

Node pruning reveals substructures

Node pruning reveals compact and optimal substructures within large networks

Manish Yadav^{1, a)} and Merten Stender¹

Chair of Cyber-Physical Systems in Mechanical Engineering, Technische Universität Berlin, Straße des 17. Juni 135, 10623 Berlin, Germany

(*Electronic mail: merten.stender@tu-berlin.de)

(*Electronic mail: manish.yadav@tu-berlin.de)

(Dated: 6 May 2025)

The structural complexity of reservoir networks poses a significant challenge, often leading to excessive computational costs and suboptimal performance. In this study, we introduce a systematic, task-specific node pruning framework that enhances both the efficiency and adaptability of reservoir networks. By identifying and eliminating redundant nodes, we demonstrate that large networks can be compressed while preserving—or even improving—performance on key computational tasks. Our findings reveal the emergence of optimal subnetwork structures from larger Erdős-Rényi random networks, indicating that efficiency is governed not merely by size but by topological organization. A detailed analysis of network structure at both global and node levels uncovers the role of density distributions, special-radius and asymmetric input-output node distributions, among other graph-theoretic measures that enhance the computational capacity of pruned compact networks. We show that pruning leads to non-uniform network refinements, where specific nodes and connectivity patterns become critical for information flow and memory retention. This work offers fundamental insights into how structural optimization influences reservoir dynamics, providing a pathway toward designing more efficient, scalable, and interpretable machine learning architectures.

Complex networks are fundamental to understanding interactions in diverse domains, from biological systems to technological infrastructures. Identifying compact and optimal substructures within these networks remains a key challenge in network science. In this study, we introduce a node pruning approach to systematically reveal critical substructures while preserving essential network properties. By applying our method to large Erdős-Rényi random networks, we demonstrate its effectiveness in uncovering underlying functional motifs and enhancing computational efficiency. These findings provide new insights into the structural organization of complex systems and offer practical implications for optimizing network-based processes.

I. INTRODUCTION

In natural and artificial systems, network efficiency often emerges not from maximizing size but from optimizing structure. Biological networks, from neural circuits to metabolic pathways, demonstrate that excessive connectivity can lead to inefficiencies, such as increased energy consumption, slower response times, and reduced adaptability. For instance, in brain development, synaptic overproduction is followed by extensive pruning, refining functional connectivity while improving cognitive efficiency¹. Similarly, metabolic networks in organisms optimize resource allocation by selectively eliminating redundant reactions, leading to more efficient bio-

chemical pathways². Even at the genetic level, regulatory networks undergo structural refinement to enhance stability and adaptability, with excessive complexity often linked to dysfunction rather than improved function³. Additionally, modular brain networks balance integration and segregation, allowing efficient information processing by reducing redundant connections while preserving critical functional pathways⁴.

How can one determine the optimal network size and structure to achieve a specific functionality, such as solving a predefined task? One approach is performance-dependent growth, where a small, minimally connected network is selectively expanded while improving task performance. This strategy, widely observed in biological development, has also been applied in artificial systems. For example, in neural network training, structured growth mechanisms allow networks to develop efficient architectures without unnecessary redundancy⁵. In reservoir computing, Yadav et al. (2025) demonstrated that recurrent networks can evolve through performance-driven expansion, selectively adding nodes and connections that improve task-specific processing while maintaining computational efficiency⁶. A similar strategy explores how artificial neural networks can evolve from small initial configurations toward minimal optimal structures that maximize efficiency for a given task⁷. These studies showcase that the formation of smaller yet efficient is possible by performance-dependently increasing the network size.

However, an equally important yet underexplored strategy is performance-dependent pruning—starting with a large, randomly connected network and systematically reducing its size while preserving or even enhancing functionality. Unlike growth-based approaches, which build complexity from simplicity, pruning focuses on eliminating unnecessary

^{a)}Corresponding Author

components from an initially overconnected system. This principle is evident across various domains: in neuroscience, synaptic pruning refines neural circuits to improve efficiency¹; in systems biology, network sparsification enhances metabolic and regulatory optimization²; and in physics, sparsification techniques identify dominant interaction structures while reducing computational overhead^{8,9}.

Artificial systems also benefit from pruning-based optimization. In machine learning, structured pruning has shown that deep neural networks can significantly reduce size without compromising performance—and in some cases, even improving generalization by reducing overfitting^{10,11}. Similarly, integrated circuits are designed by removing excess connections to improve signal transmission and reduce power consumption¹². Pruning techniques have been widely explored in machine learning (ML) and artificial intelligence (AI) to reduce model complexity, improve efficiency, and maintain predictive performance. Early pruning methods were introduced in feedforward and convolutional neural networks (CNNs) to remove redundant weights and neurons. More recent work extends pruning strategies to recurrent networks, including Echo State Networks (ESNs) and other forms of Reservoir Computing (RC).

Optimization-based approaches¹³ focus on obtaining sparse yet effective networks by iteratively removing less critical connections. Complementary to this, perturbation-based pruning¹⁴ identifies unnecessary nodes through controlled disturbances in the network, revealing structurally weak or redundant components. Both approaches aim to retain model accuracy while reducing computational cost. One challenge in pruning is maintaining a degree of adaptability in the network after reduction. Recent studies¹⁵ explore methods that allow the network to retain some plasticity even after pruning. Evolutionary approaches, such as those proposed by Liquid AI¹⁶, provide automated architecture synthesis, leveraging genetic algorithms to refine network structure dynamically.

Reservoir Computing (RC), particularly ESNs, has been a focal point for pruning research. RC is a machine learning approach that leverages the dynamics of a fixed, high-dimensional nonlinear system, known as the reservoir, to process input signals efficiently. For discrete signals, the reservoir state evolution is governed by the map

$$\mathbf{r}_{t+1} = (1 - \alpha)\mathbf{r}_t + \alpha g(\mathbf{A}\mathbf{r}_t + \mathbf{W}_i\mathbf{u}_t) \quad (1)$$

where \mathbf{r}_t represents the reservoir state at time instance t , \mathbf{u}_t is the input, \mathbf{W}_{in} is the input weight matrix, g is a nonlinear activation function, and $\alpha \in (0, 1]$ is the leakage rate that controls the update speed of the reservoir state. The reservoir, represented by the adjacency matrix \mathbf{A} , is typically implemented as a randomly connected network. It transforms the input sequence \mathbf{u} into a rich feature representation, while only the optimal linear superposition of those reservoir states \mathbf{r} is trained in the output layer

$$\mathbf{y} = \mathbf{W}_{out}\mathbf{r} \quad . \quad (2)$$

Recent findings in reservoir computing suggest that increasing the size of randomly connected reservoir networks does not always improve performance⁶. Excessive internal interactions can degrade signal propagation, introduce chaotic dynamics, or lead to computational inefficiencies. Studies have shown that large reservoir networks may suffer from signal fading or amplification, reducing their ability to effectively process information^{17–20}. These challenges highlight the need for systematic reduction strategies, such as structured pruning, to enhance efficiency and task-specific performance²¹. In a larger scope, research on optimal reservoir networks can help to build smaller but higher performing RCs, e.g. for edge computing.

Early works²² demonstrated that RC models could be pruned effectively without significant loss of computational power. The pruning of ESNs is particularly complex due to the importance of preserving network dynamics, spectral properties, and connectivity structures. Notable research^{21,23,24} has analyzed how different topologies, connectivity distributions, and weight constraints impact pruning outcomes.

Despite advancements, key structural changes during pruning in RC remain poorly understood. Specific open questions include:

- How do fundamental network properties (e.g., clustering coefficient, spectral radius, average out-degree) evolve with pruning?
- What is the preferred distribution of input-receiving and readout nodes after pruning for optimal task performance?
- How does initial network structure affect long-term stability and generalization in pruned reservoir networks?

Understanding these structural transformations is essential for refining pruning methods and designing more efficient, task-specific reservoir networks.

