https://doi.org/10.1038/s42005-024-01728-0

An exact mathematical description of computation with transient spatiotemporal dynamics in a complex-valued neural network

Check for updates

Roberto C. Budzinski[1,2,3,4,10], Alexandra N. Busch[1,2,3,4,10], Samuel Mestern[5], Erwan Martin[1,2,3,4], Luisa H. B. Liboni[1,2,3,4], Federico W. Pasini[6], Ján Mináč[1,3,4], Todd Coleman[7], Wataru Inoue[8,9] & Lyle E. Muller[1,2,3,4]

Networks throughout physics and biology leverage spatiotemporal dynamics for computation. However, the connection between structure and computation remains unclear. Here, we study a complex-valued neural network (cv-NN) with linear interactions and phase-delays. We report the cv-NN displays sophisticated spatiotemporal dynamics, which we then use, in combination with a nonlinear readout, for computation. The cv-NN can instantiate dynamics-based logic gates, encode short-term memories, and mediate secure message passing through a combination of interactions and phase-delays. The computations in this system can be fully described in an exact, closed-form mathematical expression. Finally, using direct intracellular recordings of neurons in slices from neocortex, we demonstrate that computations in the cv-NN are decodable by living biological neurons as the nonlinear readout. These results demonstrate that complex-valued linear systems can perform sophisticated computations, while also being exactly solvable. Taken together, these results open future avenues for design of highly adaptable, bio-hybrid computing systems that can interface seamlessly with other neural networks.

Spatially extended dynamics represent a powerful substrate for computation. Neural systems perform sensory computations with organized spatiotemporal dynamics traveling over maps of sensory space[1,2]. For example, waves traveling over retinotopic maps can facilitate short-term predictions of dynamic visual inputs[3]. Beyond neural systems, spatiotemporal patterns of optical or electromagnetic waves can perform sophisticated computations, such as predicting input sequences[4] or performing transformations[5].

Achieving a specific computation with spatiotemporal dynamics requires mapping a single input to a single output through a pattern of activity across a network of nodes. However, it is difficult to design the spatiotemporal dynamics required to implement a specific computation, because systems that have sufficiently rich dynamics are, in general, nonlinear. Designing computations with spatiotemporal dynamics in nonlinear systems is challenging, because control of nonlinear dynamics can be difficult to implement in practice[6].

It thus remains unclear how these systems could perform general computations through their spatiotemporal dynamics. Computational and experimental work has demonstrated that spatiotemporal dynamics can be leveraged to perform computations ranging from logic operations to speech recognition[7,8]. A precise mathematical understanding of these computations, however, remains lacking. At the heart of this problem is the difficulty in understanding how one input is mapped to a specific output through an individual dynamical trajectory in order to perform a single computational task.

In this work, we introduce a system with linear dynamics and nonlinear readout, where we can mathematically design specific computations

[1]Department of Mathematics, Western University, London, ON, Canada. [2]Western Institute for Neuroscience, Western University, London, ON, Canada. [3]Western Academy for Advanced Research, Western University, London, ON, Canada. [4]Fields Lab for Network Science, Fields Institute, Toronto, ON, Canada. [5]Graduate Program in Neuroscience, Western University, London, Canada. [6]Huron University College, London, ON, Canada. [7]Department of Bioengineering, Stanford University, Stanford, CA, USA. [8]Robarts Research Institute, Western University, London, ON, Canada. [9]Department of Physiology and Pharmacology, Western University, London, ON, Canada. [10]These authors contributed equally: Roberto C. Budzinski, Alexandra N. Busch. e-mail: lmuller2@uwo.ca

through the network dynamics. This is possible because the full computation in this system can be solved exactly. This solution, in turn, provides fundamental insight into computation with spatiotemporal dynamics. Specifically, this system allows us to describe, with a complete set of equations, a way to precisely design computations with transient spatiotemporal dynamics.

We design computations with spatiotemporal patterns of activity across a network of N nodes. The computations are implemented by patterns of phase offsets between these nodes. Writing the activity at each node in the network as a complex number $x_i(t) \in \mathbb{C}$ allows a convenient representation of phase. The dynamics of this complex-valued neural network (cv-NN) is governed by the differential equation:

$$\dot{\boldsymbol{x}}(t) = (\mathrm{i}\omega \boldsymbol{I} + \boldsymbol{K})\boldsymbol{x}(t), \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{C}^N$ is a vector that specifies the state of the network at each point in time. Here, we focus on using the network dynamics for computation; we note, however, that recent work has shown complex-valued models can also approximate the dynamics of biological spiking networks[9,10]. The matrix $\boldsymbol{K}$ contains information about the connectivity pattern, the coupling strength, and the phase-delay in the interaction term. Specifically, $\boldsymbol{K} = \epsilon e^{-\mathrm{i}\phi}\boldsymbol{A}$, where ϵ is the coupling strength and ϕ is a phase-delay. While we focus here on phase-delays in the cv-NN, we note that translating between phase-delays and time-delays is possible in networks with oscillatory dynamics[11,12]. The matrix $\boldsymbol{A}$ represents connection weights a_{ij} between nodes i and j in the network. We consider the nodes in the cv-NN to be coupled in a one-dimensional ring with periodic boundary conditions where the connection weight decays as a power-law with distance between the two nodes. In this way, while all nodes in the network are connected (excluding self-connections), the strongest interactions in the network are between neighboring nodes (see Methods, Network structure and cv-NN dynamics). Further, $\boldsymbol{I}$ is the identity matrix, i is the imaginary unit (note the distinction between the imaginary unit i and the index variable *i*.), and ω specifies the frequency at which the nodes' dynamics evolve. Throughout this work, we set $\omega = 2\pi f$ to have a natural frequency of $f = 10$ Hz. This frequency sets a timescale relevant to information processing in the brain; however, other values of f can be chosen without loss of generality. We have previously shown that this complex-valued system displays the hallmarks of canonical synchronization behavior found in oscillator networks[13–15], and while we consider the case of homogeneous natural frequencies here, the approach also generalizes to heterogeneous natural frequencies[15,16].

Phase-delays in the network interactions extend the window of time for which amplitudes in this linear network remain bounded. During this window, the network displays rich spatiotemporal dynamics, which we find can be used for computation. To implement dynamics-based computation, we set up a system with an input layer, the cv-NN, and a decoder that interprets the phase dynamics of the network (Fig. 1a). We utilize polar notation for complex numbers throughout the text, which provides a direct way to analyze the phase dynamics that is used for computation. Specifically, the cv-NN performs computations with spatiotemporal dynamics in the recurrent network, in combination with a nonlinearity in the readout.

With this formulation, we can now write the entire computation in the system with the following closed-form expression:

$$o_k(t) = \Theta_\sigma R_k \mathcal{D}_t \boldsymbol{x}(0), \tag{2}$$

where $o_k(t)$ represents the output of decoder node k, the operator $\mathcal{D}_t = e^{\mathrm{i}\omega t} e^{\boldsymbol{K}t}$ represents an exact solution for the linear dynamics in Eq. (1), starting from initial conditions $\boldsymbol{x}(0)$ (see Methods, Mathematical description of the input), the operator $R_k : \mathbb{C}^N \to \mathbb{R}$ quantifies the level of synchronization in a local patch of the network projecting to decoder unit k (see Methods, Decoding spatiotemporal dynamics, Eq. (8)), and Θ_σ is the standard Heaviside function shifted by a threshold σ. This equation captures the computation in the cv-NN in a closed-form analytical expression, in terms of linear dynamics and nonlinear readout working on the network's initial state $\boldsymbol{x}(0)$.

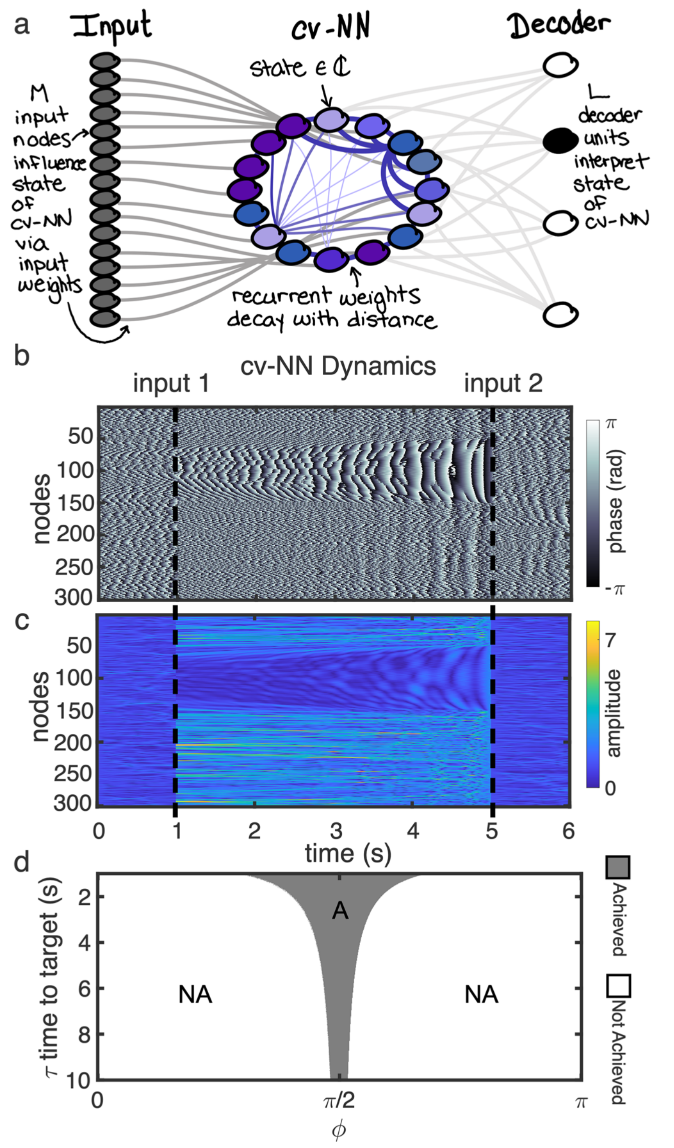

Fig. 1 | A cv-NN with linear dynamics exhibits sophisticated spatiotemporal patterns. a Our framework is composed of an input layer connected to a recurrent layer (the cv-NN) that generates dynamics which are then interpreted by a decoder. The number of nodes, weights, and decoders in the system is flexible and can be adjusted for specific applications. **b**, **c** We use the inverse of operator $\mathcal{D}$ to determine the input required to drive the network to a specific target pattern at a precise time, several seconds into the future (see Methods, Mathematical description of the input). The dynamics of the cv-NN starts in an asynchronous state due to random initial conditions. After input 1, the network evolves to a chimera state pattern. A second input leads the cv-NN back to an asynchronous state. The cv-NN exhibits both (**b**) phase and (**c**) amplitude dynamics. Throughout this work, we use the phase dynamics to perform computations. **d** The success of this process — whether or not the cv-NN achieves the target pattern at the desired time — is quantified by the similarity metric Eq. (3). There is a specific range of the delay parameter, ϕ, and τ for which the target pattern is achieved by the cv-NN dynamics.

