UNIFYING DYNAMICAL SYSTEMS AND GRAPH THEORY TO MECHANISTICALLY UNDERSTAND COMPUTATION IN NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding how biological and artificial neural networks implement computation from connectivity is a central problem in neuroscience and machine learning. In neural systems, structural and functional connectivity are known to diverge, motivating approaches that move beyond direct connections alone. Here, we show that the function of recurrent neural networks (RNNs) trained on hierarchically modular tasks, a maze navigation task, and XOR can be recovered by modelling the network as a graph and analysing the multi-hop pathways between input and output units. In particular, decomposing these pathways by hop length reveals how the network temporally routes information. This perspective reframes regularisation: if function is implemented through multi-hop communication, then standard penalties such as L1 regularisation, which act only on individual weights, constrain single-hop structure rather than the multi-hop pathways that support computation. Motivated by this view, we introduce resolvent-RNNs (R-RNNs), which constrain multi-hop pathways and thereby induce temporal sparsity beyond that achieved by standard L1 regularisation. R-RNNs achieve improved performance by inducing temporal sparsity that matches the task structure, even when the task signal is sparse. Moreover, R-RNNs exhibit stronger sparsity–function alignment, reflected in their increased robustness under strong regularisation. Together, our results identify multi-hop communication as a key principle linking structure to function in recurrent networks, and suggest that sparsity should be defined over functional pathways rather than individual parameters.

1 INTRODUCTION

Computation is often defined as a mapping from inputs to outputs or the manipulation of representations (Newell, 1980; Marr, 2010; Bengio et al., 2014; Saxe et al., 2019). But such definitions say little about how computation is implemented in networks of interacting units evolving through time. This raises a central question in neuroscience and machine learning: can we infer what a network computes from its structure?

A common assumption is that structural connectivity constrains function, such that information flow can be read directly from synaptic or weighted connections. Yet structural and functional connectivity are known to diverge in neural systems (Honey et al., 2009; Vázquez-Rodríguez et al., 2019; Park & Friston, 2013). We argue that one reason is that direct connections capture only single-step interactions, whereas computation in recurrent networks is mediated by multi-hop pathways unfolding over time. Understanding function therefore requires tools that aggregate signal propagation across multiple routes.

Graph theory and dynamical systems provide such a framework. In recurrent networks, temporal evolution can be interpreted as successive hops through connectivity, so graph-based measures such as communicability and the resolvent summarise how structure gives rise to function (Zamora-López & Gilson, 2024; Estrada & Benzi, 2014; Fakhar & Hilgetag, 2022; Crofts & Higham, 2009).

Here we use RNNs as a controlled testbed for this perspective. Across modular tasks requiring averaging, subtraction, addition, and multiplication, a maze navigation task, and XOR, we show that

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the resolvent recovers the expected routing structure, and its hop-wise decomposition reveals how information is routed across time as well as space.

Motivated by this result, we ask whether regularisation should target multi-hop pathways rather than individual weights. Standard L1 regularisation acts locally on single connections and therefore does not directly constrain the pathways that support computation. We instead regularise the resolvent, inducing temporally structured sparsity that better aligns routing with task demands and improves performance relative to L1 regularisation.

Together, these results suggest that computation in recurrent networks is better understood through multi-hop pathways than through individual connections. More broadly, this provides a concrete bridge between structure, dynamics, and function in artificial neural networks (ANNs).

2 Related Work

A central challenge in network neuroscience and ML is linking structural connectivity to function Honey et al. (2009); Vázquez-Rodríguez et al. (2019); Park & Friston (2013). Increasingly, this is framed as a communication problem: how does information propagate through a network to produce functional interactions Goñi et al. (2014); Fakhar & Hilgetag (2022)? Shortest-path models are limited because they assume global knowledge of network structure Seguin et al. (2023); Achterberg et al. (2023); Seguin et al. (2018), motivating walk-based models in which influence is mediated by ensembles of indirect routes.

One such model is communicability, which aggregates walks of all lengths,

$$C = e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}, \tag{1}$$

where A is the adjacency matrix Estrada et al. (2012); Estrada & Benzi (2014); Crofts & Higham (2009). Related broadcasting models have been shown to capture functionally relevant influence patterns in brain networks Fakhar et al. (2024) and have been used to regularise RNNs toward brain-like modular and small-world structure Achterberg et al. (2023).

Crucially, these graph measures encode assumptions about dynamics: they are meaningful when the system implements the propagation process they describe Zamora-López & Gilson (2024). For example, communicability corresponds to a continuous-time cascade, whereas the resolvent corresponds to a leaky cascade,

$$R = (I - \gamma A)^{-1} = \sum_{k=0}^{\infty} (\gamma A)^k, \tag{2}$$

with $\gamma < 1/\lambda_{\max}$ to ensure convergence. This motivates our central question: can such multi-hop measures reconstruct the spatial and temporal structure of computation in ANNs?

3 Methods

3.1 Tasks

We trained RNNs on a family of modular temporal integration tasks designed to probe different input-output routing structures. Inputs consisted of noisy features, X, grouped into modules and presented over time, with targets defined as functions of the underlying module-level features. Across tasks, the network had to recover either module means, μ, directly (module averaging as seen in Figure 2a), combine them linearly through hierarchical subtraction (Figure 2b) or addition (Figure 2c), or combine them non-linearly through multiplication (Figure 2d). Next, we used a signal on-off task to probe the network's temporal routing structure.

We next trained networks on the more biologically plausible goal–delay–choice maze task defined in Achterberg et al. (2023) (details in Appendix A.1.2).

Finally, we trained our networks on XOR to test our linear multi-hop measures on a nonlinear task. Full task definitions are provided in Appendix A.1.1, and optimal-solution derivations for the modular and maze tasks in Appendix A.2.

We trained one-layer tanh RNNs with Adam. Modular tasks used MSE loss, whereas the maze and XOR tasks used cross-entropy loss. Results are reported over 10 runs for the modular and maze tasks and 100 runs for XOR. Full data splits, architectures, training schedules, and hyperparameters are provided in Appendix A.1.1.

3.2 Computation of Input-Output Multi-Hop Measures

We can represent the weight matrices of our RNNs as an adjacency matrix W, where $W \in \mathbb{R}^{n\times n}$. Here, $n = i + h + o$ is the total number of input, hidden, and output nodes, as illustrated in Figure 1. We aggregate contributions across hop lengths by summing powers of W. One convenient way to do so is through the standard definition of the graph resolvent seen in equation 2. We adapt this by introducing a parameter α such that:

$$\gamma = \frac{\alpha}{\lambda_{max}}, \quad \text{where} \quad 0 < \alpha < 1, \tag{3}$$

and setting $A = W$ we arrive at:

$$R = (I - \alpha W^*)^{-1} = \sum_{k=0}^{\infty} (\alpha W^*)^k = I + \alpha W^* + (\alpha W^*)^2 + \cdots, \tag{4}$$

where α controls the influence of long walk effects and $W^* = \frac{W}{\lambda_{max}}$. We train RNNs on tasks with a sequence length L, so only a finite range of hop lengths can influence the output. We therefore use a truncated resolvent to avoid attributing functional influence to walks longer than those available within the task horizon:

$$R = \sum_{k=2}^{L+1} (\alpha W^*)^k. \tag{5}$$

We extract the input-to-output sub-matrix of R to obtain the input-to-output influence map, denoted as R_{io}. Specifically, $R_{io} \in \mathbb{R}^{i\times o}$ captures the signed weighted sum over walks between each input and output node, as seen in Figure 1. For all plots we use $\alpha = 0.8$. Similarly, we can compute the input-to-output routing structure associated with an individual hop length by computing W^k and extracting its input-to-output block, giving W^k_{io}.

