this partitioning, although different partitions can give different results for the entropy. Some of these methods begin with the phase space representation of the dynamical system and then coarse grain the phase space representation to create partitions. Because of the different time scales in the spiking system, coarse graining of the spiking reservoir signals leads to a loss of information; the effect of coarse graining on entropy calculations was examined in Xu et al. (2011). It was found that the permutation entropy method (Bandt and Pompe, 2002) avoided this coarse graining problem because it creates partitions based on the time ordering of the signals. Each individual node time series $r_i(t)$ was divided into windows of 4 points, and the points within the window were sorted to establish their order; for example, if the points within a window were 0.1, 0.3, –0.1 0.2, the ordering would be 2,4,1,3. Each possible ordering of points in a signal $r_i(t)$ represented a symbol $\psi_i(t)$.

At each time step t, the individual node signals were combined into a reservoir computer symbol $\Lambda(t) = [\psi_1(t), \psi_2(t), \ldots \psi_M(t)]$. With $M = 100$ nodes there were potentially a huge number of possible symbols, but the nodes were all driven by a common drive signal, so only a tiny fraction of the symbol space was actually occupied, on the order of tens of symbols for the entire reservoir computer.

If K total symbols were observed for the reservoir computer for the entire time series, then the reservoir computer entropy was

$$H = -\sum_{k=1}^{K} p\left(\Lambda_k\right) \log\left(p\left(\Lambda_k\right)\right) \tag{7}$$

where $p(\Lambda_k)$ is the probability of the k'th symbol.

5. SPIKING NODES

The spiking node is a simple 2 dimensional ordinary differential equation

$$\begin{aligned} \tfrac{du_i}{dt} &= T_C\left(-u_i^3 + u_i \times g\left(v_i, \phi_i\right)\right) \\ \tfrac{dv_i}{dt} &= T_C\left(\tfrac{1}{R_\tau}\right)\left(W_i S_V - \gamma v_i + \sum_{j=1}^{M} A_{ij} u_j\right) \\ g\left(v, \phi\right) &= \left\{ \begin{matrix} 0 \; v < \phi \\ 1 \; v > \phi \end{matrix} \right\} \\ &\text{if } u_i > 0.5 \text{ then } v_i = 0 \end{aligned} \tag{8}$$

These nodes were chosen because their simplicity made it easy to understand how their behavior depended on their parameters. The variable $u_i(t)$ is the fast variable, while $v_i(t)$ is the slow variable, and the ratio of fast to slow times is set by R_τ. The firing threshold ϕ_i was different for each node and was set to a value drawn from a uniform random distribution between 1 and 1.1.

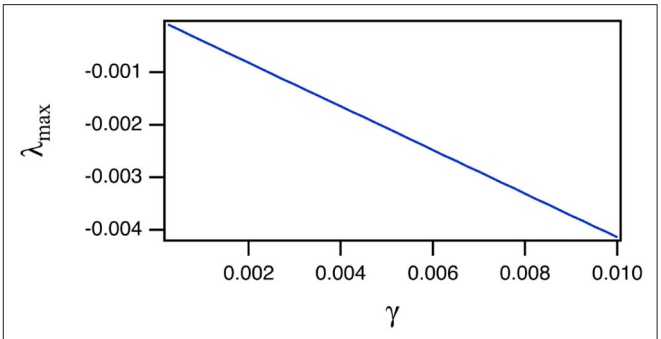

FIGURE 3 | Largest Lyapunov exponent λ_{max} for the spiking reservoir when driven by the Sprott A system as the damping constant γ varies. The largest Lyapunov exponents when driven by the other systems were almost identical, so only the result from A is shown.

The random firing thresholds created diversity in the set of nodes. There were $M = 100$ nodes in the reservoir computer.

The input signal S_V is a time series of spikes. The elements of the input vector **W** were chosen from a uniform random distribution between 0 and 1. The adjacency matrix **A** was created by randomly selecting half of its entries and setting them to values drawn from a uniform random distribution between –1 and 1. The diagonal elements of **A** were then set to zero. Because the mean value of **A** was 0, the connections between nodes were evenly divided between excitatory and inhibitory. When the connections were balanced in this way, the inhibitory connections canceled out the excitatory connections and no spiking occurred. In order to get spiking, an offset of 0.5 was added to **A**. After the offset was added, **A** was renormalized to set the spectral radius σ, the largest magnitude of its complex eigenvalues, to 0.5.

The reservoir equations were numerically integrated with a fourth order adaptive step size Runge-Kutta integration routine with a time step of 1.

5.1. Input Signals for Spiking Nodes

The input signal $s(t)$ came from the x component of one of the Sprott chaotic systems, either system A, B, C or D. In a previous project using a continuous reservoir computer to classify signals, these were the most difficult of the Sprott systems to distinguish. The Sprott systems were integrated with a time step of 0.01.

The Sprott x signals were converted to a time series of spikes. First, the input signal $s(t)$ was normalized to the signal $\tilde{s}(t)$ which