By introducing a structured pruning framework, we aim to investigate if the performance of RC can be increased by a performance-informed node removal as well as elucidate changes in the optimal pruned network structure by answering the aforementioned questions. If less nodes allow for more performance, indicators of fundamental principles governing network efficiency are present in the pruned networks and the pruning process. As an alternative to expansion-based optimization, performance-dependent pruning provides a systematic approach to reducing complexity while preserving—or even improving—functional efficiency. We show that for a range of different prediction tasks there *always* exists a reservoir network of same or better performance after pruning certain reservoir nodes.

II. METHODS

Our analysis is based on the classical reservoir computing paradigm (1) using random Erdős–Rényi graphs as reservoir network \mathbf{A} with $N = |\mathbf{A}|$ nodes. Training of the readout matrix is performed via ridge regression with regularization parameter λ . By intention, we set up the RC models such that not all nodes in the reservoir are connected to input and output layers. Specifically, we randomly choose 50% of the reservoir nodes to be input-receiving (at random input weights), and 50% random reservoir nodes to be connected to the trainable read-out layer via a read-out mask. This allows to study how the fraction of input-receiving and output-connected nodes is affected by the pruning, i.e. whether the pruning approach prefers to keep internal or input/output-connected nodes in \mathbf{A} .

The pruning approach in this work is based on iterative and performance-informed node removal from the reservoir network. Starting from the randomly generated ER-graph $\mathbf{A}^{(k=0)}$ of specified size N_{init} and density ρ_{init} , a fraction of $f_c = 0.25$ reservoir nodes is selected at random to form the set of *candidate* nodes $V_c^{(k)} = \{v_1, \dots, v_c\}$, $c = \lceil p_c \cdot N^{(k)} \rceil$. Independently, c candidate reservoir graphs $\mathbf{A}_i^{(k)}$ are generated by removing node v_i from $\mathbf{A}^{(k)}$. Corresponding entries of the read-in weights and the read-out mask are removed accordingly. The resulting RC models are trained, and the mean squared error on the test set is evaluated. The final choice about the node to finally prune from $\mathbf{A}^{(k)}$ in the current iteration k is performance-informed: The node that - when removed - gives the least performance decrease, or the maximal performance increase, will be finally pruned from the reservoir network. The next pruning iteration $k+1$ thus starts from a reservoir network of $|\mathbf{A}^{(k+1)}| = N^{(k)} - 1$ nodes, and repeats the performance-informed node removal from newly randomly sampled candidate nodes.

Pruning is halted once the performance of the pruned reservoir does not increase (allowing for the patience of five consecutive iterations with performance degradation), or if a minimum number of $N^{(K)} = 15$ reservoir nodes is achieved. The model with the lowest test set error from the K pruning iterations is reported as pruned model. The overall approach is greedy, such that the optimal pruning decision is bounded to the current pruning iteration and the set of current candidate nodes. Increasing f_c , i.e. the number of candidate nodes per iteration, scales the computational effort, but allows to find better pruning candidates and thus better-pruned models. In the limit of $f_c = 1.0$, the truly optimal pruned reservoir network will be obtained in the greedy selection process. Our results display the aggregated results of 50 pruning trials per experiment, each starting from a randomly generated ER reservoir graph, using RC hyperparameters from a preliminary hyperparameter. The main goal is to identify common properties of pruned models when starting from different initial reservoir graphs with only size, density and spectral radius being initially fixed. Properties on the graph level and on node level are tracked along pruning iterations and across candidate nodes, giving a rich view of what type of high-performing networks

arise by pruning what kind of network nodes. All computations were performed using the open-source Python pyReCo library (developed by the authors) that implements the presented pruning strategy with property logging.

III. RESULTS

The proposed pruning approach is evaluated on reservoir computers for a set of different sequence-to-sequence learning tasks. First, the memory-free translation of a sinusoidal signal into its corresponding $\pi/4$ phase-shifted copy $\sin(t) \mapsto \cos(t)$, or Sinos-1 task. In order to increase complexity, we used a complex version of the Sinos-1 task, where the sine function is mapped to its complex polynomial combination: $\sin(t) \mapsto \sin(t)\cos^3(t)$, namely the Sinos-3 task.

Next, we utilized the memory-dependent NARMA- τ tasks, where τ indicates the time lag representing the memory of past inputs²⁶. This is a benchmark task for evaluating a reservoir computer's temporal information processing capability that involves both memory of past inputs and nonlinearity, given by:

$$y(t+1) = \alpha y(t) + \beta y(t) \sum_{i=0}^{M-1} y(t-i) + \kappa x(t-M-1)x(t) + \rho \quad (3)$$

where $\alpha = 0.3$, $\beta = 0.05$, $\kappa = 1.5$, $\rho = 0.1$ and $M \leq 10$. The input $x(t)$ is drawn from a uniform distribution in the interval $[0, 0.5]$. The NARMA function is put inside an additional saturation function *tanh* for $M > 10$, to keep the output bounded between 0 and 1. In this study, we used three NARMA tasks with $\tau = \{5, 10, 15\}$ cases.

A. Performance of pruned RCs

We conducted a systematic investigation of the pruning process across the five aforementioned tasks, recording network performance alongside structural and node-level properties. The initial reservoir networks were constructed as Erdős–Rényi (ER) random graphs with $N_{init} = 50$ nodes and an initial density of $\rho_{init} = 0.05$. By design, the pruning process led to a sharp decrease in the loss function or a corresponding improvement in network performance. However, the overall performance curve exhibited a distinct trend: as pruning progressed and network size continued to decrease, the loss function initially reached a global minimum before subsequently increasing. This characteristic behavior was consistently observed across all tasks, as illustrated in Fig. 1 (a)-(d). The optimal network structure was identified at this global minimum, corresponding to the highest-performing pruned network. Overall, the best-pruned networks demonstrated superior performance compared to the original ER-random networks, as depicted in the violin