3.3 Multi-Hop Aware Regularisation: R-RNN

In supervised learning, an RNN is trained by adjusting its parameters through backpropagation and gradient descent to minimise a predefined cost function. Typically, this objective combines a task loss, with an additional weighted sparsity regularisation term, β, to promote generalisation. The overall loss is given by:

$$L = L_{\text{task}} + \beta L_{\text{sparsity}}. \tag{6}$$

A common choice for L_{sparsity} is an L1 penalty, where the network is trained to minimise the magnitude of its weights, $L_{\text{sparsity}} = \sum |W|$. In contrast, we train RNNs using a communication based regulariser, where $L_{\text{sparsity}} = \sum |R_{io}|$. Importantly, R is computed from W, rather than the normalised W^*, to prevent the network from trivially reducing R_{io} by increasing λ_{max}. This ensures that reductions in R_{io} reflect a genuine pruning of communication pathways within the network. When R_{io} is used in our regulariser, the truncated sum in equation 5 runs from $k = 1$ to $L + 1$, so that direct one-hop contributions are also penalised. This formulation yields two classes of regularised models: L1-RNNs with $L_{\text{sparsity}} = \sum |W|$, and R-RNNs with $L_{\text{sparsity}} = \sum |R_{io}|$.

For each task, we trained R-RNNs and L1-RNNs across task-specific β sweeps using 10 random seeds per setting. Full regularisation grids and training details are provided in Appendix A.5.

4 Results

4.1 Neural Networks as a Graph

We briefly summarise the graph interpretation underlying our analysis; a fuller intuition is provided in Appendix A.6. A neural network can be represented as a directed graph defined by A, where

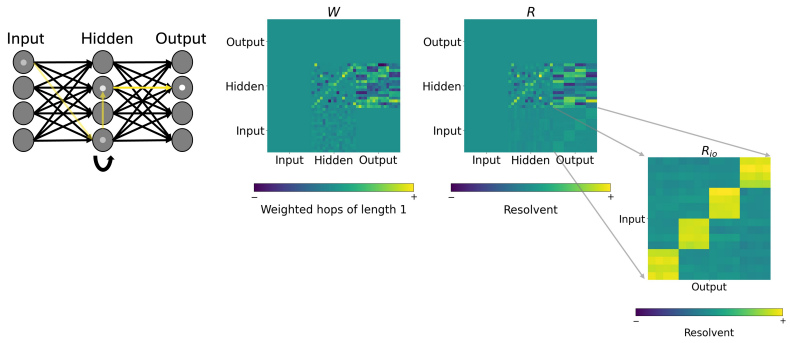

Figure 1: **Computing the multi-hop input-output map of an RNN.** We compute the total input–output influence of multi-hops in an RNN using the resolvent.

edges correspond to weighted connections between units. Powers of this matrix encode multi-hop pathways: $(A^k)_{ij}$ gives the total weighted contribution of all length k walks from node i to node j. In an RNN, recurrence allows inputs to influence outputs through multiple hop lengths over time. We therefore summarise these pathways using the aggregate input-output map R_{io}, as defined in Section 3.2 and illustrated in Figure 1. We then analyse individual hop contributions, W_{io}^k, to test whether specific hop lengths correspond to task relevant time steps.

4.2 Multi-Hop Pathways Recover Learned Input-Output Structure but the Weights do not

We aim to show that the RNNs learned solutions can be reconstructed from multi-hop measures. For the task shown in Figure 2a, we expect the functional routing structure between inputs and outputs to mirror the structure of the optimal solution, as seen in Figure 2a. This shows for the RNN to achieve the optimal MSE, it must aggregate within module information over time.

We can gain insight into how the RNN computes by viewing information as particles propagating through a directed graph as defined by the RNN's weight matrices. Inputs from each module enter the recurrent state through W_{ih}, evolve under the recurrent dynamics W_{hh} for L time steps, and are read out through W_{ho}. To ensure the model preserves each module's information without cross module contamination during recurrent processing, one might expect W_{hh} to be block structured. Because we train networks with $i = h = o = F_{\text{all}}$, the model is not forced to compress representations, so block structure in W_{hh} would be a plausible strategy. However, we do not observe this directly in W_{hh} (Figure 2a). In contrast, when we account for multi-hop interactions by computing R_{io}, the expected module structure emerges clearly (Figure 2a). These results show that weights alone do not recover the network's learned input-output structure, whereas multi-hop measures do.

We next ask whether multi-hop measures also explain tasks requiring different operations between modules. To test this, we consider tasks in which information is combined hierarchically across modules (Figure 2). In the subtraction task, the network must recover individual module means from a cumulative hierarchy, requiring successive subtraction and inhibitory pathways (Figure 2b). The addition task inverts this structure, requiring successive summation of module means (Figure 2c). For the product task, the optimal solution is less intuitive because the mapping is nonlinear, so we define it using the task Jacobian (Appendix A.2).

We clearly see that R_{io} has a higher correlation with the optimal solution for all tasks than W_{hh} (Figure 2). Because trained models achieve near optimal loss, we do not expect R_{io} to reconstruct the optimal solutions exactly for each task. As such, R_{io} offers an interpretable estimate of the information routing learned by the network, recovering the key modular structure even when the match to the analytic optimum is not exact.

4.3 Hop-Wise Decomposition Reveals Temporal Routing

So far we have shown when the RNN receives signals of the same structure for all time steps, R_{io} reconstructs the spatial structure of the optimal solution. We aim to also understand the temporal

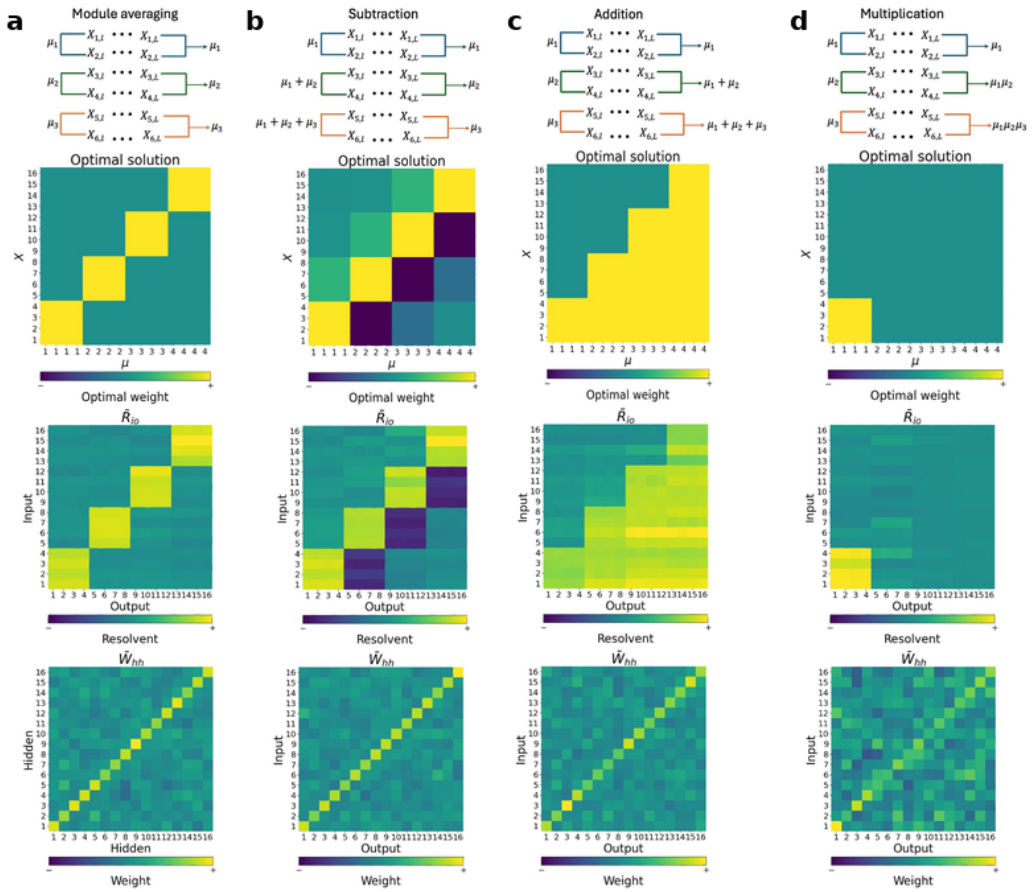

Figure 2: **Task schematics and optimal solutions: the resolvent recovers the solution across all tasks, whereas the hidden weights do not. a**, Module averaging yields strong within-module weights. We compare the optimal solution for the module averaging task with the mean R_{io} and W_{hh}, and their associated SEM. Correlation with optimal solution (Pearson): R_{io} (0.9892 ± 0.0016) and W_{hh} (0.2383 ± 0.0193) **b**, Subtraction yields negative weights between adjacent modules. Analogous comparisons for the subtraction task. Correlation with optimal solution: R_{io} (0.9903 ± 0.0003) and W_{hh} (0.1940 ± 0.0134). **c**, Addition yields hierarchical positive inter-module weights. Analogous comparisons for the addition task. Correlation with optimal solution: R_{io} (0.9351 ± 0.0105) and W_{hh} (0.098 ± 0.0143). **d**, Multiplication yields an input-output map as defined by the task Jacobian. Analogous comparisons for the multiplication task. Correlation with optimal solution: R_{io} (0.9210 ± 0.0102) and W_{hh} (0.0305 ± 0.0221).