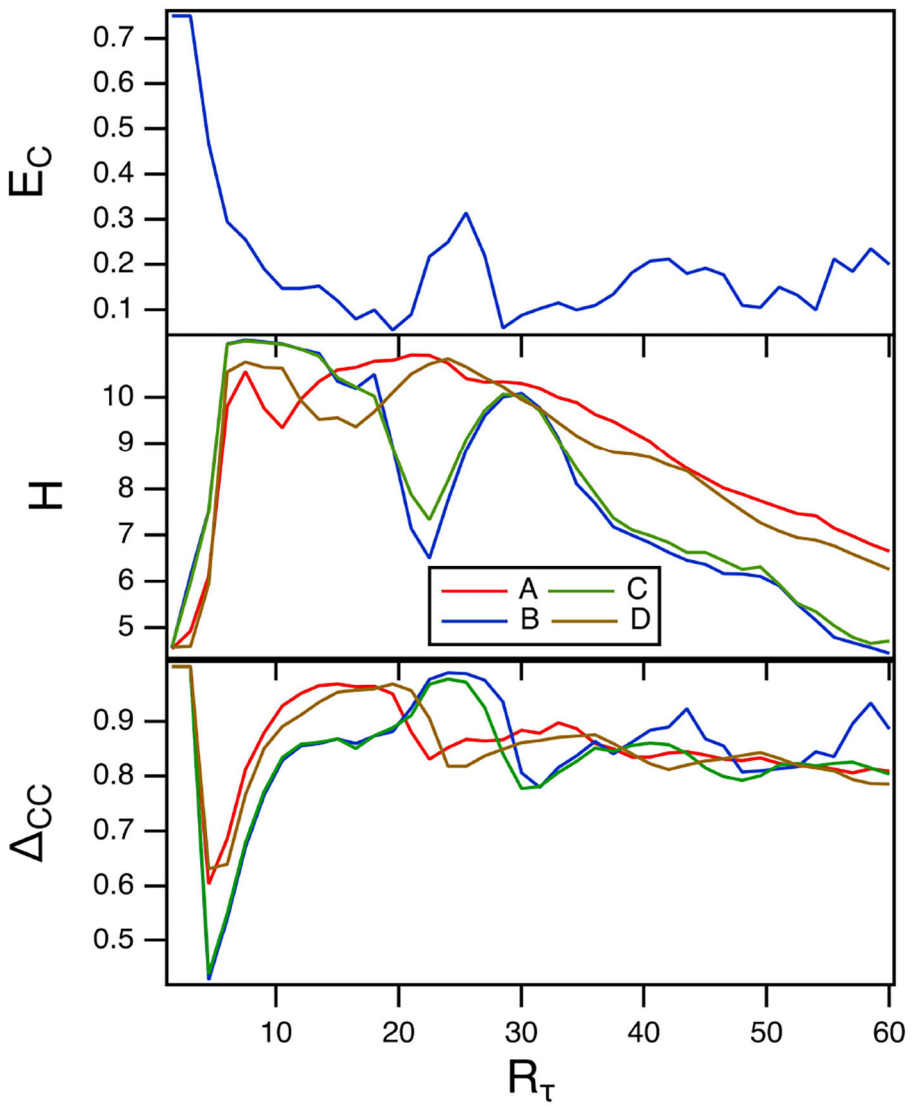

FIGURE 4 | The top plot shows the fraction of errors E_C in identifying which of the four Sprott systems was present, as a function of the fast to slow time ratio R_τ for the spiking reservoir computer. The middle plot shows the entropies for the four Sprott systems over the same parameter range. The bottom plot is the training error Δ_{CC}. For this plot, $\gamma = 0.004$ and $R_\tau = 36$.

had a maximum of 1 and a minimum of 0. The signal contains N points. A minimum and a maximum period between spikes, T_{min} and T_{max}, were chosen. For an input signal s, a spike time series S_V is produced according to

$S_V(\mathrm{i}) \leftarrow 0\ i = 1 \ldots N$
$i_0 \leftarrow 1$
$i_1 \leftarrow 1$
while do$(i_0 < N)$
  SpikePeriod $\leftarrow \lfloor s\,(i_0)\,(T_{\max} - T_{\min})\rfloor + T_{\min}$
  $i_1 \leftarrow i_0$+SpikePeriod
  $S_V(i_1) \leftarrow 1$
  $i_0 \leftarrow i_1$
end while

The floor operator $\lfloor\rfloor$ returns the largest integer that is less than the argument. For all the signals in this work, $T_{max} = 100$ and $T_{min} = 10$.

Figure 1 shows the x signal from the Sprott A system and its conversion to a time series of spikes.

5.2. Training the Spiking System

In the training phase, the reservoir computer was driven with a 100,000 point spiking input signal $S_V(t)$ from each of the four Sprott systems. The training signal for each instance was the same as the input signal, so the reservoir computer was trained to reproduce $S_V(t)$. The actual fit signal to $S_V(t)$ is $g\,(t) = \sum_{i=1}^{M} c_i u_i\,(t)$, where the fit is usually done by a ridge regression to avoid overfitting and the c_i's are the fit coefficients. The training error Δ_{CC} was calculated as the cross correlation between $S_V(t)$ and $g(t)$:

$$\Delta_{CC} = 1 - \frac{\sum_{j=1}^{N}\left[\left(g\,(j) - \bar{g}\right)\left(S_V\,(j) - \bar{S}_V\,(j)\right)\right]}{\sum_{j=1}^{N}\left(g\,(j) - \bar{g}\right)\sum_{j=1}^{N}\left(S_V\,(j) - \bar{S}_V\,(j)\right)} \tag{9}$$

where the overbar operator indicates a mean.

In the training phase, for each Sprott system a set of coefficients $\mathbf{C}_\alpha, \alpha =$ A,B,C, or D was produced. For testing the classification procedure, 100 instances of the x signal of length 5,000 points were converted to spiking signals and used to drive the reservoir computer of Equation (8). The sets of fit coefficients for the different Sprott systems in the testing phase were designated $\mathbf{K}_\beta$, $\beta =$ A,B,C, or D. The fit coefficients may be represented as a matrix

$$\mathbf{K} = \begin{bmatrix} K_{1,A} & \ldots & K_{M,A} \\ K_{1,B} & \ldots & K_{M,B} \\ K_{1,C} & \ldots & K_{M,C} \\ K_{1,D} & \ldots & K_{M,D} \end{bmatrix}. \tag{10}$$

The classification error was found by taking the Euclidean difference between the coefficients from the training and testing stages. For example, if $\beta =$ A, the errors δ_b were found as

$$\delta_b = \sqrt{\sum_{l=1}^{M}\left(K_{l,A} - C_{l,b}\right)^2} \quad b = \mathrm{A, B, C, D} \tag{11}$$

where l indicates the particular component of the coefficient. If the value of b corresponding to the minimum δ_b is not equal to A, then an error is recorded.

6. SPIKING RESERVOIR CLASSIFICATION PERFORMANCE

The following sections evaluate the fraction of errors in identifying the four Sprott systems as one of the reservoir computer parameters γ, R_τ, or T_C is varied. One parameter is varied while the others are held constant. In many studies of reservoir computers for signal prediction, all the hyperparameters are optimized simultaneously through a nonlinear optimization procedure, but to build up sufficient classification statistics requires many repeat simulations of the reservoir computer for each set of parameters, making optimization computationally very slow. Instead, the classification performance is evaluated for one parameter at a time.

6.1. Varying γ

The parameter γ is a linear damping parameter in the slow time equation for the spiking nodes (Equation 8). Smaller values of γ mean that the reservoir computer remembers inputs for a longer time. The top plot in **Figure 2** shows the fraction of errors E_C in identification as the damping term γ in Equation (8) is varied. For these simulations, $T_C = 15$ and $R_\tau = 36$. Because there are four systems, the highest probability of error is 0.75. The minimum error fraction in this plot is about 0.1. The middle plot shows the entropy for the reservoir computer driven by the four different Sprott systems over the same parameter range. **Figure 2** shows that larger entropies correspond to lower classification errors, which is expected from Crutchfield and Young (1990) and Langton (1990).

