Graph Reservoir Networks for Prediction of Spatiotemporal Systems

G. William Chapman*, J. Darby Smith*, Corinne Teeter*, and Nicole D. Jackson*
* Sandia National Laboratories Albuquerque, NM, USA; Email: gwchapm@sandia.gov

2026 Neuro Inspired Computational Elements (NICE) | 979-8-3315-5158-2/26/$31.00 ©2026 IEEE | DOI: 10.1109/NICE69539.2026.11567497

***Abstract*—Predicting large-scale spatiotemporal data poses significant challenges for traditional machine learning methods, particularly in high-performance and edge-based computing environments. Graph neural networks (GNNs) have emerged as powerful tools for modeling structured data, while reservoir computing (RC) offers efficient and state-of-the-art solutions for predicting low-rank dynamical systems. In this work, we propose a novel approach that integrates the global structured modeling capabilities of GNNs with the efficient training and predictive strengths of RC. By combining local random connectivity with global domain-informed structure, the proposed graph reservoir networks leverage both random and model-based information to accurately predict large-scale spatiotemporal dynamics. Furthermore, we demonstrate the robustness of these networks to neuromorphic hardware constraints, including temporally sparse spiking activity, quantized weights, limited fan-in, and local learning rules. These results highlight the utility of integrating reservoir computing with domain-informed inductive biases, paving the way for scalable and efficient solutions to complex dynamical systems in diverse applications.**

***Index Terms*—Reservoir Computing, Spiking Neural Networks, Graph Neural Networks, Neuromorphic Computing, Timeseries prediction**

I. Introduction

Spatiotemporal forecasting is critical for applications requiring highly accurate predictions of how a system's state evolves across both space and time. Examples include numerical weather prediction [1], earth system modeling [2], laser plasma manufacturing [3], modeling the spread of infectious or invasive species [4], and neural population dynamics in the brain [5]. These applications often involve complex, high-dimensional systems that challenge traditional machine learning methods, particularly in scenarios demanding scalability and efficiency.

Approaches to modeling spatiotemporal systems span a continuum between model-based (structured) and model-free (unstructured) methods, each offering distinct advantages and tradeoffs. Model-based methods incorporate domain knowledge, such as spatial dependencies, temporal lags, or model order, into mathematical frameworks that are then fit to observed data. In contrast, model-free methods minimize reliance on domain knowledge, instead employing unstructured approaches to extract patterns directly from data. Neural networks provide a flexible framework for integrating domain knowledge into their architecture, enabling inductive biases such as spatial invariance (e.g., convolutional neural networks) or causal temporal processing (e.g., recurrent neural networks or transformers), while allowing data-driven feature extraction which follow these inductive biases.

While inductive biases have proven effective in many domains, their utility in time series prediction, particularly for chaotic systems, has been limited. This limitation arises primarily from challenges in credit assignment for chaotic dynamics, where the relationship between inputs and outputs is highly nonlinear and sensitive to initial conditions. Reservoir computing (RC), a model-free approach, has demonstrated significant success in reconstructing chaotic dynamics [6]. RC employs a high-dimensional nonlinear embedding combined with a linear readout to approximate system behavior from observed data. However, RC faces scalability challenges for large systems, as the reservoir size must grow quadratically with the input space to ensure sufficient embedding capacity [7]. Additionally, the random connectivity within reservoirs often mixes unrelated variables, reducing predictive accuracy for structured data [8].

Recent research has explored modifications to RC that incorporate minimal domain knowledge to improve scalability and predictive accuracy for large-scale systems, such as those governed by partial differential equations [9]. For example, spatially embedded reservoirs have demonstrated reduced training time and improved performance by leveraging multiple spatially dependent reservoirs [10]. Similarly, spatially dependent reservoir connectivity has shown promise for tasks such as climate data prediction [11] and image classification [12], [13]. These approaches suggest that integrating structured domain knowledge into RC can enhance its scalability and applicability to complex systems.

Spiking neural network (SNN) equivalents of RC, known as liquid state machines (LSMs), address some of these limitations by combining random local connectivity with spiking communication and local training rules. LSMs replicate key features of biological neural systems, such as balanced excitation-inhibition, multitasking, and efficient training [12]–[14]. Their asynchronous spiking activity enables rich temporal dynamics that can be reused across multiple tasks, drawing parallels to the granular neurons of mammalian cortex [15] and the supervised learning supported by feedback to pyramidal neurons [16]. Furthermore, the spiking activity and local learning rules of LSMs makes them suitable candidates for deployment on emerging neuromorphic hardware.

In this work, we investigate how reservoir computing can be combined with domain-informed topological constraints to improve the performance of collections of reservoirs for

structured data. Specifically, we explore the integration of RC with graph neural networks (GNNs) to incorporate arbitrary domain-driven inductive biases, in the form of task-constrained graph.. By introducing graph reservoir computers (GRC), we demonstrate that combining RC with graph-based constraints improves predictive accuracy and scalability for structured spatiotemporal systems. We also show that spiking graph reservoir computers (SGRC) achieve comparable performance to GRC while leveraging spiking activity and globally sparse structured connectivity. Collectively, these results demonstrate that graph reservoirs combine the predictive capabilities on chaotic systems of unstructured reservoirs, while enabling efficient scaling, incorporation of domain-informed knowledge, and efficient deployment to hardware through locally random but globally structured and sparse communication.