routing patterns of the RNN through the lens of hops. The current time agnostic task setup hides the effects individual hop contributions have on the models temporal processing, hence we adapt the module averaging task such that the RNN receives oscillating "on" and "off" signals through time. Specifically, the model receives the same modular signal as seen in Figure 2a for even time steps (L, $L-2$, $L-4$), and then receives standard normally distributed noise at odd time steps ($L-1$, $L-3$).

How do we expect walks of different lengths to contribute to the RNNs temporal processing for this task? Another way to view this question is as the following: if a signal arrives at the RNN input at time t, what walk length k is required for it to influence the network's output? A signal arriving at the input of the one-layer RNN at $t = L-2$ must remain in the hidden state for two time steps before contributing to the output at $t = L$. It therefore follows a walk of length 4: $W_{ih} \to W_{hh} \to W_{hh} \to W_{ho}$. As such, we expect that hops of length k route inputs arriving at $t = L-(k-2)$. In other words, even numbered walks process signal inputs and odd walks process non-signal inputs. The signal has the same modular structure as the module averaging task, so we

expect the network to route signal information in the same pattern as given by the task's optimal solution (Figure 2a).

Figure 3a shows the input-output influence for $k = 2, 3, 4, 5, 6$. We observe the expected modular structure for even k, whereas it disappears for odd k. This alternating pattern indicates that W_{io}^k faithfully reconstructs the RNN's expected temporal routing structure. Interestingly, for odd k we see weak block structure that is not task relevant. This is expected, since even length walks must propagate through the recurrent layer to route signal information to the outputs with the correct modular structure. Consequently, both even and odd walks transverse the same structured W_{hh}, which can impose a residual block structure even when the timestep carries no functionally useful signal. This highlights a key difference between the Jacobian and hop based measures. The Jacobian measures input dependant sensitivity; for timesteps containing only noise, it can be 0 because the networks outputs are insensitive to those inputs. In contrast, hop based measures reveal the routing pathways through which input signals can propagate.

Because individual hops process information presented at different times, we analysed the maze task by summing the hops associated with the goal and choice periods separately. This allowed us to examine how the network processes goal and choice information independently. Figure 3b shows that the resulting goal and choice resolvents reconstruct the model's corresponding empirical input–output maps (see Appendix A.2.2).

Although our hop analysis captures only linear input–output maps, it remains informative for nonlinear tasks. In the temporal XOR task (Figure 3c), W_{io}^k displayed a stronger column motif in networks with negatively sloped decision boundaries and a stronger diagonal motif in networks with positively sloped boundaries. This relationship was consistent across trained networks, where negative motif preference scores, indicating greater similarity to the column motif, were associated with negative boundary slopes, whereas positive scores, indicating greater similarity to the diagonal motif, were associated with positive slopes. Furthermore, networks whose W_{io}^k matrices correlated more strongly with either reference motif tended to have lower boundary curvature. Thus, W_{io}^k reveals interpretable properties of the decision strategies learned by the networks, even when the underlying input–output relationship is strongly non linear.

4.4 R-RNNs Outperform L1-RNNs by Encouraging Sparsity Through Time

We now recognise that a network's spatial and temporal routing patterns can be reconstructed through multi-hop contributions, where the hop order directly encodes the temporal structure of information propagation. This challenges traditional approaches that rely solely on one-hop contributions, as captured by the weights, such as L1 regularisation.

L1 regularisation encourages network sparsity by penalising the weight magnitude. However, sparse weights do not necessarily imply sparse functional pathways, as a small number of strong connections can still induce dense multi-hop routing. In response to this, we introduce R-RNNs, which directly regularise multi-hop pathways, as outlined in the methods.

Figure 4a,b shows that, despite having larger weights, R-RNNs achieve substantially smaller resolvent magnitudes than L1-RNNs. Unlike L1 regularisation, the resolvent captures weight placement and sign, and thus communication pathways. In the maze task (Figure 4c), both models achieve similar R_{io} magnitudes, but the R-RNN shows a better test-loss vs R_{io} trade-off, reaching solutions with both low test loss and low R_{io}.

By comparing the test MSE in Figure 4, we observe that, for all tasks, the best R-RNN achieves lower test MSE loss than the best L1-RNN (details of significance are in Appendix A.4). In Figure 4a, the total hop magnitude of the R-RNN is consistently lower across all k. Crucially, the R-RNN equalises hop contributions, matching the task's routing structure in which the same signal is received at each time step (Figure 2a). This reflects multi-hop sparsity: the network suppresses redundant pathways across all temporal routes, something L1-RNNs cannot achieve by acting only on individual weights. Additionally, in Figure 4b, the R-RNN selectively suppresses hop communication at $k = 2, 4, 6$, corresponding to signal-carrying time steps (Figure 3), while leaving other hops relatively unchanged. This shows that R-RNNs induce temporally structured sparsity that aligns with the task structure even in the presence of sparse signal information. Importantly, on the maze

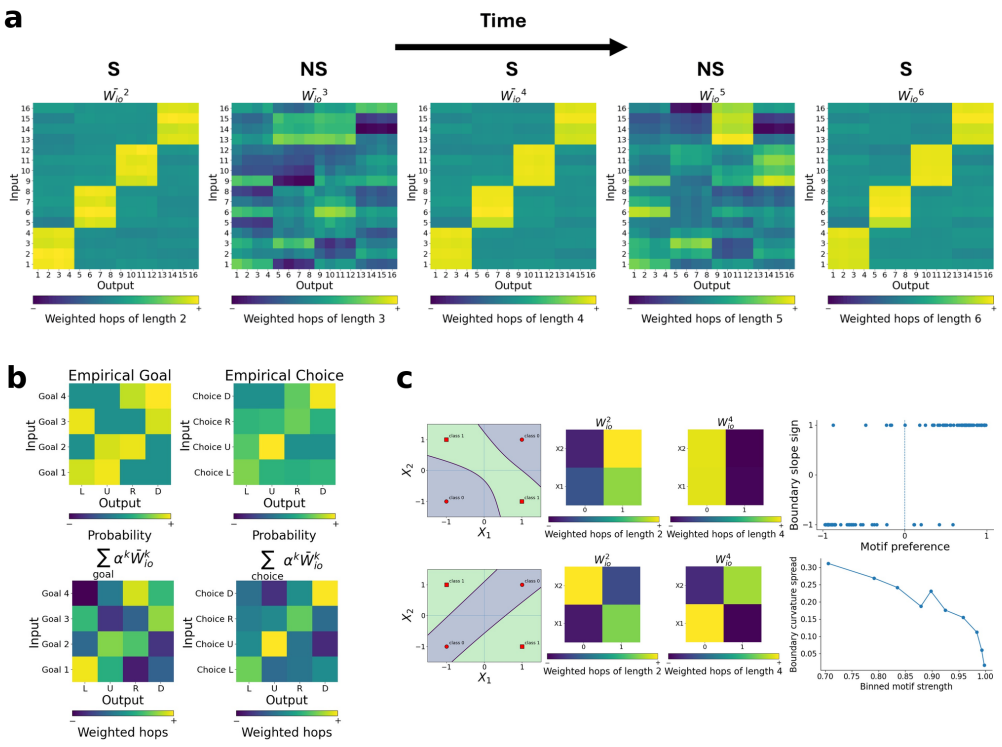

Figure 3: W_{io}^{k} **reveals how the network temporally routes information across tasks.** **a**, The network receives signal (S) and standard normally distributed noise, or no signal (NS), at alternating time steps. As k increases, W_{io}^{k} corresponds to inputs arriving progressively later in time before the final RNN output. **b**, Emperical model input–output maps for the maze task compared with goal and choice resolvents (goal: $r = 0.710 \pm 0.036$ and choice: $r = 0.886 \pm 0.014$). **c**, For the XOR task, W_{io}^{k} reveals structural motifs that correlate with the sign and curvature of the decision boundary.

