JEDI: JOINTLY EMBEDDED INFERENCE OF NEURAL DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Animal brains flexibly and efficiently achieve many behaviors with a single neural network. A core goal in modern neuroscience is to map the mechanisms of the brain's flexibility onto the dynamics underlying neural populations. However, identifying task-specific dynamical rules from limited, noisy, and high-dimensional neural recordings remains a major challenge, as experimental data often provide only partial access to brain states and dynamical mechanisms. While recurrent neural networks (RNNs) directly constrained by neural data have been effective in inferring underlying dynamical mechanisms, they are typically limited to single-task domains and struggle to generalize across behavioral conditions. Here, we introduce JEDI, a hierarchical model that captures neural dynamics across tasks and contexts by learning a shared embedding space over RNN weights. This model recapitulates individual samples of neural dynamics while sharing statistical strength across trials and conditions, uncovering shared structure across conditions in a single, unified model. Using simulated RNN datasets, we demonstrate JEDI accurately learns robust, generalizable, condition-specific embeddings, and recovers the fixed point structure of a task-trained network. Finally, we apply JEDI to motor cortex recordings during monkey reaching to extract mechanistic insight into the neural dynamics of motor control. Our work shows that joint learning of contextual embeddings and recurrent weights provides scalable and generalizable inference of brain dynamics from neural recordings.

1 INTRODUCTION

Brains are complex and nonlinear dynamical systems composed of a network of highly specialized neural circuits. The neural computations that govern complex behaviors (e.g., playing guitar or writing machine learning papers) ultimately arise from the time-varying interactions of neural populations distributed across the brain (Duncker & Sahani, 2021). With modern advances in recording technologies, experimentalists can record the activity of ever-larger populations of neurons, providing an unprecedented window into neural computation. A critical question then arises: How can we obtain mechanistic insights into the neural dynamics underlying these behaviorally-relevant neural computations?

The structure of neural connectivity is foundational to systems neuroscience where interaction weights between neurons, together with neuronal response properties and external inputs, shape network dynamics and computational capabilities. Analyzing connectivity is therefore central to understanding how circuit structure enables computation and constrains the dynamical regimes a network can support (Liu et al., 2024; Schuessler et al., 2024; Ostojic & Fusi, 2024; Mastrogiuseppe & Ostojic, 2018). Static descriptions of population activity alone do not specify these dynamics. For example, principal-component projections provide coordinates for observed population states but do not, by themselves, define the evolution law governing transitions between them. Without an accompanying dynamical model, these descriptions do not directly specify fixed points, local stability, or Lyapunov exponents. Although latent dynamical models can characterize such properties without recovering synaptic connectivity, relating them to circuit mechanisms requires additional structural constraints or evidence. Inferring connectivity thus offers a route to testing how neuronal

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

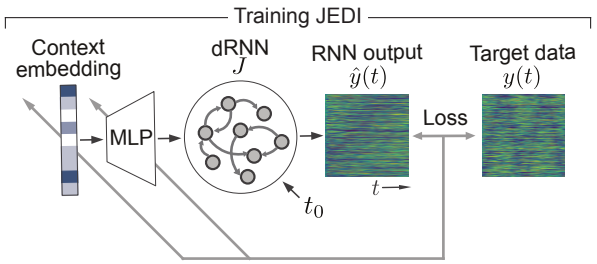

Figure 1: JEDI leverages a hypernetwork-based framework to flexibly generate RNN weights based on contextual inputs, all learned directly from a loss computed against the RNN output and time series neural recordings.

interactions generate population dynamics, particularly where direct measurement of synaptic weights is unavailable.

Over the past couple of decades, recurrent neural networks (RNNs)—which incorporate many canonical features of neural circuits such as recurrence and nonlinear interactions—have emerged as a powerful model for studying neural population activity (Sussillo & Abbott, 2009; Hess et al., 2023; Dinc et al., 2023; Valente et al., 2022; Perich et al., 2021; Durstewitz et al., 2023). RNNs trained to perform neuroscience-inspired tasks can be compared to animals performing similar tasks, allowing for indirect exploration of mechanisms in the biological brain. However, a growing class of data-constrained RNNs (dRNNs) trained to recapitulate neural recordings provides a complementary approach to make direct inference of brain computations.

Most existing dRNNs typically assume a fixed set of weights corresponding to one dynamical system for the observed data (e.g., each behavior). In contrast, biological networks flexibly operate in different dynamical regimes even within a single task (Huang et al., 2024; Turner et al., 2021). Thus, dRNN models should have similar dynamical flexibility to accurately capture neural computations necessary for complex and evolving behavioral demands. Fixed weight dRNN models lack the flexibility to account for changes in behavioral tasks or contexts. This limitation underscores the critical need for dRNN approaches that can robustly generalize across contexts by explicitly inferring distinct recurrent weights for each behavioral context.

To address this limitation, we introduce JEDI - Jointly Embedded Dynamics Inference from neural data. JEDI uses a hierarchical, hypernetwork-based framework to infer neural dynamics directly from time series of neural population recordings (Fig. 1). It leverages a shared embedding space to learn the parameters of arbitrary dRNNs that recapitulate the time series of recorded neural data (e.g. one trial of a behavioral task) using only a contextual input to the embedding. Learning in this shared model allows the model to capture common dynamics across datasets, tasks, or contexts, while providing flexibility for sample-specific dynamical features. Further, by jointly learning RNN weights and embeddings, we can infer interpretable link between weight structure and neural computations. Each fitted network operates at recorded-unit resolution, with recurrent states indexed by unit identity, allowing dynamical regimes analyses under specified input conditions without inverting a latent-to-neuron readout.

Our contributions can be summarized as follows:

- We propose a novel hypernetwork framework to model complex neural data from varying behavioral conditions and contexts in a single unified model.
- Using the learned task embeddings, we demonstrate accurate classification of dynamical regimes and generalization to unseen samples from learned tasks.
- We infer dynamical rules from the generated weights, through their spectra, Lyapunov exponents and fixed point structure, separately for every condition and from recordings alone.

2 METHODS

2.1 LOW-RANK RNNS

RNNs are universal approximators for dynamical systems, whose internal computations can be reverse engineered through the recurrent weights. We use rate-based recurrent network of N units, in which the variable r follows the dynamics:

$$\tau \frac{dr}{dt} = -r(t) + J\phi(r(t)) + \xi(t), \tag{1}$$

with state variable $\mathbf{r}(t) \in \mathbb{R}^N$, time-constant $\tau \in \mathbb{R}_{>0}$, recurrent weights $J \in \mathbb{R}^{N \times N}$, element-wise nonlinearity $\phi(x) = \textit{tanh}(x)$, gaussian white noise $\xi(t) \in R^N$ provided to each neuron. We integrate by forward Euler method.