Results

Constructing computations in the cv-NN

To describe the system in more detail, we consider computation in the cv-NN in an input-decoder framework (Fig. 1a), where the N nodes in the network receive connections from M input nodes, and project to a set of L output nodes that constitute a decoder. The set of weighted connections

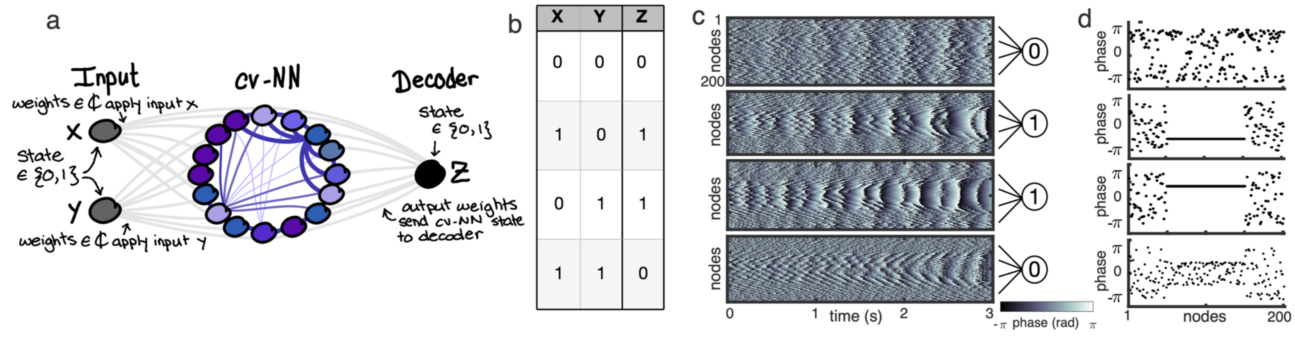

Fig. 2 | The cv-NN can perform simple computations. a Here, we consider two binary inputs X and Y applied to the cv-NN, and we use its dynamics to compute the output Z that is interpreted by the decoder. When one (or both) of the inputs turns on, the input specified by the weighted connections between the input node and the network is added to the state vector of the cv-NN. **b** Here, we implement an XOR gate. Rows in this table represent the input node state vector, $[X, Y]$, and the resulting output, Z. **c**, **d** When X = 1 and Y = 0, or X = 0 and Y = 1 (i.e. precisely one input is applied), we observe a coherent cluster in the spatiotemporal dynamics. This phase synchronized cluster is recognized by the decoder, which returns Z = 1. However, when X = 0 and Y = 0, or X = 1 and Y = 1 (i.e. neither input is applied or both inputs are applied), no phase synchronized cluster appears, and the decoder returns Z = 0.

from the input nodes to the network are collected into an $M \times N$ weight matrix $\mathbf{W}$, which we use in different configurations. In one setup, each input node could project to a single network node. In this case, $\mathbf{W}$ is the $N \times N$ identity matrix $\mathbf{I}_N$, and all input nodes drive the network with specific complex numbers. In another setup, each input node may project to the full network with varying weights. In this case, $\mathbf{W}$ is in the set of $M \times N$ matrices with complex coefficients ($\mathbf{W} \in \mathcal{M}_{M\times N}(\mathbb{C})$), and input nodes are either "on" or "off". In either case, the input to the network takes the form of a complex valued vector, which is applied through multiplication with the state vector of the cv-NN (see Methods, Mathematical description of the input). This process can be thought of as nudging the state of the network onto the trajectory in state space that will allow it to evolve with the desired dynamics. Since the dynamics of the cv-NN is governed by the operator $\mathcal{D}$, the input weights required for the system to evolve to a specific target pattern can be computed precisely using the inverse operator $\mathcal{D}^{-1}$ (see Methods, Mathematical description of the input, and Supplementary Fig. 1). Starting from random and asynchronous initial conditions, we can calculate the input needed for the system to evolve to an arbitrary state several seconds into the future (input 1, Fig. 1b, c). Applying this input to the state vector of the cv-NN brings the network to the correct state required for evolution to the target. We can then use these inputs to design specific dynamics in the cv-NN, in both amplitude and phase, and we use the phase dynamics for computation throughout this paper. It is worth noting that, while we consider only simple, instantaneous inputs to the cv-NN in this work, applying linear control theory to the complex-valued state dynamics is a straightforward extension that would allow specific nodes to receive continuous inputs over time[17], rather than a single instantaneous pulse across the whole network.

Figure 1 illustrates an example of this input-output framework in the cv-NN, depicting both the phase (Fig. 1b) and amplitude (Fig. 1c) at each node in the network. In this example, input 1 drives the network to a partially phase synchronized state 4 s into the future (Fig. 1b, Supplementary Movie 1). This partially synchronized state, where a specific group of nodes has the same phase while other nodes remain in an asynchronous state, has the key features of a chimera state[18]. Chimera states have been studied in many nonlinear systems – e.g. reaction-diffusion systems[19,20], recurrent neural networks[21], and networks of Kuramoto oscillators[18,22,23]. We report that the linear cv-NN displays a range of chimera states, from the short-lived states that we use here for computation (Fig. 1b) to chimeras that exist for long timescales (Supplementary Fig. 2). The chimera states we observe in the cv-NN are transient, and phases in cv-NN eventually collapse to synchrony after some time. Both of these features match the dynamics reported in standard nonlinear Kuramoto oscillator networks[24,25]. Our mathematical approach allows us to understand how connections and phase-delays within the cv-NN work together with driving inputs to create chimera states that range from short transients to long, temporally extended states (Supplementary Fig. 2). Because we now have a system that exhibits sophisticated dynamics such as these transient chimeras, and has a closed-form expression, we can study the input-output mappings in the cv-NN and design dynamics to perform computation.

Not all desired target states are easily achievable in the cv-NN, however. This is due to the fact that, while the operator $\mathcal{D}^{-1}$ is always defined in $\mathbb{C}^N$, when implemented in standard double precision numerical calculations, this operator may not have full rank in practice (see Methods, Mathematical description of the input). This theoretical point, in fact, has a meaningful physical interpretation: for example, if the target state is a chimera, but the network is in a fast-synchronizing regime, the network dynamics will not, in general, match the target state. With this in mind, we introduce a similarity measurement:

$$\mathcal{S} = \frac{1}{N}\left|\sum_{j=1}^{N} e^{\mathrm{i}\mathrm{Arg}[\chi_j]} e^{-\mathrm{i}\mathrm{Arg}[x_j(\tau)]}\right| \tag{3}$$

to numerically quantify the match between the target phase pattern $\mathrm{Arg}[\boldsymbol{\chi}]$, designed to appear at time τ, and the phase pattern displayed by the system at that time, $\mathrm{Arg}[\boldsymbol{x}(\tau)]$. As noted above, the cv-NN can in general collapse to a phase synchronized state (Supplementary Fig. 3), or amplitudes in the network can grow to become unbounded (or decay to zero) in finite time. We find, however, that for values of ϕ within an interval near $\pi/2$, this procedure can generate arbitrary spatiotemporal patterns up to 10 s into the future (Fig. 1d, and Supplementary Fig. 1). This is because the delay parameter ϕ affects the eigenvalues of $\boldsymbol{K}$ and thus influences the dynamics of the cv-NN (Methods, Spectral properties of the network). We also find that target patterns are easier to be obtained in networks with intermediate coupling strength ϵ, which is consistent with recent results that have shown that being near a phase transition can enhance the power of computational systems based on phase synchronized clusters[26,27]. It is worth noting that, while here we consider computations based on phase synchronized clusters in the network, this framework naturally generalizes to different phase patterns. Further, we note that, while we use a small number of decoders in this work for simplicity and ease of visualization, increasing both the number of decoders and the overlap in their connectivity could further improve the network's computational performance in future work.

Constructing logic gates and malleable short-term memory in the cv-NN

The previous result demonstrates the cv-NN can achieve a target state at a specific time window in the future. To demonstrate that these states can be used for computation, we implement the cv-NN with two possible inputs, X and Y, and one output, Z, which is decoded from the network phase dynamics (Fig. 2a, Supplementary Movie 2). This setup allows for the

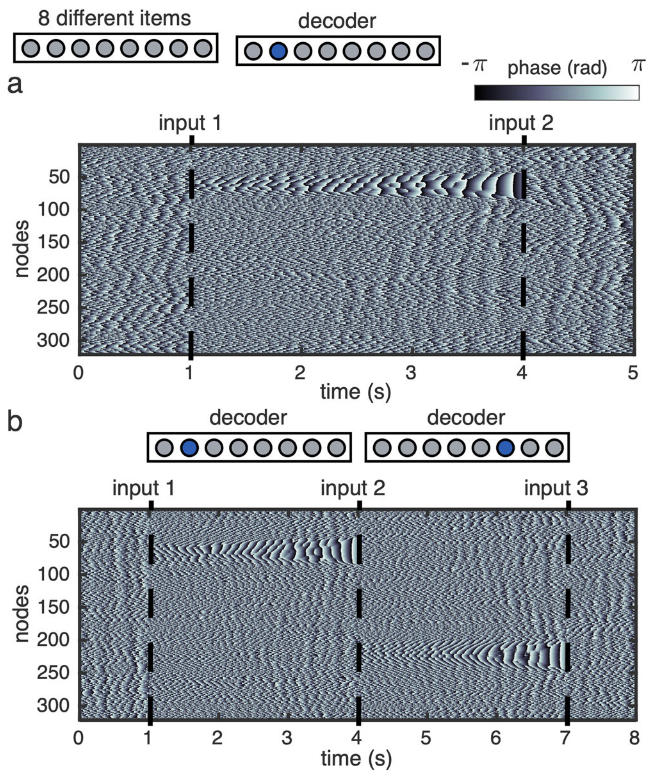

Fig. 3 | The cv-NN can perform short-term memory tasks and online updating. **a** We use our computational framework to perform a short-term memory task in which 1 of 8 possible items is to be held in working memory for 3 s. In this example, input 1 cues item 2, and the cv-NN dynamics evolves to the corresponding chimera state. Once the decoder successfully interprets the phase dynamics, a second input (input 2) then drives the network back to asynchronous behavior. **b** Online updating. In this example, due to the first cue, the network initially stores item 2 in memory. The application of input 2 (second cue) updates this memory to item 6. After the item is decoded, input 3 drives the cv-NN dynamics back to an asynchronous state.

realization of an XOR logic gate (Fig. 2b). When X and Y are both 0, the cv-NN remains asynchronous, and no chimera occurs (Z = 0) (Fig. 2c, top). When either input X or Y is 1, a synchronized cluster occurs in the center of the network (Z = 1) (Fig. 2c, middle two rows). Lastly, when X and Y are both 1, these competing inputs interfere in such a way that no synchronized cluster is observed in the network (Fig. 2c, bottom). In contrast to the way computations are often implemented in neural networks, where non-linearities at single neurons alternate with pooling across trained network connections, the interference underlying the mapping $(X = 1, Y = 1) \rightarrow Z = 0$ occurs in the linear dynamics of the cv-NN, with the nonlinearity $\Theta_\sigma R_k$ only applied once at the readout. This specific combination of linear dynamics and simple nonlinear readout allows specifying the XOR operation precisely in a closed-form expression - Eq. (2). This XOR gate is robust to noise (Supplementary Figs. 4 and 5). Having a closed-form expression opens the opportunity to generalize easily to other standard logic gates (Supplementary Fig. 6) and, potentially, more complex logic operations, in a natural way.