task, the best R-RNN achieves the highest accuracy (0.8525 ± 0.0602 SEM; $\beta = 0.002$), compared with the best L1-RNN (0.7540 ± 0.0556; $\beta = 0.0001$) and vanilla RNN (0.7880 ± 0.0464).

Furthermore, R-RNNs also perform more robustly under stronger regularisation, indicating that resolvent based sparsity is better aligned with function. As β increases, L1-RNNs exhibit a rapid degradation in test performance as total weight magnitude decreases, whereas R-RNNs maintain a small MSE loss even at high levels of resolvent sparsity (Figure 4a,b). For the maze task, decreasing R_{io} yields a greater reduction in test loss for R-RNNs than for L1-RNNs (Figure 4c). This indicates that R-RNNs preserve task relevant computation under strong regularisations across tasks. In other words, R-RNNs achieve substantial communication sparsity with only a relatively small loss in performance, demonstrating that multi-hop sparsity is more functionally aligned than weight sparsity.

5 Discussion

We demonstrate that the RNNs input-output mapping is not accounted for by a one hop model as described by W, even when W_{hh} is not forced to compress representations. Instead, multi-hop measures recover the spatial-temporal routing structure that mediates information flow from inputs to outputs. This holds across tasks which require different computations: averaging, subtraction, addition, and multiplication, which are likely important for neural information processing. This suggests that multi-hop measures may provide a useful model of communication across a broad range of tasks.

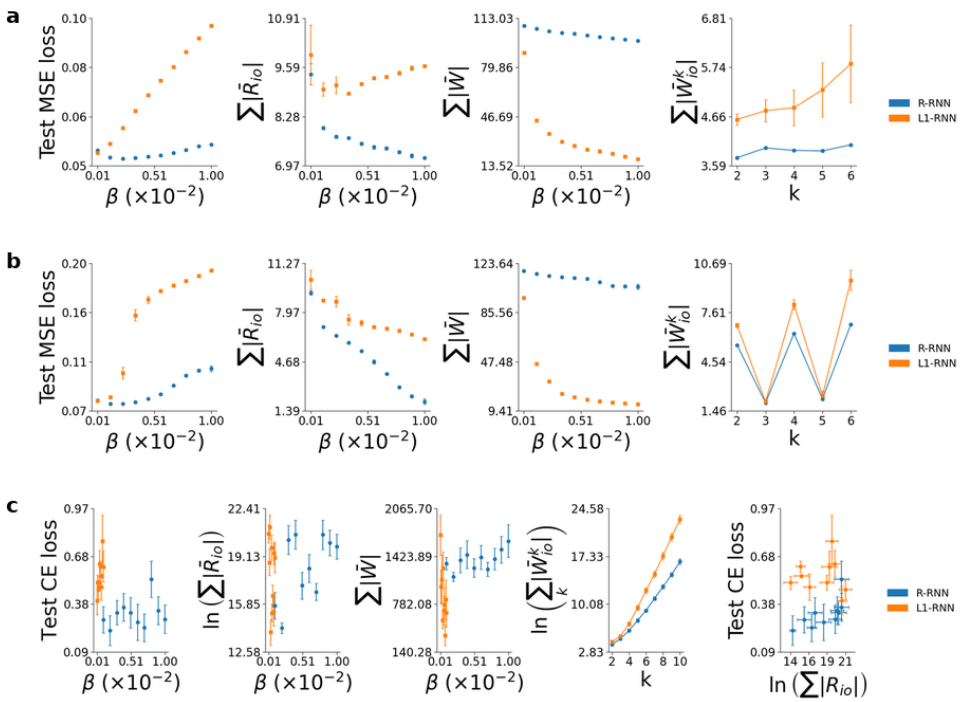

Figure 4: **R-RNNs outperform L1-RNNs by enforcing sparsity through time.** **a**, Module averaging task: we compare the test performance and L_{sparsity} terms of our trained R and L1-RNNs over β. Additionally, we compare how the total multi-hop magnitude across k varies for the best R-RNN ($\beta = 0.23 \times 10^{-2}$) and the best L1-RNN ($\beta = 0.01 \times 10^{-2}$). **b**, Oscillating on-off signal task: Analogous comparisons. The best R and L1-RNN share the same β value as in the module averaging task. **c**, Analogous comparisons for the maze task where the R and L1-RNN are plotted over a different range of β. The R-RNN shows a better test loss R_{io} scaling than the L1-RNN.

5.1 Why Multi-Hop Measures Matter for Understanding Computation

We have shown hop based measures can reveal the input-output routing pathways across both space and time. But why do we care about routing pathways if the Jacobian can tell us functional relationships between the inputs and outputs Pizarroso et al. (2022); Yeung et al. (2010); Paus (2005)? While the Jacobian quantifies input-output sensitivity, characterising a biological circuit's Jacobian across the diversity of naturalistic stimuli is impractical, as it would require extensive controlled experiments. Alternatively, hop based measures can describe a networks "routing capacity" just from its connectivity, therefore providing us with an input agnostic method to characterise which pathways a network can use. As such, we can compare network's "routing capacities" to infer how they would respond to novel stimuli, without requiring new input–output measurements from real neural data.

5.2 Structure-Function Relationships in Biological Networks

Importantly, hop measures allow us to connect a network's structure to its function. Biological networks are highly non-random, often demonstrating modular, hierarchical, or small world structures Meunier et al. (2010); Sporns & Honey (2006). These structural motifs are hypothesised to confer functional benefits like efficient information transfer and flexible learning Gu et al.; Latora & Marchiori (2001). An implicit assumption is that structural modularity is sufficient for functional modularity; however, this assumption has recently been challenged Béna & Goodman (2024). Multi-hop measures offer a fresh perspective: structural modularity only translates to functional modularity if

the walk structure respects module boundaries. Computing R_{io} for networks with varying structural modularity could reveal why the structure-function relationship breaks down in some regimes.

Naturally, these structural motifs assume specific communication patterns, which may also be over represented in biological networks. We can visualise the input-output communication patterns under different connectivity assumptions, which can help us isolate what kinds of computations are supported by structure alone. Additionally, there can exist different connectivity structures with the same R_{io}. It would be interesting to test whether structures we typically characterise as distinct nevertheless exhibit similar routing patterns. Moreover, decomposing a networks computation into individual A^k terms, we can characterise different connectivity profiles in terms of their capacity to route information over time.

5.3 Rethinking Regularisation and Network Design

If biological networks optimise for efficient communication rather than minimal synaptic strength, then sparsity should be defined over pathways rather than individual connections. In this sense, L1 regularisation can be viewed as a special case of a broader objective: it imposes sparsity at the level of individual weights, but does not constrain how signals propagate through multi-hop pathways. Our results show that this is insufficient for regularising multi-hop communication. R-RNNs achieve this by inducing sparsity in multi-hops, selectively reducing communication to match the task's routing structure. More broadly, the goal of regularisation is to balance task performance with sparsity, and we find R-RNNs better satisfy this trade-off, achieving communication sparsity with minimal loss in performance. This suggests that functionally aligned sparsity emerges when constraints are placed on how information flows, rather than solely on the underlying parameters.