The possible combinations of all J that can reproduce a given sample of neural activity is potentially large. To reduce this solution degeneracy (Huang et al., 2024), we constrain J with a low-rank penalty, compressing the recurrent weight matrix into a few dominant modes. Hence, we are interested in the case where the recurrent weight matrix has rank $R \leq N$, i.e., it can be written as $J = MN^{\mathsf{T}}$, with $M, N \in \mathbb{R}^{N \times R}$ (Mastrogiuseppe & Ostojic, 2018; Schuessler et al., 2020; Beiran et al., 2021; Dubreuil et al., 2022). In the case where the recurrent weights are unconstrained we will call the model "full rank". We note that, from population recordings alone, the exact structural connectivity matrix is largely unidentifiable. However, JEDI's primary objective is not to map exact synaptic wiring, but rather to infer the correct macroscopic computational mechanisms or the underlying dynamical rules governing the neural population.

2.2 CONTEXT-INFORMED HYPERNETWORK

We introduce conditioning through a hypernetwork (Ha et al., 2016)—a secondary network that dynamically generates the parameters of a primary network (here, a dRNN). The hypernetwork acts as a controller that receives a specific context signal (like a task instruction or a sensory goal) and uses it to dynamically reconfigure the primary network. Instead of the primary network having a single, static weights, it is conditioned to adapt its internal dynamics on the fly. Our implementation uses a feedforward layer that learns context embeddings to flexibly parameterize the downstream RNN responsible for reconstructing the dynamics of neural recordings.

$$J = f_h(c) \tag{2}$$

where f_h is a non-linear function parameterized by Θ_h and $c \in \mathbb{R}^E$ is the learnable context vector, E is the size of the context vector. Hence, equation (1) can be reformulated as :

$$\tau \frac{dr}{dt} = -r(t) + f_h(c)\phi(r(t)) + \xi(t), \tag{3}$$

From Equation (3) it is clear that context vector can significantly affect the overall dynamics for appropriate choices of $f_h(\cdot)$. Modeling choices for the context c can influence the degree of granularity of the hypernetwork. Indeed, c could be made to encode any variable, e.g. the identity of a specific trial, subject, or behavioral task. Rather than learn recurrent neural networks/weights for each trial of neural activity data separately, we generate the trial-specific weights from a single, learned hypernetwork that shares efficiently information across trials. When the recurrent weights are constrained to have low rank structure, the output of the hypernetwork are the matrices M, N. In our experiments, we initialize a new context c with zero entries for each trial. The experimenter chooses the granularity of c, and connectivity remains constant within each chosen unit. Unlike approaches that regularize embeddings toward a Gaussian distribution (Vermani et al., 2024), we impose no prior on c and learn each embedding solely through the reconstruction loss.

2.3 TRAINING OBJECTIVE

The goal is to estimate the recurrent weights to reproduce the recorded neural population activity through minimization of a loss function. We use back-propagation through time to minimize the

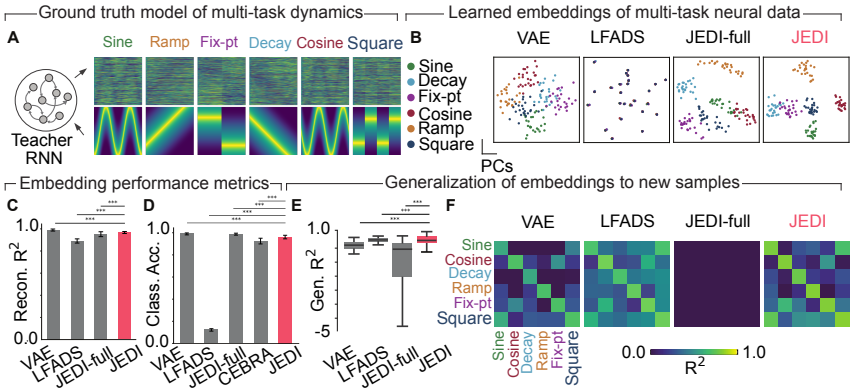

Figure 2: Quantifying the quality of the embeddings. A) Synthetic multi-task data generation setup. B) 2D PCA visualization of context embeddings. Each point corresponds to a trial, color-coded by input signal type. C) Training reconstruction R^2 across methods. D) Task-classification accuracy from the learned embeddings. CEBRA appears only here, it learns no decoder, so the reconstruction-based panels are undefined for it (see text). E) Generalization accuracy R^2 from the center of the learned embedding for each task F) Confusion matrix of generalization R^2 scores, expanding the results in Panel E. Rows represent training tasks, and columns indicate test tasks. Error bars represent ±1 standard deviation across trials. Significance: Wilcoxon's rank-sum test; *** indicates $p < 0.05$.

squared difference between the predicted and ground truth trajectories. The mean squared error (MSE) loss on each trial j can be expressed as:

$$\mathcal{L}(\Theta_h, c) = \frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(\phi(r_i^{(c)}(t)) - \phi(\hat{r}_i^{(c)}(t)) \right)^2 \tag{4}$$

where $r^{(c)}(t)$ represents the target trajectory in condition c, for neuron i and timestep t, and $\hat{r}_i^{(c)}(t)$ the corresponding predicted trajectory of the model. Similar to Von Oswald et al. (2019), we treat the context vector as a differentiable, deterministic parameter that can be optimized alongside Θ_h. At each learning step, the current context embedding $\mathbf{c}^{(j)}$ is updated along with the hypernetwork weights Θ_h to minimize the reconstruction loss. After training, the resulting context is stored and added to the collection $\{\mathbf{c}^{(j)}\}$. The nature of this collection will depend on the granularity chosen for the context embedding. In the following work, this collection represents trials from a set of tasks, dynamical contexts, or behavioral conditions, but for other applications could comprise experimental subjects or even brain regions.

Because JEDI jointly learns contextual embeddings and neural dynamics, we validate both. A synthetic multi-task benchmark with known ground truth tests whether the embeddings are robust and generalizable, and how architectural choices such as low-rank structure affect them; two further datasets test whether the inferred dynamics are recoverable by spectral and fixed-point analysis; and motor cortical recordings test the model on real data.

3 Empirical Results

3.1 JEDI learns generalizable embeddings across tasks

Our synthetic benchmark uses a teacher-student paradigm, in which a teacher network with known connectivity stands in for a neural system and JEDI, the student, is trained to reproduce its activity. The teacher is a chaotic RNN of $N = 200$ units with a frozen rank-5 connectivity matrix and gain $g = 1.8$, driven by six families of external input, oscillatory (sine, cosine, square), ramping, decaying and moving fixed-point, which we refer to as tasks; trials of the same task differ only in their random initial state (Fig. 2.A, full details in Appendix A.2). We trained JEDI to recapitulate the activity generated by the teacher RNN using two model configurations: one generating full rank RNN weights, and one constrained to produce RNN weights with rank r (set to 5 to match the ground truth data). We

configured the hypernetworks as 3-layer feedforward networks (MLPs) and compared performance of these JEDI models against the VAE and LFADS methods. Using coefficient of determination (R^2) as our metric, we found that all models achieve high reconstruction accuracy of the teacher RNN activity (Fig. 2.C). Notably, JEDI and VAE gave more consistently good fits (smaller variance across trials) than JEDI-full and LFADS.

