Recurrent neural network dynamical systems for biological vision

Wayne W.M. Soo
Department of Engineering
University of Cambridge
wmws2@cam.ac.uk

Aldo Battista
Center for Neural Science
New York University
aldo.battista@nyu.edu

Puria Radmard
Department of Engineering
University of Cambridge
pr450@cam.ac.uk

Xiao-Jing Wang
Center for Neural Science
New York University
xjwang@nyu.edu

Abstract

In neuroscience, recurrent neural networks (RNNs) are modeled as continuous-time dynamical systems to more accurately reflect the dynamics inherent in biological circuits. However, convolutional neural networks (CNNs) remain the preferred architecture in vision neuroscience due to their ability to efficiently process visual information, which comes at the cost of the biological realism provided by RNNs. To address this, we introduce a hybrid architecture that integrates the continuous-time recurrent dynamics of RNNs with the spatial processing capabilities of CNNs. Our models preserve the dynamical characteristics typical of RNNs while having comparable performance with their conventional CNN counterparts on benchmarks like ImageNet. Compared to conventional CNNs, our models demonstrate increased robustness to noise due to noise-suppressing mechanisms inherent in recurrent dynamical systems. Analyzing our architecture as a dynamical system is computationally expensive, so we develop a toolkit consisting of iterative methods specifically tailored for convolutional structures. We also train multi-area RNNs using our architecture as the front-end to perform complex cognitive tasks previously impossible to learn or achievable only with oversimplified stimulus representations. In monkey neural recordings, our models capture time-dependent variations in neural activity in higher-order visual areas. Together, these contributions represent a comprehensive foundation to unify the advances of CNNs and dynamical RNNs in vision neuroscience.

1 Introduction

Dynamical systems have long been a cornerstone in the field of neuroscience, tracing their origins back to the modeling of single neurons [1, 2]. Early biophysical models provided critical insights into the dynamic behavior of neurons. Building on this foundation, neuroscientists progressed to constructing networks of dynamical neurons [3–12]. The idea was to understand not just how individual neurons operate but how networks of neurons interact to produce complex behaviors [5–7, 12] and cognitive functions [8–10]. The rise of deep learning and artificial recurrent neural networks (RNNs) marked a significant turning point in this endeavor. Neuroscience quickly adopted RNNs due to their ability to model time-dependent processes [13–29], much like how biological circuits process information over time. In these models, each artificial neuron can behave according to some biophysical dynamics, whether through rate coding [13, 14, 18, 30] or spiking mechanisms [19, 20]. This alignment with

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

biological plausibility made RNNs a powerful tool in neuroscience. Today, advances in computational power and learning algorithms have enabled these networks to be trained on diverse and complex tasks, ranging from motor control [14, 24] to cognitive functions [15, 16, 18, 27, 29–31]. Modern RNNs are capable of learning intricate patterns in data, making them invaluable for modeling a wide array of neural processes. We provide a detailed review in Appendix A. RNNs are not the only models used in neuroscience. Convolutional neural networks (CNNs) have also been widely used, particularly in the domain of biological vision [32–41]. CNNs excel at processing spatial hierarchies in images, making them ideal for object recognition and scene understanding tasks. While there are recurrent CNNs that integrate temporal dynamics into the spatial processing capabilities of CNNs [35, 38, 42, 43], the specific advances of dynamical RNNs in neuroscience have largely remained inapplicable to vision models. To bridge this gap, we propose a hybrid architecture that integrates the continuous-time recurrent dynamics of RNNs with the spatial processing capabilities of CNNs, which we refer to as CordsNet (**Co**nvolutional **R**NN **d**ynamical **s**ystem). Briefly, our contributions and results are:

- **Dynamical expressivity analysis.** We rigorously compare CordsNet with other recurrent dynamical architectures by training them all on multiple cognitive tasks in neuroscience. We find that CordsNet can achieve the same range of dynamical regimes as other architectures.
- **New training algorithm.** Training a continuous-time model is computationally expensive. We propose a computationally cheaper algorithm to efficiently initialize CordsNets and successfully train them on standard image classification benchmarks like ImageNet.
- **Autonomous and robust inference.** We show that our trained models can perform inference autonomously and exhibit robustness to noise compared to static and discrete-time CNNs, which are actually just properties inherent to dynamical systems.
- **Analytical toolkit.** There are many ways to analyze CordsNets, but they are memory intensive. We thus develop a toolkit consisting of iterative and dimensionality-reduction methods needed to analyze the model within reasonable computational limits and show some examples.
- **Image-computable models.** We demonstrate the effectiveness of CordsNet as the front-end of an image-computable multi-area model by training such models to perform tasks using the actual stimuli seen by subjects in experiments rather than abstract inputs.
- **Prediction of neural activity.** We find that CordsNets (trained on ImageNet) are able to predict temporal signatures in the neural activity of higher-order visual areas (V4 and IT).

Together, our contributions represent a comprehensive and directed effort to bring decades of advancements in dynamical systems to vision neuroscience.

2 Model architecture

We first briefly introduce CNNs and dynamical RNNs in neuroscience. A typical convolutional layer in a CNN consists of a 2D-convolution, normalization (commonly batch normalization [44]) and a non-linearity (such as ReLU [45]), although the order of these operations can change [46]. For two consecutive convolutional layers $\mathbf{x}_{l-1}$ and $\mathbf{x}_l$, this can be written as:

$$\mathbf{x}_l = (\sigma \circ \text{Norm} \circ \text{Conv})(\mathbf{x}_{l-1}) \tag{1}$$

where σ is the non-linear activation function. For a single image, the output of a convolutional layer is characterized by three dimensions (channels, height, and width), which can be interpreted as the latent state of the CNN (Figure 1A, left). On the other hand, a continuous-time RNN dynamical system typically found in neuroscience literature [28] is described by:

$$\mathbf{T}\frac{d\mathbf{r}}{dt} = -\mathbf{r} + \sigma\left(\mathbf{W}_{\text{rec}}\mathbf{r} + \mathbf{b} + \mathbf{W}_{\text{inp}}\mathbf{h}_{\text{inp}}\right) \tag{2}$$

where $\mathbf{T}$ represents a diagonal matrix of neuron time constants, $\mathbf{r}$ represents the latent state of the RNN, $\mathbf{W}$ is the recurrent weight matrix, $\mathbf{b}$ is the bias, and $-\mathbf{r}$ is the leaky term which mimics the dynamics of biological neurons. $\mathbf{h}_{\text{inp}}$ is the external input to the network that first undergoes a linear transformation through $\mathbf{W}_{\text{inp}}$ (Figure 1A, middle). Just like in CNNs, σ is the non-linear activation function. Note that there are no normalizing operations as they are generally incompatible with continuous-time dynamical systems. The latent state of the network can be simulated forward in time by Euler's method with a suitable choice of time discretization. Unlike regular RNNs, dynamical RNNs typically evolve (and are therefore backpropagated) over hundreds of time steps [28].

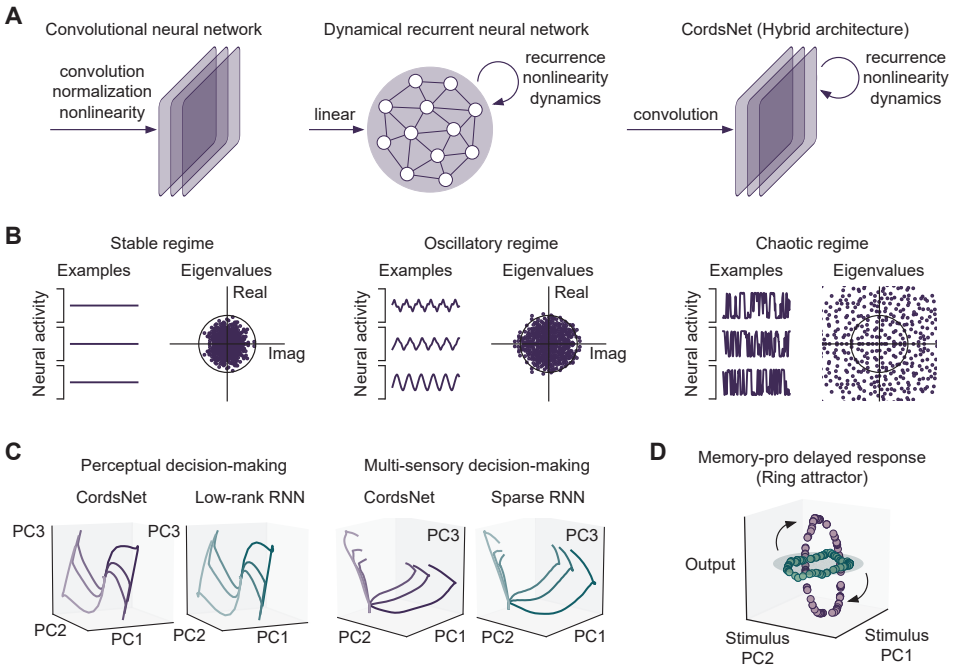

Figure 1: **A.** Overview of the proposed architecture (right) and its relation to CNNs (left) and RNNs (middle). **B.** Various dynamical regimes exhibited by randomly-initialized CordsNets. **C.** Aligned neural trajectories between a CordsNet and a low-rank RNN trained on a perceptual decision-making task (left), as well as between a CordsNet and a sparse RNN trained on a multi-sensory decision-making task (right). **D.** Example of a CordsNet trained to perform a memory-pro delayed response task by storing a circular variable in memory using a ring attractor.

In CordsNet, which is our proposed hybrid architecture, every convolutional layer is replaced by a dynamical RNN (Figure 1A, right). The activations $\mathbf{x}_l$ now represent firing rates $\mathbf{r}_l$. In order to preserve this structure, the input and recurrent weight matrices ($\mathbf{W}_{\text{inp}}$ and $\mathbf{W}_{\text{rec}}$) are now convolutions (Conv_{inp} and Conv_{rec}), while $\mathbf{h}_{\text{inp}}$ and $\mathbf{b}$ are reshaped to have convolutional structures. This can be summarized as:

$$\mathbf{T}\frac{d\mathbf{r}_l}{dt} = -\mathbf{r}_l + \sigma\left(\text{Conv}_{\text{rec}}(\mathbf{r}_l) + \mathbf{b} + \text{Conv}_{\text{inp}}(\mathbf{h}_{\text{inp}})\right) \tag{3}$$

In the rest of this work, we will show that this proposed extension is highly non-trivial and brings the best (and worst) features of dynamical RNNs into CNNs.

2.1 Dynamical characteristics

Since convolutions are linear operations, they can be expressed as a 2-D weight matrix operating on a flattened 1-D vector, just like in equation (2), which we describe in detail in Appendix B.1. However, the convolutional recurrent weight structure of CordsNet spans only a small subspace of all possible recurrent weight matrices, so there is a need to verify if such a restriction would limit the range of dynamical properties that our networks can express. We know that (sufficiently-large) Gaussian-initialized fully-connected networks using tanh activations exhibit different dynamical regimes based on initialization variance [4, 47]. We successfully replicate this in CordsNets to express stable, oscillatory and chaotic behaviors (Figure 1B) under the same conditions. Further analysis on dimensionality and autocorrelations can be found in Appendix B.2.