The cv-NN can thus perform simple spatiotemporal computations by holding target states several seconds into the future. These target states could, conceivably, enable a form of in-memory computation, which has proven to be a promising departure from traditional models of von Neumann computing architectures[28]. To explore the possibility of performing in-memory computations with target states in the cv-NN, we considered an example task in which 1 of 8 items is to be held in short-term memory for 3 s. An input to the cv-NN cues the item to be remembered, which is encoded by a specific pattern in the cv-NN dynamics, and can then be read out by decoder units with connections to nearby nodes in the network (Fig. 3a). As before, we store the item in a coherent phase cluster at a specific position in the cv-NN. This cluster then triggers the decoding unit corresponding to the item held in memory (see Methods, Decoding spatiotemporal dynamics). The network is initially asynchronous, due to random initial conditions ($t < 1$ s, Fig. 3a). Following a specific input that cues item 2 at $t = 1$ s, the network evolves to a state with a coherent phase cluster centered at decoder 2, representing the item held in memory. After the item is correctly decoded, another input is applied, and the cv-NN returns to an asynchronous state ($t > 4$ s, Fig. 3a). As with the implementation of logic gates, this framework for short-term memory is robust to noise and perturbation (Supplementary Fig. 7).

A key feature of in-memory computation is the ability to update online, a feature shared with biological working memory[29]. For instance, if someone is asked to keep a phone number in memory, it is also possible for them to update the last digit from a 1 to a 9. Online updates provide biological working memory with the flexibility to adapt to inputs and solve problems over extended time scales[30]. To demonstrate online updating, we consider a longer task where the cv-NN must switch between items 2 and 6 after 4 s (Fig. 3b, Supplementary Movie 3). The input cue needed for the switch is given by a single vector that can be computed locally in time, without requiring future information about the cv-NN's state. These results demonstrate that the cv-NN can store short-term memories with a process that can be both updated online and described with a mathematically exact solution.

Symmetric-key encryption system

Short-term states in the cv-NN can also be used to encode and transmit information between two or more sources, in a simple symmetric-key encryption format[31]. To demonstrate this, we define a *chimera alphabet*, which is a mapping of different coherent phase clusters to the 26 letters of the English alphabet (and one to a blank space, see Supplementary Note 8). We then consider the traditional scenario in which Bob sends a message to Alice, which Eve tries to intercept (Fig. 4a). Bob and Alice agree on a secret key, $\{\omega, \boldsymbol{x}(0)\}$, where ω denotes the intrinsic frequency and $\boldsymbol{x}(0)$ the initial conditions. The chimera alphabet and the network structure ($\boldsymbol{K}$) form the public structure through which the message is transmitted. Eve therefore knows the full transmission framework, including $\boldsymbol{K}$, the chimera alphabet, and the format of the ciphertext, but must attempt to guess the secret key shared between Alice and Bob. To encrypt a message, Bob first chooses a set of target times at which the encoded letters should appear in Alice's cv-NN. He then uses $\mathcal{D}^{-1}$ to compute the set of inputs ($\mathcal{I}_j$) to apply at specific input times (t_j) so that the chimera letters appear as desired (see Methods, Mathematical description of the input). Bob sends Alice the ciphertext $\{\mathcal{I}_j, t_j\}$. Alice initializes her cv-NN using the shared secret $\{\omega, \boldsymbol{x}(0)\}$, then applies the ciphertext, letting the network dynamics evolve according to the operator $\mathcal{D}$. Because Alice has the secret key, synchronized phase clusters will appear. When this happens, Alice decodes each letter of the message using the public chimera alphabet (Fig. 4b, Supplementary Movie 4). We note that the synchronized clusters appear at the target times chosen by Bob, but that these times do not need to be known by Alice and are discovered in the spatiotemporal dynamics of the cv-NN. At the same time, an eavesdropper, Eve, intercepts the ciphertext and applies the inputs to the public network to decode the encrypted message; however, because Eve does not have the secret key $\{\omega, \boldsymbol{x}(0)\}$, Eve's network does not reach the chimera states and will, in practice, evolve to synchronized states.

This example of dynamics-based encryption is robust to random attacks. Randomly guessing ω and $\boldsymbol{x}_0$ does not produce phase clusters from the alphabet (Supplementary Fig. 8). Further, the inputs $\mathcal{I}_j$ must be applied in the correct sequence and at the correct times t_j; otherwise, the target patterns are not obtained. Finally, one may question whether the synchronization of Eve's network (Fig. 4c) could offer her some insight into the private information. This is not the case, however, because in practice Bob

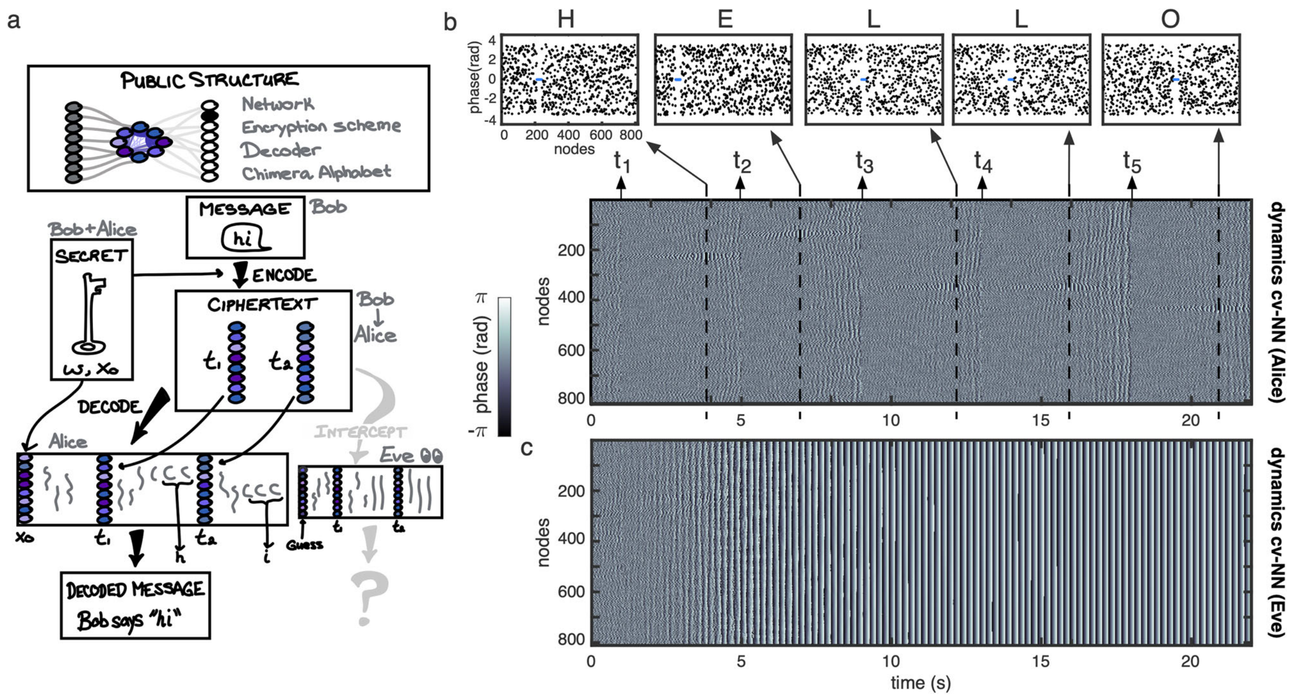

Fig. 4 | The cv-NN enables message transmission based on spatiotemporal dynamics. a Encryption scheme. Public structure: the network structure, the method for encoding and decoding messages, and the mapping between letters in the alphabet and chimera states in the cv-NN, constitute public information. Specifically, the parameters ϵ, α, ϕ and the matrix $\boldsymbol{A}$ are shared by everyone using this framework. Secret key: Bob and Alice agree in advance on the parameter ω and the initial state of the cv-NN $\boldsymbol{x}(0)$, which configures the secret key. Message: a string of letters and spaces Bob will encrypt, so that each letter appears as a chimera state in Alice's cv-NN at some time known only by Bob. To encode his message, he chooses a set of input times t_j, and uses the inverse operator $\mathcal{D}^{-1}$ to obtain the required inputs $\mathcal{I}_j$. Ciphertext: the set of inputs to the cv-NN and the times at which to apply them, which Bob computes and sends to Alice. Here, inputs to be applied at times t_1 and t_2 are depicted by sets of colored input nodes. Alice: Depiction of Alice's cv-NN when she decodes Bob's message. Alice implements the cv-NN using the secret key $\{\omega, \boldsymbol{x}(0)\}$ and applies the inputs $\mathcal{I}_j$ at the times t_j. The resulting spatiotemporal dynamics of the cv-NN can then be interpreted by the decoder using the public alphabet. In this schematic, squiggly lines represent asynchronous dynamics, and the small curved "c" shapes depict the emergence of a synchronized cluster. The positions of these clusters determine the decoded letter. Eve: Depitcion of Eve's cv-NN when she intercepts the ciphertext and tries to decode Bob's message. She applies the correct inputs at the correct times, but since she had to guess the secret key, the chimera letters do not appear in her cv-NN and she cannot decode the message. **b** An example message. Alice implements her cv-NN using the secret key $\{\omega, \boldsymbol{x}(0)\}$ and applies the ciphertext inputs $\mathcal{I}_j$ at the times t_j, denoted above the phase plot of Alice's cv-NN. These inputs cause phase-synchronized clusters to appear in Alice's cv-NN at the times marked by dashed vertical lines. The state of Alice's cv-NN at each of these times are depicted at the top of the figure, with the phase synchronized clusters highlighted in blue. The positions of these clusters are then decoded, resulting in the message HELLO. **c** Dynamics of a cv-NN built using the public information by an eavesdropper, Eve, who tries to intercept the message. Even though Eve is able to obtain the ciphertext, she is not able to decode the message: when she applies the same inputs $\mathcal{I}_j$ at the correct times t_j, she does not obtain the phase clusters in the dynamics of her cv-NN because she does not have access to the secret key $\{\omega, \boldsymbol{x}(0)\}$. Here, her cv-NN synchronizes, which can be seen in the vertical lines in the phase plot of her cv-NN.

and Alice can always extend the private information to a series of keys $\{\omega, \boldsymbol{x}(0)\}$ and jump between these keys in a sequence[32,33].