II. Methods

We express both our tasks and networks in the form of a general graph network, such that *nodes* are dynamical systems which follow internal dynamics and are coupled to inputs and other nodes through edges. This general form can be expressed as:

$$x_{t+1}^{(k,i)} = \gamma^{(k,i)}\left(x_t^{(k,i)}, \bigoplus_{j \epsilon N(i)} \phi^{(k,i)}\left(x_t^{(k-1,i)}, x_t^{(k-1,j)}, e^{(j,i)}\right)\right), \tag{1}$$

where i is the *node index*, k is the *layer index*, and e_{ji} is the directed edge *weight*. $\phi^{(k)}$ is a general function which translates the activity of nodes from the previous layer to an input to the activation function γ. The $\bigoplus$ symbol is an aggregation term, which can be either summation or averaging. This equation therefore implements node updates as a function of a node dynamic (γ) which is a function of the previous state ($x_t^{(k,i)}$) and transformations of inputs. For brevity in later sections, we refer to the network update dynamics in terms of previous state and net transformed input:

$$x_{t+1}^{(k,i)} = \gamma\left(x_t^{(k,i)}, I_t^{(k,i)}\right). \tag{2}$$

A. Tasks

We chose three exemplar tasks that include complex dynamics within nodes, and incorporate inter-node coupling. The tasks are chosen in order to test capabilities on: 1) an arbitrary graph ode with ODE-like nodes; 2) a topologically constrained chaotic system; and, 3) a topologically constrained task which requires interactions of latent variables (see Table I and Figure 1). All tasks were generated with the simple forward Euler method. Following generation, tasks were zero-centered and normalized to $[-1, 1]$ on a per-feature basis, and re-sampled with linear interpolation such that their calculated Lyapunov time constant was 1,000 frames.

Lorenz-63: We include three variations of the Lorenz-63 chaotic attractor. The baseline 'single node' case, is a standard example of a chaotic system for which reservoir computing can outperform other approaches [17]. This is a single node with three internal variables which follow the standard equations given in Equation 3, where the input ($I_t^{k,i}$) is zero. The 'multi node' case extends this task by simulating multiple nodes with different initial conditions by the same underlying dynamics. This allows each node to evolve autonomously, while testing a networks' capability to follow multiple independent trajectories. The final case introduces arbitrary numbers of nodes which continue to follow the dynamics outlined above, but allow coupling between nodes. Previous work has investigated the dynamics and characteristics of symmetrically coupled sets of two Lorenz attractors [18], [19]; our work generalizes this approach to asymmetric cases with arbitrary topologies. We specifically investigated a 20-node scenario with a random directed graph with connection probability of 0.2. This case therefore tests the capability of a network to follow multiple trajectories which are sparsely coupled.

$$\begin{aligned} \frac{dx_1}{dt} &= \sigma(x_2 - x_1) + I_t^{k,i}, \\ \frac{dx_2}{dt} &= x_1(\rho - x_3) - x_2, \\ \frac{dx_3}{dt} &= x_1 x_2 - \beta x_3. \end{aligned} \tag{3}$$

TABLE I
Summary of structural and dynamic properties for each task in this study.

Task	Complexity	Structure	Latent Variables
Lorenz-63	Medium	Random directed Graph	No
Lorenz-96	Medium	Toroidal, nearest +/- 2	No
2D KS	High	2D grid	Yes

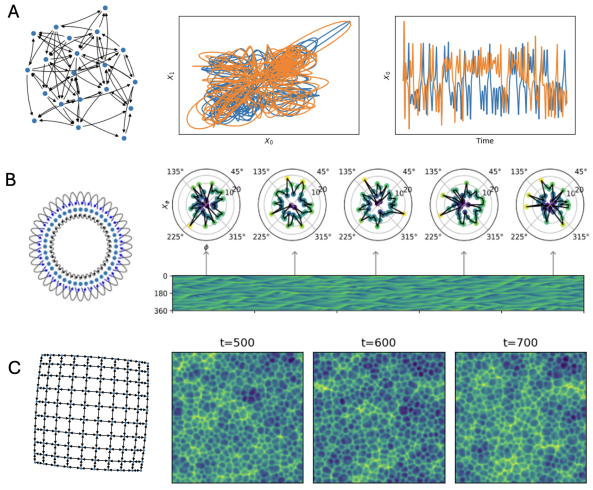

Fig. 1. Topological graph (left) and example behavior (right) for the three chosen tasks **A.** The coupled Lorenz-63 task, where each node follows the 3-variable ODE described in Equation 3, and is connected through a random graph with an average connectivity of 0.2. The first trajectory plots the first two spatial dimensions of two connected nodes. The second trajectory plots the first spatial component of the same two nodes. **B.** The Lorenz-96 task in which each node interacts with neighbors ± 2 locations on a toroid. The system is operationalized such that traveling waves of activity can be observed over time. **C.** The 2D Kuramoto-Sivashinsky (KS) task where updates at each location are governed by local gradients.