The training error Δ_{CC} when each of the four Sprott systems drive the spiking reservoir computer is plotted in the bottom plot in **Figure 2**. The training error is large when the error

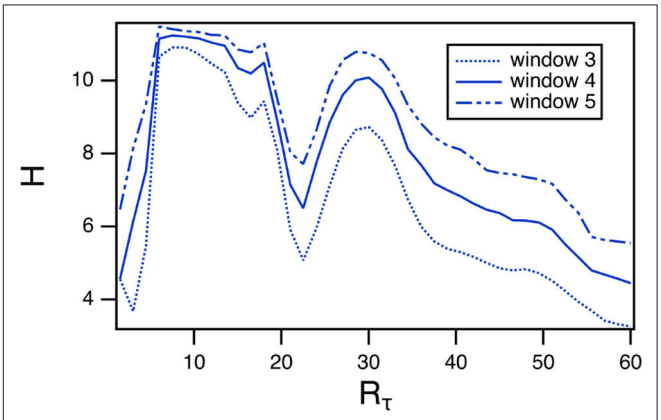

FIGURE 5 | Permutation entropy H calculated using different window lengths as a function of the fast to slow time ratio R_τ for the spiking reservoir computer driven by Sprott system B. For this plot, $\gamma = 0.004$ and $R_\tau = 36$.

in identifying the four Sprott systems is small. This is even more surprising because the entropy H for the spiking reservoir computer is larger when the training error is large. Theories of computation such as Crutchfield and Young (1990) and Langton (1990) would lead one to suspect that larger entropy would lead to better signal reproduction performance.

The reason for the large training error Δ_{CC} is explained by **Figure 3**, which shows the largest Lyapunov exponent for the spiking reservoir computer as γ varies. The Lyapunov exponent was calculated by the Gram-Schmidt method (Parker and Chua, 1989). The value of this Lyapunov exponent ranges from -1×10^{-4} to -4×10^{-3}. This range may be compared with the Lyapunov exponents for the Sprott systems in **Table 1**. For all four Sprott systems the largest Lyapunov exponent for the spiking reservoir computer is larger than the negative Lyapunov exponent for each of the Sprott systems. It has been shown in Badii et al. (1988) that such an overlap can increase the fractal dimension of a signal, and Carroll (2020a) showed that this dimension increase can lead to an increase in training error.

6.2. Varying R_τ

The parameter R_τ in the spiking nodes sets the ratio between fast and slow times. The top plot in **Figure 4** is the fraction of errors in classifying the Sprott signals as R_τ is varied. The other parameters for this plot were $\gamma = 0.004$ and $T_C = 15$. The middle plot of this figure shows the reservoir computer entropy. The lowest error fraction comes when the reservoir computer entropy is larger, as was also seen when γ was varied. The training error Δ_{CC} is in the bottom plot. The training error is large over the entire range of R_τ, so it appears that values of R_τ that minimize the classification error fraction do not result in small training errors. As in the previous section the overlap between the reservoir

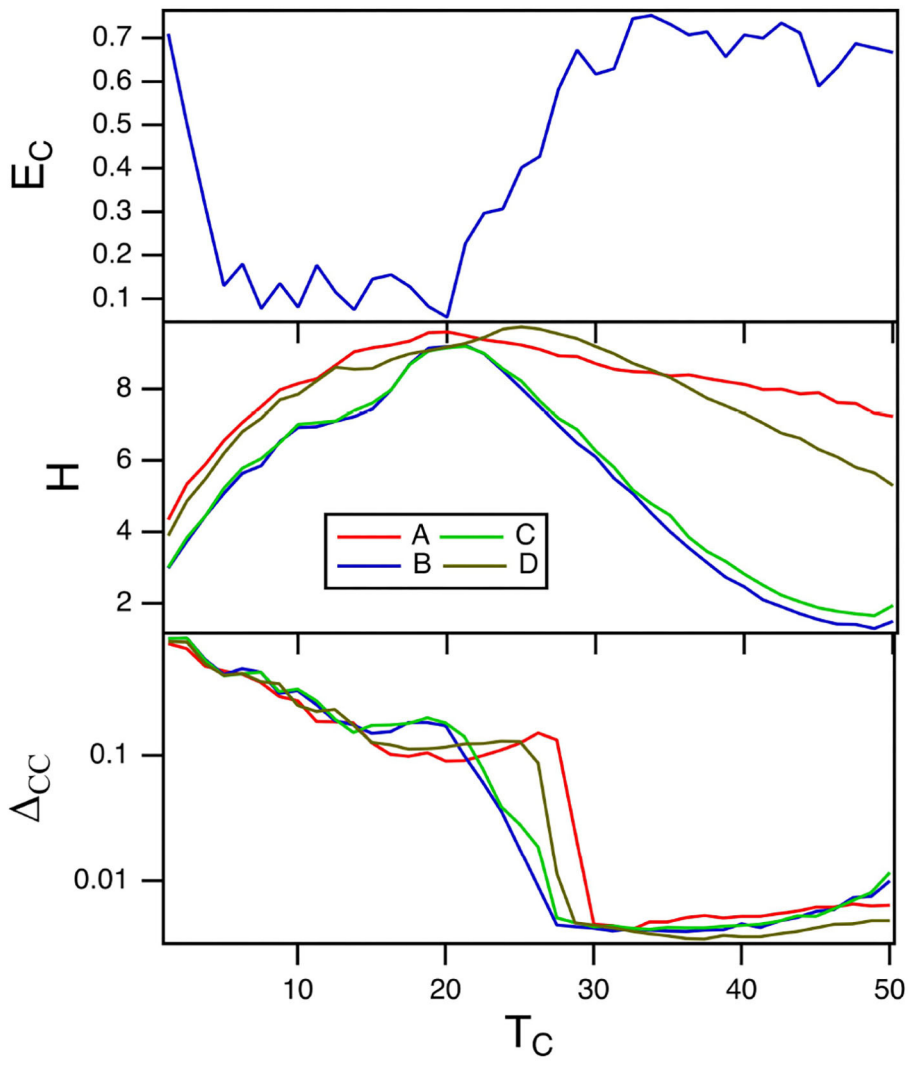

FIGURE 6 | The top plot shows the fraction of errors E_C in identifying which of the four Sprott systems was present, as a function of the time constant T_C for the spiking reservoir computer. The middle plot shows the entropies H for the four Sprott systems over the same parameter range. The bottom plot is the training error Δ_{CC}. For this plot, $\gamma = 0.004$ and $R_\tau = 36$.

computer Lyapunov spectrum and the Lyapunov exponents of the Sprott systems may be responsible for this large error. The largest Lyapunov exponent for the spiking reservoir computer for this range of parameters ranges from –0.04 to 0.

6.2.1. Effect of Window Length on Entropy

The length of the window in the permutation entropy calculation is an arbitrary parameter, so it is reasonable to ask how changing this parameter affects the results. The curve of entropy vs. R_{tau} for Sprott system B has a very distinct pattern, so the permutation entropy calculation was repeated with window lengths of 3 and 5. The results are shown in **Figure 5**. The entropy does get larger as the window length increases, but the pattern remains the same as R_τ varies.

6.3. Varying T_C

The parameter T_C for the spiking nodes matches the time scale of the reservoir computer to the time scale of the input signal. **Figure 6** shows the fraction of errors for the spiking reservoir when identifying the Sprott signals (top plot) and the reservoir computer entropy (middle plot). The training error is shown in the bottom plot. The error fraction E_C is small for a range of values of T_C, but it is useful to note that the region of largest entropy H corresponds to a low error fraction.