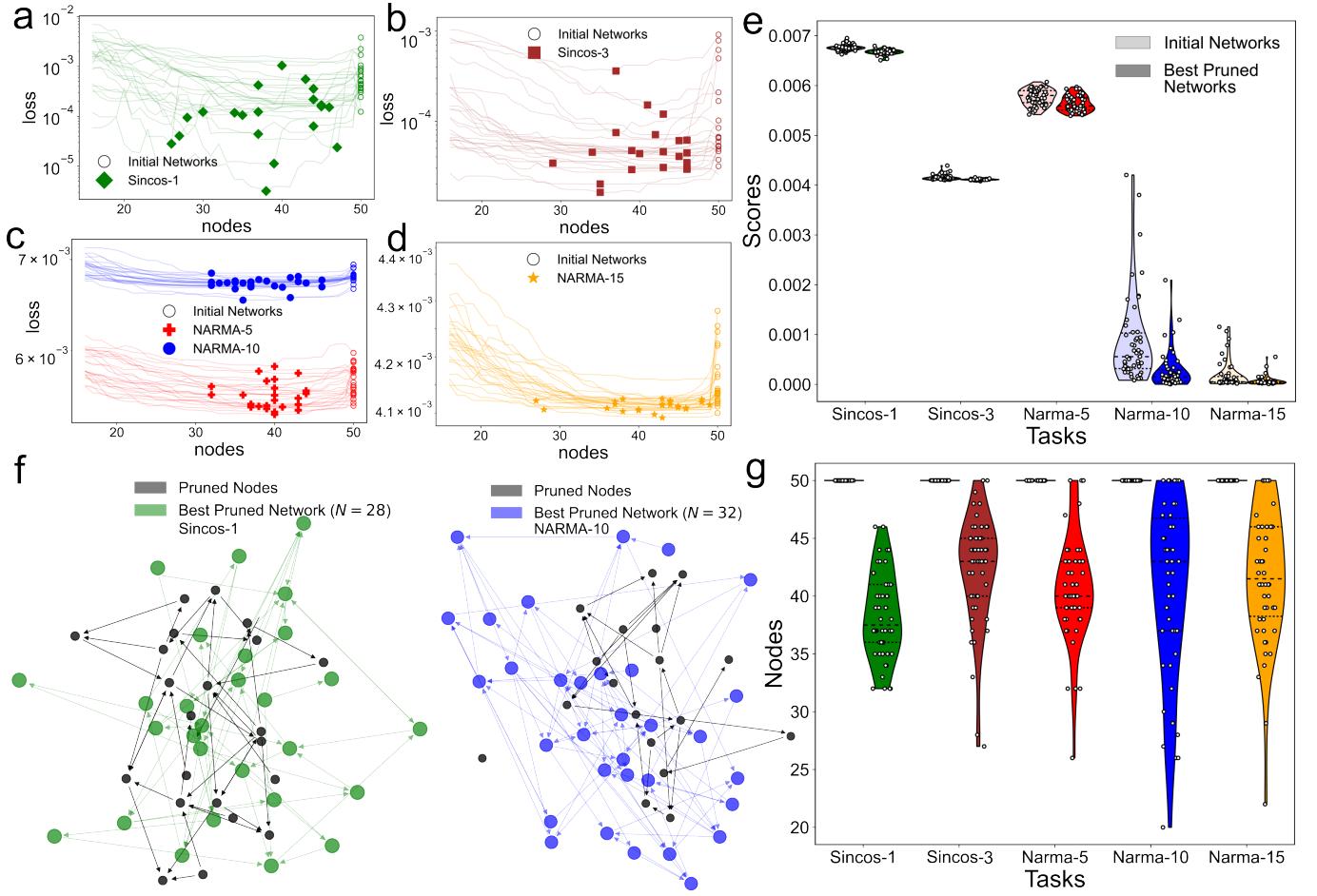


FIG. 1. Node pruning improves efficiency while reducing network size. The overall loss of the reservoir computers shown with respect to the changing reservoir size as pruning starts from initial Erdős–Rényi random networks (open circles) of size and density, $[N_{init}, p_{init}] = [50, 0.05]$ for solving (a, b) Sincos-1 and 3, (c)-(d) NARMA-5, 10 and 15 tasks. The solid markers represent the best-performing pruned network. (e) The Mean Square Errors (MSE) of the initial (*left*) and best pruned networks (*right*) are shown for each task. (f) Exemplar pruned reservoir networks for Sincos-1 (*left*) and NARMA-10 (*right*) tasks. Pruned nodes are represented with dark grey color. (g) The obtained sizes of best-performing pruned networks (*right*) are compared with the initial networks ($N_{init} = 50$) (*left*) for different tasks.

plots in Fig. 1 (e). Additionally, Fig. 1 (g) compares the sizes of these optimal pruned networks against their initial counterparts.

This initial study confirms that the pruning process effectively enhances the performance of ER-random networks by extracting a more compact yet functionally efficient subnetwork. The pruning mechanism selectively removes nodes that do not contribute to the reservoir’s computational efficacy, resulting in a refined network architecture that optimally processes information. On average, the best-pruned networks comprised $\lesssim 40$ nodes across all tasks. Notably, the most compact pruned network—obtained for the NARMA-10 task—contained only 20 nodes, representing a 60% reduction in network size while outperforming its initial ER seed network $N_{init} = 50$ size. The pruned nodes primarily included disconnected elements and dead-end nodes, as exemplified in Fig. 1 (f), where removed nodes and their corresponding edges are highlighted in gray. In the subsequent sections, we further

analyze the structural and node-level properties of these optimized pruned networks and how they differ from their initial Erdős–Rényi (ER) random seed graphs.

B. Graph-level properties

To further investigate the structural characteristics of the pruned networks, we analyzed key network properties, namely density, spectral radius, average in-degree, and clustering coefficient. The violin plots in Fig. 2 illustrate the distributions of these properties across the pruning process, capturing their variations between the initial Erdős–Rényi (ER) random networks and the best-pruned networks. These structural measures are crucial for understanding the topological evolution of the reservoir networks as they undergo pruning.

The **density** of a directed network is defined as the ratio of existing edges to the maximum possible edges:

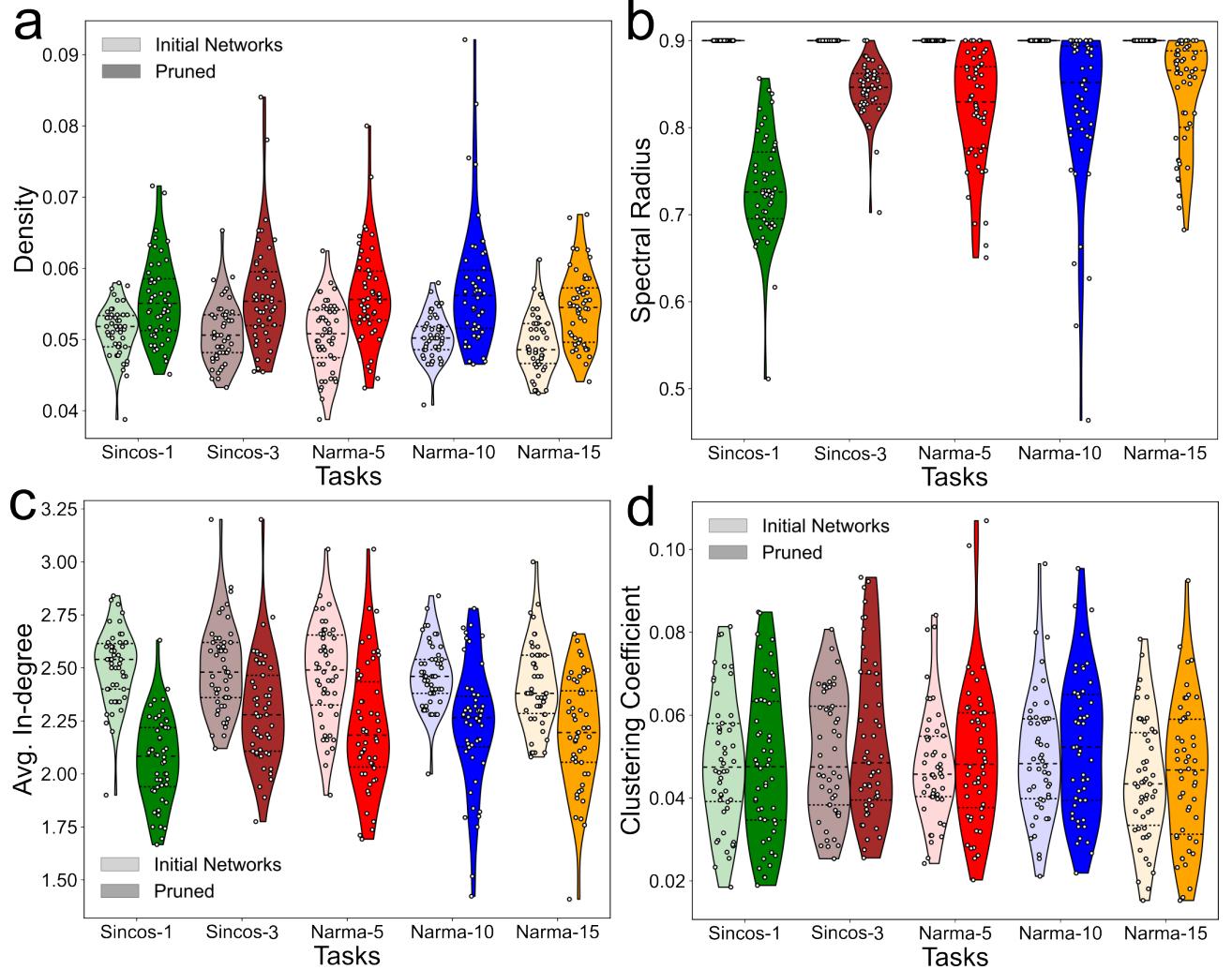


FIG. 2. Change in network structural properties after pruning. The changes in the reservoir network structure can be easily elucidated using (a) density (ρ), (b) spectral radius, (c) average in-degree and (d) clustering coefficient for 5 different tasks. The left and right violins represent the initial and pruned network properties for each task.

$$\rho = \frac{E}{N(N-1)} \quad (4)$$

where E is the total number of directed edges and N is the number of nodes in the network. This measure reflects how densely connected the network is. The **spectral radius** of a network is given by:

$$\lambda_{\max} = \max |\lambda_i| \quad (5)$$

where λ_i are the eigenvalues of the network's adjacency matrix. The spectral radius is crucial in reservoir computing, as it governs the network's dynamical stability and memory capacity. Then we calculated the **average in-degree** of the pruned networks which is given by:

$$k_{\text{in,avg}} = \frac{1}{N} \sum_{i=1}^N k_{\text{in},i} \quad (6)$$

where $k_{\text{in},i}$ represents the number of incoming connections to node i . This metric provides insight into how information is distributed across the network. And finally, the **clustering coefficient** that quantifies the tendency of nodes to form local clusters which is defined as:

$$C = \frac{1}{N} \sum_{i=1}^N C_i \quad (7)$$

where C_i is the local clustering coefficient of node i , calculated as:

$$C_i = \frac{e_i}{k_i(k_i-1)} \quad (8)$$

where e_i is the number of directed edges between the neighbors of node i , and k_i is its degree.