Multiple variants of dynamical RNNs have been trained on cognitive tasks in neuroscience [15, 28], and thus we want to know if CordsNets will produce similar solutions when trained on such tasks despite the restrictions from the convolutional weight structure. For this investigation, we choose the same set of five tasks that was previously adopted for analyzing low-rank dynamical RNNs [48], consisting of a perceptual decision-making task [49], a parametric working memory task [50], a

multi-sensory decision-making task [51], a contextual decision-making task [13] and a delayed match-to-sample task [52]. We independently train CordsNets, fully-connected RNNs, low-rank RNNs [48] and sparsely-connected RNNs [15] on these tasks across different activation functions, learning rates, network sizes and initializations. We find that trained networks of all architectures produce similar neural trajectories when aligned using canonical correlation analysis (Figure 1C), suggesting that CordsNets are able to utilize the same dynamical motifs as other established architectures in neuroscience to perform cognitive tasks. We also provide a more rigorous and quantitative analysis using recently proposed metrics on representational similarity [53] in Appendix B.3 which further supports this conclusion. In fact, from these metrics, we find that our networks produce more similar solutions to fully-connected RNNs than low-rank or sparsely-connected RNNs.

RNNs can perform tasks requiring long-term dependencies by storing information in memory. For a dynamical RNN, this is typically achieved using attractors in neural activity space [54, 55]. We show that our networks can represent well-known classes of attractors such as ring (Figure 1D), line attractors, and discrete fixed-points (see Appendix B.4). Taken together, our results suggest that the convolutional recurrent weight structure of CordsNet does not constrain its expressiveness as a recurrent dynamical system.

3 Training and results

We now focus on the functionality of CordsNets as an image recognition model. We train networks of four different sizes, named CordsNet-RX, where X $\in [2, 4, 6, 8]$ represents the number of recurrent layers. Exact model specifications can be found in Appendix C.1, where we also review important design choices. We train our models on MNIST [56], Fashion-MNIST [57], CIFAR-10 [58], CIFAR-100 and ImageNet [59], each with dataset-specific augmentations [60] as detailed in Appendix C.2. The neuron time constant is set to 10 ms (constant for all neurons), and the network is simulated with 2 ms time steps. The loss function that we aim to minimize is computed as:

```
loss = logspace(-3,0,steps=30) * CEloss(output[170:200],labels)
       + 1e-3 * MSEloss(activity[290:300],spontaneous)
```
(4)

We first simulate the networks for 200 ms (time steps `0` to `100`) without any input so that they converge to some steady state spontaneous activity (`spontaneous`). A batch of images is then presented for 200 ms (time steps `100` to `200`). During this time, the cross-entropy loss is computed for the last 60 ms (time steps `170` to `200`) and combined using a log-weighted sum. Finally, the networks are simulated for another 200 ms (time steps `200` to `300`), and the mean-squared error between activity in the last 20 ms (time steps `290` to `300`) and spontaneous activity is added to the loss. This additional term encourages the networks to return to spontaneous activity after stimulus presentation. This loss function has been carefully designed to produce a mono-stable solution, so that the network may perform accurate inference infinitely in time, which we will elaborate in the next section. We also provide a detailed ablation study of every coefficient and every term in Appendix D. Simulating these networks 300 steps across time is computationally expensive. For comparison, we compute the total multiply-accumulate operations (MACs) for 79 well-known CNN models found in `torchvision.models` library and compare them against their parameter counts (Figure 2C). Our largest model, CordsNet-R8, has approximately the same number of parameters as ResNet-18, the smallest model of the ResNet series. In contrast, our smallest model, CordsNet-R2, requires more MACs to simulate than ViT-H-14, the largest vision transformer in the model library.

While training the models by computing and minimizing the loss function in equation (4) is inevitable, we can reduce the number of training iterations needed by carefully initializing our models. We do this in three computationally cheaper steps (Figure 2B). We first train a feedforward CNN model without the recurrent component, instead replacing it with a one-time convolutional layer. We also include batch normalization here to improve training efficiency. In the second step, we fold the batch normalization [61] into the convolution operation and replace the one-time convolutional layers with dynamical linear RNNs. Batch normalization folding is done in the following way:

$$\mathrm{BN}(\mathbf{W}_{\text{conv}}\mathbf{r} + \mathbf{b}) = \underbrace{\frac{\gamma_{\text{BN}}}{\sqrt{\sigma^2_{\text{BN}} + \epsilon}}\mathbf{W}_{\text{conv}}}_{\mathbf{W}^{\text{fold}}_{\text{conv}}}\mathbf{r} + \underbrace{\frac{\gamma_{\text{BN}}}{\sqrt{\sigma^2_{\text{BN}} + \epsilon}}(\mathbf{b} - \mu_{\text{BN}}) + \beta_{\text{BN}}}_{\mathbf{b}^{\text{fold}}} \tag{5}$$

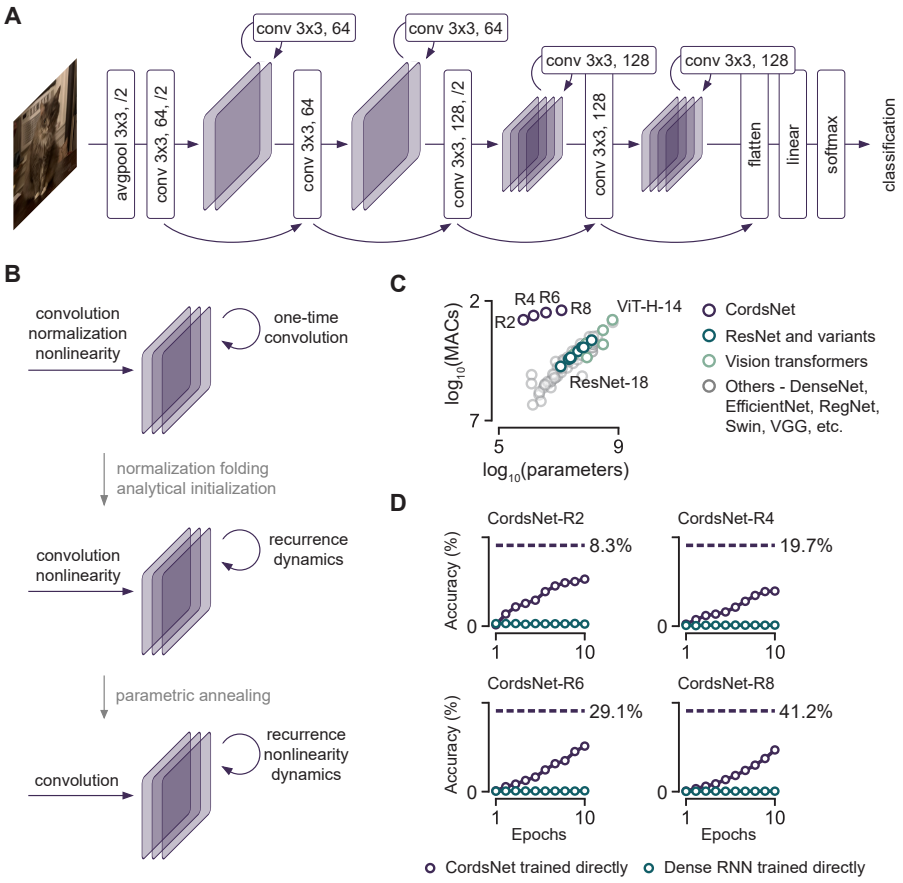

Figure 2: **A.** Architecture of CordsNet-R4. **B.** Proposed initialization method. A feedforward CNN is first trained (top). The parameters are then used to initialize and train linear RNNs (middle). Non-linearity is then introduced by annealing (bottom). **C.** Multiply-accumulate operations (MACs) of 79 CNN models (green/grey) and CordsNet (purple) plotted against parameter counts. **D.** Validation accuracy of our models trained on ImageNet using the aforementioned three steps (dashed lines), compared to training CordsNet (purple circles) and fully-connected RNNs (green circles) directly.

A dynamical linear RNN is described by equation (2), except that σ is simply the identity function. Such a network can be analytically solved for every time step t:

$$\mathbf{r}_t = e^{(\mathbf{W}_{\text{conv}}-\mathbf{I})t}\mathbf{r}_0 + \int_0^t e^{(\mathbf{W}_{\text{conv}}-\mathbf{I})(t-\tau)}(\mathbf{b}+\mathbf{W}_{\text{inp}}\mathbf{h}_{\text{inp}})d\tau \tag{6}$$

We derive its steady-state solution, which can be approximated and compared to a convolutional layer:

$$\underbrace{\mathbf{r}_\infty}_{\text{conv output}} = (\mathbf{I}-\mathbf{W}_{\text{conv}})^{-1}(\mathbf{b}+\mathbf{W}_{\text{inp}}\mathbf{h}_{\text{inp}}) \approx (\mathbf{I}+\mathbf{W}_{\text{conv}}+\mathbf{W}_{\text{conv}}^2+\dots)(\mathbf{b}+\underbrace{\mathbf{W}_{\text{inp}}\mathbf{h}_{\text{inp}}}_{\text{conv input}}) \tag{7}$$

The parameters of the linear RNN are $\mathbf{W}_{\text{conv}}$ and $\mathbf{b}$, which we optimize by minimizing the mean-squared error between its steady state $\mathbf{r}_\infty$ and the output of the convolutional layer, After optimization, we train the full linear model (end-to-end) using a reduced cost function:

```
loss_reduced = logspace(-3,0,steps=30) * CEloss(output[70:100],labels)   (8)
```

We present the image for 200 ms (time steps `0` to `100`) and compute the weighted classification loss for the last 60 ms (time steps `70` to `100`). We do not need to compute the spontaneous penalty term because linear RNNs are almost guaranteed to be mono-stable during training. Finally, in the third step, we replace the identity activation function with a parametric ReLU non-linearity [62]

Table 1: Test accuracies of CordsNets obtained using our initialization method and after fine-tuning, compared to their equivalent feedforward CNN counterparts. For controls, we trained CordsNets directly for the same amount of time taken by the initialization method (C). Fully-connected RNNs were also trained with matched parameter counts (R).