Lorenz 96: Our second exemplar task is the Lorenz 1996 [20] in which nodes are arranged on a Toroidal graph. Whereas the Lorenz-63 case allows arbitrary topologies, the Lorenz-96 enforces a nonlinear interaction with nodes at ± 2 locations, governed by Equation 4:

$$\begin{aligned} \frac{dx^i}{dt} &= -x^i + F + I_t^{k,i}, \\ I_t^{k,i} &= (x^{i+1} - x^{i-2})x^{i-1}. \end{aligned} \tag{4}$$

Kuramoto-Sivashinsky: The final exemplar task implements a 2-dimensional version of the Kuramoto-Sivashinsky [21], where a 2-dimensional Toroidal surface follows the dynamics:

$$\begin{aligned} \frac{dx}{dt} &= 0 + I_t^{k,i}, \\ I_t^{k,i} &= -\frac{1}{2}\left|\nabla x\right|^2 - \nabla^2 x - \nabla^4 x. \end{aligned} \tag{5}$$

While the Lorenz-63 and Lorenz-96 tasks contain an intra-node dynamic and inter-node connectivity, the Kuramoto-Sivashinsky has no explicit intra-node dynamics and evolves solely based on spatial partial derivatives. These derivatives however are not directly available from the input space x, and therefore represent a latent variable.

B. Network Dynamics

We investigated four network types which have been used for prediction of spatiotemporal systems: Reservoirs, Liquid State Machines, Gated Recurrent Units (GRU), and Nonlinear Vector Autoregression (NVAR). While GRUs have been investigated in combination with graph neural networks ('Graph Gated Recurrent Units'), reservoir, LSM, and NVAR networks have not been systematically combined with graph-like topologies, preventing the incorporation of graph-like priors.

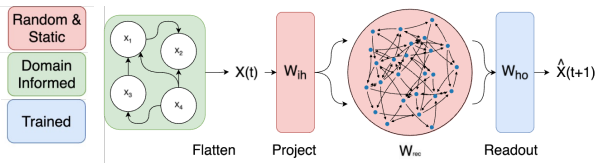

Fig. 2. In a standard reservoir, inputs to the reservoir must all come from a singular input weight matrix (W_{ih}) and the recurrent weights (W_{rec}) are uniformly sparse and random. This results in any structure in the input data being omitted when communicating the leaky-integrator units that compose the reservoir.

Figure 2 demonstrates the standard reservoir computing approach. For structured data, inputs must be 'flattened', removing and structure in the inputs, but accommodating a single input weight matrix (W_{ih}). Within the reservoir multiple 'units', for instance leaky-integrators, are connected by a sparse and spectrally-normalized random matrix (W_{hh}). The input and recurrent weights are randomly initialized according to distributions to instantiate desirable properties, such as echo-state behavior and high internal dimensionality, but do not contain trained parameters or domain information.

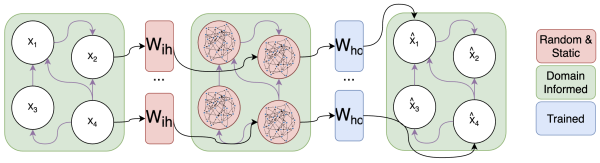

Fig. 3. In the graph reservoir approach, the hidden layer consists of multiple nodes which lie on a graph, and which *each* follow internal dynamics according to the recurrent weights. Input structure can be imposed by a topologically constrained input from the data to the appropriate reservoir nodes.

Only the readout matrix (W_{ho}) is trained, typically by ridge-regression or iterative least-squares.

Figure 3 demonstrates our proposed graph reservoir approach, which preserves domain knowledge in the form of edges between nodes in a task. To accommodate the structured inputs, we replace the single feedforward reservoir with a 'pool' of multiple reservoirs, each of which contains internal units and connectivity similar to the standard reservoir approach. Each individual reservoir therefore represents a dynamical system which occurs at each node on the task graph, and is projected into a high-dimensional random dynamical system. As with the standard reservoir approach, only the readout weights (W_{ho}) are trained.

Network dynamics follow the generic form of Equation 1, where the unit dynamics (γ) follow one of the Equations 6 through 8. All methods are implemented in PyTorch, and PyTorch Geometric. Unless otherwise stated, all initializations followed their default values in PyTorch [22] and PyTorch Geometric [23].

Network Configurations: In Equation 1, the term ϕ is a weight matrix from one node to another. For a given connection from layer $k-1$ to k, this weight may be initialized and fit separately for each edge, or it may be *convolutional*, such that each edge in a given layer is identical. For the tasks discussed above, where the underlying dynamics are known, the use of convolutional edges is an additional form of domain knowledge, and may increase network stability or minimize the number of timesteps required for training [24]. We investigate whether convolutional input, recurrent, and readout weights are beneficial in the context of graph reservoirs.

It is also possible to have connections *within* a single layer of the network (e.g., k to k), as indicated by the within pool edges in Figure 3. These lateral connections allow nodes within a reservoir to provide additional input to each other, following the same graph as the feedforward connections. The presence of lateral connections represents another form of domain knowledge, where if the observable space is thought to be exhaustive, then lateral connections are not incorporated; for tasks with latent variables (e.g., Equation 5), such connections can be incorporated.

Reservoir Dynamics: For reservoir networks, individual units followed leaky-integrator dynamics:

$$x_{t+1}^{(k,i)} = \alpha \tanh\left(I_t^{(k,i)}\right) + (1-\alpha)x_t^{(k,i)}, \tag{6}$$

where α is the leak-rate (0.15), and W_{rec} is initialized to achieve a desired sparsity (0.2) and then normalized to a given spectral radius (1.0) [7]. These initializations are chosen to ensure the echo-state property is conserved. For the experiments discussed here we utilize 500 units in reservoir nodes, with the exception of baseline reservoirs for which we utilize 500 units multiplied by the number of nodes in the input.