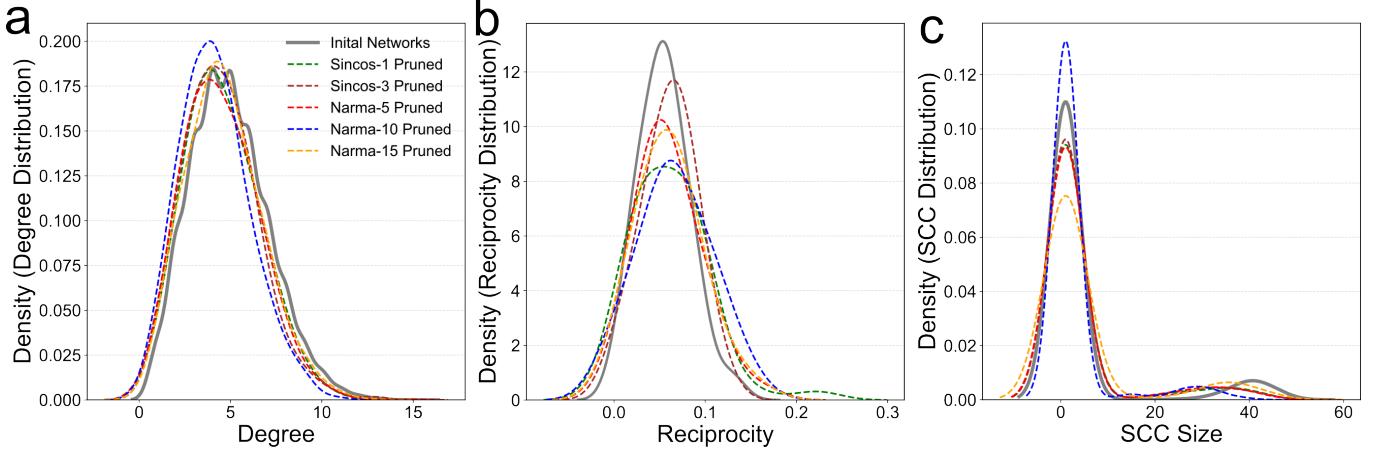


FIG. 3. Degree, Reciprocity, and SCC Distributions: Kernel Density Estimation (KDE) plots comparing the distributions of (a) degree, (b) reciprocity, and (c) strongly connected components (SCC) for the initial Erdős–Rényi (ER) random networks and the best-performing pruned networks across all cases. The distributions illustrate how pruning alters network connectivity, increasing reciprocity while reducing degree, with varying effects on SCC depending on network complexity.

The pruning process induced notable structural changes in the networks. As shown in Fig. 2(a), the density of the best-pruned networks increased across all tasks from the initial mean density of $\rho_{\text{init}} = 0.05$ to approximately 0.06. This increase suggests that the pruning process removes nodes in a way that results in a more compact network, where the remaining nodes retain or even strengthen their connectivity.

A sharp decline in the spectral radius was observed (Fig. 2(b)). The spectral radius was initially fixed at $\lambda_{\max} = 0.9$ for all networks to maintain consistency in the initial conditions. However, after pruning, it dropped to values as low as < 0.5 . Since the spectral radius governs the stability of dynamical systems, this reduction suggests that pruning systematically eliminates redundant or weakly connected nodes, leading to a network with a more controlled dynamical regime.

The average in-degree of the pruned networks also decreased (Fig. 2(c)). Since the number of nodes N decreases during pruning while density ρ slightly increases, we can express the expected number of incoming connections per node as:

$$k_{\text{in,avg}} = \rho(N - 1). \quad (9)$$

Given that ρ increases modestly while N drops significantly, the overall trend results in a lower average in-degree. This confirms that pruning removes nodes along with their incoming and outgoing connections, leading to a sparser but more structured network.

Interestingly, the clustering coefficient remained largely unchanged throughout pruning (Fig. 2(d)). This invariance suggests that while individual nodes and edges were removed, the local connectivity patterns—capturing the extent of triangular structures within the network—were preserved. This implies that the pruning mechanism selectively eliminates

nodes in a way that does not disrupt local clustering properties, reinforcing the robustness of the network’s mesoscopic organization.

Overall, these structural changes highlight the transformation of the Erdős–Rényi (ER) random networks’ macroscopic properties into a more structured, task-optimized network through pruning. In its initial state, the ER network exhibits a homogeneous connectivity distribution with a random arrangement of nodes and edges. However, the pruning process selectively eliminates redundant and weakly connected nodes, leading to a refined network with increased density, reduced spectral radius, and lower average in-degree, all while preserving local clustering properties. This transition suggests that pruning extracts a more functionally efficient core network while discarding structurally insignificant components.

These macroscopic structural changes provide insights into the global evolution of the pruned networks, they explain the underlying mechanisms driving this transformation to a greater extent. Furthermore, to gain a deeper understanding, it is essential to analyze node-level properties, which reveal how individual nodes contribute to the emergent computational efficiency of the pruned network. In the following section, we examine these node-level characteristics to uncover the finer details of the network’s reorganization and the roles played by different types of nodes in optimizing reservoir computing performance.

C. Beneficial node-level properties

We calculated node-level properties of the initial and finally obtained best-pruned networks in order to understand the substructural changes in the network when the nodes are selectively removed while improving the performance. The key properties analyzed include degree, reciprocity, and strongly connected components (SCC).

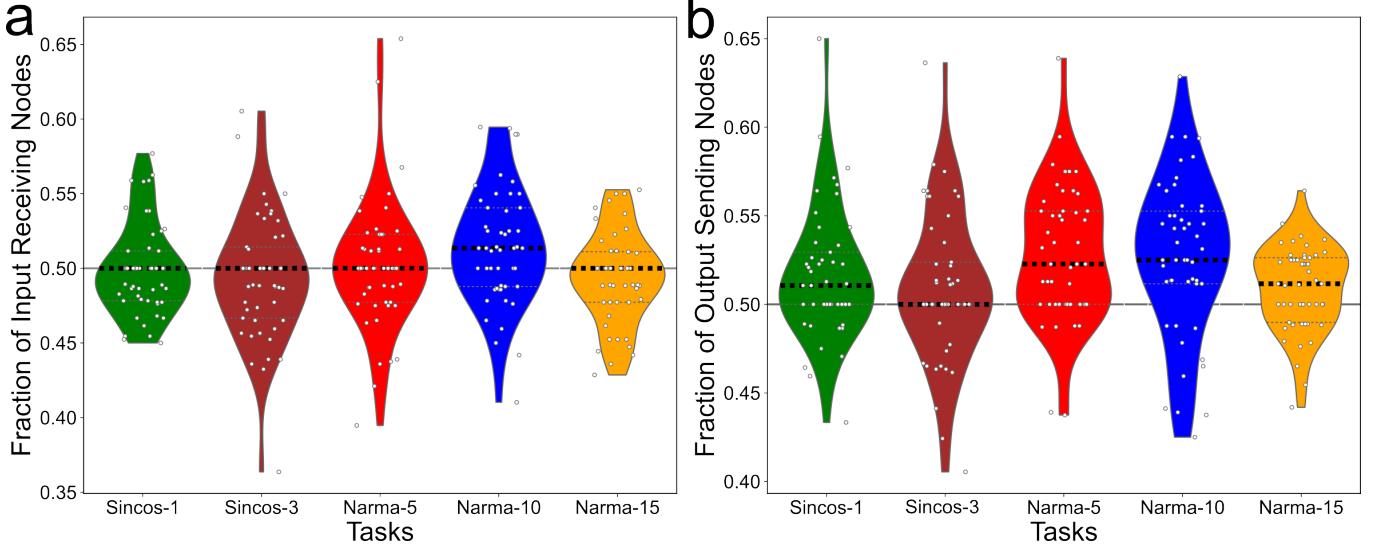


FIG. 4. Asymmetry between input-receiving and readout nodes. The fraction of (a) input-receiving and (b) readout nodes, calculated by dividing their number by the total number of pruned network nodes, is shown for various tasks. The grey line in the background indicates their fraction in the initial network, $A^{(k=0)}$.