The plot of the training error in almost looks like the inverse of the plot of error fraction. The training error Δ_{CC} is large where E_C is small and smaller where E_C is larger. The conventional picture of computing with dynamical systems (Crutchfield and Young, 1990; Langton, 1990) suggests that the training error should be smaller when the entropy is large, but **Figure 6** shows that in this situation the opposite is true.

As when γ was varied, the maximum Lyapunov exponent for the spiking reservoir computer affects the training error. The reservoir maximum Lyapunov exponent varies between -6×10^{-3} and zero as T_C varies, so the reservoir Lyapunov spectrum overlaps with the Sprott Lyapunov spectra, causing a larger training error.

These simulations of the spiking reservoir computer have shown that the best performance in classifying signals comes when the entropy of the reservoir computer is large. This is not surprising, as good signal identification depends on maximizing the difference in the response of the reservoir to different signals. What is surprising is that the performance in fitting the input signal is poor when the classification performance is good. It was shown in Carroll (2020a) that the highest entropy in a reservoir computer sometimes occurs when the Lyapunov exponent is approaching zero. This is expected (Crutchfield and Young, 1990; Langton, 1990), but as the Lyapunov exponent approaches zero it may also overlap with the Lyapunov spectrum of the input system, increasing the fractal dimension of the reservoir signals, resulting in poor signal fitting performance.

To optimize the signal classification performance of a reservoir computer without computing enough realizations to get good error statistics, it would be most useful to adjust the reservoir computer parameters to maximize the entropy. This process should only require one realization of the reservoir computer for each set of parameters, rather than many.

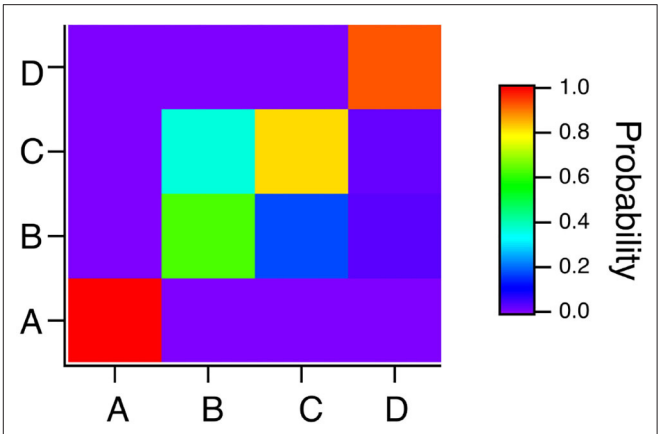

FIGURE 7 | Confusion matrix for identifying the four Sprott systems A,B,C and D with the spiking reservoir computer when $\gamma = 0.004$, $R_\tau = 36$ and $T_C = 15$. Along the *x* axis are the actual Sprott systems, while the *y* axis shows the probability that the Sprott system will be identified as that particular system.

To summarize the results for the spiking reservoir computer for the four Sprott systems, **Figure 7** is a confusion matrix for this type of reservoir computer for $\gamma = 0.004$, $T_C = 15$ and $R_\tau = 36$. The confusion matrix shows that system A is always identified correctly, but systems B and C can be hard to distinguish. System D is usually identified correctly, but it is sometimes identified as system B.

The spiking reservoir computer in this section uses a simple set of equations with a finite range of parameters. The significance of this model to biology, or to reservoir computers in general, is not known. To determine if the results in this section apply in general, a very different reservoir computer is simulated in the next section.

7. CONTINUOUS NODES

The spiking nodes in the previous sections may not be a completely accurate model of a biological system, and the results may only be true for those particular nodes. To see if the results in the previous sections are general, a different type of node is used to create a reservoir computer. If similar results are seen for these very different nodes, then the results are more likely to apply to reservoir computers in general.

The nodes in the continuous reservoir computer are a polynomial ordinary differential equations (ODE) described by

$$\frac{dr_i(t)}{dt} = \alpha \left[p_1 r_i(t) + p_2 r_i^2(t) + p_3 r_i^3(t) + \sum_{j=1}^{M} A_{ij} r_j(t) + W_i s(t) \right]. \tag{12}$$

There were $M = 100$ nodes in total. This type of node was introduced in Carroll and Pecora (2019).

As in the previous sections, the adjacency matrix **A** was created by randomly selecting half of its entries and setting

them to values drawn from a uniform random distribution between –1 and 1 and the diagonal elements of **A** were then set to zero. Unlike the previous section, no offset was added to the adjacency matrix. The adjacency matrix was renormalized to have a specified spectral radius σ. The elements of input vector **W** were again chosen from a uniform random distribution between –1 and 1. The polynomial ODE was numerically integrated with a fourth order Runge-Kutta routine with a step size of 1.

In the training stage, the input signal $s(t)$ was the x signal from one of the four Sprott systems (A,B,C,D). The Sprott systems in this section were numerically integrated with a time step of 0.1. The first 1,000 points from the $r_i(t)$ time series are discarded and the next 20,000 points are used to fit the training signal. The actual fit signal is $h(t) = \sum_{i=1}^{M} c_i r_i(t)$, where the fit is usually done by a ridge regression to avoid overfitting. The training error is

$$\Delta_{RC} = \frac{\langle x(t) - h(t) \rangle}{\langle x(t) \rangle} \tag{13}$$

where $\langle\ \rangle$ indicates a standard deviation.

The result of the training was a set of fit coefficients $\mathbf{C}_\alpha, \alpha =$ A,B,C, or D.

For testing the classification procedure, 100 instances of the x signal of length 2,000 points were used to drive the polynomial ODE reservoir computer. The sets of fit coefficients for the different Sprott systems in the testing phase were designated $\mathbf{K}_\beta$, $\beta =$ A,B,C, or D. As in the previous sections, the classification error was found by taking the Euclidean difference between the