Spiking Dynamics: For spiking-based networks, individual units followed a simple current-based leaky-integrate and fire dynamics, as implemented in Norse [25]:

$$\begin{aligned} \tau_U \frac{du_j}{dt} &= -u_j + I_t^{(k,i)} \\ \tau_m \frac{dv_j}{dt} &= -v_j + U - Z_j(t-1) \\ Z_j(t) &= v_j(t) > 1 \end{aligned} \tag{7}$$

where U represents the unit-specific current which integrates inputs (I), v is the 'membrane potential', and Z represents the units-specific spiking activity, which is passed to readout and lateral connections. Except where stated specifically, the current time constant (τ_U) was held at 5 simulation timesteps, and the membrane time constant (τ_v) was held at 10 timesteps. Additional constraints were imposed on the connectivity of spiking reservoirs, to follow separation of excitatory and inhibitory units, with 80% of units acting as excitatory units and 20% acting as inhibitory units. For spiking networks, local recurrent connections (W_{rec}) were restricted such that excitatory unit outgoing weights were non-negative, and inhibitory weights were non-positive, but followed the same overall (20% non-zero) sparsity, and normalized to a spectral radius of 1.0. Consistent with physiological evidence for inter-column connectivity [26], only excitatory units were engaged with inputs, outputs, or lateral connections between reservoirs. Note that we refer to networks following these dynamics as spiking reservoirs, rather than 'liquid state machines' as we follow general echo-state network initializations rather than more biologically inspired dynamics [27].

Nonlinear Vector Autoregression: Nonlinear vector autoregressive (NVAR) networks replace the random connectivity matrices of reservoir networks with an explicitly defined nonlinear combination of predictor variables [24]. Such networks have been shown to benefit from inductive biases by decreasing the number of training steps required to reach the same performance as independent or single large nodes [28]. By creating a lagged vector of input histories and element-wise multiplication of the resulting vectors, the networks are able to extract interactions and delays of variables, while avoiding the explicit temporal evolution of reservoir networks. The lagged vector and resultant multiplication vector can be written as

$$\begin{aligned} X(t) &= [x(t), x(t-1), ...x(t-n)], \\ Z(t) &= [x(t), X(t) \otimes X(t)], \end{aligned} \tag{8}$$

where $\otimes$ indicates the outer product of the lagged input vector. While these networks can show performance on par with echo-state configurations, they require an explicit knowledge of the relevant temporal lag and degree of nonlinear interactions. We utilized lags of up to 5 timesteps and second order interactions for this study's experiments. An additional notable example of limitations in NVAR networks is that latent interactions between spatial locations can not be modeled. Interactions in the input space (x) can be modeled by expanding the vector $X(t)$ at each spatial location to include connected input nodes (Figure 3). However there is no analog of the lateral connections between reservoirs in Figure 3. Additionally, as the size of Z will vary with the number of adjacent nodes in input space, each readout must be trained separately unless each node of the data graph has the exact same number of edges.

Gated Recurrent Unit: As a final comparison, we evaluate on Graph Gated Recurrent Units (GGRU), which utilize graph connectivity along with more standard backpropagation training. These networks follow the GRU equations:

$$\begin{aligned} r_t &= \sigma(W_{ir} x_t + W_{hr} h_{t-1} + b_r), \\ z_t &= \sigma(W_{iz} x_t + W_{hz} h_{t-1} + b_z), \\ \hat{n_t} &= \phi\left(W_{in} x_t + r_t \odot (W_{hn} h_{t-1} + b_h)\right), \\ h_t &= (1 - z_t) \odot n_t + z_t \odot h_{t-1}, \end{aligned} \tag{9}$$

where matrix-vector multiplications are replaced with graph convolutions [29].

C. Training

Reservoir-Based Networks: During training, a 500 timestep burn in period was provided, followed by a 1,000 timestep fitting period. During the fitting period the activity of reservoir units was recorded, and ridge regression ($\alpha = 1 \times 10^{-6}$) was applied to fit unit activity to target outputs. Autoregressive prediction then took place for 5,000 timesteps.

GRU-Based Networks: For the graph-GRU, readout from the GRU layers was performed with a single convolutional layer (graph convolution), using backpropagation through time on 2,000 timesteps. Loss was defined as the mean-square error on a time-step basis, and network weights were updated with the Adam [30] optimizer. Networks were trained for 1000 epochs, and evaluated on a separate validation dataset.

D. Metrics

For all combinations of network configuration and task, results are reported as the average of 10 independently initialized and trained networks. The primary metric of performance is mean square error (MSE), which is directly optimized for both reservoir and ANN-based networks. All tasks are normalized to [-1,1] on a per-feature basis, making all reported values equivalent to normalized root mean square error (NMRSE).

In addition to MSE, we examine the minimum state distance (MSD) as a complementary metric to evaluate how well a trained network approximates the underlying dynamics of a system. For a given timestep prediction, MSD is defined as the minimum Euclidean distance between the predicted state and *any* observation in the training dataset. While MSE measures the accuracy of a network's predictions at specific points in time, MSD captures how closely the network adheres to the

general dynamics of the system, even for chaotic systems where exact trajectory prediction is inherently challenging.