6 Limitations and Future Directions

The resolvent provides a useful multi-hop summary of network communication, but it also assumes a particular dynamical prior: a leaky cascade. Other systems may be better described by alternative measures, such as communicability, which corresponds to a continuous-time cascade and weights walks differently Zamora-López & Gilson (2024).

Extending these ideas beyond shallow recurrent models remains an open direction. Applying analogous multi-hop characterisations to deeper or more complex architectures (e.g.Transformers, or mixtures-of-experts) will require careful choices of what constitutes a node and an edge Joshi (2020); Bae et al. (2025); Muennighoff et al. (2025).

There may be deeper connections to information theoretic quantities. In particular, do high R_{io} pathways carry more mutual information between inputs and outputs? If so, R_{io} can be used as an additional graph metric for characterising differences in network dynamics between healthy and clinical populations Zhang et al. (2018); Wang et al. (2015). Multi-hop measures may provide a computationally efficient approximation to input-output structure compared with full input-output sensitivity analysis, which typically depends on repeated evaluations across many inputs

More broadly, the same framework could extend beyond neural networks to other physical computing systems, any setting in which dynamics unfold on a graph and multiple hops shape function Liu et al. (2023); Barham et al. (2022).

7 Conclusion

In this work, we show that a network's computation can be viewed through the lens of a graph, where the input-output map can be reconstructed by incorporating multi-hop information, which is not apparent from the raw weight structure alone. This perspective offers a new route for linking structure and function in both biological and artificial networks, and provides a way to probe and compare internal routing strategies for interpretability beyond activation based measures. We show that regularising these multi-hop input–output maps in RNNs improves test performance by inducing temporally structured sparsity. Ultimately, our results point toward a more unified view of computation grounded in structure, dynamics, and the walks that connect them.

A Appendix

A.1 Task Definitions

A.1.1 Modular Tasks

Our RNNs are initially trained on a task that requires them to average noisy modular features over time. We first define the task's hidden parameters given by:

$$\mu_h \sim \mathcal{N}(0, \sigma_\mu^2), \tag{7}$$

and then the module parameters

$$\mu_m = G_{hm}\mu_h, \tag{8}$$

where G_{hm} defines the module structure. In our tasks we have two module structures given by

$$G_{hm}^{\text{mod}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad G_{hm}^{\text{add}} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \tag{9}$$

which are shown for tasks with $M = 3$ modules. Next, we define our feature parameters as

$$\mu_f = G_{mf}\mu_m, \tag{10}$$

and $G_{mf} = I_M \otimes \mathbf{1}_F$, where F is the number of features per module. Given the feature parameters, we then take normally distributed noisy samples given by:

$$X \sim \mathcal{N}(\mu_f, \sigma_\epsilon^2 I_F), \tag{11}$$

and N independent samples are taken over L time steps, such that $X \in \mathbb{R}^{N \times L \times F_{\text{all}}}$, where $F_{\text{all}} = FM$ denotes the total number of features. For the signal on-off task used in Sections 4.3 and 4.4, standard normally distributed noise is injected at odd time steps (i.e. $L-1$, $L-3$, $\dots$). For each task, the network receives X as input and must predict the following target parameters:

$$\mu_t = f_t(\mu_m), \tag{12}$$

where f_t is some function that maps our task parameters to the network's target parameters.

By using different choices of G_{hm} and f_t, we can define our set of tasks as shown in Figure 2. First, we define the modular averaging task, which uses G_{hm}^{mod} and $f_t(\mu_m) = \mu_m$ (Figure 2a). Next, we define the subtraction and addition tasks. The subtraction task uses G_{hm}^{add} with $f_t(\mu_m) = A^{\text{sub}}\mu_m$ (Figure 2b), while the addition task uses G_{hm}^{mod} with $f_t(\mu_m) = A^{\text{add}}\mu_m$. We define A^{sub} and A^{add} as:

$$A^{\text{sub}} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \quad \text{and} \quad A^{\text{add}} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \tag{13}$$

Finally, we define the multiplication task, which uses G_{hm}^{mod} with $f_t(\mu_m)$ as:

$$\begin{pmatrix} \mu_1 \\ \mu_1\mu_2 \\ \mu_1\mu_2\mu_3 \end{pmatrix}, \tag{14}$$

as shown in Figure 2d. Across all modular tasks, the module level targets are shared among features within the same module, and N independent samples are drawn to form $\mu_t \in \mathbb{R}^{N \times F_{\text{all}}}$.

A.1.2 MAZE TASK

The maze task is represented as a square grid in which the network begins at the centre and must select the direction that provides the shortest path to a specified goal, following the task described by Achterberg et al. (2023) and illustrated in Figure 5.

The network receives an input tensor of dimensions $N \times L \times 4$. The final dimension represents the four possible movement directions on the grid: up, left, right, and down. During the goal step, one direction is represented using a one-hot encoded vector. During the delay step, the network receives a vector containing only zeros. During the choice step, two possible directions are presented, and the network must select the direction that enables the most efficient route to the previously presented goal.

The network is trained on a predefined set of goal and choice combinations, as detailed in Table 1.

Goal	**Choices**	**Correct Choices**
1	UR, LR	U, L
2	UL, RD	U, R
3	RD, UL	D, L
4	LD, LR	D, R

Table 1: All goal and choice configurations presented to the network for the maze task. Choice combinations containing L, together with their corresponding correct choices, are highlighted in red.

For the maze task, we trained the RNNs for 20 epochs. Each trial comprised of 20 goal presentation time steps, a 10-step delay, and 20 choice time steps. When training the R-RNN, we regularised the input–output resolvent using $k = 10$.

A.1.3 XOR TASK

For this task, we trained the network on a temporal XOR paradigm. The network received one of four possible input conditions, corresponding to the different combinations of 1 and -1 presented across two input channels. To introduce a temporal component, the selected input combination was repeated over multiple time steps, and the network produced its class prediction at the final time step.

A.2 OPTIMAL AND EMPIRICAL SOLUTIONS

A.2.1 MODULAR TASKS

In this section, we will discuss how we arrived at the optimal solutions as seen in Figure 2. We can represent our module averaging, subtraction and addition tasks as a linear regression problem:

$$X = A\mu + \epsilon, \tag{15}$$

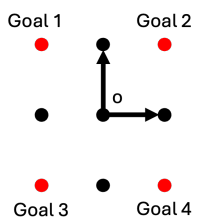

Figure 5: **Visual representation of the maze task**. The network begins at the centre of the maze, denoted by o, and is presented with one of four possible goals, shown in red. It must then select between two available directions to follow the shortest path to the specified goal.

where $X \in \mathbb{R}^n$, such that $n = F_{all} \times L$, defines our feature vector for all modules over time, $A \in \mathbb{R}^{n \times M}$ defines our task structure, $\mu \in \mathbb{R}^M$ defines our module means, and $\epsilon \in \mathbb{R}^n$ defines the feature noise. Using the standard result of the posterior mean under Bayesian linear regression we obtain:

$$\hat{\mu} = W_{\text{optimal}} X, \quad \text{where} \quad W_{\text{optimal}} = (A^\top A + \frac{\sigma_\epsilon^2}{\sigma_\mu^2} I)^{-1} A^\top, \tag{16}$$

such that $W_{\text{optimal}} \in \mathbb{R}^{M \times n}$. We then reshape such that $W_{\text{optimal}} \in \mathbb{R}^{M \times F_{\text{all}} \times L}$, average over the temporal dimension L, and duplicate the resulting module level solutions. This produces a final matrix $W_{\text{optimal}} \in \mathbb{R}^{F_{\text{all}} \times F_{\text{all}}}$, as shown in Figure 2.