A good contextual embedding should learn separable and interpretable structure across tasks. To evaluate if the per-trial embeddings (Fig. 2.B) learned the multi-task structure, we trained a Naive Bayes classifier on the learned representations and evaluated its accuracy on held-out trials. All models except LFADS achieved near-perfect classification accuracy (Fig. 2.D), indicating that the dynamical regimes induced by different input types are readily separable in latent space. LFADS IC encoder embeddings lack task structure on multi-task data because the model's controller network absorbs task-relevant temporal information through the CI encoder, leaving IC encoder uninformative about condition identity. CEBRA (Schneider et al., 2023) appears in Fig. 2.D alone, for two reasons. It is a contrastive method and learns no decoder, so it can neither reconstruct activity nor generate it from a fixed embedding; classification, which needs embeddings alone, is the one metric that applies to it. Its embeddings also arrive by a different route, via an amortized convolutional encoder applied to activity and mean-pooled over time, rather than the per-trial free parameters JEDI optimizes against reconstruction loss. Fig. 2.D therefore compares representations obtained under different constraints, and we treat it as a reference point rather than a like-for-like benchmark.

We evaluated across-task generalization by sampling the mean embedding for each task from training data and used this embedding to generate predictions on held-out test data across all tasks. A model that captures only task-specific identity will have high values along the diagonal (generalization within task), whereas a model that uncovers shared structure across tasks will show off-diagonal generalization. While some tasks were dynamically distinct and should have no generalization (e.g. ramp vs sine), others have common structure which could be leveraged by the embedding (e.g. the oscillatory tasks). JEDI outperformed other models in across-task generalization (Fig. 2.E). For example, embeddings derived from sine generalize to cosine, and embeddings from ramp transfer to decay(Fig. 2.F). Interestingly, JEDI-full performed substantially worse, indicating that structural low-rank constraints in the RNN weight outputs of JEDI enable learning generalizable representations in the embedding space. We next tested the robustness of the embeddings by adding gaussian noise of increasing variance to the learned values for each trial and decoding the resulting activity. Because JEDI's embeddings were not constrained to gaussian distributions like the VAEs and LFADS, we scaled the perturbation magnitude according to the specific variance of each model's task embeddings. JEDI-full and LFADS exhibit sharp performance declines (Fig. 3) in the presence of the perturbations. While VAE embeddings collapse at high noise level, JEDI outperforms all other models during low-to-moderate perturbations, indicating that the model learned robust, structured representations in the embedding.

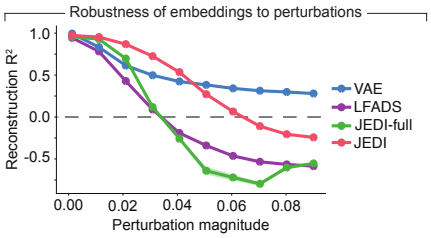

Figure 3: Impact of increasing embedding noise on model performance using reconstruction R^2 . The added noise was scaled according to the standard deviations of embeddings for each model

3.2 JEDI UNCOVERS EFFECTIVE SPECTRAL PROPERTIES OF NEURAL DYNAMICS

We next explore JEDI's ability to infer dynamical properties of neural data by reverse-engineering the RNN weights. We devised a variant of the previous experiment where the teacher RNN was driven by sinusoidal inputs of increasing frequencies (Fig. 4.A). After training on multiple samples of each frequency, we analyzed the spectrum of the learned JEDI RNN weights. The imaginary part of the eigenvalues indicates rotational velocity, while the real part of eigenvalues indicates stability in associated eigen directions.

We found that eigenvalues formed distinct clusters within the complex plane based on the input frequency. Crucially, the spectra expanded along the imaginary axis as input frequency increased, directly capturing the higher oscillatory content inherent in the driving signal. In contrast, the real components remained relatively invariant, suggesting that JEDI selectively adapts the rotational

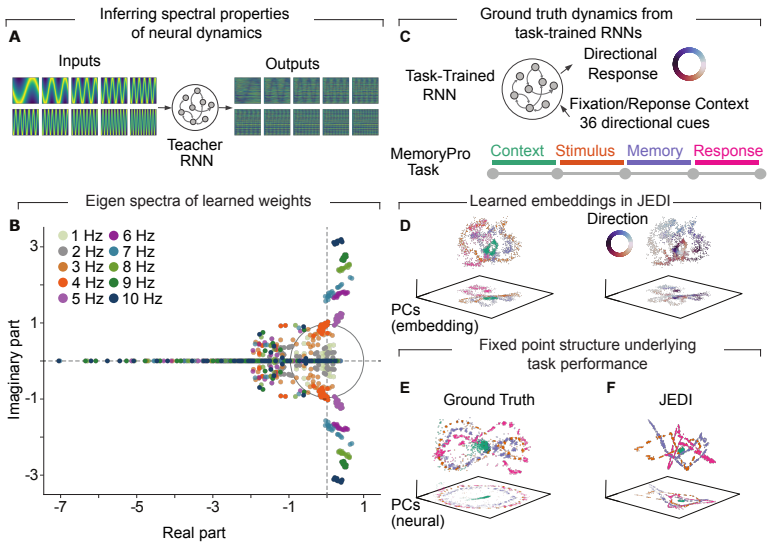

Figure 4: Investigating the properties of learned weights. A) We drove the chaotic Teacher RNN with sinusoidal inputs at different frequencies (1–10 Hz). B) The spectra of the weights inferred with JEDI exhibit a characteristic expansion along the imaginary axis as input frequency increases. C) We fit JEDI to a network trained to perform the MemoryPro task to identify fixed point structure in task-trained networks. This task has four contextual trial phases. D) Context embedding learned by JEDI colored by trial phase (left) and response direction (right). E-F) Fixed point structure for the four trial phases for the ground-truth task-trained network and inferred by JEDI.

dynamics of the network to match the data's spectral properties without compromising global stability. Notably, this precise spectral alignment was absent in models trained via standard full-rank initialization (see Appendix S5), which failed to recover the underlying dynamical structure. Furthermore, while we focus here on controlled sinusoidal inputs, JEDI demonstrates similar spectral consistency in more complex settings; for a comparison with the original spectra derived from multi-task learning paradigms (see Appendix S3). In Appendix E.3 we vary the teacher's rank over two orders of magnitude and find that JEDI recovers a generating rank.