Using MSD, we define a derived metric called Time to Divergence (TD), which quantifies the number of timesteps until a network produces a prediction with a state distance greater than an arbitrary threshold, set to 0.1 in our experiments. TD provides an estimate of how long a network can produce valid state predictions before deviating significantly from the underlying dynamics. For example, a network that learns the attractors of the Lorenz-63 system but fails to match the true trajectory may exhibit high MSE due to lobe-switching errors, while maintaining low MSD and high TD, indicating that the network has captured the system's general structure.

While MSE is a widely used metric in machine learning literature, its limitations for chaotic systems are well-documented. The chaotic nature of the tasks studied here often results in high MSE values, even when the network successfully learns the underlying dynamics. In contrast, MSD remains stable over the prediction period, allowing TD to reach arbitrarily high values, which indicates that the network has learned the system's dynamics, even if trajectory-specific errors persist.

The distinction between these metrics has important real-world implications. MSE is a strong indicator of the accuracy and trustworthiness of a prediction, making it suitable for applications requiring precise state estimation. MSD and TD, on the other hand, are better suited for evaluating how well a model captures physical constraints and general system behavior, which is critical for applications such as digital twins or simulations of chaotic systems [31].

III. Results & Discussion

A. Performance

Network performance, as measured by mean squared error is summarized in Table III-A across all tasks.

Baselines: The standard reservoir results show that a single large reservoir is unable to capture the dynamics of either the multiple independent Lorenz-63 nodes or the coupled Lorenz-63 system. However, assigning multiple independent reservoirs to each independent node in the multi-Lorenz produces an MSE in line with expectations, while being unable to capture the dynamics of the coupled system. These results, while unsurprising, demonstrate that a set of uncoupled reservoirs is unable to match the dynamics of a coupled system, and require some information from the topologically connected nodes. The GGRU approach however is able to match all four tasks with an MSE less than 1, indicating that the domain informed neural network approach outperforms naive reservoirs, even when the dynamics of a single node are traditionally better performed by RC than ANN approaches. NVAR performed in general slightly better than standard reservoir approaches, but the quadratic growth in memory as a function of input variables prevented fitting for the high dimensional KS task.

Graph Reservoirs: Across the four tasks, graph reservoir approaches perform better than, or as well as, baseline approaches. In all cases convolutional approaches outperform linearly independent configurations (e.g., 'linear' versus 'convolutional'). The introduction of lateral connections between reservoir nodes only improved performance in the Kuramoto-Sivashinsky task. This can be explained by the presence of the spatial derivatives in Equation 5, which are not present in the observable space, but can be extracted in the interactions within reservoirs. Spiking graph reservoirs had similar performance to the equivalently structured graph reservoirs across all tasks.

Time to Divergence: Table III-A summarizes time to divergence across all configurations. While the overall trends of the MSE results hold here, it is notable that for many of the graph-based approaches, the networks did not diverge from a valid state prediction within the 5000 timesteps (5 Lyapunov time constant) prediction period. This suggests that the domain-informed topology is able to counteract erroneous interactions which may otherwise lead to learning incorrect dynamics.

B. Scaling and Computational Efficiency

Beyond numerical accuracy, the graph reservoirs presented here have the potential to enable efficient scaling to large datasets, due to their combination of relatively dense and random local connectivity with globally sparse and domain informed topology. Additionally, the spiking graph reservoirs utilize local membrane potentials within individual units, while communicating within-and-between nodes by temporally sparse spiking activity, making them potential candidates for deployment to neuromorphic hardware, with additional modifications outlined below. Experiments presented below discuss run-time and memory constraints for spiking graph reservoirs, as evaluated on a single single NVIDIA A100 40-GB GPU. Reported results are performed only on the randomly coupled Lorenz-63 system, due to the ability to independently modulate node connectivity and number of nodes, while the other two systems have a fixed connectivity structure.

Run Time: Standard reservoirs increase approximately quadratically in the number of hidden units required to support an increase in input dimensionality. Similarly, graph-based RNNs increase approximately $\mathcal{O}(n \log n)$ in the number of hidden states, while growing quadratically in training time as a function of number of timesteps. In contrast, the number of units required *per node* for GRC is constant for a given node dimensionality, while the number of nodes increases directly with the size of the task graph. Both run-time and memory increase quadratically as a function the number of units, but linearly with the number of nodes (Figure 4). It is therefore desirable to have more nodes of smaller size than to have a single large reservoir. It is also worth noting that in our PyTorch-based framework, nodes are processed serially, while SNN-specific simulators [32] or neuromorphic hardware [33] process discrete neural populations in parallel, and could

TABLE II

PERFORMANCE ACROSS ALL TASKS AND NETWORK CONFIGURATIONS, AS MEASURED BY MEAN SQUARED ERROR DURING THE PREDICTION PERIOD. BEST PERFORMING NETWORKS, MEASURED BY MSE, AND HIGHLIGHTED IN YELLOW. NETWORKS WITH MSE OUTSIDE OF THE UNIT CIRCLE ARE INDICATED ONLY WITH MSE >1. N/A INDICATES THAT THE NETWORK WAS UNABLE TO RUN, DUE TO PROHIBITIVE MEMORY REQUIREMENTS.