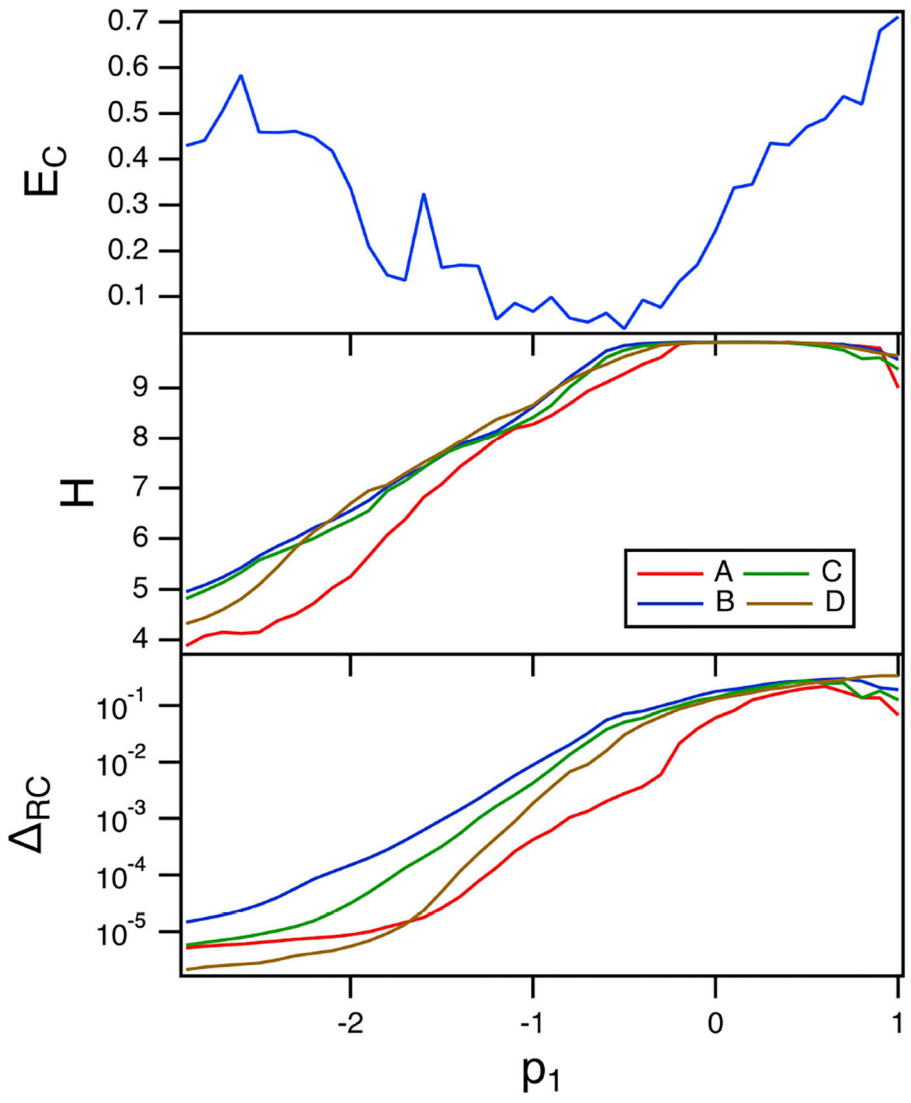

FIGURE 8 | The top plot shows the fraction of errors E_C in identifying which of the four Sprott systems was present, as a function of the linear parameter p_1 for the polynomial ODE reservoir computer. The middle plot shows the entropies for the four Sprott systems over the same parameter range. The bottom plot shows the training error Δ_{RC}. For this plot, $\alpha = 0.3$ and the spectral radius $\sigma = 0.8$.

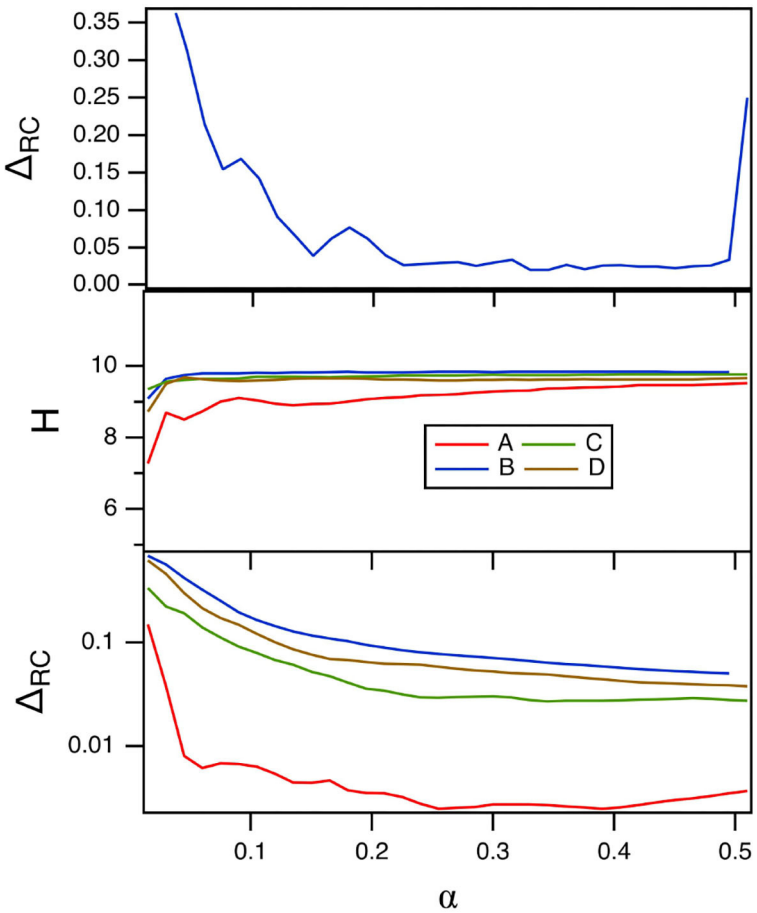

FIGURE 9 | The top plot shows the fraction of errors E_C in identifying which of the four Sprott systems was present, as a function of the time scale parameter α for the polynomial ODE reservoir computer. The middle plot shows the entropies for the four Sprott systems over the same parameter range, while the bottom plot is the training error Δ_{RC}. For this plot, $p_1 = -0.5$ and the spectral radius $\sigma = 0.8$.

coefficients from the training and testing stages. For example, if $\beta = \mathrm{A}$, the errors δ_b were found as

$$\delta_b = \sqrt{\sum_{l=1}^{M} \left(K_{l,A} - C_{l,b}\right)^2} \quad b = \mathrm{A, B, C, D} \tag{14}$$

where l indicates the particular component of the coefficient. If the value of b corresponding to the minimum δ_b is not equal to A, then an error is recorded.

7.1. Variation of p_1

The top plot in **Figure 8** is a plot of the fraction of errors in identifying the four Sprott systems as the linear parameter p_1 in the polynomial ODE reservoir is varied. For this plot, $\alpha = 0.3$ and the spectral radius $\sigma = 0.8$.

The minimum error in identifying the four Sprott systems using the polynomial ODE reservoir computer comes at a value of p_1 near the maximum reservoir computer entropy. This is the same behavior the was seen for the spiking reservoir computer. The best parameter for classifying the Sprott signals is not the best parameter for fitting the Sprott x signal, as **Figure 8** shows.

The training error when p_1 varies is large even though the entropy of the reservoir computer is near its maximum. Once again, the maximum Lyapunov exponent for the reservoir computer is large enough to overlap with the Lyapunov spectrum of the Sprott systems, causing an increase in the fractal dimension of the reservoir signals. The largest Lyapunov exponent for the polynomial reservoir computer as p_1 is varies ranges between –0.7 and 0. **Table 1** shows that the reservoir computer Lyapunov exponent overlaps with the Lyapunov spectra of systems B,C and D, and overlaps with the Lyapunov spectrum of system A for part of the range of p_1.