3.3 JEDI CAN RECOVER GROUND TRUTH FIXED POINT STRUCTURE PRESENT IN THE NEURAL DYNAMICS

We explored the ability of JEDI to infer mechanisms of neural computation using the framework proposed by Yang et al. (2019), who trained RNNs to perform a range of neuroscience-like tasks. Following the protocol of Versteeg et al. (2025), we used the task-trained RNN to simulate trials for the MemoryPro task across four distinct periods: context, stimulus, memory, and response, see (Fig. 4.C). The fixed-point structure of the MemoryPro task was previously characterized by Driscoll et al. (2024), providing a clear ground truth target of dynamical mechanisms to infer by JEDI (see Appendix A.1 for comprehensive details).

Table 1: Procrustes disparity against the ground-truth fixed-point ring, by MemoryPro phase (lower is better). Baselines: Meta-Dynamical SSM (Vermani et al., 2024) and LtrRNN (Pellegrino et al., 2023).

Phase	JEDI	Meta-SSM	LtrRNN
context	0.906	**0.557**	0.569
Stimulus	**0.307**	0.729	0.736
Memory	**0.410**	0.770	0.939
Response	**0.147**	0.912	0.788

Next, to reverse-engineer the trial-specific computations, we identified slow-moving points within the model's hidden state space—commonly referred to as fixed points (Sussillo & Barak, 2013)—derived from JEDI's trained weights. These fixed points were projected into separate low-dimensional principal component (PC) spaces and visualized collectively for analysis. We observed that the fixed-point structure extracted by JEDI closely mirrored the shape of the original task-trained RNN

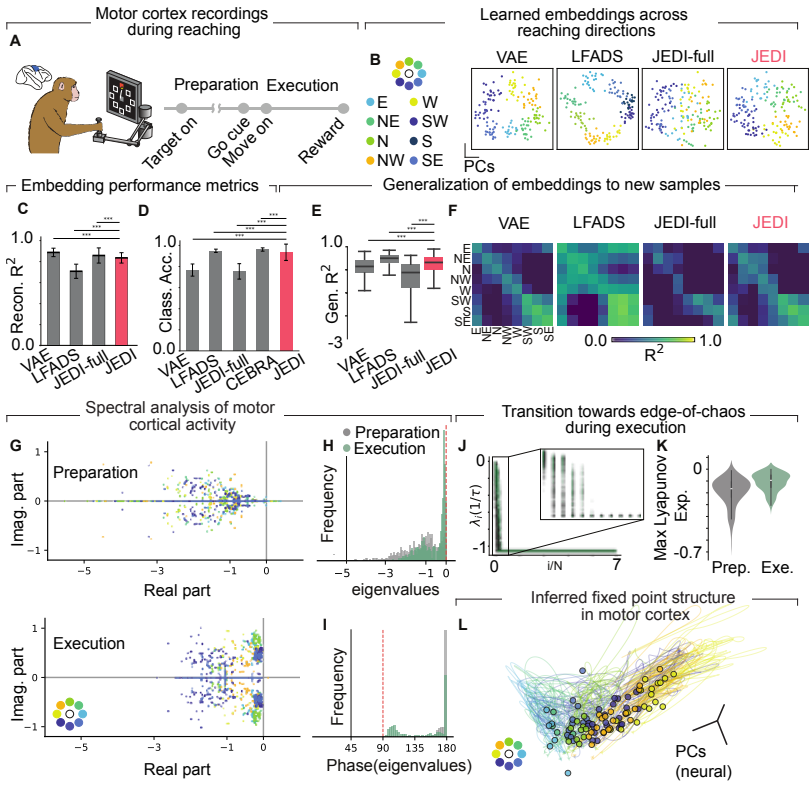

Figure 5: JEDI applied to monkey motor cortex data during reaching. A) JEDI applied to motor and premotor cortex during a center-out reaching task; schematic adapted from Gallego et al. (2020). B) 2D PCA of the learned embeddings, each point one trial, colored by reach direction. C) Reconstruction R^2 across trials. D) Reach-direction classification accuracy. E) Generalization R^2 from each embedding cluster centre. F) Confusion matrix for generalization across contexts, expanding Panel E. G) Jacobian spectra of learned JEDI weights during preparation (top) and movement (bottom), colored by the 8 reach directions. H) Spectra change quantified by the real part of each eigenvalue. I) Eigenvalue phase, a descriptive summary of the spectrum's angular spread. J) Lyapunov exponents across all learned weights. K) Distribution of maximum Lyapunov exponents across trials, preparation vs execution. L) Stable fixed points during execution, colored by reach direction, with single-trial neural trajectories. Error bars are ±1 standard deviation across trials; Significance: Wilcoxon's rank-sum, *** $p < 0.05$.

(Fig. 4.E&F). To evaluate geometric alignment, we mapped each model's fixed-point ring onto the task-trained network's ring via Procrustes analysis (accounting for rotation, scale, and translation). The resulting residual mismatch is reported as a disparity score (lower being better; 1). We then compared against closely related baselines (Vermani et al., 2024; Pellegrino et al., 2023).

After training on the large number of trials and contexts, JEDI yielded a low reconstruction loss and high R^2(0.94). We first examined whether the learned context embeddings preserved the functional organization of the task. We found that JEDI's embeddings capture the task's logical structure. Specifically, we found distinct, phase-specific rings organized according to stimulus direction (Fig. 4.D left), confirming that JEDI recovers the geometric relationships between input conditions (Fig. 4.D right).

The reconstruction was comparable across all three (R^2: 0.923, 0.935, 0.817), so these differences are not fit quality. JEDI recovered the ring markedly better during stimulus, memory and response, and worse during context. RNN states evolved from a single, central fixed point that cued the context towards geometrically related, target-specific fixed points, forming an approximate ring attractor. This is consistent with the fixed-point characteristics previously reported in Driscoll et al. (2024) for RNNs performing this task. This alignment with established structures confirms that JEDI's

embeddings and weights effectively capture meaningful features of computations underlying neural activity.

3.4 JEDI FLEXIBLY MODELS POPULATION RECORDINGS FROM MONKEY MOTOR CORTEX

We then applied JEDI to infer dynamical properties from real neural recordings. We fit JEDI models on motor cortical recordings from macaque monkeys performing a center-out reaching task (Perich et al., 2018). Neural population activity was recorded simultaneously from the primary motor and premotor cortex with two electrode arrays. The monkey was trained to perform an instructed delay task, which involved a movement preparation phase followed by a go cue instructing the monkey to reach to one of eight targets (movement phase) (see Fig. 5.A). After training, JEDI learned a clear ring-like structure in the embedding corresponding to the eight reach directions (Fig. 5.B), with comparable performance in fit quality, classification accuracy, and generalization as the other, embedding-specific methods (Fig. 5.C-E). This ring cannot be an artefact of supervision. Each embedding is free, initialized at zero and optimized only against that trial's reconstruction loss, and no term in Equation 4 sees direction, target or trial label, so any organization by direction was recovered from the activity itself. To explore this, we compared movement preparation (a covert process without explicit motor output) to movement execution (an overt process that moves the arm). These two processes have been shown to involve distinct neural computations (Elsayed et al., 2016), and we hypothesized that they should consequently have distinct dynamical properties which JEDI can uncover.