For the multiplication task (Figure 2d) we define the module level task structure with input vector I, where $I = (\mu_1, \mu_2, \ldots, \mu_M)^\top$, and output vector O, where $O = (\mu_1, \mu_1\mu_2, \ldots, \prod_{k=1}^{M} \mu_k)^\top$. The inputs cannot be represented as a linear combination of the network outputs, as governed by equation 15. Instead, we calculate the input-output sensitivity by computing the task's Jacobian, J, where each element is given by:

$$J_{ij} = \frac{\partial O_j}{\partial I_i}. \tag{17}$$

Since $O_j = \prod_{k=1}^{j} \mu_k$, we obtain:

$$J_{ij} = \begin{cases} \dfrac{O_j}{\mu_i}, & i \leq j, \\ 0, & i > j. \end{cases} \tag{18}$$

In practice, the module level means are sampled from normal distributions as shown in equation 7, and given that the network is trained on N independent samples, we have $\mathbb{E}[\mu_i] = 0$. For example, when $M = 3$, the Jacobian displayed in reverse row order takes the form

$$J = \begin{pmatrix} 0 & 0 & \mu_1\mu_2 \\ 0 & \mu_1 & \mu_1\mu_3 \\ 1 & \mu_2 & \mu_2\mu_3 \end{pmatrix}, \tag{19}$$

where, in expectation, all multiplicative terms vanish, leaving only $J_{1,1}$ populated. By duplicating this module level Jacobian structure across features, we obtain the corresponding optimal solution, as shown in Figure 2d.

A.2.2 MAZE TASK

To compare R_{io} with the model's observed input–output behaviour, we constructed separate goal and choice conditioned prediction matrices. For each trial, the goal identity was determined from the active goal channel during the goal period, while choice availability was identified from the active choice channels during the choice period. We then averaged the model's output probabilities separately for each goal and for trials in which each choice was available, producing 4×4 matrices whose columns corresponded to the four possible output choices as seen in Figure 3b. These matrices were computed for each trained model and then averaged across models.

The empirical goal and choice matrices in Figure 3b can be understood intuitively from the structure of the task. For example, when Goal 1 is presented, the network tends to assign greater output probability to L and U. As shown in Figure 5, these are the two actions that can lead from the centre of the maze to Goal 1. The same reasoning applies to the response patterns associated with the other goals.

The empirical choice matrix can similarly be interpreted using Table 1. Across the five task conditions in which L is available, L is the correct response in two cases, whereas U, R, and D are each correct in one case. Thus, the expected response proportions are $2/5$ for L and $1/5$ for each of the other actions. Consistent with this distribution, Figure 3b shows that when L is among the available choices, the network assigns more probability to L than to U, R, or D, with the latter three receiving approximately equal probabilities. This indicates that the network has learned the choice distributions underlying the task and that multi-hops capture the same structure.

A.3 XOR MOTIF PREFERENCE AND MOTIF STRENGTH DEFINITIONS

For each model and hop, we quantified the similarity of the two by two input–output matrix to two reference motifs: a diagonal motif,

$$T_{\text{diag}} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix},$$

and a column motif,

$$T_{\text{col}} = \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix}.$$

Motif similarity was calculated separately at each k as the absolute Pearson correlation between the flattened input–output matrix and the flattened reference motif. These absolute correlations were then averaged across hops for each model. This produced two average correlation values for each network: one for the diagonal motif, $\overline{\rho}_{\text{diag}}$, and one for the column motif, $\overline{\rho}_{\text{col}}$. We defined the motif preference as seen in Figure 3C as:

$$D = \overline{\rho}_{\text{diag}} - \overline{\rho}_{\text{col}},$$

where positive values indicated a stronger diagonal motif and negative values indicated a stronger column motif. We plotted this score against the sign of the model's mean decision boundary slope.

For the boundary curvature spread vs binned motif strength plot seen in Figure 3C, we calculated the boundary curvature from the variability of the decision boundary orientation. Slopes were converted to orientation angles using $\theta_i = \arctan(s_i)$. Boundary curvature spread was quantified as the standard deviation of these angles. Motif strength was defined as: $S = \max\left(\overline{\rho}_{\text{diag}}, \overline{\rho}_{\text{col}}\right)$.

A.4 COMPARISON OF R-RNN AND L1-RNN TEST LOSS

In Figure 4a,b the differences between the best R and L1-RNNs test loss is not visully obvious. Therefore, we test whether the R and L1-RNNs differed in their generalisation performance by comparing their test losses across 10 matched runs. The network with the lowest validation loss, for each model class, was selected before its test performance was evaluated.

On the module averaging task (Figure 4a), the validation selected R-RNNs achieved a mean test loss of $\text{MSE} = 0.049145 \pm 0.000124$ compared with $\text{MSE} = 0.050745 \pm 0.000171$ for the validation-selected L1-RNNs. This corresponded to a 3.15% reduction in test loss. A paired statistical comparison confirmed that the difference was significant, with $t(9) = 7.54$ and $p = 3.55 \times 10^{-5}$.

On the signal on/off task, the validation-selected R-RNNs achieved a mean test loss of $\text{MSE} = 0.078409 \pm 0.000117$, compared with $\text{MSE} = 0.080864 \pm 0.000356$ for the validation-selected L1-RNNs. This represented a 3.04% reduction in test loss, with $t(9) = 8.16$ and $p = 1.89 \times 10^{-5}$.

Thus, although the absolute differences in test loss appeared small visually, the R-RNNs consistently achieved lower test loss across matched runs, and the differences were statistically significant for both tasks.

A.5 IMPLEMENTATION DETAILS

All models were single-layer tanh RNNs trained with Adam ($\text{lr} = 0.01$) and PyTorch's default initialisation. Modular tasks used a $64{:}16{:}20$ train–validation–test split and were trained for 100 epochs with MSE loss. Maze and XOR used an $87{:}4{:}9$ split and used a cross-entropy loss; maze was trained for 20 epochs and XOR for 10 epochs. We performed 10 runs for the modular and maze tasks and 100 for XOR. For the modular tasks, an L1 penalty of 0.001 was applied to W_{hh}, except for multiplication. All modular tasks used $F_{\text{all}} = 16$, $M = 4$, $i = h = o = F_{\text{all}}$, $L = 5$, and $N = 2000$.

For the results in Figure 4a, b, each setting was trained for 200 epochs using 10 repeats. Both models were evaluated at 10 equally spaced values of β in $[0.0001, 0.01]$. For the maze task, L1-RNNs used 10 values over β in $[0.0001, 0.001]$, while R-RNNs used 10 values over β in $[0.0001, 0.01]$. We used $\alpha = 0.8$ for all R-RNNs.

Unless otherwise stated, all reported uncertainties and error bars denote ± 1 standard error of the mean (SEM) across 10 independent runs with different random initialisations. All models were trained locally on CPU using an Apple M1 MacBook Pro. Training 10 repeats for 100 epochs, as described in Section 3.1, took roughly 15 s. Training R-RNN and L1-RNN models for 200 epochs across β values, as described in Section 3.3, took roughly 2 mins.

A.6 Neural Networks as a Graph Extended

In graph theory, a network can be represented by nodes and edges, where the structural connectivity is defined by its adjacency matrix, A, as seen in Figure 6a. Alternatively, from a dynamical systems perspective, we can view A as representing the number of particle walks of length 1 between each node of the graph. An interesting property of graphs is that A^k tells you how many walks of length k there are between two nodes, as seen in Figure 6a. For example, $A^2_{13} = 1$ indicates that there is one possible route of length 2 between nodes 1 and 3. From this we see that a network's structure (A) implicitly defines a family of walk patterns (A^k), therefore, if networks transmit information through multiple hops, different network structures give rise to non-trivial differences in network function. For example, Figure 6b shows the A of two structurally different networks; however, both share the same A^2 pattern, implying similar function. This observation raises the question of whether representing ANNs as graphs and modelling their multi-hop structure can provide a link between the network's structure and its function.

By treating an ANN as a directed graph we can construct its adjacency matrix which contains all possible connections. Figure 6c shows the adjacency matrix for two multi-layer-perceptrons (MLPs), A_1 and A_2. Both exhibit populated input-hidden and hidden-output blocks, and their weight distributions are identical (6c). As an MLP has no recurrent connections, a particle can traverse the network only through higher order hops of length two, corresponding to input $\rightarrow$ hidden $\rightarrow$ output paths. To characterise this two hop structure, we compute A^2 for each MLP. As expected, only the input–output block is populated. Despite identical weight distributions, the resulting input–output maps differ, reinforcing the non-triviality of hop structure.