7.2. Variation of α

The parameter α varies the time scale of the polynomial ODE reservoir computer. The main effect of α is to match the frequency response of the reservoir computer to the frequency spectrum of the input signal. **Figure 9** shows the fraction of errors in identifying the four Sprott systems, E_C in the top plot, and the

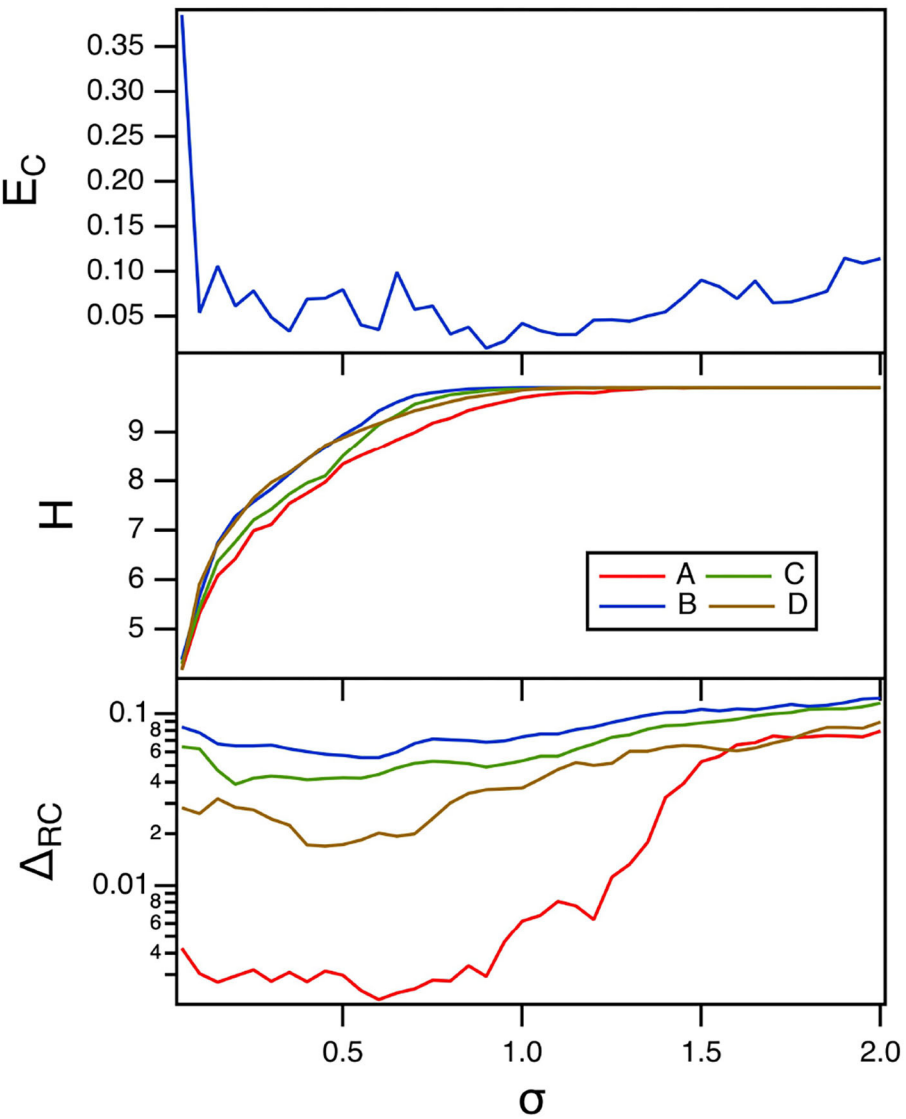

FIGURE 10 | The top plot shows the fraction of errors E_C in identifying which of the four Sprott systems was present, as a function of the spectral radius σ for the polynomial ODE reservoir computer. The bottom plot shows the entropies for the four Sprott systems over the same parameter range. The training error Δ_{RC} is shown in the bottom plot. For this plot, $p_1 = -0.5$ and the time constant $T_0 = 0.3$.

reservoir computer entropy H in the middle plot. The bottom plot is the training error. For this figure, $p_1 = -0.5$ and $\sigma = 0.8$.

Figure 9 shows that the error in identifying the Sprott systems is low for most of the range of α. For $\alpha \geq 0.5$, the polynomial ODE reservoir computer becomes unstable. The reservoir computer entropy is high over the entire range of α. The training error as α is varied appears to be small, but comparing with the training error shown in **Figure 8** shows that the training error is not as small as it could be for the polynomial ODE reservoir computer. Once again, the Lyapunov exponent spectrum of the reservoir computer overlaps the Lyapunov exponent spectrum of the Sprott systems. The largest Lyapunov exponent for the reservoir computer for these parameters ranges from –0.05 to 0, numbers that may be compared to the Sprott Lyapunov exponents in **Table 1**.

7.3. Variation of Spectral Radius σ

The final parameter to be varied for the polynomial ODE reservoir computer is the spectral radius σ of the adjacency matrix. Carroll (2020b, 2021b) showed that increasing the interaction between reservoir computer nodes by increasing the spectral radius increased the entropy. The top plot in **Figure 10** shows the error fraction in identifying the Sprott systems as the spectral radius is swept. The time constant α for this plot was 0.3, while the parameter p_1 was –0.5. The middle plot in this figure shows the entropy of the reservoir computer.

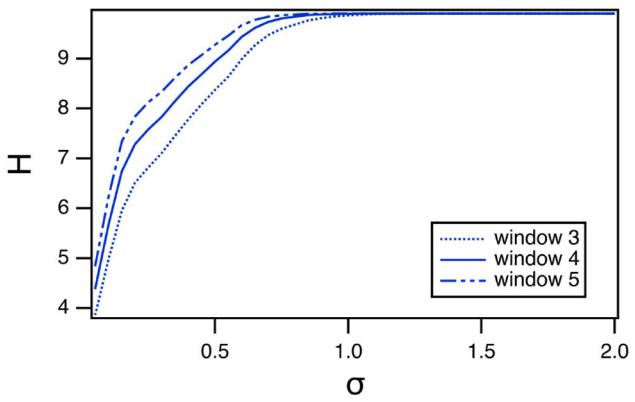

FIGURE 11 | Permutation entropy H as a function of the spectral radius σ for the polynomial ODE reservoir computer driven by Sprott system B, as the window length varies. For this plot, $p_1 = -0.5$ and the time constant $T_0 = 0.3$.

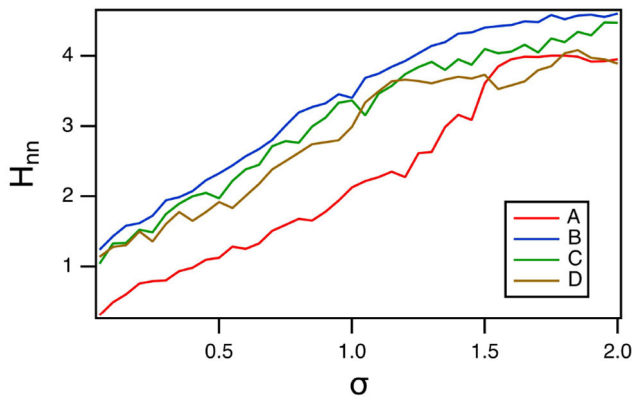

FIGURE 12 | Nearest neighbor entropy H_{nn} as a function of the spectral radius σ for the polynomial ODE reservoir computer driven by Sprott systems A,B,C, and D, as the window length varies. For this plot, $p_1 = -0.5$ and the time constant $T_0 = 0.3$.