Model	Dataset	Control (R)	Control (C)	Initialization	Fine-tuned	CNN
R2	MNIST	97.45	98.07	97.85	98.48	98.56
	F-MNIST	74.17	77.75	85.60	88.12	88.44
	CIFAR-10	45.68	44.44	62.83	71.83	73.89
	CIFAR-100	16.19	10.00	24.65	39.56	40.42
	ImageNet	0.22	5.55	8.27	14.23	15.57
R4	MNIST	97.81	99.26	98.82	99.24	99.59
	F-MNIST	84.39	87.63	91.64	92.68	93.65
	CIFAR-10	48.79	60.64	81.38	88.76	90.29
	CIFAR-100	17.78	16.93	44.28	60.24	63.95
	ImageNet	0.31	11.67	19.67	33.78	36.28
R6	MNIST	98.16	99.32	99.26	99.38	99.76
	F-MNIST	86.62	90.82	93.36	94.62	95.32
	CIFAR-10	52.68	76.82	88.57	91.32	93.87
	CIFAR-100	22.88	42.56	56.98	71.32	75.70
	ImageNet	0.22	19.33	29.14	50.14	52.06
R8	MNIST	87.51	99.40	99.37	99.36	99.74
	F-MNIST	87.51	91.82	94.26	95.88	96.13
	CIFAR-10	53.74	83.15	91.57	94.56	95.99
	CIFAR-100	25.11	51.34	66.89	77.32	78.44
	ImageNet	0.26	23.21	41.24	57.90	63.16

with parameter a, where $\text{PReLU}(x) = \max(0, x) + a \times \min(0, x)$. We train the networks starting from $a = 1$ and gradually reducing a until $a = 0$ at the end of training. This provides us with an initialization of the original model for further fine-tuning. We additionally perform two control experiments, where we train both fully-connected RNNs (with matched parameter counts) and CordsNets directly without our initialization method. In both controls, the number of training epochs is determined by matching the time required for CordsNets to be trained directly with the time required for CordsNets to be initialized using our method. Additional details about this time benchmark can be found in Appendix C.3. Test accuracies of all experiments can be found in Table 1 (for ImageNet, the validation accuracy is shown instead). We draw three main conclusions:

- **Initialization vs Control (C).** Our initialization method consistently produces models with higher test accuracies compared models that were directly trained on every dataset except for MNIST, highlighting the effectiveness of our initialization approach. Figure 2D compares the ImageNet validation accuracy of CordsNets trained directly (purple circles) across all epochs against the validation accuracy of the models initialized using our method (purple dashed lines). MNIST classification is simple and does not require extensive feature extraction; it may therefore benefit from a more straightforward training approach.
- **Control (R) vs Control (C).** When trained for the same number of epochs, CordsNets R6/R8 consistently outperforms their parameter-matched fully-connected RNN counterparts on all datasets by a significant margin. Fully-connected RNNs perform remarkably poor on ImageNet due to their poor scaling with image size (Figure 2D, green circles). However, the results become more ambiguous when it comes to smaller models on simpler datasets. We attribute this to the flexibility of fully-connected models which allow them to attain early and fast gains in performance, especially when learning simpler features in smaller datasets.
- **Fine-tuned vs CNN.** Our fine-tuned CordsNets are able to attain accuracies that are slightly lower than (but reasonably close to) their feedforward CNN counterparts, suggesting that we have successfully trained a continuous-time dynamical system with competitive image processing capabilities. Closing the performance gap with CNNs remains a goal for future work.

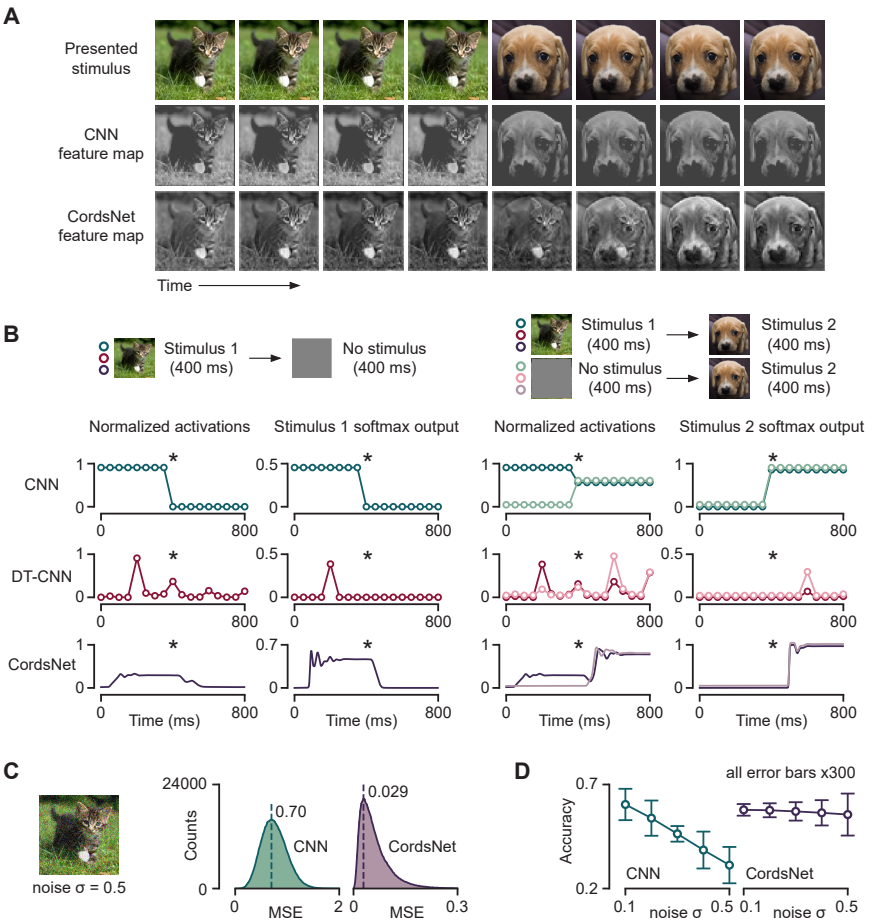

Figure 3: **A.** Evolution of a single interpretable feature map over time in the first layer of a feedforward CNN (middle) and CordsNet-R8 (bottom). **B.** Neural activity and softmax output of a feedforward CNN (green), discrete-time CNN (red), and CordsNet-R8 (purple) in response to various stimuli sequences. **C.** Mean-squared deviation from noiseless activations of the output layer across 50000 noisy images. **D.** ImageNet validation accuracies over 5 noise levels.

4 Model analysis

As a continuous-time model, the feature maps of CordsNets will change over time. We handpick a single interpretable feature map from CordsNet-R8 that depicts a gradual evolution over time when the input image is changed (Figure 3A). To emphasize the rich temporal dynamics that trained CordsNets express, we consider an additional comparison with CORNet-RT [35], a class of discrete-time recurrent CNN trained on ImageNet. In addition to architectural differences, the two models are trained differently. CordsNet has been trained to classify images over some time interval (and expected to perform accurately infinitely across time), while CORNet-RT has been trained to classify for a single time step (and expected to be accurate for that particular time step).

We compare the temporal activities of the two models (as well as the feedforward CNN) in two scenarios. In the first scenario, an image is presented for 400 ms and then removed for the next 400 ms (Figure 3B, left). The feedforward CNN, having no concept of time, responds with the same activations across time and drops to zero when there is no input. The activity in CORNet-RT varies over time, and spikes briefly at the time step in which it is trained to make an accurate classification of the input image. It is not able to make an accurate prediction at all other times. In contrast, the activity in CordsNet rises and stays at a fixed level for as long as the image is presented and returns to some baseline activity when the image is removed. In the second scenario, an image is presented for 400 ms, followed by a different image for 400 ms (Figure 3B, right). For its particular time step,

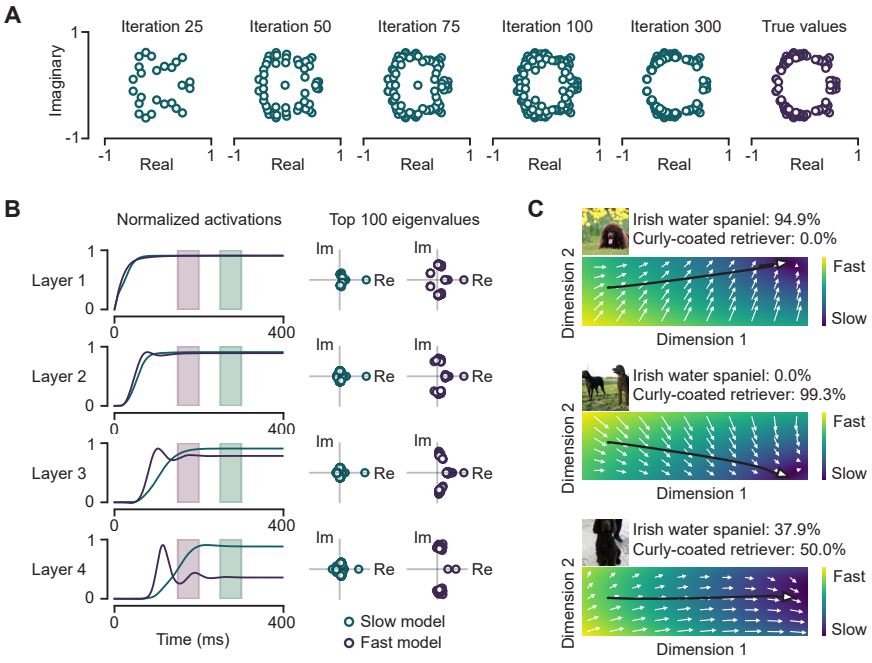

Figure 4: **A.** Arnoldi iteration [63] applied to a convolutional recurrent weight matrix. **B.** Model activations in CordsNet-R4 trained to classify images in time intervals $[140\text{ ms}, 200\text{ ms}]$ (purple) and $[240\text{ ms}, 300\text{ ms}]$ (green), along with their top 100 eigenvalues (right). **C.** Neural trajectories in the final layer of CordsNet-R8 projected onto two dimensions when presented with 3 different images.

CORNet-RT correctly classifies the first image but cannot correctly classify the second image, even though it can correctly classify the second image if it were presented first. CordsNet can correctly classify both images and once again maintains the correct classification throughout the duration of the stimulus. These results showcase the autonomous nature of CordsNet compared to discrete-time CNNs – it can perform inference infinitely in time, self-reset to a baseline state, and react flexibly to stimuli changes, all while being governed by a single differential equation.

In the feedforward CNN, when temporally uncorrelated white noise is applied to the input image at each time step, model activations deviate from baseline noiseless values, which results in a decrease in classification accuracy (Figure 3C-D, green). When the same noisy images are instead presented to CordsNet, its deviations are attenuated by more than an order of magnitude compared to those of the feedforward CNN. It also robustly maintains its classification accuracy at high noise levels (Figure 3C-D, purple). This is a result of natural noise attenuation from continuous-time dynamics and filtering by the recurrent weights. We can rewrite the dynamics as an update equation to depict these effects:

$$\mathbf{r}_{t+1} = \left(1 - \frac{\Delta t}{T}\right)\mathbf{r}_t + \underbrace{\frac{\Delta t}{T}}_{\text{attenuation}} f(\overbrace{\mathbf{W}\mathbf{r}_t}^{\text{filtering (next step)}} + \mathbf{b} + \mathbf{W}_{\text{inp}}\mathbf{h}_{\text{inp}} + \text{noise}) \tag{9}$$

We next present an analysis of our trained models from a dynamical systems point of view. For a dynamical RNN, this typically involves performing some form of matrix decomposition on its recurrent weight matrix or applying dimensionality-reduction techniques on neural trajectories. For CordsNet, both of these approaches are particularly challenging due to the size of the networks. Memory limitations prevent the full recurrent weight matrix to be expanded from its kernel representation. One solution that we found is to compute the eigenvalues of the recurrent weight matrix directly from its kernel form using Arnoldi iteration [63] (Figure 4A), which we use to uncover an important dynamical motif present in our trained networks. We notice that the dynamical characteristics of our networks are different when we train them to correctly classify images at different times. When trained to classify early, the networks exhibit an oscillatory behavior, but not when trained to classify late (Figure 4B, left). We attribute this to the effect of transient overshooting, where being in the