Biological neurons can decode dynamics in this artificial network

The previous results demonstrate the cv-NN can perform computations, store short-term memories, and can enable secure message passing, but is it possible to utilize these computations in practice? To address this question, we next test whether biological neurons can decode the spatiotemporal dynamics underlying computation in the cv-NN. To do this, we injected the cv-NN dynamics directly as a current into a biological cell via an intracellular recording electrode (Fig. 5a, and Methods, cv-NN dynamics as inputs to biological neurons), and then used the resulting spikes generated by the cell as the decoder (see Methods, Experimental details). We then implemented the short-term memory task where 1 of 8 items is to be held in memory (Fig. 3). As before, the cv-NN holds an item in short-term memory through the position of the phase coherent cluster. We then systematically injected dynamics from subsets of the cv-NN as a current into the biological neuron in separate trials, effectively using the biological cell in place of the 8 decoding units used previously (Fig. 5b). Inputs from the subset of the network corresponding to the remembered item sum constructively and cause the biological neuron to fire (black trace, Fig. 5c), while inputs from outside the coherent phase cluster sum destructively and do not cause the neuron to fire (gray trace, Fig. 5c). Over several trials, the biological neuron repeatedly spiked successfully for the remembered item and not for other inputs (Fig. 5d). These results are robust for different short-term memory items, different scaling factors to translate the cv-NN dynamics into a biological current (Supplementary Fig. 9), and are consistent with a standard mathematical model of the neuron (Supplementary Fig. 10). It is worth noting that, in contrast to standard mathematical models of single neurons, which always fire a spike at a fixed threshold potential and instantaneously reset, biological neurons have variable thresholds that change dynamically in time and with different input[34]. Even under these conditions, however, computations in the cv-NN can be successfully implemented by real biological cells.

The possibility of designing an interface between a biological cell and this recurrent neural network can be understood as a key outcome of our mathematical approach. In general, it could be difficult to design an interface that would allow a biological neuron to decode the output of an artificial recurrent neural network that had been trained to perform a certain task. We believe the fact that designing such an interface is both straightforward and efficient highlights the utility of our approach. This simplicity can, in

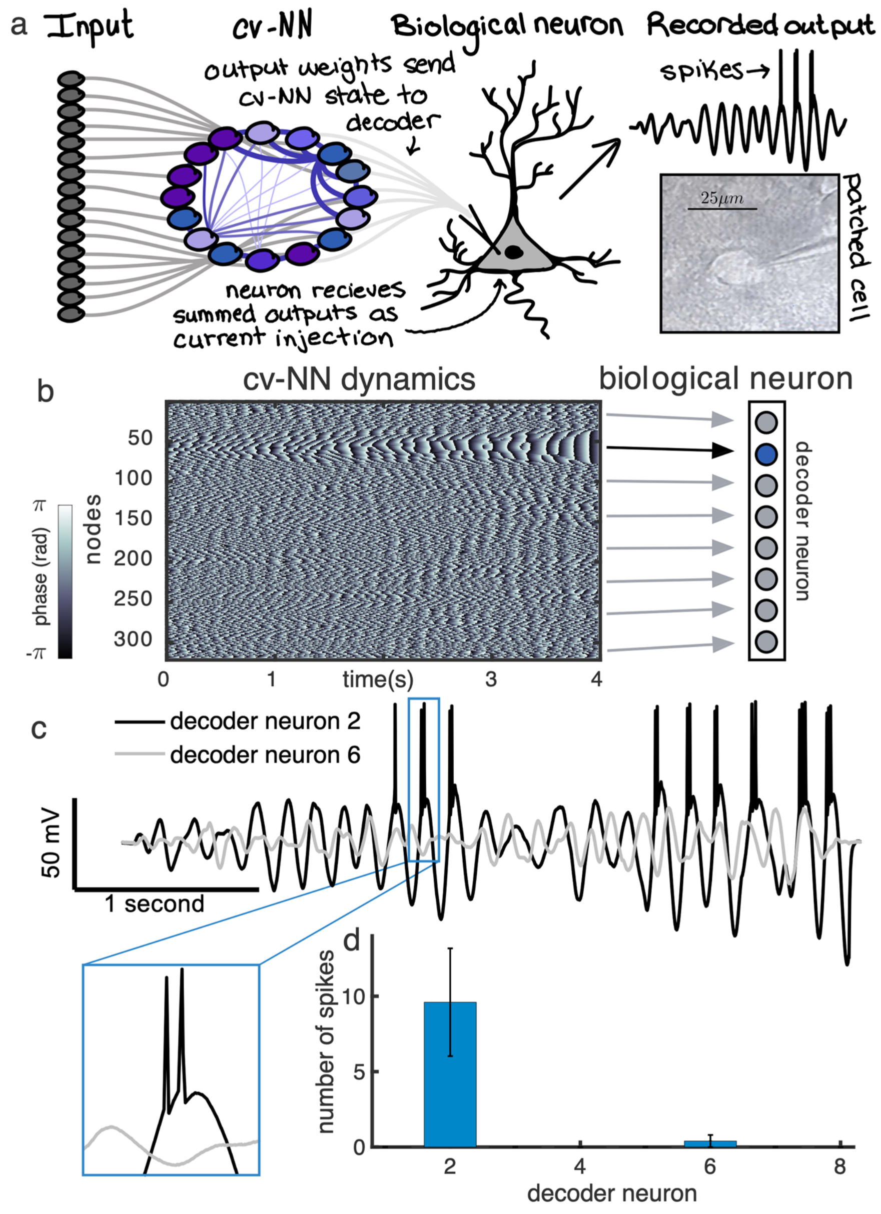

Fig. 5 | Biological neurons can successfully decode the dynamics of the cv-NN. a We implement the cv-NN using real, biological neurons as the decoders. To do so, we inject the dynamics of the cv-NN as a current into a biological neuron, whose resulting physiological signal is used for decoding. In detail, the dynamics in a local patch of the cv-NN are summed into an aggregate activity pattern (Methods, cv-NN dynamics as inputs to biological neurons). This aggregate pattern is then injected via an electrode into the neuron acting as one of the decoders. An example of this procedure is shown in the microscope image, where circular shapes represent cell bodies, and the darker, triangular region shows the electrode patched onto the decoder neuron. **b** As an example, we consider the short-term memory task represented in Fig. 3. The location of the phase synchronized cluster indicates which item is being remembered (item 2 in this case). **c** The current input created from the segment of the network corresponding to the phase synchronized cluster causes the biological neuron decoder to fire repeatedly (black trace). However, when the current input corresponds to an asynchronous segment of the network, the neuron does not fire (gray trace). **d** This procedure is repeated for several different trials, which shows successful decoding.

turn, open possibilities to design interfaces between artificial and biological neural networks in future work.

Discussion

In this work, we have introduced a network that can perform computations while also being exactly solvable. The advantage of this approach is that the entire computation can be written in an exact, closed-form mathematical expression. This expression allows us to design specific computations in the cv-NN, as well as understand the complete dynamical trajectory that transforms specific inputs into specific outputs during an individual computation. To the best of our knowledge, these results represent the first complete mathematical description of how a network utilizes a specific spatiotemporal pattern for an individual computation.

The cv-NN introduced here is a modification to the standard architecture of recurrent neural networks. Specifically, the cv-NN has nodes with complex-valued state, linear interactions within the network, and a nonlinear readout stage. Recurrent neural networks are known to have powerful computational capacity[35–37]. However, they are also known to be difficult to train[38,39], and to interpret once trained[40]. One adaptation of standard recurrent neural networks is the reservoir computing framework, where computations are performed by a recurrent network with fixed connections and a trained linear readout[41–43]. The insight of this approach is that training only the readout can provide a dramatic simplification of RNN training[41], and the reservoir computing framework has led to substantial advances in predicting chaotic time series[43,44].

Our cv-NN bears similarity both to standard recurrent neural networks and to reservoir computing, but introduces two main adaptations. First, switching the order of nonlinear and linear steps in the network makes possible an exact solution for the entire mapping from input to output. Specifically, instead of feeding input into a recurrent neural network with nonlinear interactions and linear readout, here inputs project into a network with *linear* interactions and *nonlinear* readout. Because the nonlinearity only happens at the last stage of the computation, inputs can create sophisticated spatiotemporal dynamics in the recurrent network, while the dynamics at that stage remain exactly solvable. Thus, while some nonlinearity is known to be an essential component for computation[45,46], changing the standard architecture of nonlinear and linear stages in recurrent neural networks may allow for new insights. Recent numerical simulations and experimental work have also found that linear dynamics in the recurrent network and nonlinear readout can be useful for computation and time-series prediction[4,47–51]. In this work, we utilize a similar architecture, with linear dynamics and nonlinear readout, to obtain an exact mathematical solution for the computation performed in this system. This solution, in turn, makes possible the second main adaptation from the recurrent neural network and reservoir paradigms: instead of training either the recurrent or readout weights to minimize error, here we use the exact solution to directly construct computations in the system. The direct construction offers clear and precise mathematical insight into how a networked system can transform a single input into a specific spatiotemporal pattern in its internal state, and then to generate a specific output through nonlinear readout. Relating these direct constructions to training paradigms for recurrent neural networks, including backpropagation through time and reservoir computing, will be a subject of interest for future work.

Constructing computations through linear network dynamics may at first seem counter-intuitive, because linear dynamics are often thought to be too simple to produce sophisticated spatiotemporal behavior[6,52]. For this reason, research in both neural networks and dynamics-based computation has largely focused on nonlinear systems[43,44,53–55]. Nonlinear dynamics are used throughout machine learning for training RNNs[35,56] because saturating nonlinearities can keep the network activity within bounded intervals[57,58]. Nonlinear dynamics are also used extensively in physics for training reservoir computers to predict chaotic dynamics[43,44,59,60]. Further, it is increasingly appreciated that nonlinear oscillator networks can be trained to perform sophisticated computations[61–66]. In these systems, nonlinear dynamics are thought to be essential for computation because they provide a rich diversity of dynamical behavior that can be leveraged for spatiotemporal computation[52,67]. However, a complete mathematical description of spatiotemporal computation in these systems remains challenging.

Several different analytical techniques have provided insights into these systems. Computing with spatiotemporal dynamics has been studied in reaction-diffusion systems[68]. Mathematical approaches have been worked out for excitable lattices with nearest-neighbor interactions, where each possible spatiotemporal pattern can be computed by hand[69]. Mean-field approaches, including dynamical mean-field theory (DMFT), have provided fundamental insights into the macroscopic statistics of nonlinear dynamics in RNNs[70,71], by providing analytical expressions for the structure of autocorrelations and the transitions to chaos in these systems[57,71–73]. In order to fully understand how neural networks perform computations with spatiotemporal dynamics, however, it is necessary to understand how an individual dynamical trajectory, driven by an input, can lead to an output in a single instance of computation. The key missing piece has always been how to formulate, in a mathematical manner, precisely how computation emerges from the interaction between nonlinearity and the pattern of connections in an individual network.