Reciprocity measures the symmetry of connections in a network and is defined as:

$$R = \frac{\sum_{i,j} A_{ij} A_{ji}}{\sum_{i,j} A_{ij}}.$$

As shown in Table I, pruning generally improves reciprocity across the cases, with increases of **11.51%** in Sincos-1, **11.68%** in Sincos-3, and **6.49%** in Narma-5. This suggests more balanced node interactions post-pruning. In Narma-10 and Narma-15, the increase is more significant, with improvements of **14.00%** and **17.89%**, respectively, indicating that pruning leads to a more efficient and balanced structure, especially in more complex networks.

Degree, representing the number of connections each node has, generally decreases after pruning, as seen in Table I. The largest reduction occurs in Narma-10 with a **17.27%** decrease, reflecting a more significant pruning effect on connectivity in larger networks. Other cases, such as Sincos-1 (**-7.82%**) and Narma-5 (**-9.30%**), show a decrease in degree, suggesting that pruning simplifies the network structure by reducing the number of connections. This may contribute to improved network performance by focusing on the most essential connections. This trend is further supported by the KDE (Kernel Density Estimation) plots shown in Fig.3, which illustrate how the degree distribution shifts between the initial and pruned networks. The KDE curves reveal a noticeable narrowing of the degree distribution after pruning, indicating that pruning focuses on a more limited set of connections, which aligns with the degree reduction observed.

Strongly Connected Components (SCC) measure the network's cohesiveness, with higher values indicating better robustness. SCC is defined as:

$$\text{SCC} = \sum_i \sum_j \delta(A_{ij}, A_{ji}),$$

where $\delta(x,y)$ measures directed cycles. As shown in Table I, pruning tends to reduce SCC in smaller networks, such as Sincos-1 (**-12.48%**) and Narma-5 (**-4.25%**). However, in more complex cases like Narma-10, SCC increases by **20.85%**, suggesting that pruning can strengthen the cohesiveness of larger networks by eliminating weaker connections and reinforcing stronger ones. The KDE plot in Fig.3 also reveals the shift in SCC distribution post-pruning. It shows that the distribution for the pruned networks tends to be more concentrated around higher values in some cases, especially in larger networks, corresponding to the observed increase in SCC for Narma-10.

Upon further analysis, we identified an emergent node-specific characteristic in the pruned networks: a self-organized asymmetric distribution of input-receiving and readout nodes. Initially, in all our simulations, nodes in the Erdős–Rényi (ER) random networks were randomly designated as input-receiving and readout nodes with an independent probability of 50%. As a result, the initial reservoir networks, on average, had half of their nodes assigned as input-receiving and half as readout nodes, with approximately 25% of the total nodes serving both roles.

However, in the final pruned networks, this initially symmetric distribution became asymmetric. Across all tasks, the fraction of readout nodes consistently exceeded 0.5 in the pruned networks, as illustrated in Fig. 4. Surprisingly, the proportion of input-receiving nodes remained at 0.5 even after pruning. This indicates that nodes not serving as readout nodes were preferentially removed, suggesting that the pruned networks enhance efficiency by selectively

| Case | Degree (Init/Final) | % Change | Reciprocity | % Change | SCC | % Change |
|----------|---------------------|----------|-----------------|----------|---------------|----------|
| Sincos-1 | 4.932 / 4.546 | -7.82% | 0.0548 / 0.0611 | 11.51% | 11.54 / 10.10 | -12.48% |
| Sincos-3 | 4.827 / 4.446 | -7.90% | 0.0537 / 0.0599 | 11.68% | 11.66 / 10.58 | -9.26% |
| Narma-5 | 4.944 / 4.484 | -9.30% | 0.0560 / 0.0597 | 6.49% | 10.36 / 9.92 | -4.25% |
| Narma-10 | 5.015 / 4.149 | -17.27% | 0.0582 / 0.0663 | 14.00% | 10.36 / 12.52 | 20.85% |
| Narma-15 | 4.994 / 4.624 | -7.40% | 0.0514 / 0.0606 | 17.89% | 10.08 / 8.41 | -16.60% |

TABLE I. The percent change between the initial and pruned network properties.

retaining more readout nodes than input nodes while reducing overall network size.

In summary, as depicted in Table I and visualized in the KDE plots (Fig.3), pruning tends to simplify networks by reducing degrees and increasing reciprocity, with mixed effects on SCC depending on network complexity. Smaller networks generally show reduced SCC after pruning, while larger networks may benefit from stronger connectivity and improved cohesion. This indicates that pruning not only reduces the size of the network but also enhances its structural efficiency. Additionally, pruning leads to a self-organized asymmetric distribution of input-receiving and readout nodes. While the initial networks have a symmetric assignment, with approximately 50% of nodes designated as input-receiving and 50% as readout nodes, the final pruned networks consistently retain a higher fraction of readout nodes while maintaining the input-receiving fraction at 0.5. This suggests that non-readout nodes are selectively removed, allowing the pruned networks to maintain efficiency with a reduced structure while prioritizing readout functionality.

D. Effect of Initial Network Conditions on Pruned Substructures

In this section, we explore the effect of initial conditions on pruned substructures by testing different Erdős-Rényi random networks for sizes [50, 75, 100] and densities [0.05, 0.1]. The results, summarized in Table II and illustrated in Figure 5, highlight how these initial configurations influence reservoir properties such as spectral radius, average in-degree, average out-degree, and clustering coefficient (CC) after pruning.

Smaller, sparser networks (50 nodes, density 0.05) experience substantial reductions in network properties, particularly spectral radius and degree metrics, compared to larger, denser networks (75 and 100 nodes, density 0.1). The spectral radius, average in-degree, out-degree, and CC all decrease significantly in these smaller networks, suggesting that they are more sensitive to pruning. In contrast, larger and denser networks are more resilient, showing smaller changes in their structural properties after pruning, indicating that they are better able to maintain their core structure.

For task-specific results, the SinCos-1 task with 75 nodes and a density of 0.1 shows a 10.67% decrease in CC after pruning. In contrast, Narma-5 with 100 nodes and a density of 0.1 experiences minimal changes in CC, with a slight increase observed post-pruning. This reflects the greater stability of larger networks. For more complex tasks like Sincos-3,

pruning leads to a notable increase in CC when pruning networks with 50 nodes and a density of 0.05. Conversely, Narma-10 with 50 nodes and a density of 0.05 experiences a 23.22% decrease in CC, illustrating the typical reduction in connectivity for smaller networks. Narma-15 with 75 nodes and a density of 0.1 shows a slight increase in CC (0.16%) post-pruning, indicating some variation in pruning effects even within larger networks. These task-specific results emphasize how network size and task complexity interact to influence the impact of pruning.