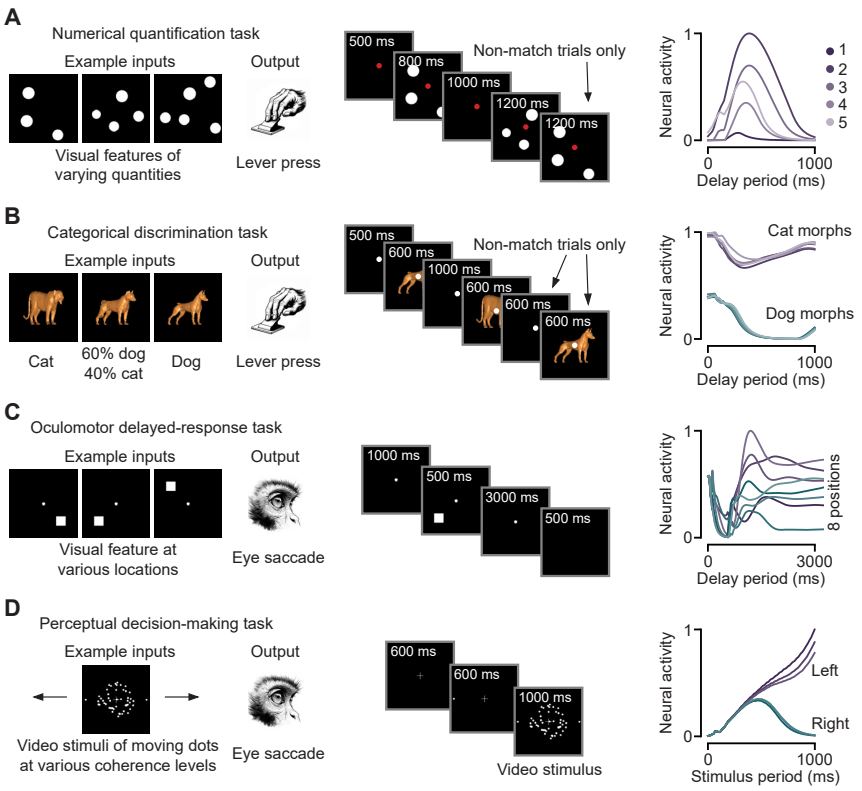

Figure 5: A multi-area model consisting of CordsNet-R8 connected to a fully-connected RNN trained on various cognitive tasks in neuroscience. **A.** Numerical quantification task where monkeys are required to remember the number of visual features on the screen. **B.** Discrimination task where monkeys must discern whether an image is predominantly cat or dog. **C.** Delayed-response task where monkeys have to saccade to the location of a previously shown stimulus. **D.** Evidence integration task where monkeys are required to tell if random dots are moving left or right.

oscillatory regime results in an early overshoot in activity, thereby speeding up the propagation of the signal. In the linear approximation, according to equation (6), this would manifest if the eigenvalues of $(\mathbf{W}_{\text{conv}} - \mathbf{I})$ of the model trained for faster inference contain larger imaginary components. We confirm this to be the case (Figure 4B, right). It is also possible to perform dimensionality reduction within reasonable memory limits, but these methods require some adaptation for convolutional operations. Therefore, we release a toolkit consisting of the aforementioned functions that are specifically tailored for convolutional weights. These convenient tools unlock many possible approaches for analyzing CordsNets. For example, we identify a particular dimension in CordsNet-R8 which represents whether the model perceives an image as an Irish water spaniel or a curly-coated retriever (Figure 4C).

5 Applications

CordsNet is an ideal front-end of any RNN dynamical system that requires image-computability. To demonstrate this, we connect the final layer of CordsNet-R8 with a fully-connected RNN (with 512 neurons), and train only the fully-connected RNN on a set of cognitive tasks that explicitly requires visual information processing (Figure 5). RNN dynamical systems that have previously been trained on these tasks have always modeled the visual stimuli as unrealistic abstract inputs. In contrast, we use the actual images the monkeys see as the input to our multi-area model. This is done either by obtaining the stimuli set from the authors of the experiments [64, 65], or generated according to the specifications described in the original experimental papers [66, 67]. In our trained models, we found interpretable neurons in the fully-connected RNN layer, such as neurons tuned to stimulus quantity (Figure 5A, right), cats or dogs (Figure 5B, right), spatial positions on the screen (Figure 5C, right) and motion direction (Figure 5D, right).

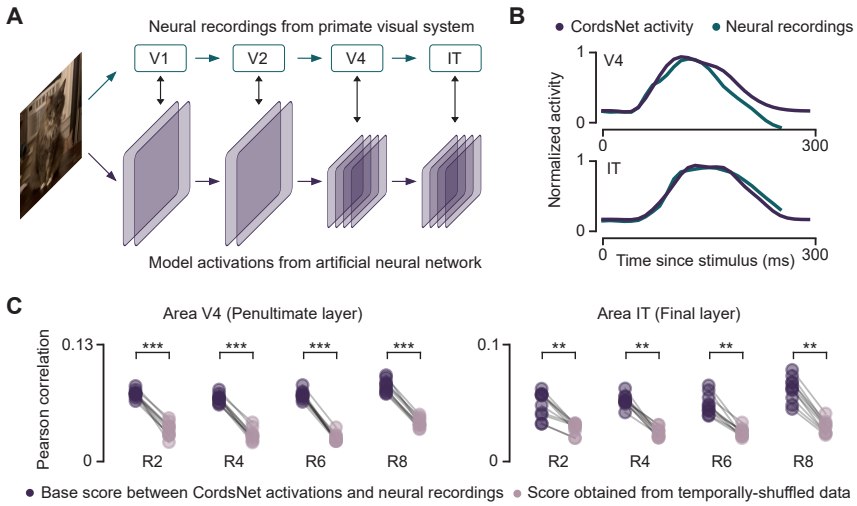

Figure 6: **A.** Framework of Brain-Score (Vision). **B.** Normalized CordsNet-R8 activity in the last two layers compared with experimentally-recorded [68] normalized activity in visual areas V4 and IT. **C.** Similarity metrics between CordsNet model activations and neural data (dark purple) and temporally-shuffled neural data (light purple). (paired t-test, ** $p < 10^{-4}$, *** $p < 10^{-7}$)

Brain-Score is a benchmarking framework designed to evaluate the performance of artificial neural networks in terms of their ability to model and predict neural and behavioral data from the brain [36]. In particular, model activations from CNN layers have been compared with time-averaged neural data from various parts of the primate visual system (Figure 6A). With our continuous-time model, we want to know if our models are able to model temporal signatures in brain activity. To do so, we omit the time-averaging step when computing brain-scores (see Appendix E for details). Using the same training-test splits, we additionally fit the same model activations with temporally-shuffled neural data. Due to the lagged response in neural data, we have to temporally shift our model activations such that the time in which activity rises in response to an input is aligned with neural data. There are occasions where no shifting is necessary, such as in the final two layers of CordsNet-R8 (Figure 6B) on V4 and IT neural activity [68]. All CordsNet models score higher on unshuffled data across V4 and IT (paired t-test, all p-values $< 10^{-4}$, Figure 6C), suggesting that the models are capturing temporal structures within the neural data. For the sake of transparency, we also state that we have performed the same test on a different dataset with V1 and V2 activities [69], but obtained inconclusive results.

6 Discussion and conclusion

We have presented CordsNet, a hybrid architecture combining the strengths of CNNs and dynamical RNNs to process visual information with continuous-time dynamics. CordsNet exhibits various dynamical behaviors, such as oscillations and chaos (Figure 1B). In our analysis of its image processing capabilities, we showcase its ability to classify images infinitely in time (Figure 3B, right), as well as its robustness to noise (Figure 3D, right). We have also effectively used CordsNet as a means to build image-computable dynamical systems capable of performing cognitive tasks (Figure 5). Finally, CordsNet has been successful at modeling the temporal signatures of neural activity in higher-order visual areas like V4 and IT (Figure 6C).

Limitations. The main limitation of CordsNet lies in its substantial memory requirements compared to its small parameter count (Figure 2C). Another limitation is the difficulty of analyzing CordsNet from a dynamical systems perspective, primarily due to how it is unfeasible to convert the recurrent kernel weights into its corresponding full-size recurrent weight matrix.

Conclusion. CordsNet bridges a crucial gap between dynamical systems and vision neuroscience. While most of cognitive neuroscience continues to build upon decades of RNN research, vision neuroscience remains dominated by CNN models due to the inability of RNNs to efficiently process visual information. CordsNet has resolved this limitation by being a dynamical RNN with image processing capabilities.

Author contributions

W.S. designed the architecture. W.S, A.B. and P.R. trained and analyzed the networks. All authors designed the study, took part in discussions, interpreted the results, and wrote the paper.

Acknowledgments and Disclosure of Funding

This work was supported by the NIH grant R01MH062349, Office of Naval Research grant N00014, James Simons Foundation grant NC-GB-CULM-00003138 and NYU High Performance Computing. A.B. was supported by the Swartz Foundation.

References

[1] Hodgkin, A. L. & Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology* **117**, 500–544 (1952).

[2] Knight, B. W. Dynamics of encoding in a population of neurons. *The Journal of General Physiology* **59**, 734–766 (1972).

[3] van Vreeswijk, C. & Sompolinsky, H. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science* **274**, 1724–1726 (1996).

[4] Sompolinsky, H., Crisanti, A. & Sommers, H. J. Chaos in random neural networks. *Physical Review Letters* **61**, 259–262 (1988).

[5] Seung, H. S. How the brain keeps the eyes still. *Proceedings of the National Academy of Sciences* **93**, 13339–13344 (1996).

[6] Bulsara, A. R., Elston, T. C., Doering, C. R., Lowen, S. B. & Lindenberg, K. Cooperative behavior in periodically driven noisy integrate-fire models of neuronal dynamics. *Physical Review E* **53**, 3958–3969 (1996).

[7] Zhang, K. Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory. *Journal of Neuroscience* **16**, 2112–2126 (1996).

[8] Amit, D. J. & Brunel, N. Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cerebral Cortex* **7**, 237–252 (1997).

[9] Amit, D. J. & Brunel, N. Dynamics of a recurrent network of spiking neurons before and following learning. *Network: Computation in Neural Systems* **8**, 373–404 (1997).

[10] Brunel, N. & Sergi, S. Firing frequency of leaky intergrate-and-fire neurons with synaptic current dynamics. *Journal of Theoretical Biology* **195**, 87–95 (1998).

[11] Vogels, T. P., Rajan, K. & Abbott, L. F. Neural network dynamics. *Annu. Rev. Neurosci.* **28**, 357–376 (2005).

[12] Sussillo, D. & Abbott, L. F. Generating coherent patterns of activity from chaotic neural networks. *Neuron* **63**, 544–557 (2009).

[13] Mante, V., Sussillo, D., Shenoy, K. V. & Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503**, 78–84 (2013).

[14] Sussillo, D., Churchland, M. M., Kaufman, M. T. & Shenoy, K. V. A neural network that finds a naturalistic solution for the production of muscle activity. *Nature neuroscience* **18**, 1025–1033 (2015).

[15] Song, H. F., Yang, G. R. & Wang, X.-J. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLoS computational biology* **12**, e1004792 (2016).

[16] Song, H. F., Yang, G. R. & Wang, X.-J. Reward-based training of recurrent neural networks for cognitive and value-based tasks. *Elife* **6**, e21492 (2017).

[17] Cueva, C. J. & Wei, X.-X. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770* (2018).

[18] Masse, N. Y., Yang, G. R., Song, H. F., Wang, X.-J. & Freedman, D. J. Circuit mechanisms for the maintenance and manipulation of information in working memory. *Nature neuroscience* **22**, 1159–1167 (2019).

[19] Kim, R., Li, Y. & Sejnowski, T. J. Simple framework for constructing functional spiking recurrent neural networks. *Proceedings of the national academy of sciences* **116**, 22811–22820 (2019).