By leveraging recent analytical work from our group that introduced an operator-based description of nonlinear oscillator networks[13–16], we have developed a cv-NN whose complex-linear dynamics can be exactly solved, but also produces rich spatiotemporal patterns that can be used for computation. The asymptotic behavior of the linear dynamics is, obviously, quite simple, in that the amplitudes of all nodes either diverge to infinity or collapse to zero. A key technical insight, however, is to utilize phase-delays in the network interactions to change the spectrum of the operator governing the complex-linear network dynamics. We find that, by tuning these delays in a manner guided by our analytical approach, we can create long-lived amplitude transients in the complex-linear cv-NN. Notably, during these amplitude transients, the network displays sophisticated dynamics in its phase. Using these patterns of phase as a dynamical reservoir, we can, in turn, design computations from a specific matrix equation. Both the origins of the phase dynamics in the cv-NN and how to extend the computations that can be constructed in this way will be the subject of future work. The key advance in the present work is to show that constructions such as this are possible. By fundamentally re-ordering the components of the standard neural network framework, guided by our analytical approach, this work provides a missing piece required to understand computation in these systems.

Methods

Network structure and cv-NN dynamics

Equation (2) expresses the whole computation performed by our cv-NN in a closed-form solution, in terms of operators acting on the initial state of the network. To simplify these operators, we express the evolution of the cv-NN in matrix from in Eq. (1). We can also express the dynamics of each node in the cv-NN explicitly:

$$\dot{x}_i(t) = \mathrm{i}\omega + \sum_{\mathrm{j}=1}^{\mathrm{N}} \epsilon \mathrm{e}^{-\mathrm{i}\phi} \mathrm{a}_{\mathrm{ij}} \mathrm{x}_{\mathrm{j}}(\mathrm{t}). \tag{4}$$

This expression makes clear that each node i interacts with each other node j in a way that is modulated by the coupling strength ϵ, phase-delay ϕ, and connectivity a_{ij}. In our work, the cv-NN structure is given by a one-dimensional ring with periodic boundary conditions and connection weights a_{ij} that follow a power rule:

$$a_{ij} = \frac{1}{\varsigma(\alpha)(d_{ij})^{\alpha}}. \tag{5}$$

Here, d_{ij} is the distance between nodes i and j (defined as $d_{ij} = \min(|i - j|, N - |i - j|)$ for $i \neq j$), α is the power-law exponent (which controls the decay), and $\varsigma(\alpha)$ is a normalization term given by:

$$\varsigma(\alpha) = \sum_{j=1, j\neq i}^{N} \frac{1}{(d_{ij})^{\alpha}}. \tag{6}$$

The resulting weighted adjacency matrix $\boldsymbol{A}$ is symmetric and circulant. In this case, each node in the network is connected to all other nodes (without self-connections, so that $a_{ii} = 0$), but the connection weight decays with the edge distance between nodes i and j. This connection scheme means that the

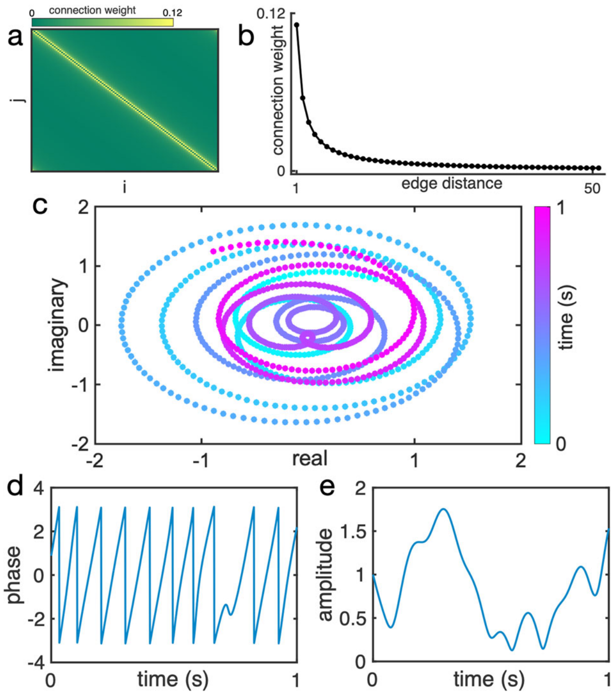

Fig. 6 | Example of the network connectivity and the dynamics of one node in the cv-NN. a Distance-dependent weighted adjacency matrix that we consider for the cv-NN. **b** The connection weight decays as the edge distance increases. **c** The dynamics of one element in the cv-NN is shown in the complex-plane with time represented in color-code. Moreover, we can observe (**d**) the argument and (**e**) the amplitude of the dynamics as a function of time.

matrix $\boldsymbol{K} = \epsilon e^{-i\phi}\boldsymbol{A}$ is also circulant. Recall that $\boldsymbol{K}$ provides information about the interactions between nodes, since ϵ is the coupling strength and ϕ is the phase-lag in the system. In this work, we consider distance-dependent networks with $\alpha = 1.0$. The weighted adjacency matrix for this case is represented in Fig. 6a, where the connection weight decays as the edge distance increases (Fig. 6b).

The cv-NN dynamics evolve according to Eq. (1). An example of the dynamics of one element in the cv-NN is depicted in Fig. 6. Here, we show the dynamics of this example node plotted in the complex-plane (Fig. 6c), with time represented in color-code. We also depict the phase (Fig. 6d) and the amplitude (Fig. 6e) of the node's dynamics, respectively, as a function of time.

Mathematical description of the input

The dynamics of the cv-NN is given by Eq. (1), whose solution is given by $\boldsymbol{x}(t) = e^{i\omega t}e^{\boldsymbol{K}t}\boldsymbol{x}(0)$. The evolution of the cv-NN dynamics is specified by the operator $\mathcal{D}$ applied on $\boldsymbol{x}(0)$. Because the operator $\mathcal{D}$ defines an exact solution for the dynamics of the cv-NN, we can find, analytically, the initial state $\boldsymbol{x}(0)$ that will evolve to an arbitrary target pattern $\boldsymbol{\chi}$ at time $t = \tau$. To do so, we can use inverse operator $\mathcal{D}^{-1}$ applied to the target state $\boldsymbol{\chi}$ to find the initial state $\boldsymbol{x}(0)$:

$$\boldsymbol{x}(0) = \mathcal{D}_\tau^{-1}\boldsymbol{\chi} = \frac{\boldsymbol{\chi}}{e^{i\omega\tau}e^{\boldsymbol{K}\tau}}, \tag{7}$$

where the matrix $\boldsymbol{K}$ has a defined inverse. The operator $\mathcal{D}^{-1}$ is always defined in $\mathbb{C}^N$ (except in the case the matrix $\boldsymbol{K}$ is singular - see Supplementary Note 1 for mathematical details). However, due to the numerical implementation, some of these states are, in practice, hard to obtain. To exemplify this point, we consider an implementation of our method with standard double numerical precision. Figure 7a shows the range of parameters ϕ and the time into the future where a chimera state can be obtained in this condition (we note that this panel is identical to Fig. 1d). We then numerically evaluate the rank of the operator $\mathcal{D}^{-1}$ under the same conditions, where we observe that, this operator is (numerically) full rank for the same range of parameter where the chimera states are successfully obtained (Fig. 7b). This range of parameters thus highlights the conditions where arbitrary spatiotemporal patterns can be easily and robustly obtained in our cv-NN.

With this approach, we are now able to compute the input necessary to drive the network to a given pattern at a specific time in the future. Suppose the network starts in a random initial state and evolves according to Eq. (1). If, at some point in its evolution, we decide that we want the cv-NN to display a target state, $\boldsymbol{\chi}$, some τ seconds into the future, we can use Eq. (7) to determine which state would lead to that pattern at that time. We then compute the difference between the current state of the cv-NN and the state necessary for evolution to the target. This difference can then be applied as an input to quickly bump the network towards the correct starting point, so that the desired trajectory then unfolds in time. An example is shown in Supplementary Fig. 1.

This framework is flexible to different implementations of the cv-NN. Depending on the task or computation, the cv-NN architecture may take a variety of forms: input nodes can connect to all or only part of the network, with weights that can be complex, real, or binary; and these nodes can either have a binary state, on or off, or can take different complex values. In all cases, the input to the network ultimately takes the form of a vector in $\mathbb{C}^N$, which is multiplied componentwise by the cv-NN state vector. In Fig. 8, we demonstrate two specific examples. In the first set up, $M < N$ nodes are each connected to all N nodes in the cv-NN with varying weights (Fig. 8a). These connections are specified by a weighted adjacency matrix with complex coefficients. When an input node turns on, the vector corresponding to its connection weights is applied as input to the network. This set up is useful for computations like logic gates, where the number of inputs to be applied is much smaller than the number of nodes in the network. In a second example, N input nodes are each connected to a single node in the cv-NN,

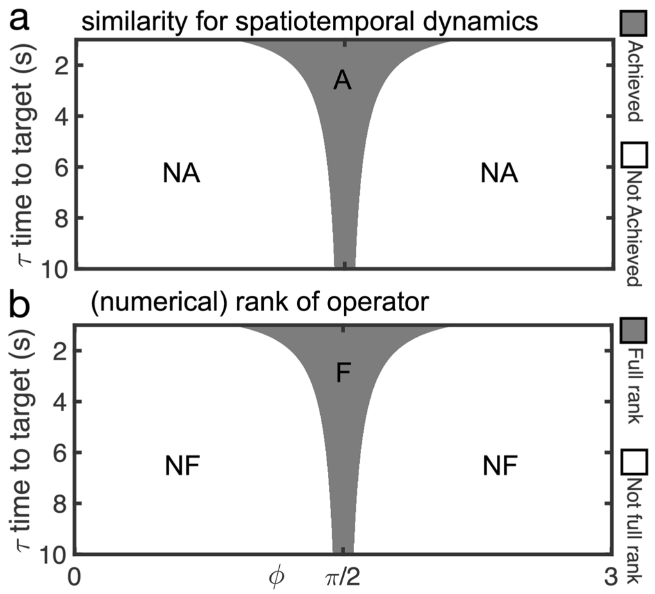

Fig. 7 | Numerical implementation of the operator $\mathcal{D}^{-1}$. a We are able to analytically define the necessary input to drive a chimera state τ seconds into the future. This approach is found to be successful for a given range of τ and ϕ (note that this panel is identical to Fig. 1d). **b** While the operator $\mathcal{D}^{-1}$ is always defined (except when the matrix $\boldsymbol{K}$ is singular), a numerical investigation of the rank of this operator reveals that, with standard double precision, the operator is only (numerically) full rank for the same range of parameters where the chimeras are successfully obtained.

with connection weight 1 (Fig. 8b). At each point in time, the state of each input node (in $\mathbb{C}$) denotes the input to be applied by that node. This set up works well for flexibly applying inputs, as in the memory updating task. In both cases, we can use our mathematical approach represented by Eq. (7) to precisely obtain the inputs that lead the cv-NN dynamics to a desired state in the future.