Spectral analysis of the trained recurrent matrices generated by JEDI (but not JEDI-full, see Appendix S4) revealed clear changes in dynamical structure across the two behavioral phases. During preparation, eigenvalues lay well to the left of $\mathrm{Re}(\lambda) = 1$, with prominent excursions along the real axis (Fig. 5.G top), consistent with the ramping dynamics that take the network from quiescence to readiness. During movement, the spectrum shifted toward the boundary, with groups emerging whose real parts approached $\mathrm{Re}(\lambda) = 1$ (Fig. 5.G bottom). The shift toward the boundary therefore indicates that movement engages higher-gain, longer-lived modes than preparation, while the claim that the system operates near criticality rests on the Lyapunov exponents below, which are computed from the state-dependent Jacobian along trajectories and need no eigenvalue-boundary convention. Reassuringly, these dynamics were identical across all trials, consistent with the assumption that the underlying dynamical system driving movement does not change for different reach directions.

We further explored whether execution-related dynamics represent a shift towards criticality (edge of stability). We analyzed the Lyapunov exponents (Vogt et al., 2022) of the inferred weights to quantify the long-horizon behavior of the system, between preparation and execution. Values above zero indicate a tendency towards chaos, values below zero indicate stability, and values near zero are consistent with criticality or edge-of-chaos. We found that the transition between preparation and movement corresponded to an increase in the maximum Lyapunov exponent towards zero (Fig. 5.J &K), placing execution close to the edge of chaos, a regime associated with efficient computation (Legenstein & Maass, 2007). This marginal stability reflects the motor system's trade-off between robustness and expressivity (Russo et al., 2018).

We lastly assessed the fixed point structure inferred by JEDI to test possible mechanisms governing the evolution of neural trajectories that underlie reaching movements in the motor cortex (Gallego et al., 2017). Using the procedures described in the task-trained RNN experiment, we identified stable fixed points in the neural population activity (Fig. 5.L). These fixed points were clustered by reach direction, consistent with different attractor zones for different reaching conditions. Intriguingly, we found that neural trajectories identified by PCA ended at these stable fixed points; this end point corresponds to the end of the monkey's reach.

The fixed points uncovered by JEDI, together with the spectra and Lyapunov exponents, posits that neural dynamics for reaching are governed by marginally-stable trajectories towards stable attractors. This mechanistic insight highlights the strength of JEDI for analyzing neural data.

4 RELATED WORK

RNNs are ubiquitous in computational neuroscience, explaining neural computation through underlying dynamical principles (Vyas et al., 2020). Prior work highlighted how contexts and task

variables shape computation (Yang et al., 2019; Driscoll et al., 2024; Costacurta et al., 2024; Williams et al., 2025), but typically uses fixed rather than learned contextual inputs and trains networks *de novo* on specific tasks. These works do not infer weight structure directly from recorded neural activity, our primary goal. Another common application of neural networks for neuroscience is latent state inference, including LFADS (Pandarinath et al., 2018) and XFADS (Dowling et al., 2024). CEBRA (Schneider et al., 2023) learns latent embeddings jointly from neural and behavioral data, paralleling our use of context to align across conditions. Unlike these methods, our method aims to infer interpretable weight structure to reproduce the full time series of neural population recordings.

Methods such as CURBD (Perich et al., 2021) and CORNN (Dinc et al., 2023) directly learn from time series neural data. Others model neural activity in the latent space with low-rank RNNs (Pals et al., 2024; Valente et al., 2022). Our method shares this low-rank assumption but extends it: we fit at full neural resolution and explicitly link weight structure to learned context embeddings. LtrRNN (Pellegrino et al., 2023) also shares structure across trials, combining shared connectivity factors with GP-smoothed trial coefficients that follow trial order, capturing gradual within-session change; JEDI instead learns unordered embeddings supporting comparison across tasks, conditions or subjects, which LtrRNN provides no space for. Meta-Dynamical SSM of Vermani et al. (2024) uses a hypernetwork to generate session-specific additive changes to shared latent dynamics. JEDI instead generates trial-specific low-rank connectivity directly in neuron-indexed space without a separate latent-to-neuron readout.

Other work combines learned contexts with neural data modeling: meta-learning frameworks (Cotler et al., 2023), hierarchical time-series models (Brenner et al., 2024), and Bayesian models of decision criteria (Vloeberghs et al., 2025) share our goal of extracting contextual structure (Kirchmeyer et al., 2022). Task-specific embedding layers also enable shared SSM decoders (Ryoo et al., 2025), though these lack JEDI's interpretability.

5 Discussion

Summary. We introduced JEDI, a hierarchical hypernetwork model that generates low-rank autonomous RNNs to fit neural data, testing whether contextual embeddings that parameterize recurrent weights yield interpretable, generalizable models of neural computation. JEDI recovers both the contextual relationships between experimental conditions and the dynamical mechanisms underlying them. We showed that low-rank hypernetworks simultaneously (i) produce embeddings that classify dynamical regimes and generalize to new data, and (ii) generate interaction matrices whose eigenvalue spectra match known dynamical signatures in synthetic systems and reveal new motifs in real recordings. Low-rank constraints act as an inductive bias improving robustness and generalization, while spectral analysis of the learned interactions yields mechanistic insight into the rules governing computation. Prior low-rank RNN work established that structured connectivity shapes computation; we extend this by linking context embeddings to weight organization through hypernetworks, bridging latent-variable and weight-space analyses.

Future work should incorporate biologically plausible connectivity priors where known *a priori*, extend the framework to incomplete or multimodal observations, and apply JEDI across larger datasets, tasks and species to reveal conserved organizational principles.

Limitations. Rank must be specified manually; nuclear norm regularization (Scarvelis & Solomon, 2024) may relax this. Long sequences remain prone to vanishing and exploding gradients, suggesting stabilized architectures (e.g. CORNN (Rusch & Mishra, 2020)), and partial observation may introduce mechanistic mismatch (Qian et al., 2024). Without explicit time-varying inputs JEDI cannot separate externally driven activity from intrinsic dynamics, and context is fixed over an experimenter-defined interval, so connectivity changes within it are neither detected nor represented; explicit inputs and change-point detection would address both. Finally, the hypernetwork emits parameters for a fixed neuron count, so one model cannot span sessions or subjects with differing unit counts. Session-specific read-in/read-out layers (Pandarinath et al., 2018) or unit-count-agnostic tokenization (Azabou et al., 2023) could accommodate this, though preserving neuron-indexed connectivity needs further design.