[20] Kim, R. & Sejnowski, T. J. Strong inhibitory signaling underlies stable temporal dynamics and working memory in spiking neural networks. *Nature neuroscience* **24**, 129–139 (2021).

[21] Cueva, C. J. *et al.* Low-dimensional dynamics for working memory and time encoding. *Proceedings of the National Academy of Sciences* **117**, 23021–23032 (2020).

[22] Keller, A. J. *et al.* A disinhibitory circuit for contextual modulation in primary visual cortex. *Neuron* **108**, 1181–1193 (2020).

[23] Kleinman, M., Chandrasekaran, C. & Kao, J. A mechanistic multi-area recurrent network model of decision-making. *Advances in neural information processing systems* **34**, 23152–23165 (2021).

[24] Saxena, S., Russo, A. A., Cunningham, J. & Churchland, M. M. Motor cortex activity across movement speeds is predicted by network-level strategies for generating muscle activity. *Elife* **11**, e67620 (2022).

[25] Soo, W. & Lengyel, M. Training stochastic stabilized supralinear networks by dynamics-neutral growth. *Advances in Neural Information Processing Systems* **35**, 29278–29291 (2022).

[26] Stroud, J. P., Watanabe, K., Suzuki, T., Stokes, M. G. & Lengyel, M. Optimal information loading into working memory explains dynamic coding in the prefrontal cortex. *Proceedings of the National Academy of Sciences* **120**, e2307991120 (2023).

[27] Goudar, V., Peysakhovich, B., Freedman, D. J., Buffalo, E. A. & Wang, X.-J. Schema formation in a neural population subspace underlies learning-to-learn in flexible sensorimotor problem-solving. *Nature Neuroscience* **26**, 879–890 (2023).

[28] Soo, W., Goudar, V. & Wang, X.-J. Training biologically plausible recurrent neural networks on cognitive tasks with long-term dependencies. *Advances in Neural Information Processing Systems* **36**, 32061–32074 (2023).

[29] Fascianelli, V. *et al.* Neural representational geometries reflect behavioral differences in monkeys and recurrent neural networks. *Nature Communications* **15**, 6479 (2024).

[30] 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. *Nature Neuroscience* **22**, 297–306 (2019).

[31] Liu, Y. & Wang, X.-J. Flexible gating between subspaces in a neural network model of internally guided task switching. *Nature Communications* **15**, 6497 (2024).

[32] Yamins, D. L. *et al.* Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the national academy of sciences* **111**, 8619–8624 (2014).

[33] Yamins, D. L. & DiCarlo, J. J. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience* **19**, 356–365 (2016).

[34] Nayebi, A. *et al.* Task-driven convolutional recurrent models of the visual system. *Advances in neural information processing systems* **31** (2018).

[35] Kubilius, J. *et al.* Cornet: Modeling the neural mechanisms of core object recognition. *BioRxiv* (2018).

[36] Schrimpf, M. *et al.* Brain-score: Which artificial neural network for object recognition is most brain-like? *BioRxiv* (2018).

[37] Kubilius, J. *et al.* Brain-like object recognition with high-performing shallow recurrent anns. *Advances in neural information processing systems* **32** (2019).

[38] Kar, K., Kubilius, J., Schmidt, K., Issa, E. B. & DiCarlo, J. J. Evidence that recurrent circuits are critical to the ventral stream's execution of core object recognition behavior. *Nature neuroscience* **22**, 974–983 (2019).

[39] Zhuang, C. *et al.* Unsupervised neural network models of the ventral visual stream. *Proceedings of the National Academy of Sciences* **118**, e2014196118 (2021).

[40] Lindsay, G. W. Convolutional neural networks as a model of the visual system: Past, present, and future. *Journal of cognitive neuroscience* **33**, 2017–2031 (2021).

[41] Nayebi, A., Rajalingham, R., Jazayeri, M. & Yang, G. R. Neural foundations of mental simulation: Future prediction of latent representations on dynamic scenes. *Advances in Neural Information Processing Systems* **36** (2024).

[42] Liang, M. & Hu, X. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3367–3375 (2015).

[43] Nayebi, A. *et al.* Recurrent connections in the primate ventral visual stream mediate a trade-off between task performance and network size during core object recognition. *Neural Computation* **34**, 1652–1675 (2022).

[44] Ioffe, S. & Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[45] Nair, V. & Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning*, 807–814 (2010).

[46] Franceschi, D. D. & Jang, J. H. Demystifying batch normalization: Analysis of normalizing layer inputs in neural networks. In *Optimization and Learning: Third International Conference, OLA 2020, Cádiz, Spain, February 17–19, 2020, Proceedings 3*, 49–57 (Springer, 2020).

[47] Mastrogiuseppe, F. & Ostojic, S. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron* **99**, 609–623 (2018).

[48] Dubreuil, A., Valente, A., Beiran, M., Mastrogiuseppe, F. & Ostojic, S. The role of population structure in computations through neural dynamics. *Nature neuroscience* **25**, 783–794 (2022).

[49] Gold, J. I. & Shadlen, M. N. The neural basis of decision making. *Annu. Rev. Neurosci.* **30**, 535–574 (2007).

[50] Romo, R., Brody, C. D., Hernández, A. & Lemus, L. Neuronal correlates of parametric working memory in the prefrontal cortex. *Nature* **399**, 470–473 (1999).

[51] Raposo, D., Kaufman, M. T. & Churchland, A. K. A category-free neural population supports evolving demands during decision-making. *Nature neuroscience* **17**, 1784–1792 (2014).

[52] Miyashita, Y. & Chang, H. S. Neuronal correlate of pictorial short-term memory in the primate temporal cortexyasushi miyashita. *Nature* **331**, 68–70 (1988).

[53] Williams, A. H., Kunz, E., Kornblith, S. & Linderman, S. Generalized shape metrics on neural representations. *Advances in Neural Information Processing Systems* **34**, 4738–4750 (2021).

[54] Battista, A. & Monasson, R. Capacity-resolution trade-off in the optimal learning of multiple low-dimensional manifolds by attractor neural networks. *Physical Review Letters* **124**, 048302 (2020).

[55] Driscoll, L. N., Shenoy, K. & Sussillo, D. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience* 1–15 (2024).

[56] Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine* **29**, 141–142 (2012).

[57] Xiao, H., Rasul, K. & Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[58] Krizhevsky, A. & Hinton, G. Learning multiple layers of features from tiny images. *The CIFAR-10/CIFAR-100 dataset* (2009).

[59] Deng, J. *et al.* Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255 (Ieee, 2009).

[60] Cubuk, E. D., Zoph, B., Shlens, J. & Le, Q. V. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719* **2**, 7 (2019).

[61] Jacob, B. *et al.* Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2704–2713 (2018).

[62] He, K., Zhang, X., Ren, S. & Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034 (2015).

[63] Arnoldi, W. E. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics* **9**, 17–29 (1951).

[64] Freedman, D. J., Riesenhuber, M., Poggio, T. & Miller, E. K. Categorical representation of visual stimuli in the primate prefrontal cortex. *Science* **291**, 312–316 (2001).

[65] Nieder, A., Freedman, D. J. & Miller, E. K. Representation of the quantity of visual items in the primate prefrontal cortex. *Science* **297**, 1708–1711 (2002).

[66] Roitman, J. D. & Shadlen, M. N. Response of neurons in the lateral intraparietal area during a combined visual discrimination reaction time task. *Journal of neuroscience* **22**, 9475–9489 (2002).

[67] Funahashi, S., Bruce, C. J. & Goldman-Rakic, P. S. Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *Journal of neurophysiology* **61**, 331–349 (1989).

[68] Majaj, N. J., Hong, H., Solomon, E. A. & DiCarlo, J. J. Simple learned weighted sums of inferior temporal neuronal firing rates accurately predict human core object recognition performance. *Journal of Neuroscience* **35**, 13402–13418 (2015).

[69] Freeman, J., Ziemba, C. M., Heeger, D. J., Simoncelli, E. P. & Movshon, J. A. A functional and perceptual signature of the second visual area in primates. *Nature neuroscience* **16**, 974–981 (2013).

Recurrent neural network dynamical systems for biological vision

Supplementary material

A Recurrent neural network dynamical systems in neuroscience

Single neuron biophysical models. The dynamics of single neurons must be captured to some reasonable fidelity in order to develop reliable models of networks. Two of the most widely studied models for single neurons are the Hodgkin-Huxley model [1] and the leaky integrate-and-fire model [2]. The Hodgkin-Huxley model was developed to explain the ionic mechanisms underlying action potential generation and propagation in neurons, using the squid giant axon as the experimental model. However, the complexity of this model limits its use in large-scale simulations involving thousands of neurons. To address this challenge, simplified models like the leaky integrate-and-fire model have been developed. This model abstracts away the ion channel dynamics and instead focuses on capturing the core behavior of a spiking neuron.

Handcrafted networks. Before the accessibility of deep learning tools, neuroscientists have proposed models of RNNs by manually constructing their recurrent weights. One such example is the Hopfield network [3], which stores patterns and recalls them even from incomplete or corrupted inputs. This is achieved by constructing its weights based on a Hebbian learning rule that helps form attractors corresponding to the stored patterns [4]. Handcrafted continuous-time RNN dynamical systems have also been suggested to explain various observations in neuroscience literature, such as the functioning of head-direction cells [5] and how the brain maintains stable eye positions [6].

Another important subfield in neuroscience studies the dynamical properties of RNN dynamical systems where the recurrent weights are randomly generated from some (controlled) distribution [7, 8]. This was motivated by the need to study large-scale network behavior before the advent of deep learning. Various dynamical motifs, such as oscillatory and chaotic regimes, have been identified in RNNs from such models through analyses of autocorrelations and Lyapunov exponents.

Trained networks. Once artificial RNNs are adapted for continuous-time dynamics and applied to biologically relevant tasks, they can be used to study various aspects of brain function. One approach is to compare the emergent properties of RNN activity with the patterns observed in real neurons. For instance, RNNs can generate low-dimensional dynamical structures, such as line attractors, which correspond to stable states or trajectories in neural state space [9]. This approach was first applied to investigate how the prefrontal cortex integrates sensory inputs to guide context-dependent decisions, focusing on how neurons in this brain region handle the selection of relevant information while ignoring irrelevant inputs [10].

Deep equilibrium models are a class of artificial neural networks that are remarkably similar to RNN dynamical systems [11]. At its core, deep equilibrium models rely on fixed points of single layers (equivalent to infinitely stacking the same layer) instead of relying on stacking multiple distinct layers. However, the similarities start to break down when considering the order of computations in the presence of multiple layers. In RNNs with multiple layers, the activations of every layer are updated sequentially at each time step. In the case of deep equilibrium models, a single layer is first simulated to its fixed point before any computation in the next layer is performed.

In a different approach, continuous-time RNN dynamical systems were trained on a wide variety of cognitive tasks simultaneously [12]. The work investigates how such RNNs organize their internal architecture and how their recurrent units become functionally specialized for different aspects of cognition. The analysis of these functional clusters revealed that tasks that required similar cognitive processes, such as decision-making across sensory modalities, recruited overlapping clusters of neurons. This indicates that the network reused certain clusters across tasks that shared underlying computational principles.