There is a broad range of σ in **Figure 10** for which the identification error is small even though the entropy varies considerably. Nevertheless, having a large value of entropy still leads to small identification error, so maximizing the entropy is a useful way to minimize the identification error.

The training error for the polynomial reservoir computer is plotted in the bottom plot of **Figure 10**. While the training error is not large, it does not get as small as for the lowest values of p_1 in **Figure 8**. The largest Lyapunov exponent for the reservoir computer for this range of σ ranges from –0.12 to –0.01. Comparing to **Table 1**, the largest Lyapunov exponent for the reservoir computer overlaps with the Lyapunov spectrum for Sprott systems B, C, and D, but not for system A, which may be why the training error for system A is lower than for the other three systems.

7.3.1. Vary Window Length

As with the spiking reservoir computer, it is possible that the window length used to calculate the permutation entropy can affect the results. **Figure 11** shows how the window length affects the value of the permutation entropy for Sprott system B, as the spectral radius varies. The magnitude of the permutation entropy increases with window length, but the pattern of variation does not change.

Because the polynomial ODE reservoir computer did not have a large a variation in time scales as the spiking reservoir computer, entropy methods that used coarse graining in phase space could also be used. In **Figure 12**, the nearest neighbor entropy (Kraskov et al., 2004) is estimated. For this entropy, the set of reservoir signals $r_i(t), i = 1 \ldots M$ is considered as an M dimensional vector $\mathbf{R}(t)$. A number of index points on the vector are randomly chosen, and for each index point $\mathbf{R}(t_n)$, the distance to the k nearest neighbor is ε_n. The nearest neighbor entropy H_{nn} is then calculated as

$$H_{nn} = \psi(N) - \psi(k) + \langle \ln \varepsilon_n \rangle \quad (15)$$

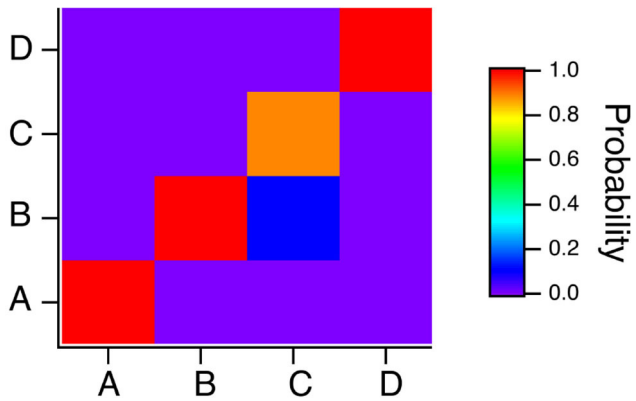

FIGURE 13 | Confusion matrix for identifying the four Sprott systems A,B,C and D with the polynomial ODE reservoir computer when $p_1 = -0.5$, the time constant $T_0 = 0.3$ and the spectral radius $\sigma = 0.8$. Along the x axis are the actual Sprott systems, while the y axis shows the probability that the Sprott system will be identified as that particular system.

where ψ is the digamma function, N is the number of points in the multidimensional time series and $\langle\rangle$ indicates a mean. The distances ε_n are normalized by the standard deviation of $\mathbf{R}(r)$, and the number of neighbors was $k = 10$.

Comparing with **Figure 10**, larger values of the nearest neighbor entropy correspond to smaller values of the classification error E_C. **Figure 12** shows that different methods of calculating entropy are useful for minimizing the classification error.

8. CONCLUSIONS

The classification results for the polynomial ODE reservoir computer are summarized in the confusion matrix of **Figure 13** for which $p_1 = -0.5$, the time constant $\alpha = 0.3$ and the spectral radius $\sigma = 0.8$. Systems A and B are identified without error, but

the reservoir computer has trouble distinguishing systems B and C. **Figure 7** showed that the spiking reservoir computer also had trouble with these same two systems.

While it may not be practical to use a full nonlinear optimization to optimize the reservoir computer parameters for classifying time series signals, sweeping the parameters one at a time yields useful insights. It was noted for both a spiking reservoir computer and a reservoir computer based on a polynomial ordinary differential equation that the best classification performance coincides with the largest entropy for the reservoir computer. This result is not unexpected, but it is useful to confirm it using a simple method to determine entropy. The best classification performance did not always occur at the best parameters for signal reproduction.

The entropy concept used here appears to be useful, but Kanders et al. (2017) studied different measures of complexity in a spiking neural network and found that the most complex (or critical) states did not always occur when the network Lyapunov exponent was largest, so the principles found here for getting the best performance from a reservoir computer may not hold in all cases.

These results open up the possibility of optimizing a reservoir computer for signal classification by maximizing the entropy of the reservoir computer based on a single realization of the signal to be classified. Optimizing based on the classification error would require many, possibly hundreds of realizations of the signals to be classified, requiring considerable computation.

It was shown in Carroll (2020a,b) that increasing the interaction between reservoir computer nodes can increase the entropy. If the interaction is too strong, the reservoir computer may become chaotic or unstable, in which case the reservoir computer signals may increase without bound. Carroll (2021b) showed that instability may be avoided by adjusting other reservoir computer parameters so that the largest Lyapunov of the reservoir computer remains negative.

Other methods of parameter optimization in reservoir computers, such as Yperman and Becker (2017), will also work as long as the optimization criteria used is the entropy.

Neither of the reservoir computers in this study were designed with biological relevance in mind, but the fact that the requirements for the best classification performance in both types of reservoir computer were similar means that biological systems may also work best for classifying inputs when their entropy is maximized.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author/s.

AUTHOR CONTRIBUTIONS

The author confirms being the sole contributor of this work and has approved it for publication.

FUNDING

This work was supported by the Naval Research Laboratory's basic research program.