Decoding spatiotemporal dynamics

The computations performed by our framework are based on the spatiotemporal dynamics of the cv-NN. The example computations demonstrated in this text involve clusters of phase synchronized nodes. To interpret these dynamics, we use a simple decoder that identifies the position of the cluster. Each of the L decoding units is connected to a group of nearby nodes in the network. As the cv-NN dynamics evolve, each decoding unit evaluates the level of synchronization in the group of nodes to which it is connected using the Kuramoto order parameter:

$$R_k[\boldsymbol{x}(t)] = \frac{1}{N_L}\left|\sum_{j=1+(k-1)N_L}^{kN_L} \exp\left(i\mathrm{Arg}[x_j(t)]\right)\right| \tag{8}$$

where k represents the group index, and N_L is the number of elements in each group. This quantifier is close to 1 when the dynamics within the group is coherent (synchronized), and it is close to zero otherwise. Following Eq. (2), we then apply Θ_σ to $R_k[\boldsymbol{x}(t)]$, which is a standard Heaviside function shifted by a threshold σ. This produces a a binary output, either 0 if $R_k[\boldsymbol{x}(t)]$ is below the threshold or 1 if $R_k[\boldsymbol{x}(t)]$ is above the threshold. An example is represented in Fig. 9, where we observe a chimera state in the dynamics of the cv-NN (left). In this case, we divide the network into four groups ($L = 4$) and evaluate the phase synchronization level of each group, given by Eq. (8) and plotted as a function of time (right, blue line). We then use the level of synchronization at a specified time to identify the synchronized cluster and decode the spatiotemporal patterns in the cv-NN. With these results we can perform computations by applying a Heaviside function shifted by a threshold σ – Eq. (2)– which leads to a

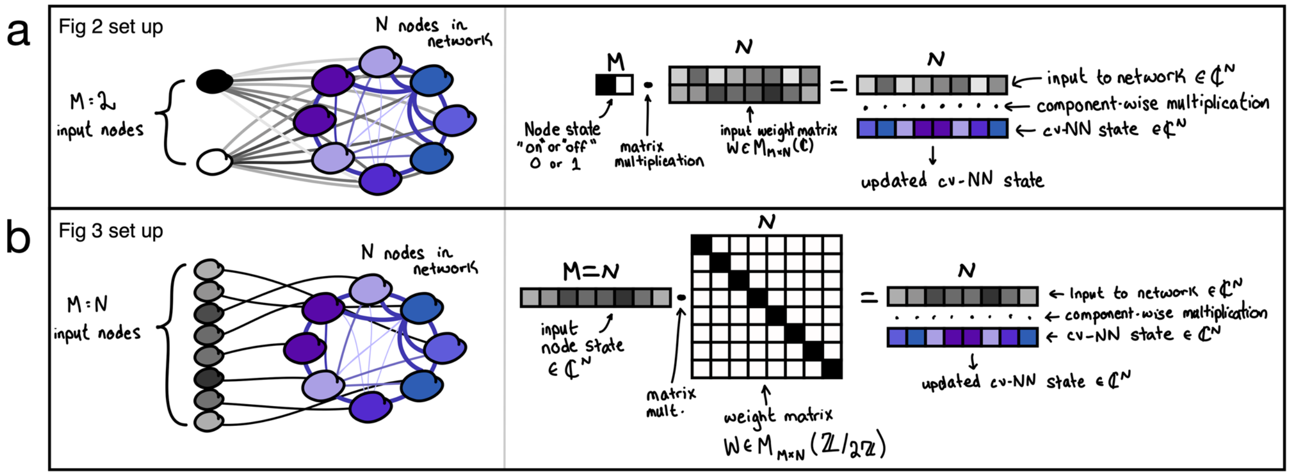

Fig. 8 | Schematic representation of input application to the cv-NN. The specific configuration of input nodes and weights depends on the context of the task and can be adapted to individual applications. **a** In this example set up, $M = 2$ input nodes are each connected to all N nodes in the cv-NN with varying complex-valued weights that are described in an $M \times N$ weight matrix. Here, input nodes have binary states that specify whether each input is on or off. When an input node is on, the weights projecting from that input node to the cv-NN determine the input that is applied. More precisely, at the time of an input, the binary vector describing the states of the input nodes is multiplied by the complex-valued input weight matrix to obtain an input vector in $\mathbb{C}^N$. This input vector is applied to the cv-NN by componentwise multiplication with the state of the network at the time of the input. This is the case considered in the set up of Fig. 2. **b** In another possible set up, $M = N$ input nodes are each connected to one node in the cv-NN, as described by a binary weight matrix. Here, each input node has a state in $\mathbb{C}$, and acts only on the one cv-NN node to which it is connected. Again, the input to the network takes the form of a vector in $\mathbb{C}^N$, with each component given by input applied by one input node. This is the set up considered in Fig. 3. In both cases, we can use our exact mathematical expression to obtain the precise input necessary to drive the cv-NN to evolve to a desired pattern in the future.

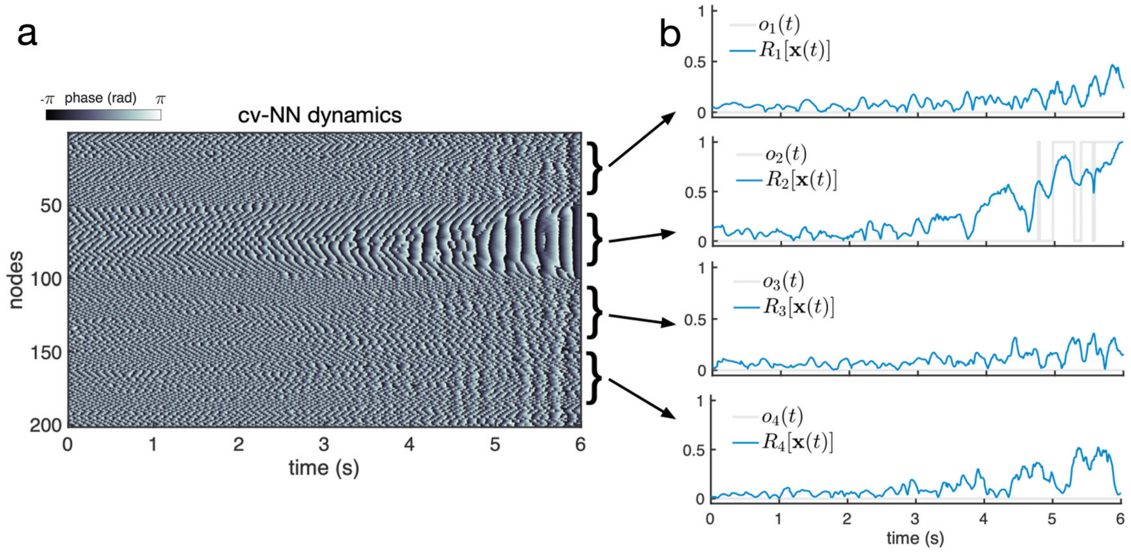

Fig. 9 | We use synchronization level of each part of the cv-NN to decode the spatiotemporal dynamics. a An example of a chimera state can be observed in the dynamics of the cv-NN. **b** We divide the network into four groups $L = 4$ and evaluate the synchronization of each one using the operator $R_k[\boldsymbol{x}(t)]$ (blue line). We then use a Heaviside function shifted by a threshold σ applied to this signal to obtain a binary output (gray line), which is then used for computation.

binary output, either 0 if R_k is below the threshold or 1 if R_k is above the threshold (right, gray line).

Spectral properties of the network

We can express the dynamics of the cv-NN in terms of the eigenvalues and eigenvectors of the matrix $\boldsymbol{K}$ (provided $\boldsymbol{K}$ is diagonalizable):

$$\boldsymbol{x}(t) = e^{\mathrm{i}\omega t}\left(c_1 e^{\lambda_1 t}\boldsymbol{v}_1 + c_2 e^{\lambda_2 t}\boldsymbol{v}_2 + \cdots + c_N e^{\lambda_N t}\boldsymbol{v}_N\right), \tag{9}$$

where c_j can be determined by the initial state, λ_j is the jth eigenvalue, and $\boldsymbol{v}_j$ is the jth eigenvector, with $j \in [1, N]$. Further, in our context the matrix $\boldsymbol{K}$ is circulant, thus the eigenvalues and eigenvectors and their ordering can be obtained by the Circulant Diagonalization Theorem (CDT). In this case, the analytical expression to obtain the eigenvalues is given by:

$$\lambda_k(\boldsymbol{H}) = \sum_{j=1}^{N} h_j \exp\left[-\frac{2\pi\mathrm{i}}{N}(k-1)(j-1)\right], \tag{10}$$

where $\boldsymbol{h}$ is the generating vector, and $k, s \in [1, N]$. The expression to obtain the eigenvectors is given by:

$$[\boldsymbol{v}_k]_s = \frac{1}{\sqrt{N}}\exp\left[-\frac{2\pi\mathrm{i}}{N}(k-1)(s-1)\right], \tag{11}$$

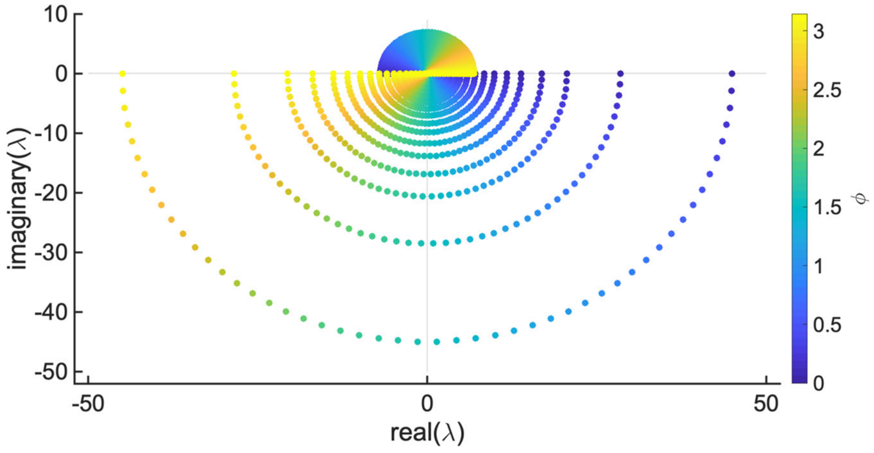

Fig. 10 | Eigenvalues of *K* in the complex plane. The eigenvalues are obtained through Eq. (10) for different values of ϕ, which are represented in color-code. We observe that the term $e^{-i\phi}$ leads to a rotation of the eigenvalues in the complex plane.