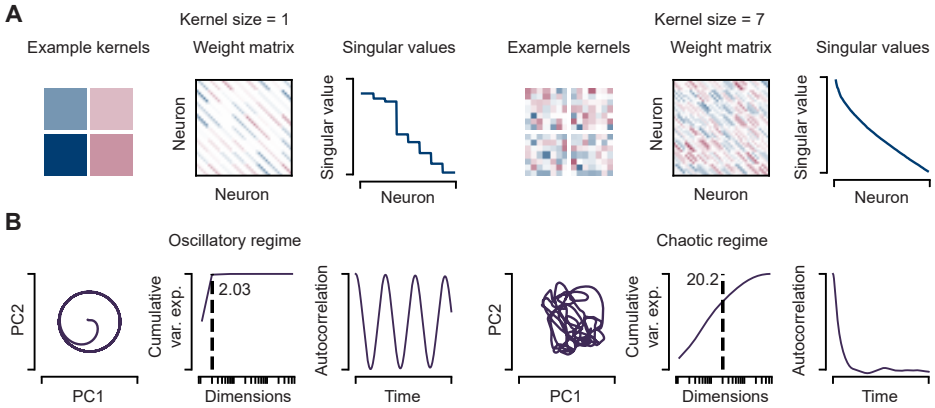

Figure S1: **A.** Examples of convolutional kernels, their resultant flattened weight matrices as described in equation (2) and their distribution of singular values.**B.** Gaussian-initialized CordsNets exhibiting an oscillatory regime (left) and a chaotic regime (right). In each regime, we plot the population trajectory of the space of the first two principal components (left), the cumulative variance explained and participation ratio (middle) and activity autocorrelation (right).

B Analysis of dynamical characteristics

B.1 Converting a convolution into a 2-D weight matrix

CordsNets replace the recurrent and input weight matrices of dynamical RNNs with convolutional operations. This is possible due to the fact that convolutions are linear operations, and can therefore be rewritten as a 2-D matrix operating on a flattened 1-D vector. Here, we review the steps required to this on a single channel. Consider an input matrix $\mathbf{X}$ of size $H_X \times W_X$ and a convolutional kernel $\mathbf{K}$ of size $H_K \times W_K$. Applying the convolutional kernel $\mathbf{K}$ onto the input $\mathbf{X}$ would result in an output $\mathbf{Y}$ with size $(H_X - H_K + 1) \times (W_X - W_K + 1)$, assuming no padding and a stride of 1. The (i, j)-th element of $\mathbf{Y}$ can be computed as:

$$Y_{i,j} = \sum_{m=0}^{H_K-1} \sum_{n=0}^{W_K-1} X_{i+m,j+n} K_{m,n} \tag{1}$$

This can be flattened into a linear operation:

$$\mathbf{Y}^{\text{flat}} = \mathbf{K}^{\text{flat}} \mathbf{X}^{\text{flat}} \tag{2}$$

where $\mathbf{X}^{\text{flat}}$ is a $H_X W_X \times 1$ vector representing the flattened input. $\mathbf{K}^{\text{flat}}$ would therefore have size $(H_X - H_K + 1)(W_X - W_K + 1) \times H_X W_X$. The elements of $\mathbf{K}^{\text{flat}}$ can be determined by gathering the terms of $\mathbf{K}$ that are multiplied with a specific $X_{i+m,j+n}$ in equation (1). This is visualized in Figure S1A, along with the singular values of $\mathbf{K}^{\text{flat}}$. We see that for sufficiently large kernel sizes, the singular values decay smoothly just like in full-rank matrices.

B.2 Randomly-initialized CordsNets

Gaussian-initialized dynamical RNNs with tanh activation functions have been extensively studied in neuroscience literature [7, 13]. Depending on the variance of the Gaussian initialization, fully-connected RNNs can exhibit three distinct dynamical regimes: stable, oscillatory and chaotic. In the stable regime, the network converges to a stable fixed point and remains stationary. In the oscillatory regime, the network evolves over time in a periodic manner in low-dimensional activity space (Figure S1B, left). We estimate activity dimensionality from its participation ratio (PR):

$$\text{PR} = \frac{\left(\sum_i \lambda_i\right)^2}{\sum_i \lambda_i^2} \tag{3}$$

where λ_i represents the i-th eigenvalue of the activity covariance matrix [14, 15]. The activity autocorrelation also reflects this periodicity (Figure S1B, left). In contrast, the chaotic regime

is characterized by aperiodic and unpredictable behavior. This regime is identified by a positive Lyapunov exponent, indicating exponential divergence of nearby trajectories. Network activity has a much higher dimensionality compared to the oscillatory regime, and the autocorrelation decays quickly to zero (Figure S1B, right).

B.3 Comparison with other architectures

We compare the solutions found by training CordsNets, fully-connected RNNs, low-rank RNNs [16] and sparsely-connected RNNs [17] on five cognitive tasks across different activation functions, learning rates, network sizes and initializations. In addition to CordsNets, the other architectures in our comparison are:

- Fully-connected RNNs have full-rank weight matrices of size $N \times N$, where N is the number of neurons.
- Low-rank RNNs have low-rank weight matrices with rank $R \ll N$. This is implemented by decomposing the matrix into two smaller matrices $\mathbf{P}$ and $\mathbf{Q}$ with sizes $N \times R$ and $R \times N$, so that $\mathbf{W}_{\text{low-rank}} = \mathbf{PQ}$.
- Sparsely-connected RNNs have sparse weight matrices, implemented by an element-wise mask $\mathbf{W}_{\text{sparse}} = \mathbf{W} \odot \mathbf{M}$, where every element in the mask $\mathbf{M}$ is 1 with some probability p and 0 otherwise.

The cognitive tasks are the same tasks previously adopted in the analysis of low-rank RNNs [16]:

- The **perceptual decision-making** (PDM) task consists of a fixation epoch of 100 ms, followed by a stimulus epoch of 800 ms, a delay epoch of 100 ms and a decision epoch of 20 ms. During the stimulus epoch, a noisy signal (Gaussian with standard deviation 0.1) drawn uniformly from $\{\pm 0.4, \pm 0.2, \pm 0.1\}$ is presented as input to the networks. The output of the networks during the decision epoch should indicate the sign of the signal.
- The **parametric working memory** task (PWM) consists of a fixation epoch of 100 ms, a stimulus epoch of 100 ms, a delay epoch randomly drawn from 500 ms to 2000 ms, a second stimulus epoch of 100 ms and a decision epoch of 100 ms. In each stimulus epoch, a signal drawn uniformly from $\{10, 11, ..., 34\}$ is presented to the network. The output of the networks during the decision epoch should compute the normalized differences in values of the two signals.
- The **contextual decision-making** task (CDM) consists of a fixation epoch of 100 ms, a context epoch of 350 ms, a stimulus epoch of 800 ms, a second context epoch of 500 ms and a decision epoch of 20 ms. There are four inputs into the networks: 2 cues and 2 noisy signals drawn uniformly from $\{\pm 0.4, \pm 0.2, \pm 0.1\}$. For a given trial, one of the cues is set at 0.1, while the other is set to 0. The networks are expected to output the sign of one of signals depending on which cue is non-zero.
- The **multi-sensory decision-making** task (MDM) has the same task structure and network inputs as CDM, except the 2 noisy signals have the same sign. Similarly, the networks are expected to output the signal of either signal during the decision epoch.
- The **delayed match-to-sample** task (DMS) consists of a fixation epoch of 100 ms, a stimulus epoch of 500 ms, a delay epoch randomly drawn from 500 ms to 3000 ms, a second stimulus epoch of 500 ms and a decision epoch of 1000 ms. In both stimulus epochs, one of two possible signals is presented, and the output of the networks during the decision epoch should indicate whether the signals presented in both stimulus epochs are the same.

Table S1: Number of trainable parameters in the recurrent weights of different network architectures across three different sizes. To match parameter counts, we vary the kernel size of CordsNets, weight matrix rank of low-rank RNNs and weight matrix sparsity of sparsely-connected RNNs.

Neurons	CordsNet	Low-Rank RNN	Sparse RNN
125	400 (kernel size 4)	500 (rank 2)	500 (sparsity 0.032)
216	900 (kernel size 5)	864 (rank 2)	864 (sparsity 0.0185)
512	2304 (kernel size 6)	2048 (rank 2)	2048 (sparsity 0.008)

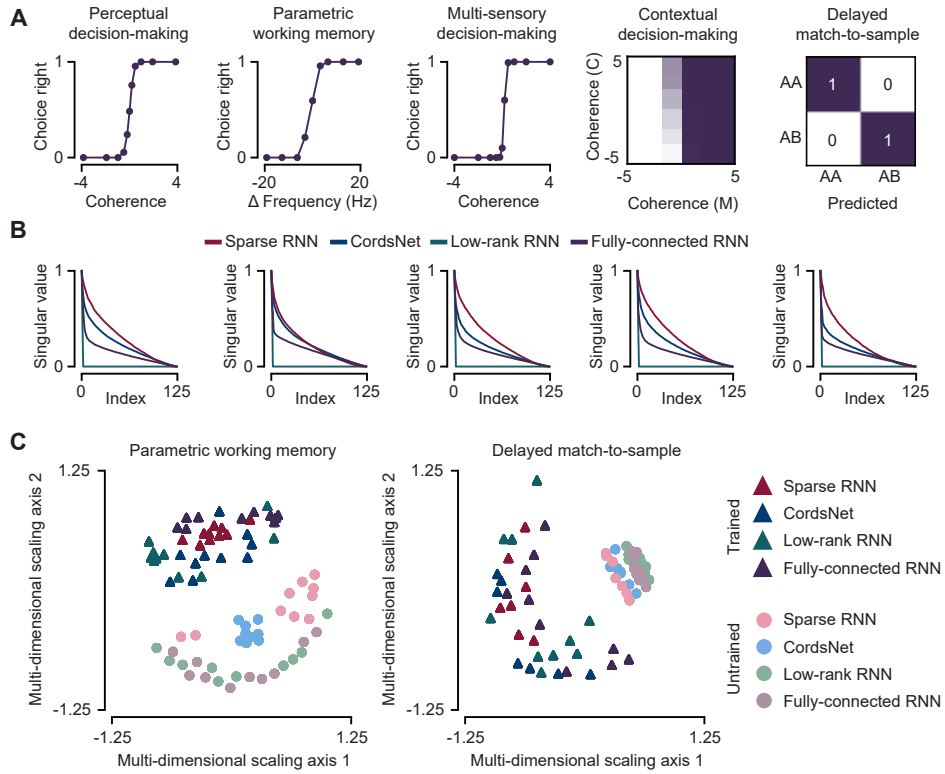

Figure S2: **A.** Examples of psychometric curves of CordsNets trained to solve five cognitive tasks. **B.** Normalized singular value distribution for different architectures on each respective task. **C.** Multi-dimensional scaling plots of distance matrices obtained after aligning trajectories using Procrustes analysis for the parametric working memory task (left) and the delayed match-to-sample task (right).

For each architecture-task pair, we trained 10 networks across three learning rates $(10^{-2}, 10^{-3}, 10^{-4})$, three activation functions (ReLU, tanh and softplus) and three network sizes (Table S1). We successfully trained all networks on all tasks. Figure S2A shows examples of psychometric curves for CordsNet on all five tasks. We also computed the singular values of the recurrent weights for all architectures (Figure S2B), and consistently found that CordsNets have singular value distributions that are the most similar to fully-connected RNNs across all hyperparameters.