The **spectral radius**, initially fixed at 0.9, is a key indicator of network stability and dynamics. Across tasks, pruning leads to a decrease in spectral radius, with smaller networks experiencing more significant reductions. For SinCos-1, pruning reduces the spectral radius by 10.36% for 50 nodes (density 0.05) and by 5.71% for 100 nodes (density 0.1). This suggests that pruning reduces the network's ability to maintain stable dynamics, especially in smaller networks. In Sincos-3, pruning causes a smaller decrease in spectral radius compared to SinCos-1, with a 4.22% decrease for 50 nodes (density 0.05) and a 4.01% decrease for 100 nodes (density 0.1), indicating that larger networks are more stable post-pruning. For Narma-5, the spectral radius decreases moderately, while in more complex tasks like Narma-10, pruning results in a significant decrease in spectral radius, particularly for smaller networks (17.07% for 50 nodes and density 0.05). This highlights the greater impact of pruning on smaller networks and tasks requiring higher dynamical capacity. Larger networks, such as Narma-15, show more stability post-pruning, with a smaller decrease (5.26%) observed for 100 nodes (density 0.1).

As seen in Fig.5 (last row), the spectral radius always drastically decreases while pruning the network, especially for smaller networks. This suggests that smaller networks, particularly those with lower density, are more likely to lose their ability to maintain stable dynamical behavior. The initial spectral radius of 0.9 ensures that all networks start with a stable dynamical range, and the observed reductions in spectral radius, particularly for tasks like Narma-10, highlight the challenges of preserving dynamical properties after pruning. However, larger networks demonstrate greater stability, indicating that pruning has a lesser impact on their spectral radius and overall stability.

In terms of the **average in-degree**, pruning generally reduces the connectivity of the network. For SinCos-1 with 50 nodes and density 0.05, the average in-degree decreases by 8.54%, while for Narma-5 with 100 nodes and density 0.1, the reduction is 6.77%. The reduction in connectivity is more

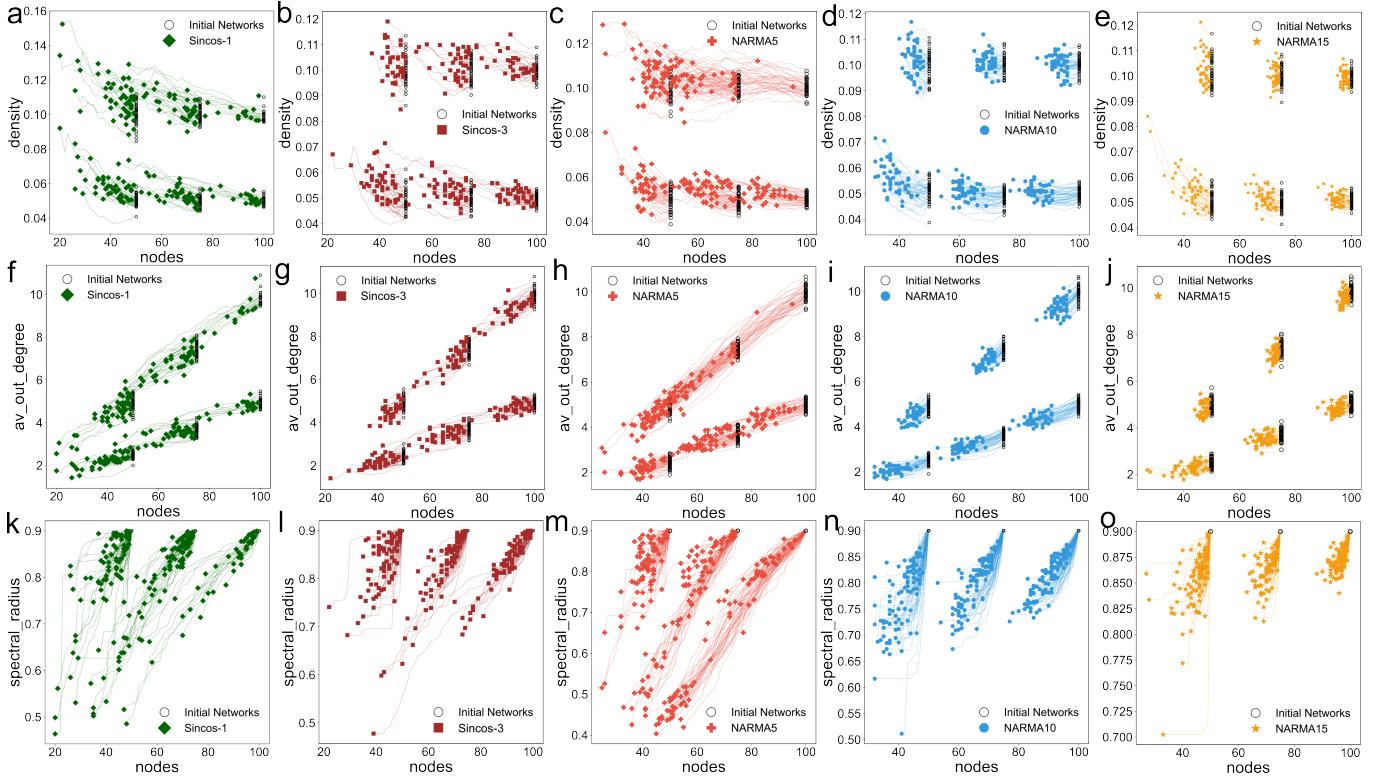


FIG. 5. Effect of initial network organization on pruned substructure. Initial Erdős-Rényi random reservoir networks $A^{(k=0)}$ are generated to have different sizes $N_{init} = [50, 75, 100]$ and densities $\rho_{init} = [0.05, 0.1]$. These are pruned for 5 different Sincos and NARMA tasks. The changes in their densities (first row), average out-degree (middle row) and spectral radius (last row) are shown in different panels. The initial networks are shown with open black circles and the best pruned networks with solid symbols. The pruning trajectories (faded lines) are shown here until the best pruned networks.

pronounced in smaller networks, indicating that pruning tends to remove less connected nodes, which affects the network's capacity to propagate information. In more complex tasks like Narma-10, pruning leads to an 18.15% decrease in average in-degree for 50 nodes (density 0.05), demonstrating the larger impact pruning has on smaller networks.

Finally, the **clustering coefficient** (CC) shows varied changes across tasks and network sizes. Generally, pruning leads to a decrease in CC, particularly in smaller networks. For instance, in SinCos-1 with 75 nodes and a density of 0.1, the CC decreases by 10.67%, while for Narma-5 with 100 nodes and a density of 0.1, a slight increase of 0.16% is observed. In Sincos-3, pruning causes a significant increase in CC for 50 nodes and a density of 0.05 (50.38%), suggesting that for some tasks and configurations, pruning can enhance local clustering. In contrast, Narma-10 shows a typical decrease in CC post-pruning, especially in smaller networks, highlighting the varying effects pruning has depending on the task complexity and network size.

In conclusion, pruning tends to reduce network stability, as evidenced by decreases in spectral radius, average in- and out-degree, and clustering coefficient, particularly in smaller networks. Larger networks, however, show greater resilience, with smaller decreases in spectral radius and clustering coef-

ficient. Tasks like Narma-5 and Sincos-3 show a more stable network structure post-pruning, while tasks like Narma-10 experience greater reductions in connectivity and dynamical capacity. These findings emphasize the need to carefully consider initial network conditions and task complexity when designing efficient reservoir networks.

IV. DISCUSSION AND OUTLOOK

This study systematically investigates the structural transformation of Erdős-Rényi (ER) random networks through a performance-dependent pruning process in the context of reservoir computing (RC). Our results demonstrate that pruning selectively eliminates redundant nodes while preserving key structural properties essential for effective information processing. This process results in a refined network with increased density, reduced spectral radius, and a lower average in-degree while maintaining the clustering coefficient. The observed improvements in network performance highlight the emergence of a compact, efficient computational structure from an initially random topology.

The findings have significant implications for Reservoir Computing (RC), Network Science, and Nonlinear Dynamics community where understanding the evolution of complex

networks is crucial. In RC, an optimized network structure enhances memory capacity and prediction accuracy, making it beneficial for time-series forecasting, signal processing, and dynamical system modeling^{17,27}. In **network science**, this work sheds light on how connectivity patterns evolve through optimization mechanisms, which has implications for communication networks, social networks, and artificial intelligence architectures. In **nonlinear dynamics**, where RC has been widely used to model chaotic systems, this study contributes a new theoretical perspective on how pruning affects dynamical properties^{28,29}.