Network Type	Configuration	Multi-Lorenz63	Coupled Lorenz63	Lorenz96	KS
Standard Reservoir	Single	>1	>1	0.24	>1
	Multi-Reservoir	0.35	>1	**0.04**	>1
NVAR	Single	0.73	>1	0.22	N/A
	Multi-NVAR	0.33	0.27	0.06	>1
Graph NN	GRU	0.39	0.65	0.32	0.83
Graph Reservoir	Linear	0.35	0.32	0.06	>1
	Linear + Lateral	0.42	0.7	0.20	>1
	Convolutional	**0.31**	**0.25**	0.05	0.73
	Convolutional + Lateral	0.42	0.46	0.13	**0.59**
Spiking Graph Reservoir	Linear	0.37	0.31	0.06	>1
	Linear + Lateral	0.43	0.75	0.23	>1
	Convolutional	0.37	0.26	0.06	0.81
	Convolutional + Lateral	0.43	0.49	0.17	0.64

TABLE III

PERFORMANCE ACROSS ALL TASKS AND NETWORK CONFIGURATIONS, AS MEASURED BY TIME TO DIVERGENCE. ASTERISKS INDICATE THAT THE PREDICTION DID NOT DEVIATE FROM A GROUND-TRUTH STATE WITHIN 5,000 TIMESTEPS.

Network Type	Configuration	Multi-Lorenz63	Coupled Lorenz63	Lorenz96	KS
Standard Reservoir	Single	703	10	1047	3
	Multi-Reservoir	*	93	*	209
NVAR	Single	209	104	1526	N/A
	Multi-NVAR	*	*	2032	17
Graph NN	GRU	*	*	*	*
Graph Reservoir	Linear	*	3297	*	942
	Linear + Lateral	*	2260	*	27
	Convolutional	*	*	*	3874
	Convolutional + Lateral	*	*	*	*
Spiking Graph Reservoir	Linear	*	2926	*	602
	Linear + Lateral	*	1962	*	32
	Convolutional	*	*	*	*
	Convolutional + Lateral	*	*	*	*

therefore run in constant time as a function of number of nodes.

Quantization: Increasingly, the size of machine learning models, along with their robustness to quantization compared to traditional numerical approaches, has resulted in the development of HPC and edge-based accelerators which utilize low-precision in-memory computing rather than floating point operations. This includes neuromorphic platforms such as Loihi2 [34], which operates on 8-bit weights while communicating in either binary or graded spikes. While reservoir computers utilize on a distributed neural population similar to many deep learning models, they also maintain a real-valued membrane potential which represents nonlinear mixing of input variables across various lags, and may therefore be more sensitive to numerical precision. To asses this, we implemented both GRC and SGRC using 8-bit weights, sub-threshold potentials, and 8-bit activations (1-bit for SGRC). For GRC, we found that across network configurations, mean-square error on the Coupled Lorenz-63 task was within 10% of the values (Table III-A). For SGRC, the difference was smaller, with all MSE within 3% of the floating-point performance.

Excitation-Inhibition Balance: SNN and neuromorphic platforms often utilize an event-driven communication frame-

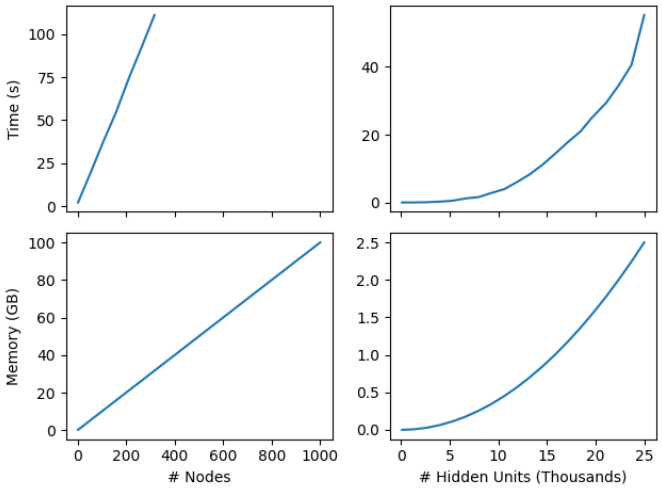

Fig. 4. Run-time (top) and memory (bottom) as a function of the number of nodes (left) and number of hidden units per node in a spiking reservoir as described in section II.B. Both run-time and memory increase linearly as a function of the number of nodes, and quadratically as a function of the number of units per node.

work between populations of neurons [35], rather than dense tensor communication. As a result, such platforms are most efficient when the number of connections between neural populations, and the number of events between populations, can be minimized [36]. The SGRC firing rate, with 20% inhibitory neurons as described in the methods, results in a mean-firing probability of 0.17, indicating that approximately one-fifth of neurons fire on a given timestep, which is significantly higher than the approximately 5% activity of liquid-state networks [14]. However, we observed that when nodes in our graph approach were configured with LSM lateral connections the resulting spiking patterns were irregular and resulted in poor performance. To mitigate the communication bottleneck associated with high inter-node transmission, we varied the percentage of inhibitory neurons from 20% to 80%, and observed a negative relationship between inhibitory population size and overall firing rate (evaluated by OLS, $\beta = -0.183$ change in mean firing probability per fraction inhibitory, $p<0.001$), while mean-square error was within 10% of baseline for inhibitory populations up to 50%, corresponding to a mean firing probability of 11.5%. This demonstrates that E-I balance can be tuned to reach an appropriate balance of accuracy and inter-node communication.

Fan-In & Fan-Out: Similar to the overall rate of inter-node spikes, the number of units involved in inter-node lateral connections can increase the communication overhead of event-based networks. While the SGRC maintains consistent within-node connectivity, inter-node connectivity is controlled by the average number of units involved in internode connectivity and the number of edges per node. To test performance under constrained connectivity, we evaluated the SGRC with linear and convolutional lateral connections constrained to a randomly selected set of units in 100-unit increments. For linear lateral connections performance was MSE was less than 10% higher for 2000 or more units, and convolutional connections showed similar performance for as few as 800 units.