In contrast to the MLP, introducing recurrence through the hidden–hidden connections in our RNNs allows particles to propagate through higher order hops. To capture the cumulative effect of these multi-hop pathways, we calculate the resolvent as defined in the methods and seen in Figure 6d).

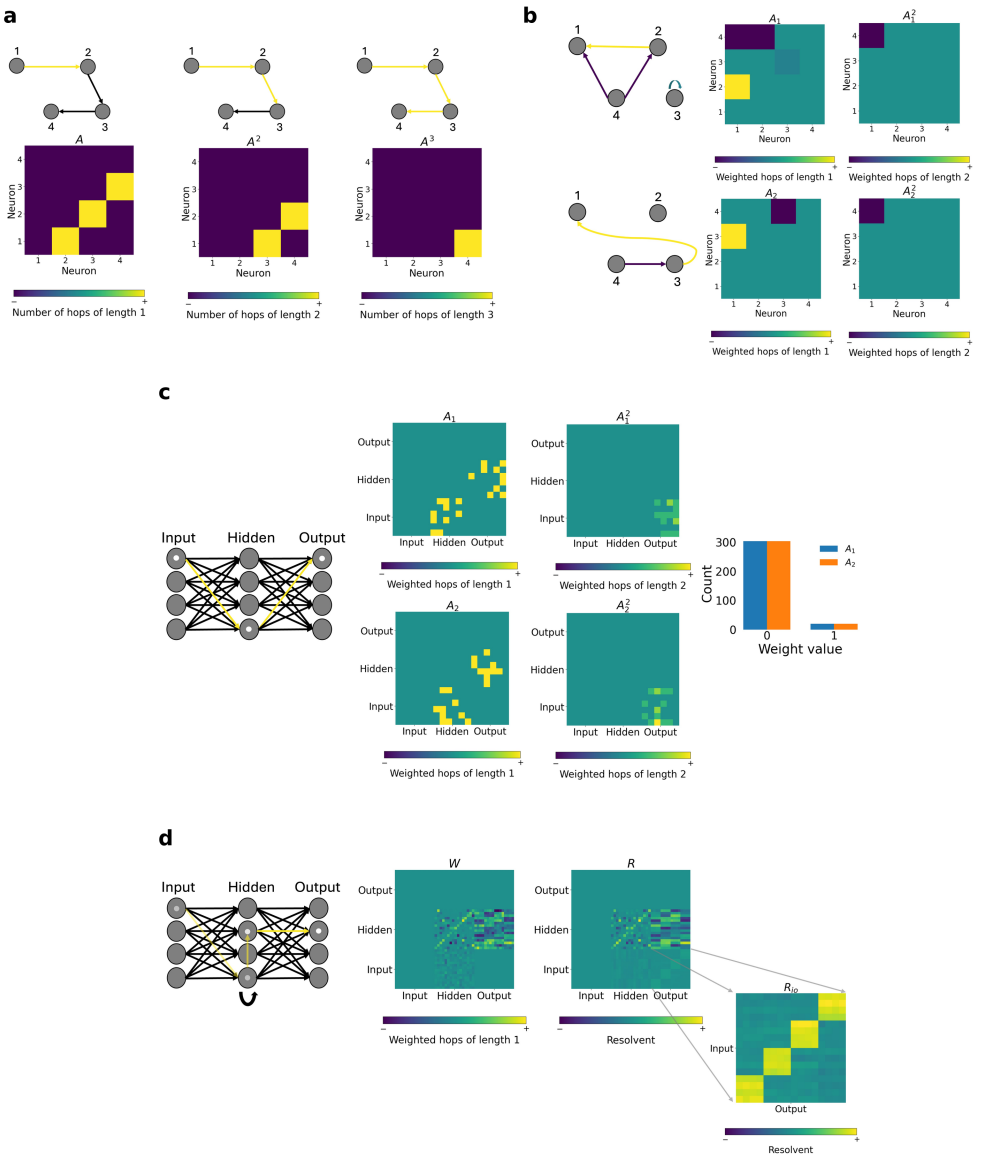

Figure 6: **Networks represented as graphs and their multi-hop structures.** **a**, Arbitrary graphs with one pathway highlighted, illustrated by their corresponding A^k matrix. **b**, Two networks with distinct structures share the same two-hop pattern. **c**, Two MLPs with the same weight distributions have different two-hop patterns. **d**, We compute the total input–output influence across multiple hops in an RNN using the resolvent.

AI USE STATEMENT

In this work, we used generative AI tools for no tasks requiring mandatory disclosure. We did not use generative AI tools to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide ingredients for or assist with mathematical proofs, propose or refine hypotheses, design or provide feedback on the research methodology or experiments, implement the proposed methods, assist with translation, clean or reformat datasets, support qualitative or thematic data analysis, or interpret results. No other tasks requiring mandatory disclosure were applicable to this work. Additionally, we used generative AI tools to create or edit software code, brainstorm, and improve the readability of the manuscript. We reviewed all AI-assisted work. We inspected and tested AI-assisted code, independently evaluated suggestions produced during brainstorming, and checked all language edits for accuracy and consistency with the intended scientific meaning. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 describes the implementation of the resolvent, together with the basic task definitions and model parameters. Additional implementation details are provided in Appendix A.5, while complete descriptions of all tasks are provided in Appendix A.1. Derivations of the optimal solutions for the modular tasks and empirical solutions for the maze task are presented in Appendix A.2. The motif preference and motif strength measures used in Figure 3c are defined in Appendix A.3. Finally, Appendix A.4 reports the statistical analysis supporting the significant difference in test loss between the R-RNNs and L1-RNNs shown in Figure 4a,b.

REFERENCES

Jascha Achterberg, Danyal Akarca, D. J. Strouse, John Duncan, and Duncan E. Astle. Spatially embedded recurrent neural networks reveal widespread links between structural and functional neuroscience findings. *Nature Machine Intelligence*, 5(12):1369–1381, December 2023. ISSN 2522-5839. doi: 10.1038/s42256-023-00748-9. URL https://www.nature.com/articles/s42256-023-00748-9.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-Recursions: Learning Dynamic Recursive Depths for Adaptive Token-Level Computation, July 2025. URL http://arxiv.org/abs/2507.10524. arXiv:2507.10524 [cs].

Paul Barham, Aakanksha Chowdhery, Jeff Dean, Sanjay Ghemawat, Steven Hand, Dan Hurt, Michael Isard, Hyeontaek Lim, Ruoming Pang, Sudip Roy, Brennan Saeta, Parker Schuh, Ryan Sepassi, Laurent El Shafey, Chandramohan A. Thekkath, and Yonghui Wu. Pathways: Asynchronous Distributed Dataflow for ML, March 2022. URL http://arxiv.org/abs/2203.12533. arXiv:2203.12533 [cs].

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives, April 2014. URL http://arxiv.org/abs/1206.5538. arXiv:1206.5538 [cs].

Gabriel Béna and Dan F. M. Goodman. Dynamics of specialization in neural modules under resource constraints, October 2024. URL http://arxiv.org/abs/2106.02626. arXiv:2106.02626.

Jonathan J. Crofts and Desmond J. Higham. A weighted communicability measure applied to complex brain networks. *Journal of the Royal Society, Interface*, 6(33):411–414, April 2009. ISSN 1742-5689. doi: 10.1098/rsif.2008.0484.

Ernesto Estrada and Michele Benzi. Walk-based measure of balance in signed networks: Detecting lack of balance in social networks. *Physical Review E*, 90(4):042802, October 2014. doi: 10.1103/PhysRevE.90.042802. URL https://link.aps.org/doi/10.1103/PhysRevE.90.042802.

Ernesto Estrada, Naomichi Hatano, and Michele Benzi. The Physics of Communicability in Complex Networks. *Physics Reports*, 514(3):89–119, May 2012. ISSN 0370-1573. doi: 10.1016/j.physrep.2012.01.006. URL http://arxiv.org/abs/1109.2950. arXiv:1109.2950 [physics].