REFERENCES

Mehdi Azabou, Vinam Arora, Venkataramana Ganesh, Ximeng Mao, Santosh Nachimuthu, Michael Mendelson, Blake Richards, Matthew Perich, Guillaume Lajoie, and Eva Dyer. A unified, scalable framework for neural population decoding. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Manuel Beiran and Ashok Litwin-Kumar. Prediction of neural activity in connectome-constrained recurrent networks. *bioRxiv*, 2024.

Manuel Beiran, Alexis Dubreuil, Adrian Valente, Francesca Mastrogiuseppe, and Srdjan Ostojic. Shaping Dynamics With Multiple Populations in Low-Rank Recurrent Networks. *Neural Computation*, 33(6):1572–1615, 2021.

Manuel Brenner, Elias Weber, Georgia Koppe, and Daniel Durstewitz. Learning interpretable hierarchical dynamical systems models from time series data. *arXiv preprint arXiv:2410.04814*, 2024.

Julia Costacurta, Shaunak Bhandarkar, David Zoltowski, and Scott Linderman. Structured flexibility in recurrent neural networks via neuromodulation. *Advances in Neural Information Processing Systems*, 37:1954–1972, 2024.

Jordan Cotler, Kai Sheng Tai, Felipe Hernández, Blake Elias, and David Sussillo. Analyzing populations of neural networks via dynamical model embedding. *arXiv preprint arXiv:2302.14078*, 2023.

Fatih Dinc, Adam Shai, Mark Schnitzer, and Hidenori Tanaka. CORNN: Convex optimization of recurrent neural networks for rapid inference of neural dynamics. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Matthew Dowling, Yuan Zhao, and Memming Park. exponential family dynamical systems (xfads): Large-scale nonlinear gaussian state-space modeling. *Advances in Neural Information Processing Systems*, 37:13458–13488, 2024.

Laura N Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience*, 27(7):1349–1363, 2024.

Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. The role of population structure in computations through neural dynamics. *Nature Neuroscience*, 25(6): 783–794, 2022.

Lea Duncker and Maneesh Sahani. Dynamics on the manifold: Identifying computational dynamical activity from neural population recordings. *Current opinion in neurobiology*, 70:163–170, 2021.

Daniel Durstewitz, Georgia Koppe, and Max Ingo Thurm. Reconstructing computational system dynamics from neural data with recurrent neural networks. *Nature Reviews Neuroscience*, 24(11): 693–710, 2023.

Gamaleldin F Elsayed, Antonio H Lara, Matthew T Kaufman, Mark M Churchland, and John P Cunningham. Reorganization between preparatory and movement population responses in motor cortex. *Nature communications*, 7(1):13239, 2016.

Rainer Engelken and Fred Wolf. Lyapunov spectra of chaotic recurrent neural networks. *Physical Review Research*, 5(4):043044, 2023.

Juan A Gallego, Matthew G Perich, Lee E Miller, and Sara A Solla. Neural manifolds for the control of movement. *Neuron*, 94(5):978–984, 2017.

Juan A Gallego, Matthew G Perich, Raeed H Chowdhury, Sara A Solla, and Lee E Miller. Long-term stability of cortical population dynamics underlying consistent behavior. *Nature neuroscience*, 23 (2):260–270, 2020.

Matthew D Golub and David Sussillo. Fixedpointfinder: A tensorflow toolbox for identifying and characterizing fixed points in recurrent neural networks. *Journal of open source software*, 3(31): 1003, 2018.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Florian Hess, Zahra Monfared, Manuel Brenner, and Daniel Durstewitz. Generalized teacher forcing for learning chaotic dynamics. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, 2023.

Ann Huang, Satpreet H Singh, Flavio Martinelli, and Kanaka Rajan. Measuring and controlling solution degeneracy across task-trained recurrent neural networks. *arXiv preprint arXiv:2410.03972*, 2024.

Matthieu Kirchmeyer, Yuan Yin, Jérémie Donà, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to new physical systems via context-informed dynamics model. In *International Conference on Machine Learning*, pp. 11283–11301. PMLR, 2022.

Robert Legenstein and Wolfgang Maass. Edge of chaos and prediction of computational performance for neural circuit models. *Neural networks*, 20(3):323–334, 2007.

Yuhan Helena Liu, Aristide Baratin, Jonathan Cornford, Stefan Mihalas, Eric Shea-Brown, and Guillaume Lajoie. How connectivity structure shapes rich and lazy learning in neural circuits. *ICLR*, pp. arXiv–2310, 2024.

Francesca Mastrogiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623.e29, 2018.

Srdjan Ostojic and Stefano Fusi. Computational role of structure in neural activity and connectivity. *Trends in Cognitive Sciences*, 28(7):677–690, 2024.

Matthijs Pals, A Erdem Sağtekin, Felix Pei, Manuel Gloeckler, and Jakob H Macke. Inferring stochastic low-rank recurrent neural networks from neural data. *Advances in Neural Information Processing Systems*, 37:18225–18264, 2024.

Chethan Pandarinath, Daniel J. O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D. Stavisky, Jonathan C. Kao, Eric M. Trautmann, Matthew T. Kaufman, Stephen I. Ryu, Leigh R. Hochberg, Jaimie M. Henderson, Krishna V. Shenoy, L. F. Abbott, and David Sussillo. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, 2018.

Arthur Pellegrino, N Alex Cayco Gajic, and Angus Chadwick. Low tensor rank learning of neural dynamics. *Advances in Neural Information Processing Systems*, 36:11674–11702, 2023.

Matthew G Perich, Juan A Gallego, and Lee E Miller. A neural population mechanism for rapid learning. *Neuron*, 100(4):964–976, 2018.

Matthew G. Perich, Charlotte Arlt, Sofia Soares, Megan E. Young, Clayton P. Mosher, Juri Minxha, Eugene Carter, Ueli Rutishauser, Peter H. Rudebeck, Christopher D. Harvey, and Kanaka Rajan. Inferring brain-wide interactions using data-constrained recurrent neural network models. bioRxiv:2020.12.18.423348, 2021.

Matthew G Perich, Devika Narain, and Juan A Gallego. A neural manifold view of the brain. *Nature Neuroscience*, 28(8):1582–1597, 2025.

William Qian, Jacob A Zavatone-Veth, Benjamin S Ruben, and Cengiz Pehlevan. Partial observation can induce mechanistic mismatches in data-constrained models of neural dynamics. *Advances in Neural Information Processing Systems*, 37:67467–67510, 2024.

Kanaka Rajan, LF Abbott, and Haim Sompolinsky. Stimulus-dependent suppression of chaos in recurrent neural networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 82 (1):011903, 2010.

T Konstantin Rusch and Siddhartha Mishra. Coupled oscillatory recurrent neural network (cornn): An accurate and (gradient) stable architecture for learning long time dependencies. *arXiv preprint arXiv:2010.00951*, 2020.