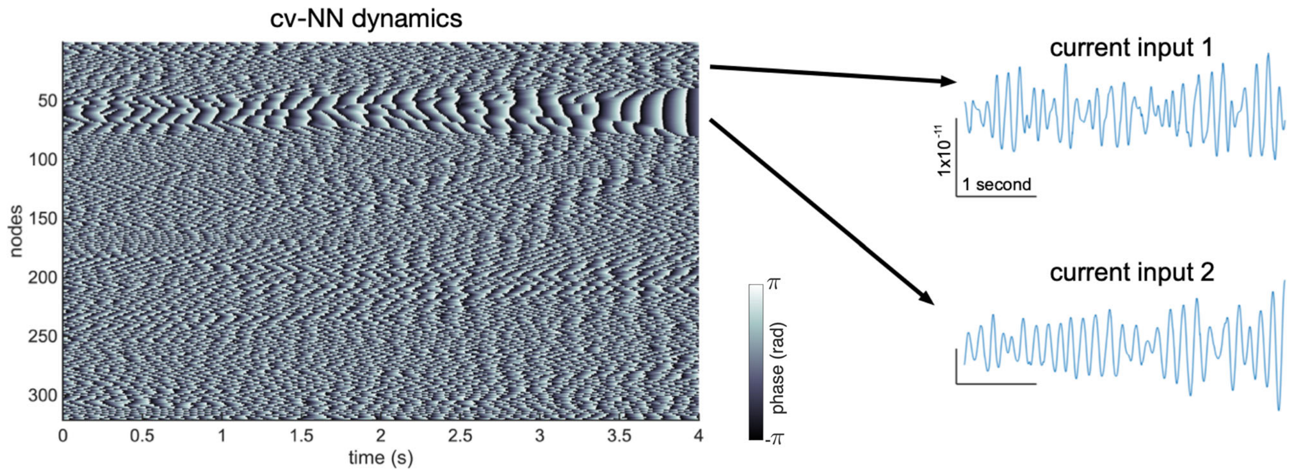

Fig. 11 | Dynamics of the cv-NN transformed into current inputs to biological neurons. The cv-NN dynamics shows a chimera state. We then use Eq. (12) to obtain the inputs to be applied as currents to the biological neurons. We observe that current input 2 has a higher amplitude compare to input 1, for example. This due to the phase synchronized cluster in the cv-NN in the position of the input 2. The inputs coming from the other parts of the network that are also in an asynchronous state have similar amplitude as input 1. In the plots of the inputs, the vertical bar indicates 1×10^{-11} (amplitude measured in Ampere) and the horizontal one represents 1 s.

Note that the CDT provides a natural ordering of the eigenvalues and eigenvectors of a circulant matrix.

In particular, Eq. (9) shows that the configuration of the eigenvalues and eigenvectors defines the long-term behavior of $\boldsymbol{x}$. Further, $\boldsymbol{K} = \epsilon e^{-i\phi}\boldsymbol{A}$, which shows that ϵ scales the eigenvalues and ϕ leads to a rotation in the complex plane. Figure 10 depicts the eigenvalues of the matrix $\boldsymbol{K}$ in the complex plane for different values of $\phi \in [0, \pi]$, which are represented in color-code. For $\phi < \pi/2$, the eigenvalue λ_1 has the largest real part. This eigenvalue is associated to $\boldsymbol{v}_1 = [1, 1, \cdots, 1]$, which has argument given by $\text{Arg}[\boldsymbol{v}_1] = [0, 0, \cdots, 0]$, corresponding to phase synchrony. Thus, when $\phi < \pi/2$, the network transitions to synchrony. However, as ϕ approaches $\pi/2$, the real part of the eigenvalues decreases and the transient time increases, allowing the system to depict a richer diversity of dynamical patterns. For $\phi = \pi/2$ all eigenvalues are purely imaginary, and the coupling is neutral. For $\pi/2 < \phi \leq \pi$, the real part of λ_1 is negative, and therefore, the system cannot reach phase synchronization. This result highlights the fact that there is a finite range of ϕ in which the cv-NN can achieve sophisticated spatio-temporal patterns.

cv-NN dynamics as inputs to biological neurons

As an application of our framework, we implement the 8 item short-term memory task using real, biological neurons as the decoders. We set up the network as in Fig. 3, with each of $L = 8$ decoder units connected to a separate segment of the network (or group of nodes). Together, they decode the stored memories by reading out the location of the phase synchronized cluster. To translate the network dynamics into a biologically relevant signal, we evaluate the effective signal of each group of nodes. Specifically, we sum the inputs from the network that each decoder unit would receive. We then apply these signals as current inputs to a biological neuron. The current input to the cell can be expressed as:

$$I_i(t) = \rho \sum_{j=1+(i-1)N_L}^{iN_L} \cos\Big(\text{Arg}[x_j(t)]\Big), \qquad (12)$$

where $i \in [1, L]$, N_L is the number of elements in each group, and the sum over j is performed within each group, i.e. from $j = 1 + (i-1)N_L$ to $j = iN_L$. Moreover, ρ is a scaling factor (measured in Ampere), which controls the amplitude of the current to a biologically relevant scale, and is commonly used in patch clamp procedures.

Here, we show an example of cv-NN dynamics and the inputs that are applied to neurons as currents in our experiments (Fig. 11). In all plots of the inputs, the vertical bar indicates 1×10^{-11} (amplitude measured in Ampere) and the horizontal one represents 1 s. We can observe that input 2 has a higher amplitude, due to the coherent dynamics in the cv-NN.

Experimental details

The patch clamp recordings were performed on pyramidal neurons of the medial prefrontal cortex of a 5 months old C57BL/6 mouse. All experimental procedures were performed in accordance with the Canadian Council on Animal Care guidelines and approved by the University of Western Ontario Animal Use Subcommittee (AUP: 2022-071). *Acute brain slice preparation:* Animals were intraperitoneally injected with pentobarbital (100 mg/kg) and intracardially perfused with an ice cold oxygenated N-methyl-D-glucamine (NMDG) solution containing 92 mM NMDG, 93 mM HCl, 2.5 mM KCl, 1.2 mM NaH2PO4, 30 mM NaHCO3, 20 mM HEPES, 25 mM D-Glucose, 5 mM sodium ascorbate, 2 mM thiourea, 3 mM sodium pyruvate, 10 mM MgCl2, 0.5 mM CaCl2. The brain was quickly removed, submerged into NMDG solution and coronally sectioned (thickness 250 μM) using a vibratome (VT1200S, Leica Microsystems). After placement into oxygenated NMDG solution at 32 °C, slices recovered for 15 min. Subsequently, the slices were placed for 45 min into HEPES holding solution (92 mM NaCl, 2.5 mM KCl, 1.2 mM NaH2PO4, 30 mM NaHCO3, 20 mM HEPES, 25 mM D-Glucose, 5 mM sodium ascorbate, 2 mM thiourea, 3 mM sodium pyruvate, 2 mM MgCl2, 2 mM CaCl2). *Electrophysiological recordings:* Sections were visualized using an upright microscope (SliceScope, Scientifica) equipped with infrared Dodt gradient contrast and continuously perfused with oxygenated aCSF solution (1 ml/min, 32 °C) containing 126 mM NaCl, 2.5 mM KCl, 1.2 mM NaH2PO4, 26 mM NaHCO3, 12.5 mM D-Glucose, 1 mM MgCl2, 2 mM CaCl2. Borosilicate glass pipettes (Sutter Instruments) were pulled in a P1000 Micropipette Puller (Sutter Instruments) and filled with a intracellular solution containing 108 mM potassium gluconate, 2 mM MgCl2, 8 mM sodium gluconate, 8 mM KCl, 1 mM K2-EGTA, 4 mM K2-ATP, 0.3 mM Na3-GTP and 10 mM HEPES. The resistance of the pipettes was between 3 and 6 MΩ, and the access resistance was less than 20 MΩ. Synaptic transmission was blocked using picrotoxin (0.1 mM) and kynurenic acid (1 mM). Recordings were performed in whole cell patch-clamp configuration using a Multiclamp 700B amplifier (Molecular Devices) and digitized using Digidata 1550B (Molecular Devices).

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

Experimental data from the neuronal recordings presented in Fig. 5 are available at https://doi.org/10.5281/zenodo.12511142.

Code availability

An open-source code repository for this work is available on GitHub: https://github.com/mullerlab/budzinskiEAexact.