REFERENCES

Antonik, P., Gulina, M., Pauwels, J., and Massar, S. (2018). Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and cryptography. *Phys. Rev. E* 98, 012215. doi: 10.1103/PhysRevE.98.012215

Appeltant, L., Soriano, M. C., der Sande, G. V., Danckaert, J., Massar, S., Dambre, J., et al. (2011). Information processing using a single dynamical node as complex system. *Nat. Commun.* 2:468. doi: 10.1038/ncomms1476

Badii, R., Broggi, G., Derighetti, B., Ravani, M., Ciliberto, S., Politi, A., et al. (1988). Dimension increase in filtered chaotic signals. *Phys. Rev. Lett.* 60, 979–982. doi: 10.1103/PhysRevLett.60.979

Bandt, C., and Pompe, B. (2002). Permutation entropy: a natural complexity measure for time series. *Phys. Rev. Lett.* 88, 174102. doi: 10.1103/PhysRevLett.88.174102

Canaday, D., Griffith, A., and Gauthier, D. J. (2018). Rapid time series prediction with a hardware-based reservoir computer. *Chaos* 28, 123119. doi: 10.1063/1.5048199

Carroll, T. L. (2020a). Do reservoir computers work best at the edge of chaos? *Chaos* 30, 121109. doi: 10.1063/5.0038163

Carroll, T. L. (2020b). Path length statistics in reservoir computers. *Chaos* 30, 083130. doi: 10.1063/5.0014643

Carroll, T. L. (2021a). Adding filters to improve reservoir computer performance. *Physica D Nonlinear Phenomena* 416:132798. doi: 10.1016/j.physd.2020.132798

Carroll, T. L. (2021b). Low dimensional manifolds in reservoir computers. *Chaos* 31, 043113. doi: 10.1063/5.0047006

Carroll, T. L., and Pecora, L. M. (2019). Network structure effects in reservoir computers. *Chaos* 29, 083130. doi: 10.1063/1.5097686

Crutchfield, J. P., and Young, K. (1990). "Computation at the onset of chaos," in *Complexity, Entropy, and the Physics of Information*, ed W. H. Zurek (TRedwood City, CA: Addison-Wesley), 223–269.

der Sande, G. V., Brunner, D., and Soriano, M. C. (2017). Advances in photonic reservoir computing. *Nanophotonics* 6, 561–576. doi: 10.1515/nanoph-2016-0132

Dion, G., Mejaouri, S., and Sylvestre, J. (2018). Reservoir computing with a single delay-coupled non-linear mechanical oscillator. *J. Appl. Phys.* 124, 152132. doi: 10.1063/1.5038038

Hart, A., Hook, J., and Dawes, J. (2020). Embedding and approximation theorems for echo state networks. *Neural Netw.* 128, 234–247. doi: 10.1016/j.neunet.2020.05.013

Herteux, J., and Rath, C. (2020). Breaking symmetries of the reservoir equations in echo state networks. *Chaos* 30, 123142. doi: 10.1063/5.0028993

Jaeger, H. (2001). *The Echo State Approach to Analysing and Training Recurrent Neural Networks-With an Erratum Note.* German National Research Center for Information Technology GMD Technical Report 148, 34.

Jaeger, H., and 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., Demuynck, K., Neve, W. D., and Martens, J.-P. (2018). On the application of reservoir computing networks for noisy image recognition. *Neurocomputing* 277, 237–248. doi: 10.1016/j.neucom.2016.11.100

Kanders, K., Lorimer, T., and Stoop, R. (2017). Avalanche and edge-of-chaos criticality do not necessarily co-occur in neural networks. *Chaos* 27, 047408. doi: 10.1063/1.4978998

Kraskov, A., Stögbauer, H., and Grassberger, P. (2004). Estimating mutual information. *Phys. Rev. E* 69, 066138. doi: 10.1103/PhysRevE.69.066138

Langton, C. G. (1990). Computation at the edge of chaos: phase transitions and emergent computation. *Physica D Nonlinear Phenomena* 42, 12–37. doi: 10.1016/0167-2789(90)90064-V

Larger, L., Soriano, M. C., Brunner, D., Appeltant, L., Gutierrez, J. M., Pesquera, L., et al. (2012). Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing. *Optics Express* 20, 3241–3249. doi: 10.1364/OE.20.003241

Lu, Z., Hunt, B. R., and Ott, E. (2018). Attractor reconstruction by machine learning. *Chaos* 28, 061104. doi: 10.1063/1.5039508

Lu, Z., Pathak, J., Hunt, B., Girvan, M., Brockett, R., and Ott, E. (2017). Reservoir observers: model-free inference of unmeasured variables in chaotic systems. *Chaos* 27, 041102. doi: 10.1063/1.4979665

Lukoševičius, Jaeger, H., and Schrauwen, B. (2012). Reservoir computing trends. *Kunstliche Intell.* 26, 365–371. doi: 10.1007/s13218-012-0204-5

Lymburn, T., Walker, D. M., Small, M., and Jungling, T. (2019). The reservoir's perspective on generalized synchronization. *Chaos* 29, 093133. doi: 10.1063/1.5120733

Natschlaeger, T., Maass, W., and Markram, H. (2002). The "liquid computer": a novel strategy for real-time computing on time series. *Special Issue Foundat. Inform. Proc. Telematik* 8, 39–43.

Parker, T. S., and Chua, L. O. (1989). *Practical Numerical Algorithms for Chaotic Systems*. New York, NY: Springer-Verlag.

Penrose, R. (1955). A generalized inverse for matrices. *Math. Proc. Cambridge Philos. Soc.* 51, 406–413. doi: 10.1017/S0305004100030401

Schurmann, F., Meier, K., and Schemmel, J. (2004). "Edge of chaos computation in mixed-mode vlsi-a hard liquid," in *Advances in Neural Information Processing Systems 17* (MIT Press), 1201–1208.

Sprott, J. C. (1994). Some simple chaotic flows. *Phys. Rev. E. Stat. Phys. Plasmas Fluids Relat. Interdiscip Top.* 50, R647–R650. doi: 10.1103/PhysRevE.50.R647

Stoop, R., Saase, V., Wagner, C., Stoop, B., and Stoop, R. (2013). Beyond scale-free small-world networks: Cortical columns for quick brains. *Phys. Rev. Lett.* 110, 108105. doi: 10.1103/PhysRevLett.110.108105

Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., et al. (2019). Recent advances in physical reservoir computing: a review. *Neural Netw.* 115, 100–123. doi: 10.1016/j.neunet.2019.03.005

Tikhonov, A. N. (1943). On the stability of inverse problems. *Comptes Rendus de l'Acadèmie des sci. de l'URSS* 39, 176–179.

Xiong, W., Faes, L., and Ivanov, P. C. (2017). Entropy measures, entropy estimators, and their performance in quantifying complex dynamics: effects of artifacts, nonstationarity, and long-range correlations. *Phys. Rev. E* 95, 062114. doi: 10.1103/PhysRevE.95.062114

Xu, Y., Ma, Q. D. Y., Schmitt, D. T., Bernaola-Galv' an, P., and Ivanov, P. C. (2011). Effects of coarse-graining on the scaling behavior of long-range correlated and anti-correlated signals. *Physica A* 390, 4057–4072. doi: 10.1016/j.physa.2011.05.015

Yperman, J., and Becker, T. (2017). Bayesian optimization of hyper-parameters in reservoir computing. *arXiv*. 1611.05193v3.

Zimmermann, R. S., and Parlitz, U. (2018). Observing spatio-temporal dynamics of excitable media using reservoir computing. *Chaos* 28, 043118. doi: 10.1063/1.5022276

Conflict of Interest: The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.