Local Online Learning: Ridge regression relies on a large buffer of activity histories and is access to linear general linear algebra solvers, making it incompatible with a online learning in neuromorphic hardware. Other iterative learning rules, such as recursive least squares (RLS), rely only on instantaneous activity of reservoirs units and target outputs, and have been demonstrated in both continuous [17] and spiking [14] approaches. We tested both GRC and SGRC configurations with RLS, and found no systematic differences in MSE compared to ridge-regression.

IV. Conclusions

This work demonstrates that domain-informed structure can be combined with the random computation of reservoir computers to enable long-term, realistic prediction of highly complex spatiotemporal data. Furthermore, while these networks eventually diverge from ground-truth trajectories, the incorrect predictions still lie near a valid prediction, suggesting that their mode of failure is more realistic than a single large reservoir. The results however indicate that there is no single configuration of task-informed structure which results in the best performance, and that instead networks must be designed to incorporate inductive biases which are appropriate for the task. Overall however, the graph reservoir approach has the potential to combine long-term prediction capabilities of reservoirs to outperform alternative graph recurrent neural network approaches.

While the current results are promising, they are currently limited to static graphs which are defined a-priori. Standard graph neural networks, such as the GGRU, support automated methods for determining the underlying graph, but which rely on learning through backpropagation, and are therefore incompatible with the graph reservoir approaches used here. Future work may investigate how automated edge discovery can be implemented in graph reservoirs, for instance through evolutionary approaches [37], enabled by the fast training periods for GRC compared to GGRUs.

The numerical accuracy, robustness to hardware constraints, and scaling properties of spiking graph reservoirs indicate that they may be ideal candidates for solving large systems of graph-like data, such as spatially-embedded PDEs [9], climate data [38], or biological simulations [39], [40] in both neuromorphic hardware and HPC.

Acknowledgment

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. The authors own all right, title and interest in and to the article and is solely responsible for its contents. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this article or allow others to do so, for United States Government purposes. The DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan https://www.energy.gov/downloads/doe-public-access-plan. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

References

[1] P. Bauer, A. Thorpe, and G. Brunet, "The Quiet Revolution of Numerical Weather Prediction," *Nature*, vol. 525, no. 7567, pp. 47–55, 2015.

[2] M. Yu, Q. Huang, and Z. Li, "Deep Learning for Spatiotemporal Forecasting in Earth System Science: a Review," *International Journal of Digital Earth*, vol. 17, no. 1, p. 2391952, 2024.

[3] T. Wu, T. Wu, S. Zhou, Q. Liao, and P. Lu, "Deep Learning-based Spatiotemporal Sequence Forecasting of Physical Fields in Tin Droplet Laser-produced Plasma," *Plasma Science and Technology*, 2025.

[4] S. Ganesan and D. Subramani, "Spatio-temporal Predictive Modeling Framework for Infectious Disease Spread," *Scientific Reports*, vol. 11, no. 1, p. 6741, 2021.

[5] T. Le and E. Shlizerman, "Stndt: Modeling neural population activity with spatiotemporal transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 17 926–17 939, 2022.

[6] M. Lukoševičius and H. Jaeger, "Reservoir Computing Approaches to Recurrent Neural Network Training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, Aug. 2009.

[7] M. Cucchi, S. Abreu, G. Ciccone, D. Brunner, and H. Kleemann, "Hands-on Reservoir Computing: A Tutorial for Practical Implementation," *Neuromorphic Computing and Engineering*, vol. 2, no. 3, p. 032002, 2022.

[8] A. Hart, J. Hook, and J. Dawes, "Embedding and Approximation Theorems for Echo State Networks," *Neural Networks*, vol. 128, pp. 234–247, Aug. 2020.

[9] M. Yan, C. Huang, P. Bienstman, P. Tino, W. Lin, and J. Sun, "Emerging opportunities and challenges for the future of reservoir computing," *Nature Communications*, vol. 15, no. 1, p. 2056, Mar. 2024.

[10] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach," *Physical review letters*, vol. 120, no. 2, p. 024102, 2018.

[11] T. Arcomano, I. Szunyogh, J. Pathak, A. Wikner, B. R. Hunt, and E. Ott, "A Machine Learning-Based Global Atmospheric Forecast Model," *Geophysical Research Letters*, vol. 47, no. 9, p. e2020GL087776, May 2020.

[12] A. Biswas, S. Ashok Medhe, R. Singhal, and U. Ganguly, "Temporal and Spatial Reservoir Ensembling Techniques for Liquid State Machines," in *2024 International Conference on Neuromorphic Systems (ICONS)*, Jul. 2024, pp. 78–81.

[13] N. Soures and D. Kudithipudi, "Deep Liquid State Machines with Neural Plasticity for Video Activity Recognition," *Frontiers in neuroscience*, vol. 13, p. 686, 2019.

[14] W. Nicola and C. Clopath, "Supervised Learning in Spiking Neural Networks with FORCE Training," *Nature Communications*, vol. 8, no. 1, 2017.