Abigail A Russo, Sean R Bittner, Sean M Perkins, Jeffrey S Seely, Brian M London, Antonio H Lara, Andrew Miri, Najja J Marshall, Adam Kohn, Thomas M Jessell, et al. Motor cortex embeds muscle-like commands in an untangled population response. *Neuron*, 97(4):953–966, 2018.

Avery Hee-Woon Ryoo, Nanda H. Krishna, Ximeng Mao, Mehdi Azabou, Eva L. Dyer, Matthew G. Perich, and Guillaume Lajoie. Generalizable, real-time neural decoding with hybrid state-space models, 2025. URL https://arxiv.org/abs/2506.05320.

David Saad and Sara A Solla. Exact solution for on-line learning in multilayer neural networks. *Physical Review Letters*, 74(21):4337, 1995.

Christopher Scarvelis and Justin M Solomon. Nuclear norm regularization for deep learning. *Advances in Neural Information Processing Systems*, 37:116223–116253, 2024.

Steffen Schneider, Jin Hwa Lee, and Mackenzie Weygandt Mathis. Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, 617(7960):360–368, 2023.

Friedrich Schuessler, Alexis Dubreuil, Francesca Mastrogiuseppe, Srdjan Ostojic, and Omri Barak. Dynamics of random recurrent networks with correlated low-rank structure. *Physical Review Research*, 2(1):013111, 2020.

Friedrich Schuessler, Francesca Mastrogiuseppe, Srdjan Ostojic, and Omri Barak. Aligned and oblique dynamics in recurrent neural networks. *Elife*, 13:RP93060, 2024.

Hyunjune Sebastian Seung, Haim Sompolinsky, and Naftali Tishby. Statistical mechanics of learning from examples. *Physical review A*, 45(8):6056, 1992.

Haim Sompolinsky, Andrea Crisanti, and Hans-Jurgen Sommers. Chaos in random neural networks. *Physical review letters*, 61(3):259, 1988.

David Sussillo and L. F. Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63:544–557, 2009.

David Sussillo and Omri Barak. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural computation*, 25(3):626–649, 2013.

Elia Turner, Kabir V Dabholkar, and Omri Barak. Charting and navigating the space of solutions for recurrent neural networks. *Advances in Neural Information Processing Systems*, 34:25320–25333, 2021.

Adrian Valente, Jonathan W Pillow, and Srdjan Ostojic. Extracting computational mechanisms from neural data using low-rank rnns. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Ayesha Vermani, Josue Nassar, Hyungju Jeon, Matthew Dowling, and Il Memming Park. Meta-dynamical state space models for integrative neural data analysis. *arXiv preprint arXiv:2410.05454*, 2024.

Christopher Versteeg, Jonathan D McCart, Mitchell Ostrow, David M Zoltowski, Clayton B Washington, Laura Driscoll, Olivier Codol, Jonathan A Michaels, Scott W Linderman, David Sussillo, and Chethan Pandarinath. Computation-through-dynamics benchmark: Simulated datasets and quality metrics for dynamical models of neural activity. *bioRxiv*, 2025.

Robin Vloeberghs, Anne E Urai, Kobe Desender, and Scott W Linderman. A bayesian hierarchical model of trial-to-trial fluctuations in decision criterion. *PLOS Computational Biology*, 21(7): e1013291, 2025.

Ryan Vogt, Maximilian Puelma Touzel, Eli Shlizerman, and Guillaume Lajoie. On lyapunov exponents for rnns: Understanding information propagation using dynamical systems tools. *Frontiers in Applied Mathematics and Statistics*, 8:818799, 2022.

Johannes Von Oswald, Christian Henning, Benjamin F Grewe, and João Sacramento. Continual learning with hypernetworks. *arXiv preprint arXiv:1906.00695*, 2019.

Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual review of neuroscience*, 43(1):249–275, 2020.

Ezekiel Williams, Alexandre Payeur, Avery Hee-Woon Ryoo, Thomas Jiralerspong, Matthew G Perich, Luca Mazzucato, and Guillaume Lajoie. Expressivity of neural networks with random weights and learned biases. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=5xwx1Myosu.

Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

SUPPLEMENTARY MATERIAL: JEDI: JOINTLY EMBEDDED INFERENCE OF NEURAL DYNAMICS

Anonymous authors

Table of Contents

A Additional details on Datasets **2**
- A.1 Synthetic Memory Pro task . . . 2
- A.2 Synthetic multi-task teacher network . . . 2
- A.3 Motor Cortex Recordings . . . 2
- A.4 Multiple frequency Noisy Sine data . . . 3

B Model Architecture and Training details **3**
- B.1 VAE . . . 3
- B.2 LFADS . . . 3
- B.3 CEBRA . . . 4
- B.4 Meta-Dynamical State Space Model . . . 4
- B.5 LtrRNN . . . 5
- B.6 Our proposed model . . . 5

C Fixed point finding and visualization **5**
- C.1 Quantifying fixed-point geometric recovery . . . 6

D Details of Lyapunov Exponents calculation **6**

E Supplementary Experiments and Details **6**
- E.1 Hyperparameter Sweep . . . 6
- E.2 Effect of data rank on performance of Jedi . . . 7
- E.3 Does JEDI recover condition-specific rank? . . . 7
- E.4 Embedding and initialisation protocol . . . 8
- E.5 Effect of hypernetwork capacity and explicit regularization . . . 8
- E.6 Additional results on spectral analysis . . . 10
- E.7 Additional fixed points results . . . 11

F Compute Resources **11**

A Additional details on Datasets

A.1 Synthetic Memory Pro task

We followed the training procedure outlined in the Computation through Dynamics benchmark (Versteeg et al., 2025) and selected the MemoryPro task from the suite (Yang et al., 2019), as its fixed-point structure was previously characterized by Driscoll et al. (2024). In this task, the RNN learned to respond in the same direction as a stimulus after a memory period by minimizing the squared loss between its 3-dimensional output and the target using backpropagation through time. The input space consisted of 18 dimensions, including a 1-d fixation signal, 2-d stimulus vectors encoding the circular variable θ as $A\sin(\theta)$ and $A\cos(\theta)$, and 15-d rule inputs that remained active throughout the trial to indicate the task type. We used a tanh activation function for the RNN and treated each task period—context, stimulus, memory, and response—as an autonomous dynamical system with a distinct set of fixed points induced by the piecewise constant inputs.We structured the experimental trials by initiating a 75-timestep context period where only fixation and rule inputs were active, followed by the stimulus period and a subsequent memory period that shared identical inputs with the context phase. When the fixation input dropped to zero during the response period, the network generated its directional output, which we monitored across 36 distinct stimulus angles. To ensure a robust dataset, we generated 20 variations for each angle through random sampling, resulting in a final collection of 2,880 trials. We then fit our model to these activations by initializing 2,880 context embeddings and training them together to capture the underlying task dynamics and specific patterns of activity induced by the trial phases.