More importantly, we want to compare the neural trajectories of all architectures and quantify how close they are to the trajectories of fully-connected RNNs. For each task and across every hyperparameter setting, there are:

$$(2 \text{ trained/untrained}) \times (4 \text{ architectures}) \times (10 \text{ random initializations}) = 80 \text{ networks} \qquad (4)$$

to be considered. Between each pair of networks, we align their neural trajectories using Procrustes alignment and compute the resultant Procrustes distance, which ultimately gives us a 80×80 distance matrix for one particular task and hyperparameter setting. This matrix can be visualized using multi-dimensional scaling [18, 19] (Figure S2C). We observe that networks of all architectures have similar aligned trajectories when they are either all trained or all untrained. This suggests

Table S2: Mean distance compared to fully-connected RNN of each architecture-task pair.

Task	Sparse RNN	CordsNet	Low-Rank RNN
PDM	**0.62** ($\pm$0.02)	0.64 ($\pm$0.03)	0.77 ($\pm$0.02)
PWM	0.84 ($\pm$0.02)	**0.75** ($\pm$0.02)	0.85 ($\pm$0.01)
MDM	0.75 ($\pm$0.02)	**0.73** ($\pm$0.03)	0.76 ($\pm$0.02)
CDM	**0.69** ($\pm$0.01)	0.74 ($\pm$0.03)	0.66 ($\pm$0.01)
DMS	0.94 ($\pm$0.04)	**0.86** ($\pm$0.05)	0.92 ($\pm$0.07)

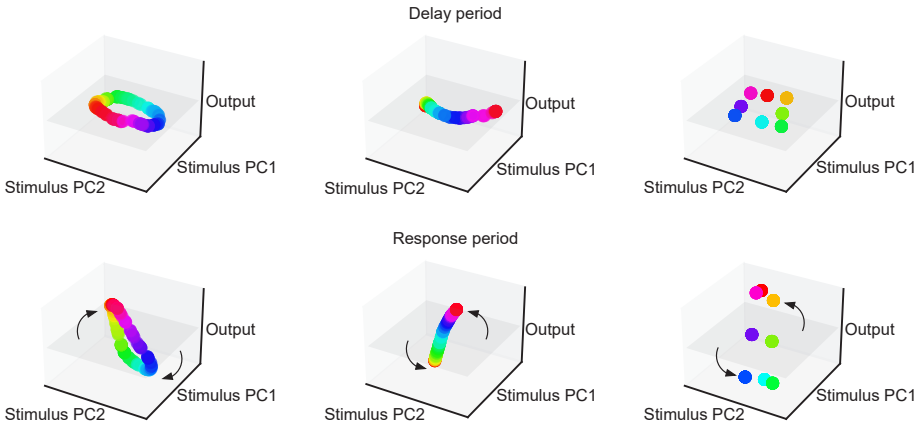

Figure S3: Evolution of neural activity across time in CordsNets trained to perform a memory-pro delayed-response task [20]. During the delay epoch, neural activity converges to different fixed/slow points depending on the stimulus that was presented in the previous epoch. Depending on the type of stimulus presented, the trained networks may exhibit ring (left), line (middle) or point attractors (right) during the delay epoch. In the response epoch, neural activity rotates to the output axis and approximately preserves the same geometry [20].

that the all network architectures are solving the tasks in similar ways. Finally, we compute the mean distances of each network architecture to fully-connected RNNs (Table S2) averaged across all tasks and hyperparameters, and find that on average CordsNets have the shortest mean distance to fully-connected RNNs. We conclude that the convolutional recurrent structure of CordsNets does not significantly restrict dynamical expressitivity.

B.4 Attractor formation

A key dynamical feature of RNNs is their ability to retain information about past stimuli over time through attractor states, enabled by carefully tuned recurrent weights. We want to check if the convolutional structure of CordsNets would prevent these attractors from forming. To do this, we train CordsNets on a memory-pro delayed-response task [20]. The task consists epochs of random duration, starting with a fixation epoch lasting between 300 ms to 700 ms, a stimulus epoch lasting between 200 ms to 1600 ms, a delay epoch lasting between 200 ms to 1600 ms, and a response epoch lasting between 300 ms to 700 ms. The random epoch durations have been known to promote attractor formation. During the stimulus epoch, the network input can represent a circular variable, a continuous linear variable, or a variable with 8 discrete states. The network output in the response epoch should match the initial input. For each type of input, we successfully trained CordsNets to generate ring, line, and point attractors, respectively, during the delay epoch (Figure S3, top). During the response epoch, neural activity shifts out of the output null space, creating a rotational effect that approximately preserves the structure of the attractors. These results confirm that CordsNets are able to manifest attractors despite their restricted weight structures.

C Training for image classification

C.1 Model architecture

The architectures of the CordsNets trained to perform image classification are described in Table S3. We generally follow the layer structure of ResNet-18 [21], but with two key modifications: we omit normalization and replace max pooling with average pooling, as averaging is a linear operation and thus more compatible with the dynamics of RNNs. Another important design choice that we made is to keep residual connections in our models. The brain performs cognition in a highly distributed manner, leveraging a multitude of interconnected regions that work in concert to process information, solve problems, and generate behaviors. Residual connections avoid the simplicity of a single-path structure, and allow for more sophisticated pathways for information to travel through the network.

Table S3: CordsNet architectures for image classification. In each block, the top convolution represents a feedforward transformation, while the bottom convolution represents the recurrent weights in the recurrent dynamical system.

Output size	CordsNet-R2	CordsNet-R4	CordsNet-R6	CordsNet-R8
112×112	3x3 average pool, stride 2			
56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
28×28		$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
14×14			$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
7×7				$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
1×1	average pool, linear, softmax			

C.2 Training details

For all datasets used in our study, we selected a set of image augmentation techniques that are widely recognized as universally beneficial [22]. Additional details can be found in Table S4

- **MNIST.** `RandomAffine` (5 degrees random rotation, 5% translation, 0.05 scaling factor and 5 degrees shear angle), `ColorJitter` (10% brightness, 10% contrast), `ElasticTransform` (scaling factor 20, smoothness 5)
- **Fashion-MNIST.** `RandomHorizontalFlip`, `RandomAffine` (5 degrees random rotation, 5% translation, 0.05 scaling factor and 5 degrees shear angle), `ColorJitter` (10% brightness, 10% contrast), `CutMix` (50%), `MixUp` (50%)
- **CIFAR-10/CIFAR-100.** `RandomHorizontalFlip`, `RandomAugment`, `CutMix`, `MixUp`
- **ImageNet.** `RandomHorizontalFlip`, `RandomAugment`

Table S4: Training specifics for each step of our proposed method for an efficient initialization of CordsNet. In step 3, the PReLU parameter a is varied from 0.95 to 0 in 20 steps of 0.05. For Imagenet, 0.1 epochs corresponds to 16000 iterations.

Dataset	Epochs	Optimizer	Learning rate	Batch size
Step 1 – Feedforward CNN				
ImageNet	3×30	SGD	`1e-1`, `1e-2`, `1e-3`	256
Others	3×100	SGD	`1e-1`, `1e-2`, `1e-3`	256
Step 2 – Linear RNN dynamical system				
ImageNet	30	AdamW	`1e-5`	32
Others	100	AdamW	`1e-5`	32
Step 3 – Parametric annealing				
ImageNet	20×0.1	AdamW	`1e-5`	8
Others	20×1	AdamW	`1e-5`	8
Step 4 – Fine tune				
ImageNet	20	AdamW	`1e-5`	8
Others	20	AdamW	`1e-5`	8

Table S5: Time required (in hours) to complete each step of our proposed initialization method, as benchmarked on a server with 2x RTX 4090 GPUs. The time required to train CordsNet for one full epoch is shown as a control, and the equivalent number of epochs (by time taken) using our method is computed in the final column.

Model	Dataset	Step 1	Step 2	Control	Epochs needed
R2	MNIST	5.99	22.82	1.46	39.70
	F-MNIST	4.05	20.65	1.47	36.77
	CIFAR-10	4.20	19.03	1.25	38.53
	CIFAR-100	4.06	19.86	1.24	39.25
	ImageNet	26.14	145.9	31.87	8.40
R4	MNIST	6.06	38.05	2.86	35.40
	F-MNIST	4.29	36.75	2.88	34.21
	CIFAR-10	4.22	31.85	2.40	35.02
	CIFAR-100	4.26	32.34	2.34	35.67
	ImageNet	27.44	245.2	62.27	7.38
R6	MNIST	6.06	50.99	4.31	33.24
	F-MNIST	4.50	49.95	4.26	32.77
	CIFAR-10	4.39	43.22	3.60	33.23
	CIFAR-100	4.44	43.45	3.55	33.49
	ImageNet	28.79	334.2	90.85	7.00
R8	MNIST	6.10	67.44	5.65	33.00
	F-MNIST	4.84	65.76	5.63	32.85
	CIFAR-10	4.82	56.53	4.73	32.97
	CIFAR-100	4.70	57.50	4.71	33.19
	ImageNet	31.20	439.2	121.04	6.89

C.3 Time benchmark

We train every model-dataset combination with a variable number of GPUs depending on memory requirements, as shown.

- **CordsNet-R2.** 8x RTX 4090 (ImageNet), 2x RTX 4090 (others)
- **CordsNet-R4.** 4x H100 80GB (ImageNet), 4x RTX 4090 (others)
- **CordsNet-R6.** 4x H100 80GB (ImageNet), 4x RTX 4090 (others)
- **CordsNet-R8.** 8x H100 80GB (ImageNet), 8x RTX 4090 (others)

One of the control experiments that we have proposed is to train a separate model in the same amount of time taken by our proposed method. In order to obtain an accurate estimate, we run every step of every model on a single server with 2x RTX 4090 GPUs. For each step, we simulate the training for approximately 5 minutes and extrapolated the time required for the entire step to be complete. We then computed the number of epochs needed to be simulated by our control experiment by:

$$\text{Control epochs} = \frac{\text{Time taken for step 1} + \text{Time taken for step 2}}{\text{Time taken for 1 control epoch}} + \text{Epochs in step 3} \tag{5}$$

This is because the parametric annealing step in step 3 involves training with the full loss function, which is essentially equivalent to 1 control epoch. The benchmark times are reported in Table S5. The duration of each epoch is influenced by the specific image augmentation techniques we have chosen to employ. The time taken for the MNIST dataset is particularly high due to the `ElasticTransform` augmentation, which is notably time-consuming. Training on the MNIST datasets takes longer than on the CIFAR datasets because of the larger MNIST training sets. For simplicity and also to account for variability, we choose to keep the number of control epochs at 10 for ImageNet, and 40 for the other datasets.