Beyond artificial networks, **biological systems** exhibit similar pruning phenomena, reinforcing the broader applicability of the framework introduced in this study. In **neuronal networks**, synaptic pruning plays a fundamental role in cognitive development, where unnecessary synaptic connections are eliminated to optimize brain function^{30,31}. This process mirrors our observation that pruning reduces network size while improving performance, as seen in the decline and subsequent optimization of the loss function. The removal of redundant nodes in our networks parallels the way the brain refines its connectivity to enhance computational efficiency.

Similarly, in **gene regulatory networks**, non-essential genes or regulatory interactions are selectively silenced over evolutionary time to improve the efficiency of cellular responses³². Our results show a drop in spectral radius and a decline in average in-degree, suggesting that removing weakly connected nodes improves the network's functional efficiency—just as biological systems streamline regulatory interactions to optimize control over gene expression. In **protein interaction networks**, certain interactions are lost as a system adapts to environmental changes, leading to optimized metabolic and signaling pathways². The increase in density observed in our pruned networks suggests that, despite a reduction in size, the retained nodes maintain or strengthen their functional interconnectivity, similar to how essential protein interactions persist even as redundant ones disappear.

From a real-world perspective, pruning-like mechanisms are observed in various infrastructural and technological networks. In **communication networks**, inefficient nodes or links are decommissioned to improve efficiency, much like our study's pruning process removes underperforming nodes while enhancing the reservoir's predictive capabilities. In **power grids**, redundant connections are removed to reduce energy losses while maintaining resilience, similar to how pruning preserves network connectivity despite node removal. Likewise, in **transportation systems**, route optimizations eliminate underused pathways while maintaining network connectivity—aligning with our observation that clustering coefficients remain largely unchanged, ensuring the pruned network retains its essential structural motifs.

These parallels suggest that the structural changes induced by pruning in our study align with fundamental optimization strategies observed across biological and engineered systems. By systematically quantifying how pruning refines network efficiency, our work provides a theoretical foundation to understand how complex systems evolve toward more efficient forms.

This work establishes a theoretical foundation for performance-dependent pruning in reservoir computing and complex networks, opening several avenues for future research. Given the parallels with neuronal pruning, this approach could be tested on biologically inspired spiking neural networks to explore how optimized reservoir structures emerge naturally in the brain. Furthermore, the insights from this study can be extended to fields such as **epidemiology** (modeling how redundant connections in disease-spread networks affect transmission dynamics), **financial networks** (identifying critical nodes in economic systems), and **AI explainability** (understanding how deep networks can be pruned without loss of function).

Reservoir computing (RC) has gained significant attention for its ability to efficiently process temporal data with minimal training overhead^{17,27}. While much research in the RC community has focused on optimizing hyperparameters¹⁸, spectral properties³³, and learning rules³⁴, little attention has been paid to the structural efficiency of reservoir networks. Most studies employ standard random networks (Erdős–Rényi, scale-free, or small-world) without systematically analyzing whether these topologies are the most efficient for a given task. Our study addresses this gap by demonstrating that an initially unoptimized random network can be pruned into a more efficient structure without loss of performance, and often with performance enhancement.

Another largely unexplored direction in RC is using it as a framework to study network evolution and pruning mechanisms. While network pruning is well known in deep learning^{10,35}, its role in recurrent architectures like RC remains underexplored. By leveraging pruning, we show that redundant structures in reservoir networks can be systematically removed to obtain a more efficient computational core, making RC not only a powerful tool for learning but also a framework for studying the emergent properties of complex systems.

This study also complements the evolving network approach in reservoir computing proposed by Yadav et al.⁶, which takes a *bottom-up* perspective by building performance-dependent networks from scratch through an evolutionary process. Their work focuses on how networks can be grown adaptively to optimize performance, whereas this study follows a *top-down* approach, starting from an initially large network and systematically reducing it to its most efficient form. Together, these two approaches provide a holistic view of how networks can be both grown and pruned to achieve optimal computational structures.

A key insight from this work is that network pruning can reveal the essential structural motifs required for optimal RC performance, while Yadav et al.'s approach demonstrates how these structures can emerge through adaptive growth. Future research could integrate these two methodologies—combining evolutionary network growth with performance-dependent pruning—to refine reservoir topologies dynamically. Such an approach could be particularly valuable in adaptive learning systems, where networks continuously evolve in response to changing tasks and environments.

By bridging insights from network science, machine learning, and neuroscience, this work opens new avenues for understanding the interplay between structure and function in complex adaptive systems.

ACKNOWLEDGMENTS

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under the Special Priority Program (SPP 2353: Daring More Intelligence – Design Assistants in Mechanics and Dynamics, project number 501847579).

DATA AVAILABILITY STATEMENT

All the Python codes for the numerical simulations and the data generated in this study are publicly available at the following repository: <https://github.com/Cyber-Physical-Systems-in-Mech-Eng/PrunedRC>.

Appendix A: Quantification of the effect of different initial conditions

The effect of different initial Erods-Renyi random networks on the emergent best-pruned subnetwork properties are provided in the following Table II. A combination of different nodes $N_{init} = [50, 75, 100]$ and densities $\rho_{init} = [0.05, 0.1]$ are used to generate a total of six different kinds of initial ER-random networks $\mathbf{A}^{(k=0)}$ for pruning experiment for each of the 5 tasks. The table shows the mean percent change of density, spectral radius, average in-degree, average out-degree and clustering coefficient (CC) of the final pruned networks from that of the initial networks.

Appendix B: Pruning model parameters

The following parameters are fixed and used for all the experiments in this study in order to maintain consistency: Minimum nodes till the pruning continues, $N^K = 15$, patience (=5) defines the continuation of pruning process even after reaching a (local) minimum of the test set score, fraction of input-receiving and readout nodes is 0.25, set of candidate nodes used to find a node to be ruined, $f_c=0.25$, spectral radius = 0.9, mean square error (MSE) is used for obtaining the performance.

¹L. D. Selenon, “A role for synaptic plasticity in the adolescent development of executive function,” *Translational Psychiatry* **3**, e238 (2013).

²A.-L. Barabási and Z. N. Oltvai, “Network biology: Understanding the cell’s functional organization,” *Nature Reviews Genetics* **5**, 101–113 (2004).

³U. Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits* (Chapman & Hall/CRC, 2007).

⁴O. Sporns and R. F. Betzel, “Modular brain networks,” *Annual Review of Psychology* **67**, 613–640 (2016).

⁵G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, “Long short-term memory and learning-to-learn in networks of spiking neurons,” in *NeurIPS*, Vol. 31 (2018).

⁶M. Yadav, S. Sinha, and M. Stender, “Performance-dependent network evolution for efficient information processing,” *Physical Review E* **111**, 014320 (2025).

⁷A. Radhakrishnan, J. F. Lindner, S. T. Miller, S. Sinha, and W. L. Ditto, “When less is more: evolving large neural networks from small ones,” *arXiv preprint arXiv:2501.18012* (2025), <https://arxiv.org/abs/2501.18012>.

⁸J. W. Rocks, N. Pashine, I. Bischoffberger, C. P. Goodrich, A. J. Liu, and S. R. Nagel, “Designing allosteric-inspired response in mechanical networks,” *Proceedings of the National Academy of Sciences (PNAS)* **116**, 2506–2511 (2019).

⁹D. A. Spielman and S.-H. Teng, “Spectral sparsification of graphs,” *SIAM Journal on Computing* **40**, 981–1025 (2011).

¹⁰S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural networks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 28 (2015).

¹¹P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource-efficient inference,” in *International Conference on Learning Representations (ICLR)* (2017).

¹²C. Mead, *Analog VLSI and Neural Systems* (Addison-Wesley, 1989).

¹³I. Hussain and T. Li, “An optimization-based algorithm for obtaining an effective sparse network,” *Chaos* **33**, 033103 (2023).