Received: 4 March 2024; Accepted: 2 July 2024;
Published online: 15 July 2024

References

1. Ermentrout, G. B. & Kleinfeld, D. Traveling electrical waves in cortex: insights from phase dynamics and speculation on a computational role. *Neuron* **29**, 33 (2001).
2. Muller, L., Chavane, F., Reynolds, J. & Sejnowski, T. J. Cortical travelling waves: mechanisms and computational principles. *Nat. Rev. Neurosci.* **19**, 255 (2018).
3. Benigno, G. B., Budzinski, R. C., Davis, Z. W., Reynolds, J. H. & Muller, L. Waves traveling over a map of visual space can ignite short-term predictions of sensory input. *Nat. Commun.* **14**, 3409 (2023).
4. Vandoorne, K. et al. Experimental demonstration of reservoir computing on a silicon photonics chip. *Nat. Commun.* **5**, 3541 (2014).
5. del Hougne, P. & Lerosey, G. Leveraging chaos for wave-based analog computation: demonstration with indoor wireless communication signals. *Phys. Rev. X* **8**, 041037 (2018).
6. Jiang, J. & Lai, Y.-C. Irrelevance of linear controllability to nonlinear dynamical networks. *Nat. Commun.* **10**, 3961 (2019).
7. Adamatzky, A. & Costello, B. D. L. Experimental logical gates in a reaction-diffusion medium: The xor gate and beyond. *Phys. Rev. E* **66**, 046112 (2002).
8. Fernando, C. & Sojakka, S. Pattern recognition in a bucket. In *European Conference on Artificial Life* (Springer, 2003) pp. 588–597.
9. Schaffer, E. S., Ostojic, S. & Abbott, L. F. A complex-valued firing-rate model that approximates the dynamics of spiking networks. *PLoS Computational Biol.* **9**, e1003301 (2013).
10. Pietras, B., Gallice, N. & Schwalger, T. Low-dimensional firing-rate dynamics for populations of renewal-type spiking neurons. *Phys. Rev. E* **102**, 022407 (2020).
11. Jeong, S. O., Ko, T. W. & Moon, H. T. Time-delayed spatial patterns in a two-dimensional array of coupled oscillators. *Phys. Rev. Lett.* **89**, 154104 (2002).
12. Ko, T. W. & Ermentrout, G. B. Effects of axonal time delay on synchronization and wave formation in sparsely coupled neuronal oscillators. *Phys. Rev. E* **76**, 056206 (2007).
13. Budzinski, R. C. et al. Geometry unites synchrony, chimeras, and waves in nonlinear oscillator networks. *Chaos: Interdiscip. J. Nonlinear Sci.* **32**, 031104 (2022).
14. Budzinski, R. C., Graham, J. W., Mináč, J., & Muller, L. E. Theory of transient chimeras in finite Sakaguchi-Kuramoto networks. Preprint at https://arxiv.org/abs/2311.01382 (2023).
15. Budzinski, R. C. et al. Analytical prediction of specific spatiotemporal patterns in nonlinear oscillator networks with distance-dependent time delays. *Phys. Rev. Res.* **5**, 013159 (2023).
16. Muller, L., Mináč, J. & Nguyen, T. T. Algebraic approach to the Kuramoto model. *Phys. Rev. E* **104**, L022201 (2021).
17. Liu, Y.-Y., Slotine, J.-J. & Barabási, A.-L. Controllability of complex networks. *Nature* **473**, 167 (2011).
18. Abrams, D. M. & Strogatz, S. H. Chimera states for coupled oscillators. *Phys. Rev. Lett.* **93**, 174102 (2004).
19. Tinsley, M. R., Nkomo, S. & Showalter, K. Chimera and phase-cluster states in populations of coupled chemical oscillators. *Nat. Phys.* **8**, 662 (2012).
20. Totz, J. F., Rode, J., Tinsley, M. R., Showalter, K. & Engel, H. Spiral wave chimera states in large populations of coupled chemical oscillators. *Nat. Phys.* **14**, 282 (2018).
21. Masoliver, M., Davidsen, J. & Nicola, W. Embedded chimera states in recurrent neural networks. *Commun. Phys.* **5**, 205 (2022).
22. Panaggio, M. J. & Abrams, D. M. Chimera states: coexistence of coherence and incoherence in networks of coupled oscillators. *Nonlinearity* **28**, R67 (2015).
23. Kotwal, T., Jiang, X. & Abrams, D. M. Connecting the kuramoto model and the chimera state. *Phys. Rev. Lett.* **119**, 264101 (2017).
24. Wolfrum, M. & Omel'chenko, E. Chimera states are chaotic transients. *Phys. Rev. E* **84**, 015201 (2011).
25. Shanahan, M. Metastable chimera states in community-structured oscillator networks. *Chaos: Interdiscip. J. Nonlinear Sci.* **20**, 013108 (2010).
26. Wang, L., Fan, H., Xiao, J., Lan, Y. & Wang, X. Criticality in reservoir computer of coupled phase oscillators. *Phys. Rev. E* **105**, L052201 (2022).
27. Feketa, P., Meurer, T. & Kohlstedt, H. Structural plasticity driven by task performance leads to criticality signatures in neuromorphic oscillator networks. *Sci. Rep.* **12**, 15321 (2022).
28. Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R. & Eleftheriou, E. Memory devices and applications for in-memory computing. *Nat. Nanotechnol.* **15**, 529 (2020).
29. Morris, N. & Jones, D. M. Memory updating in working memory: The role of the central executive. *Br. J. Psychol.* **81**, 111 (1990).
30. O'Reilly, R. C. Biologically based computational models of high-level cognition. *Science* **314**, 91 (2006).

31. Delfs, H. & Knebl, H. Symmetric-key cryptography. In *Introduction to Cryptography: Principles and Applications* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2015) pp. 11–48
32. Mitola, J. & Maguire, G. Q. Cognitive radio: making software radios more personal. *IEEE Personal. Commun.* **6**, 13 (1999).
33. Wang, B. & Liu, K. J. R. Advances in cognitive radio networks: A survey. *IEEE J. Sel. Top. Signal Process.* **5**, 5 (2010).
34. Platkiewicz, J. & Brette, R. A threshold equation for action potential initiation. *PLoS Computational Biol.* **6**, e1000850 (2010).
35. Yang, G. R., Joglekar, M. R., Song, H. F., Newsome, W. T. & Wang, X. J. Task representations in neural networks trained to perform many cognitive tasks. *Nat. Neurosci.* **22**, 297 (2019).
36. Medsker, L. & Jain, L. C., *Recurrent neural networks: design and applications* (CRC press, 1999)
37. Yu, Y., Si, X., Hu, C. & Zhang, J. A review of recurrent neural networks: Lstm cells and network architectures. *Neural Comput.* **31**, 1235 (2019).
38. Pascanu, R., Mikolov, T. & Bengio, Y. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning* (Pmlr, 2013) pp. 1310–1318.
39. Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* **5**, 157 (1994).
40. Guidotti, R. et al. A survey of methods for explaining black box models. *ACM Comput. Surv.* **51**, 1 (2018).
41. Jaeger, H. & Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78 (2004).
42. Cucchi, M., Abreu, S., Ciccone, G., Brunner, D. & Kleemann, H. Hands-on reservoir computing: a tutorial for practical implementation. *Neuromorphic Comput. Eng.* **2**, 032002 (2022).
43. Tanaka, G. et al. Recent advances in physical reservoir computing: A review. *Neural Netw.* **115**, 100 (2019).
44. Pathak, J., Hunt, B., Girvan, M., Lu, Z. & Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.* **120**, 024102 (2018).
45. Zador, A. M. The basic unit of computation. *Nat. Neurosci.* **3**, 1167 (2000).
46. Lloyd, S. Any nonlinear gate, with linear gates, suffices for computation. *Phys. Lett. A* **167**, 255 (1992).
47. Vinckier, Q. et al. High-performance photonic reservoir computer based on a coherently driven passive cavity. *Optica* **2**, 438 (2015).
48. Laporte, F., Katumba, A., Dambre, J. & Bienstman, P. Numerical demonstration of neuromorphic computing with photonic crystal cavities. *Opt. Express* **26**, 7955 (2018).
49. Lugnan, A. et al. Photonic neuromorphic information processing and reservoir computing. *APL Photonics* **5**, 020901 (2020).
50. Ma, C., Laporte, F., Dambre, J. & Bienstman, P. Addressing limited weight resolution in a fully optical neuromorphic reservoir computing readout. *Sci. Rep.* **11**, 3102 (2021).
51. Gauthier, D. J., Bollt, E., Griffith, A. & Barbosa, W. A. Next generation reservoir computing. *Nat. Commun.* **12**, 5564 (2021).
52. Kia, B., Lindner, J. F. & Ditto, W. L. Nonlinear dynamics as an engine of computation. *Philos. Trans. R. Soc. A* **375**, 20160222 (2017).
53. Kia, B. et al. Nonlinear dynamics based machine learning: Utilizing dynamics-based flexibility of nonlinear circuits to implement different functions. *Plos One* **15**, e0228534 (2020).
54. Murali, K., Ditto, W. L. & Sinha, S. Reconfigurable noise-assisted logic gates exploiting nonlinear transformation of input signals. *Phys. Rev. Appl.* **18**, 014061 (2022).
55. Choudhary, A. et al. Physics-enhanced neural networks learn order and chaos. *Phys. Rev. E* **101**, 062207 (2020).
56. Kim, J. Z. & Bassett, D. S. A neural machine code and programming framework for the reservoir computer. *Nat. Mach. Intell.* **5**, 622 (2023).
57. Sompolinsky, H., Crisanti, A. & Sommers, H.-J. Chaos in random neural networks. *Phys. Rev. Lett.* **61**, 259 (1988).
58. Kadmon, J. & Sompolinsky, H. Transition to chaos in random neuronal networks. *Phys. Rev. X* **5**, 041030 (2015).
59. Krishnagopal, S., Girvan, M., Ott, E., & Hunt, B. R. Separation of chaotic signals by reservoir computing. Chaos **30**, 023123 (2020).
60. Vlachas, P. R. et al. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural Netw.* **126**, 191 (2020).
61. Izhikevich, E. M. Computing with oscillators. *Neural Netw.* **5255**, 1 (2000).
62. Raychowdhury, A. et al. Computing with networks of oscillatory dynamical systems. *Proc. IEEE* **107**, 73 (2018).
63. Heeger, D. J. & Mackey, W. E. Oscillatory recurrent gated neural integrator circuits (organics), a unifying theoretical framework for neural dynamics. *Proc. Natl Acad. Sci.* **116**, 22783 (2019).
64. Ricci, M. et al. Kuranet: systems of coupled oscillators that learn to synchronize. Preprint at https://arxiv.org/abs/2105.02838 (2021).
65. Zanin, M., Papo, D. & Boccaletti, S. Computing with complex-valued networks of phase oscillators. *Europhys. Lett.* **102**, 40007 (2013).
66. Csaba, G. & Porod, W. Coupled oscillators for computing: A review and perspective. *Appl. Phys. Rev.* **7**, 011302 (2020).
67. Zanin, M., Papo, D., Sendina-Nadal, I. & Boccaletti, S. Computation as an emergent feature of adaptive synchronization. *Phys. Rev. E* **84**, 060102 (2011).
68. Adamatzky, A., Costello, B. D. L. & Asai, T. *Reaction-diffusion computers* (Elsevier, 2005)
69. Adamatzky, A. Universal dynamical computation in multidimensional excitable lattices. *Int. J. Theor. Phys.* **37**, 3069 (1998).
70. Helias, M. & Dahmen, D., *Statistical field theory for neural networks*, Vol. 970 (Springer, 2020)
71. Keup, C., Kühn, T., Dahmen, D. & Helias, M. Transient chaotic dimensionality expansion by recurrent networks. *Phys. Rev. X* **11**, 021064 (2021).
72. Bordelon, B. & Pehlevan, C. Population codes enable learning from few examples by shaping inductive bias. *Elife* **11**, e78606 (2022).
73. Aljadeff, J., Stern, M. & Sharpee, T. Transition to chaos in random networks with cell-type-specific connectivity. *Phys. Rev. Lett.* **114**, 088101 (2015).

Acknowledgements
This work was supported by BrainsCAN at Western University through the Canada First Research Excellence Fund (CFREF), the NSF through a NeuroNex award (#2015276), the Natural Sciences and Engineering Research Council of Canada (NSERC) grant R0370A01, Compute Ontario (computeontario.ca), Digital Research Alliance of Canada (alliancecan.ca), and the Western Academy for Advanced Research. R.C.B. gratefully acknowledges the Western Institute for Neuroscience Clinical Research Postdoctoral Fellowship.

Author contributions
Conceptualization: R.C.B., A.N.B., L.E.M.; Methodology: R.C.B., A.N.B., L.H.B.L., F.W.P., L.E.M.; Experiments and analyses: R.C.B., A.N.B., S.M., E.M., W.I., L.E.M.; Visualization: R.C.B., A.N.B., L.E.M.; Discussion: R.C.B., A.N.B., S.M., E.M., L.H.B.L., F.W.P., J.M., T.C., W.I., L.E.M.; Funding acquisition: W.I., L.E.M.; Supervision: W.I., L.E.M.; Writing: R.C.B., A.N.B., L.E.M.; Revision: R.C.B., A.N.B., S.M., E.M., L.H.B.L., F.W.P., J.M., T.C., W.I., L.E.M.

Competing interests
The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42005-024-01728-0.

Correspondence and requests for materials should be addressed to Lyle E. Muller.

Peer review information *Communications Physics* thanks the anonymous reviewers for their contribution to the peer review of this work. A peer review file is available.

Reprints and permissions information is available at http://www.nature.com/reprints