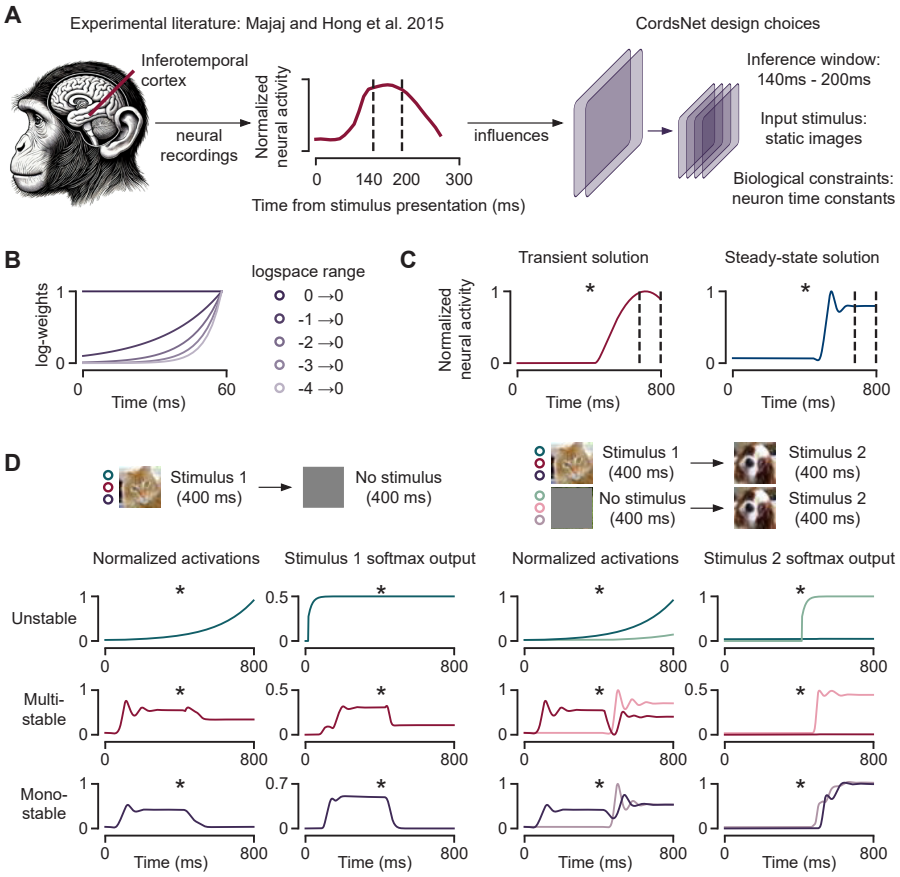

Figure S4: **A.** Building artificial neural network models of the biological visual system based on experimental literature [23]. The inference window of our loss function is derived from neural recordings. **B.** Logarithmic scale applied to the cross-entropy loss terms at different time steps. **C.** Trained networks can either classify images in a transient state where network activity is changing within the time window (left) or a stationary state where activity is unchanged (right). **D.** Without the spontaneously penalty term in the loss function, networks can exhibit three classes of solutions: an unstable solution where network activity blows up after the inference window (top, green), a multi-stable solution where the network remains stable after the first inference window but fails to classify subsequent images (middle, red), and a mono-stable solution that correctly classifies any number of inputs presented sequentially across time (bottom, purple).

D Loss function ablation study

The loss function we ultimately selected for training our networks on image classification is closely related to the standard template of a cross-entropy loss term with some optional regularization terms. We have a log-weighted sum of cross-entropy losses over 30 time steps and a spontaneous penalty term (which is a form of regularization). We first reintroduce the loss function here:

$$\texttt{loss} = \overbrace{\texttt{logspace(-3,0,steps=30)}}^{\text{log-weighting (Figure S4B)}} \texttt{ * CEloss(output[}\overbrace{\texttt{170:200}}^{\text{Inference window (Figure S4A)}}\texttt{],labels)} + \underbrace{\texttt{1e-3 * MSEloss(activity[290:300],spontaneous)}}_{\text{Spontaneous penalty (Figure S4C)}} \tag{6}$$

As stated in the main text, the network is simulated for 100 time steps (interpreted as 2 ms per time step) without any input to allow the network to arrive at a steady state spontaneous activity level (`spontaneous`). The inference time window of `[170:200]` corresponds to 140 ms to 200 ms after stimulus presentation. This time window was selected based on experimental recordings showing

Table S6: Ablation studies for the `logscale` range used to weigh the cross-entropy loss terms across time (top) and the coefficient of the spontaneous penalty term (bottom).

`logscale` range		
Range	Steady-state solution	Transient solution
$0 \rightarrow 0$	8	2
$-1 \rightarrow 0$	10	0
$-2 \rightarrow 0$	10	0
$-3 \rightarrow 0$	10	0
$-4 \rightarrow 0$	10	0
Spontaneous penalty coefficient		
Coefficient	Mono-stable solution	Other solutions
0	7	13
10^{-5}	16	4
10^{-4}	20	0
10^{-3}	20	0
10^{-2}	20	0
10^{-1}	20	0

heightened neural activity in the inferotemporal cortex of macaque monkeys following stimulus presentation (Figure S4A). Just like in conventional supervised learning settings, we calculate the cross-entropy loss across all 30 time steps within the selected window and sum them after weighing them with a logarithmic scale across time (Figure S4B). The purpose of this term is to ensure that we do not get solutions where the accuracy peaks in the middle of the inference window and drops off towards the end of the window, which we refer to as a transient solution. Instead, we want a network where neural activity is stable throughout, which we refer to as a steady-state solution (Figure S4C). To verify this, we train 10 CordsNet-R4s on CIFAR-10 across various `logscale` ranges. We find that varying the range of the `logscale` does not significantly impact the trained networks, as long as it is used (Table S6).

A fundamental characteristic of the brain is that it runs continuously, unlike artificial networks that reset (or shut down) after each inference. Therefore, after our network correctly classifies an image, we want it to accurately classify subsequent images, starting from the steady-state activity produced by the previous image. Intuitively, we reason that this property can only be attained by a mono-stable network, where there are no other fixed points in the vicinity of the activity space around its spontaneous activity level. The main property of a mono-stable network is that it returns to the same spontaneous activity level (before stimulus onset) after the presented image has been removed. This motivates the spontaneous penalty term in our loss function. Without this term, we find that trained networks can either be unstable (Figure S4D, green) where network activity blows up after the first inference window, multi-stable (Figure S4D, red) where the network remains stable after first inference but does not return to the original spontaneous activity level, or mono-stable ((Figure S4D, purple) which is the desired property. We train 20 CordsNet-R4s on CIFAR-10 across 6 different spontaneous penalty coefficient values. We find that we need a sufficiently large coefficient (Table S6) to prevent unwanted solutions (unstable and multi-stable) from emerging in our trained networks.

E Fitting to neural data

The neural similarity metric computed in the main text is simply the original Brain-Score [24] extended to fit across all time steps rather than time averaged quantities. Let N_t be the number of time steps, N_i be the number of images, N_1 be the number of neurons in CordsNet and N_2 be the number of neurons in the experimental recording. In each trial, we use a 90/10 train-test split (for a total of 10 splits), such that for each training split, we fit a $[N_t N_i,\, N_1]$ matrix of CordsNet neural activity with a $[N_t N_i,\, N_2]$ matrix of experimental data [23]. For each test split, we compute the Pearson correlation coefficients of all N_2 neurons, and select the median value. We then computed the mean coefficient across all 10 splits. We do not apply any noise ceiling correction. We then repeat

the fitting process after shuffling the neural data in the time axis. The entire process of fitting shuffle and unshuffled data is repeated over 20 trials. Finally, we perform a paired t-test across the 20 data points to look for any statistically significant differences between the scores when fitting on shuffled and unshuffled data.

F Code availability

Code for training and analyzing CordsNets, along with selected trained checkpoints, can be found at:

`https://github.com/wmws2/cordsnet`

References

[1] Hodgkin, A. L. & Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology* **117**, 500–544 (1952).

[2] Knight, B. W. Dynamics of encoding in a population of neurons. *The Journal of General Physiology* **59**, 734–766 (1972).

[3] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* **79**, 2554–2558 (1982).

[4] Battista, A. Low-dimensional continuous attractors in recurrent neural networks: from statistical physics to computational neuroscience. *Université Paris sciences et lettres* PhD thesis (2020).

[5] Zhang, K. Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory. *Journal of Neuroscience* **16**, 2112–2126 (1996).

[6] Seung, H. S. How the brain keeps the eyes still. *Proceedings of the National Academy of Sciences* **93**, 13339–13344 (1996).

[7] Sompolinsky, H., Crisanti, A. & Sommers, H. J. Chaos in random neural networks. *Physical Review Letters* **61**, 259–262 (1988).

[8] van Vreeswijk, C. & Sompolinsky, H. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science* **274**, 1724–1726 (1996).

[9] Battista, A. & Monasson, R. Capacity-resolution trade-off in the optimal learning of multiple low-dimensional manifolds by attractor neural networks. *Physical Review Letters* **124**, 048302 (2020).

[10] Mante, V., Sussillo, D., Shenoy, K. V. & Newsome, W. T. Context-dependent computation by recurrent dynamics in prefrontal cortex. *Nature* **503**, 78–84 (2013).

[11] Bai, S., Kolter, J. Z. & Koltun, V. Deep equilibrium models. *Advances in neural information processing systems* **32** (2019).

[12] 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. *Nature Neuroscience* **22**, 297–306 (2019).

[13] Mastrogiuseppe, F. & Ostojic, S. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron* **99**, 609–623 (2018).

[14] Gao, P. *et al.* A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv* (2017).

[15] Clark, D. G., Abbott, L. & Litwin-Kumar, A. Dimension of activity in random neural networks. *Physical Review Letters* **131**, 118401 (2023).

[16] Dubreuil, A., Valente, A., Beiran, M., Mastrogiuseppe, F. & Ostojic, S. The role of population structure in computations through neural dynamics. *Nature neuroscience* **25**, 783–794 (2022).

[17] Song, H. F., Yang, G. R. & Wang, X.-J. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLoS computational biology* **12**, e1004792 (2016).

[18] Williams, A. H., Kunz, E., Kornblith, S. & Linderman, S. Generalized shape metrics on neural representations. *Advances in Neural Information Processing Systems* **34**, 4738–4750 (2021).

[19] Ostrow, M., Eisen, A., Kozachkov, L. & Fiete, I. Beyond geometry: Comparing the temporal structure of computation in neural circuits with dynamical similarity analysis. *Advances in Neural Information Processing Systems* **36** (2024).

[20] Driscoll, L. N., Shenoy, K. & Sussillo, D. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience* 1–15 (2024).

[21] He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).

[22] Cubuk, E. D., Zoph, B., Shlens, J. & Le, Q. V. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719* **2**, 7 (2019).

[23] Majaj, N. J., Hong, H., Solomon, E. A. & DiCarlo, J. J. Simple learned weighted sums of inferior temporal neuronal firing rates accurately predict human core object recognition performance. *Journal of Neuroscience* **35**, 13402–13418 (2015).

[24] Schrimpf, M. *et al.* Brain-score: Which artificial neural network for object recognition is most brain-like? *BioRxiv* (2018).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer:[Yes]

   Justification: We have made 6 claims. The claim on dynamical expressitivity analysis can be found in Section 2, the claim on new training algorithm can be found in section 3, the autonomous nature of our model can be observed in Figure 3, the analytical toolkit is described in Section 4, the image-computable models can be found in Figure 5, and the neural data analysis can be found in Figure 6.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have included a dedicated part in our discussion to discuss limitations, where we reference other relevant parts of the paper where we presented the limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All training details required to reproduce our results can be found in Appendices B and C.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes, the link to our code is provided in the supplementary.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: For all experiments, we have provided details in both the main text and the supplementary.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We have done a paired t-test for our claims in Figure 6.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We have provided hardware specifications and time benchmarks in Appendix C.

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have properly cited all sources of experimental data that we have used.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]