Kayson Fakhar and Claus C. Hilgetag. Systematic perturbation of an artificial neural network: A step towards quantifying causal contributions in the brain. *PLOS Computational Biology*, 18(6):e1010250, June 2022. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1010250. URL https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1010250.

Kayson Fakhar, Fatemeh Hadaeghi, Caio Seguin, Shrey Dixit, Arnaud Messé, Gorka Zamora-López, Bratislav Misic, and Claus Hilgetag. *A General Framework for Characterizing Optimal Communication in Brain Networks*. June 2024. doi: 10.1101/2024.06.12.598676.

Joaquín Goñi, Martijn P. van den Heuvel, Andrea Avena-Koenigsberger, Nieves Velez de Mendizabal, Richard F. Betzel, Alessandra Griffa, Patric Hagmann, Bernat Corominas-Murtra, Jean-Philippe Thiran, and Olaf Sporns. Resting-brain functional connectivity predicted by analytic measures of network communication. *Proceedings of the National Academy of Sciences*, 111(2):833–838, January 2014. doi: 10.1073/pnas.1315529111. URL https://www.pnas.org/doi/abs/10.1073/pnas.1315529111.

Shi Gu, Marcelo G. Mattar, Huajin Tang, and Gang Pan. Emergence and reconfiguration of modular structure for artificial neural networks during continual familiarity detection. *Science Advances*, 10(30):eadm8430. ISSN 2375-2548. doi: 10.1126/sciadv.adm8430. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11277393/.

C. J. Honey, O. Sporns, L. Cammoun, X. Gigandet, J. P. Thiran, R. Meuli, and P. Hagmann. Predicting human resting-state functional connectivity from structural connectivity. *Proceedings of the National Academy of Sciences*, 106(6):2035–2040, February 2009. doi: 10.1073/pnas.0811168106. URL https://www.pnas.org/doi/10.1073/pnas.0811168106.

Chaitanya Joshi. Transformers are Graph Neural Networks, February 2020. URL https://graphdeeplearning.github.io/post/transformers-are-gnns/.

Vito Latora and Massimo Marchiori. Efficient Behavior of Small-World Networks. *Physical Review Letters*, 87(19):198701, October 2001. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.87.198701. URL https://link.aps.org/doi/10.1103/PhysRevLett.87.198701.

Siting Liu, Yuan Pu, Peiyu Liao, Hongzhong Wu, Rui Zhang, Zhitang Chen, Wenlong Lv, Yibo Lin, and Bei Yu. FastGR: Global Routing on CPU–GPU With Heterogeneous Task Graph Scheduler. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(7):2317–2330, July 2023. ISSN 1937-4151. doi: 10.1109/TCAD.2022.3217668. URL https://ieeexplore.ieee.org/abstract/document/9931135.

David Marr. *Vision: a computational investigation into the human representation and processing of visual information*. MIT Press, Cambridge, Mass, 2010. ISBN 978-0-262-51462-0 978-0-262-28961-0.

David Meunier, Renaud Lambiotte, and Edward T. Bullmore. Modular and Hierarchically Modular Organization of Brain Networks. *Frontiers in Neuroscience*, 4:200, December 2010. ISSN 1662-4548. doi: 10.3389/fnins.2010.00200. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC3000003/.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, Yuling Gu, Shane Arora, Akshita Bhagia, Dustin Schwenk, David Wadden, Alexander Wettig, Binyuan Hui, Tim Dettmers, Douwe Kiela, Ali Farhadi, Noah A. Smith, Pang Wei Koh, Amanpreet Singh, and Hannaneh Hajishirzi. OLMoE: Open Mixture-of-Experts Language Models, March 2025. URL http://arxiv.org/abs/2409.02060. arXiv:2409.02060 [cs].

Allen Newell. Physical symbol systems. *Cognitive Science*, 4(2):135–183, April 1980. ISSN 0364-0213. doi: 10.1016/S0364-0213(80)80015-2. URL https://www.sciencedirect.com/science/article/pii/S0364021380800152.

Hae-Jeong Park and Karl Friston. Structural and Functional Brain Networks: From Connections to Cognition. *Science*, 342(6158):1238411, November 2013. doi: 10.1126/science.1238411. URL https://www.science.org/doi/full/10.1126/science.1238411.

Tomáš Paus. Inferring causality in brain images: a perturbation approach. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1457):1109–1114, May 2005. ISSN 0962-8436. doi: 10.1098/rstb.2005.1652. URL https://doi.org/10.1098/rstb.2005.1652.

Jaime Pizarroso, José Portela, and Antonio Muñoz. NeuralSens: Sensitivity Analysis of Neural Networks. *Journal of Statistical Software*, 102:1–36, April 2022. ISSN 1548-7660. doi: 10.18637/jss.v102.i07. URL https://doi.org/10.18637/jss.v102.i07.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116 (23):11537–11546, June 2019. doi: 10.1073/pnas.1820226116. URL https://www.pnas.org/doi/full/10.1073/pnas.1820226116.

Caio Seguin, Martijn P. van den Heuvel, and Andrew Zalesky. Navigation of brain networks. *Proceedings of the National Academy of Sciences*, 115(24):6297–6302, June 2018. doi: 10.1073/pnas.1801351115. URL https://www.pnas.org/doi/abs/10.1073/pnas.1801351115. Company: National Academy of Sciences Distributor: National Academy of Sciences Institution: National Academy of Sciences Label: National Academy of Sciences.

Caio Seguin, Olaf Sporns, and Andrew Zalesky. Brain network communication: concepts, models and applications. *Nature Reviews Neuroscience*, 24(9):557–574, September 2023. ISSN 1471-0048. doi: 10.1038/s41583-023-00718-5. URL https://www.nature.com/articles/s41583-023-00718-5.

Olaf Sporns and Christopher J. Honey. Small worlds inside big brains. *Proceedings of the National Academy of Sciences*, 103(51):19219–19220, December 2006. doi: 10.1073/pnas.0609523103. URL https://www.pnas.org/doi/10.1073/pnas.0609523103.

Bertha Vázquez-Rodríguez, Laura E. Suárez, Ross D. Markello, Golia Shafiei, Casey Paquola, Patric Hagmann, Martijn P. van den Heuvel, Boris C. Bernhardt, R. Nathan Spreng, and Bratislav Misic. Gradients of structure–function tethering across neocortex. *Proceedings of the National Academy of Sciences*, 116(42):21219–21227, October 2019. doi: 10.1073/pnas.1903403116. URL https://www.pnas.org/doi/abs/10.1073/pnas.1903403116.

Zhe Wang, Ahmed Alahmadi, David Zhu, and Tongtong Li. Brain functional connectivity analysis using mutual information. In *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 542–546, December 2015. doi: 10.1109/GlobalSIP.2015.7418254. URL https://ieeexplore.ieee.org/document/7418254/.

Daniel S. Yeung, Ian Cloete, Daming Shi, and Wing W. Y. Ng. *Sensitivity Analysis for Neural Networks*. Natural Computing Series. Springer, Berlin, Heidelberg, 2010. ISBN 978-3-642-02531-0 978-3-642-02532-7. doi: 10.1007/978-3-642-02532-7. URL http://link.springer.com/10.1007/978-3-642-02532-7.

Gorka Zamora-López and Matthieu Gilson. An integrative dynamical perspective for graph theory and the study of complex networks, February 2024. URL http://arxiv.org/abs/2307.02449. arXiv:2307.02449.

Wei Zhang, Viktoria Muravina, Robert Azencott, Zili D. Chu, and Michael J. Paldino. Mutual Information Better Quantifies Brain Network Architecture in Children with Epilepsy. *Computational and Mathematical Methods in Medicine*, 2018(1):6142898, 2018. ISSN 1748-6718. doi: 10.1155/2018/6142898. URL https://onlinelibrary.wiley.com/doi/abs/10.1155/2018/6142898. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2018/6142898.