[15] W. Maass, T. Natschläger, and H. Markram, "Real-Time Computing without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[16] M. E. Larkum, J. Wu, S. A. Duverdin, and A. Gidon, "The Guide to Dendritic Spikes of the Mammalian Cortex in Vitro and in Vivo," *Neuroscience*, p. S030645222200063X, Feb. 2022.

[17] D. Sussillo and L. F. Abbott, "Generating Coherent Patterns of Activity from Chaotic Neural Networks," *Neuron*, vol. 63, no. 4, pp. 544–557, Aug. 2009.

[18] H.-C. Ma, C.-C. Chen, and B.-W. Chen, "Dynamics and Transitions of the Coupled Lorenz System," *Physical Review E*, vol. 56, no. 2, p. 1550, 1997.

[19] M. Camplani, B. Cannas, and P. Carboni, "Dynamic Behaviour of Two Coupled Lorenz Systems," *IFAC Proceedings Volumes*, vol. 42, no. 7, pp. 62–66, 2009.

[20] E. N. Lorenz, "Predictability: A problem partly solved," in *Proc. Seminar on Predictability*, vol. 1. Reading, 1996, pp. 1–18.

[21] A. Kalogirou, E. E. Keaveny, and D. T. Papageorgiou, "An in-depth numerical study of the two-dimensional Kuramoto–Sivashinsky equation," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 471, no. 2179, p. 20140932, Jul. 2015.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Dec. 2019.

[23] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," Apr. 2019.

[24] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. Barbosa, "Next Generation Reservoir Computing," *Nature Communications*, vol. 12, no. 5564, 2021.

[25] C. Pehle and J. E. Pedersen, "Norse - A Deep Learning Library for Spiking Neural Networks," Jan. 2021.

[26] S. Haeusler and W. Maass, "A Statistical Analysis of Information-Processing Properties of Lamina-Specific Cortical Microcircuit Models," *Cerebral Cortex*, vol. 17, no. 1, pp. 149–162, 2007.

[27] J. Kaiser, R. Stal, A. Subramoney, A. Roennau, and R. Dillmann, "Scaling up Liquid State Machines to Predict over Address Events from Dynamic Vision Sensors," *Bioinspiration & biomimetics*, vol. 12, no. 5, p. 055001, 2017.

[28] W. A. Barbosa and D. J. Gauthier, "Learning Spatiotemporal Chaos Using Next-Generation Reservoir Computing," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, no. 9, 2022.

[29] D. Buterez, J. P. Janet, S. J. Kiddle, D. Oglic, and P. Liò, "Graph Neural Networks with Adaptive Readouts," Nov. 2022.

[30] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017.

[31] L.-W. Kong, Y. Weng, B. Glaz, M. Haile, and Y.-C. Lai, "Reservoir Computing as Digital Twins for Nonlinear Dynamical Systems," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 33, no. 3, 2023.

[32] F. Wang, S. Kulkarni, B. Theilman, F. Rothganger, C. Schuman, S.-H. Lim, and J. B. Aimone, "Scaling neural simulations in STACS," *Neuromorphic Computing and Engineering*, vol. 4, no. 2, p. 024002, Jun. 2024.

[33] H. A. Gonzalez, J. Huang, F. Kelber, K. K. Nazeer, T. Langer, C. Liu, M. Lohrmann, A. Rostami, M. Schöne, B. Vogginger, T. C. Wunderlich, Y. Yan, M. Akl, and C. Mayr, "SpiNNaker2: A Large-Scale Neuromorphic System for Event-Based and Asynchronous Machine Learning," Jan. 2024.

[34] G. Orchard, E. P. Frady, D. B. D. Rubin, S. Sanborn, S. B. Shrestha, F. T. Sommer, and M. Davies, "Efficient Neuromorphic Signal Processing with Loihi 2," in *2021 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2021, pp. 254–259.

[35] A. Patino-Saucedo, H. Rostro-Gonzalez, T. Serrano-Gotarredona, and B. Linares-Barranco, "Event-Driven Implementation of Deep Spiking Convolutional Neural Networks for Supervised Classification Using the SpiNNaker Neuromorphic Platform," *Neural Networks*, vol. 121, pp. 319–328, 2020.

[36] W. Severa, F. Wang, Y. Ho, F. Rothganger, A. Daram, and E. Gonzalez, "Benchmarking Spiking Network Partitioning Methods on Loihi 2," in *Proceedings of the Great Lakes Symposium on VLSI 2025*. New Orleans LA USA: ACM, Jun. 2025, pp. 898–904.

[37] C. D. Schuman, T. E. Potok, S. Young, R. Patton, G. Perdue, G. Chakma, A. Wyer, and G. S. Rose, "Neuromorphic Computing for Temporal Scientific Data Classification," in *Proceedings of the Neuromorphic Computing Symposium*, 2017, pp. 1–6.

[38] K. Goode, D. Ries, and K. McClernon, "Characterizing climate pathways using feature importance on echo state networks," Oct. 2023.

[39] L. E. Suárez, A. Mihalik, F. Milisav, K. Marshall, M. Li, P. E. Vértes, G. Lajoie, and B. Misic, "Connectome-Based Reservoir Computing with the Conn2res Toolbox," *Nature Communications*, vol. 15, no. 1, p. 656, Jan. 2024.

[40] F. Wang, B. H. Theilman, F. Rothganger, W. Severa, C. M. Vineyard, and J. B. Aimone, "Neuromorphic Simulation of Drosophila Melanogaster Brain Connectome on Loihi 2," Aug. 2025.