¹⁴Z. Yan and H. Wang, “A perturbation-based approach to identifying unnecessary nodes in neural networks,” *Chaos* **33**, 063119 (2023).

¹⁵L. Jiang and R. Zhao, “Keeping some adaptivity in the network after pruning,” *Chaos* **34**, 012345 (2024).

¹⁶R. Hasani, M. Lechner, and R. Grosu, “Automated architecture synthesis via targeted evolution,” *Liquid AI Research* (2023).

¹⁷H. Jaeger, “The ‘echo state’ approach to analyzing and training recurrent neural networks,” *Tech. Rep. GMD Technical Report 148* (German National Research Center for Information Technology, 2001).

¹⁸D. Verstraeten, B. Schrauwen, M. D’Haene, and D. Stroobandt, “An experimental unification of reservoir computing methods,” *Neural Networks* **20**, 391–403 (2007).

¹⁹I. B. Yıldız, H. Jaeger, and S. J. Kiebel, “Re-visiting the echo state property,” *Neural Networks* **35**, 1–9 (2012).

²⁰C. Gallicchio, A. Micheli, and L. Pedrelli, “Deep reservoir computing: A critical experimental analysis,” *Neurocomputing* **268**, 87–99 (2017).

²¹M. Lukosevicius, H. Jaeger, and B. Schrauwen, “Reservoir computing trends,” in *Lecture Notes in Computer Science*, Vol. 718 (Springer, 2012) pp. 384–397.

²²B. Schrauwen, D. Verstraeten, and J. Van Campenhout, “Pruning reservoir computing networks,” *Neurocomputing* **72**, 1534–1546 (2009).

²³Y. Can and D. Gauthier, “Structural evolution in reservoir computing networks,” *Nature Machine Intelligence* **6**, 202–215 (2024).

²⁴L. Schiller, T. Hüser, and B. Lindner, “The effect of pruning on the performance of echo state networks,” *Chaos* **30**, 045678 (2020).

²⁵<https://github.com/Cyber-Physical-Systems-in-Mech-Eng/pyReCo>.

²⁶A. Atiya and A. Parlos, “New results on recurrent network training: unifying the algorithms and accelerating convergence,” *IEEE Transactions on Neural Networks* **11**, 697 (2000).

²⁷W. Maass, T. Natschlaeger, and H. Markram, “Real-time computing without stable states: A new framework for neural computation based on perturbations,” *Neural Computation* **14**, 2531 (2002).

²⁸J. Pathak, B. R. Hunt, M. Girvan, Z. Lu, and E. Ott, “Model-free prediction of large spatiotemporally chaotic systems from data,” *Science* **360**, 786–790 (2018).

²⁹J. Gauthier, “Emergent properties of machine learning in dynamic systems,” *Nature Communications* **12**, 5439 (2021).

³⁰J. Huttenlocher, “The nature of cognitive development: A psychological perspective,” *Psychological Review* **86**, 331–347 (1979).

³¹G. Chechik *et al.*, “An integrative theory of memory and perception,” *Journal of Cognitive Neuroscience* **10**, 410–430 (1998).

³²U. Alon, “Network motifs: Theory and experimental approaches,” *Nature Reviews Genetics* **7**, 4–16 (2006).

³³A. Rodan and P. Tino, “Minimum complexity echo state networks,” *IEEE Transactions on Neural Networks* **22**, 131–144 (2011).

| Tasks | N_{init} | ρ_{init} | $\Delta\rho$ (%) | Δ Spectral radius (%) | Δ Avg. in-degree (%) | Δ Avg. out-degree (%) | Δ CC (%) |
|----------|------------|---------------|------------------|------------------------------|-----------------------------|------------------------------|-----------------|
| SinCos-1 | 50 | 0.05 | 4.23% | -10.36% | -8.54% | -8.54% | -2.41% |
| SinCos-1 | 50 | 0.1 | 0.73% | -4.18% | -5.44% | -5.44% | 3.03% |
| SinCos-1 | 75 | 0.05 | 0.60% | -1.80% | -2.11% | -2.11% | -10.67% |
| SinCos-1 | 75 | 0.1 | 0.11% | -7.84% | -6.65% | -6.65% | 0.62% |
| SinCos-1 | 100 | 0.05 | 2.77% | -0.66% | -0.34% | -0.34% | 0.84% |
| SinCos-1 | 100 | 0.1 | 1.20% | -5.71% | -5.95% | -5.95% | 3.90% |
| SinCos-3 | 50 | 0.05 | 9.53% | -4.22% | -8.35% | -8.35% | 50.38% |
| SinCos-3 | 50 | 0.1 | -0.23% | -11.98% | -10.41% | -10.41% | -4.90% |
| SinCos-3 | 75 | 0.05 | 3.67% | -11.28% | -10.34% | -10.34% | -5.58% |
| SinCos-3 | 75 | 0.1 | 2.18% | -2.17% | -1.96% | -1.96% | -1.00% |
| SinCos-3 | 100 | 0.05 | 2.23% | -0.67% | -0.86% | -0.86% | -2.77% |
| SinCos-3 | 100 | 0.1 | 1.12% | -4.01% | -3.99% | -3.99% | 1.70% |
| Narma-5 | 50 | 0.05 | 0.65% | -16.12% | -13.73% | -13.73% | -4.01% |
| Narma-5 | 50 | 0.1 | 1.73% | 0.00% | -0.34% | -0.34% | 2.30% |
| Narma-5 | 75 | 0.05 | 3.79% | -11.25% | -11.64% | -11.64% | -3.20% |
| Narma-5 | 75 | 0.1 | 0.90% | -2.41% | -1.83% | -1.83% | -1.89% |
| Narma-5 | 100 | 0.05 | 6.09% | -7.41% | -6.77% | -6.77% | 0.16% |
| Narma-5 | 100 | 0.1 | 8.44% | -11.53% | -11.28% | -11.28% | 3.87% |
| Narma-10 | 50 | 0.05 | 8.39% | -17.07% | -18.15% | -18.15% | -23.22% |
| Narma-10 | 50 | 0.1 | 2.35% | -13.02% | -10.18% | -10.18% | 1.67% |
| Narma-10 | 75 | 0.05 | 4.14% | -19.39% | -16.97% | -16.97% | 22.05% |
| Narma-10 | 75 | 0.1 | 0.29% | -3.72% | -3.77% | -3.77% | -1.47% |
| Narma-10 | 100 | 0.05 | -0.18% | -12.54% | -11.27% | -11.27% | -7.67% |
| Narma-10 | 100 | 0.1 | -0.91% | -10.00% | -9.92% | -9.92% | -0.96% |
| Narma-15 | 50 | 0.05 | 4.38% | -3.93% | -4.14% | -4.14% | -10.15% |
| Narma-15 | 50 | 0.1 | 2.29% | -1.72% | -1.89% | -1.89% | 1.05% |
| Narma-15 | 75 | 0.05 | 0.39% | -3.67% | -5.04% | -5.04% | -6.38% |
| Narma-15 | 75 | 0.1 | -0.36% | -3.32% | -3.06% | -3.06% | 0.16% |
| Narma-15 | 100 | 0.05 | 2.73% | -5.26% | -6.60% | -6.60% | -0.03% |
| Narma-15 | 100 | 0.1 | 0.23% | -2.89% | -3.82% | -3.82% | 1.60% |

TABLE II. The changes in reservoir network properties for different initial Erdős-Rényi random networks.

³⁴B. Schrauwen *et al.*, “The echo state network: A powerful tool for the modeling of dynamic systems,” *Neural Networks* **21**, 1212–1221 (2008).

³⁵Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a back-propagation network,” in *IEEE Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, Vol. 1 (1990) pp. 541–544.

³⁶Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE* **86**, 2278–2324

(1998).

³⁷D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature* **323**, 533–536 (1986).

³⁸M. Lukoševičius, “A practical guide to applying echo state networks,” in *Neural Networks: Tricks of the Trade*, Lecture Notes in Computer Science (LNCS), Vol. 7700 (Springer, Heidelberg, 2012) 2nd ed., pp. 659–686.