A.2 Synthetic multi-task teacher network

To evaluate the performance of the model, we studied a teacher-student paradigm (Saad & Solla, 1995; Seung et al., 1992; Beiran & Litwin-Kumar, 2024) where a teacher network is used as a proxy for a neural system with known recurrent connectivity. The student (our model) is trained to mimic the teacher. We generated synthetic data from a chaotic teacher RNN with pre-defined recurrent connectivity and input structure (Fig. 2.A). The RNN was composed of N=200 neurons, which evolved according to the dynamical update:

$$\tau \frac{dr}{dt} = -r(t) + gJ\phi(r(t)) + W_{ext}U(t), \tag{5}$$

where $r(t) \in \mathbb{R}^N$ is the RNN state at time t, ϕ is nonlinear activation function, τ is the neural time constant, and $U(t)$ is time varying external input. $J \in \mathbb{R}^{N\times N}$ is the recurrent connectivity matrix kept frozen across all trials and tasks. To impose low-rank (Perich et al., 2025) structure we draw factors $M, N \in \mathbb{R}^{N\times R}$ with i.i.d. $\mathcal{N}(0, 1/N)$ entries and set $J = MN^{\mathsf{T}}$, giving rank exactly $R = 5$; drawing all N^2 entries independently would instead give a full-rank matrix. The gain parameter g determines the strength of the recurrent connections, and thus whether ($g > 1$) or not ($g < 1$) the network produces spontaneous activity with non-trivial dynamics (Rajan et al., 2010; Sompolinsky et al., 1988). We set $g = 1.8$ to produce chaotic dynamics shaped by the external inputs. $W_{ext} \in R^{N\times N}$ is input matrix that maps input signals to the recurrent matrix.

To probe JEDI's ability to capture diverse dynamical regimes in a single model, we drove the teacher RNN with a range of external input signals $U(t)$ that shaped the temporal evolution of the RNN (Fig. 2.A). Inputs included oscillatory (sine, cosine, square), ramping, decaying, and moving fixed-point (step-like) patterns. These inputs were directly applied to a subset of neurons in the chaotic RNN (50% of the population), selected via an input weight matrix $W_{\text{ext}} \in \mathbb{R}^N$. We refer to the input types as tasks, and simulated the variability across trials by repeating each input from different random initial RNN states. We simulated each trial for 2 seconds following a 0.1 second burn-in period. The resulting activity serves as ground-truth trials of a known dynamical system, providing a testbed for assessing the generalization and interpretability of our model.

A.3 Motor Cortex Recordings

Two monkeys were trained to perform a two-dimensional center-out reaching task using a planar that controlled a cursor on a screen. In each trial, the monkey moved to a central start position, waited through a variable delay, and then reached toward one of eight randomly selected targets arranged

uniformly in a circle. A go cue signaled movement initiation, and successful trials required reaching the target within 1 second and holding for 0.5 seconds to receive a reward. We tested the applicability of the proposed approach on this data containing $c = 160$ distinct recorded trials (8 reach directions $\times$ 20 trials each) of $T = 150$ from $N = 117$ neurons. The hypernetwork used for these recordings is trained entirely from scratch on the monkey data and is not fine-tuned or warm-started from any model trained on synthetic data.

A.4 Multiple frequency Noisy Sine data

Similar to multi-task synthetic teacher task framework, we generated chaotic sinusoidal datasets by driving chaotic teacher RNN with 10 increasing frequencies across 20 unique initializations of initial states. This procedure produced a comprehensive dataset of 200 trials characterized by complex, non-linear dynamics. We subsequently utilized these trajectories to train and evaluate both JEDI and JEDI-full, assessing their capacity to capture and reconstruct high-dimensional chaotic representations.

B Model Architecture and Training details

The latent dimension for all the methods is set to 16. We selected the hyperparameters based on the grid search. Every method must reproduce the full $T \times N$ trajectory of a trial from a 16-dimensional representation of that trial, what differs between the methods is the decoder. The VAE expands its 16 numbers through an unconstrained feedforward mapping straight to the trajectory. JEDI's context embeddings instead generate the two $N \times R$ factors of a low-rank connectivity matrix, which is then integrated autonomously for T steps under Equation 3. The $2 \times N \times R$ weights are not a second bottleneck the model may store information in independently, since they are a deterministic function of the same 16-dimensional code. The information available to each decoder at test time is matched, and we chose VAEs as the reference precisely because compressing trajectories into such a code is their design purpose, which makes matching them a meaningful bar.

B.1 VAE

We used a two-layer Multi-Layer Perceptron(MLP) for both the encoder and decoder. The encoder transforms the flattened input of shape $(T \times features)$ into a latent vector of dimension 16 though hidden layer of 128 hidden units. The decoder mirrors this architecture, decoding from the latent space through a hidden layer of 128 units back to the input dimension.

The reconstruction loss is computed using the reparameterized latent vector, and the total loss includes a standard KL divergence term. During training, we set the batch size to 30 and trained the model for 1500 epochs, selecting the best model using validation loss. We used the Adam optimizer with an initial learning rate of 10^{-3}. Tanh activation was used in the decoder output.

B.2 LFADS

We employed the LFADS (Latent Factor Analysis via Dynamical Systems) framework as a sequential variational autoencoder instantiated in PyTorch Lightning. The encoder consisted of two independent bidirectional Gated Recurrent Units (GRUs) with cell-state clipping at ±5.0: an initial-condition (IC) encoder and a controller-input (CI) encoder, each with a hidden size of 16 per direction. The IC encoder processed the full neural activity sequence and mapped its concatenated forward and backward final hidden states (dimension 32) through a linear projection to produce the mean and log-variance of a 16-dimensional Gaussian posterior over initial conditions ($q(g_0 \mid x)$), subject to a minimum posterior variance floor of 10^{-4}. The CI encoder produced a time-resolved representation of the data that was time-lagged by one step to create a causal controller input sequence. The decoder comprised a controller GRU (hidden size 16) that received the CI representation concatenated with the current factor state(total input dimension 48) and emitted a 4-dimensional controller output (z_t) at each time step, sampled from a Gaussian posterior whose KL divergence was penalized against learnable first-order autoregressive prior with initial time constant $\tau = 10$ and noise variance $\sigma^2 = 0.1$. A generator GRU (hidden size 16) received the controller output as input, initialized from the inferred initial condition via a linear mapping, and produced hidden states that were projected

through a kernel-normalized linear layer to a 16-dimensional factor space. A final session-specific FanInLinear readout (weights initialized with standard deviation $1/\sqrt{16}$) mapped factors to the neural observation space—200 neurons for synthetic data and 117 neurons for monkey reaching data—with mean-squared error (MSE) used as the reconstruction objective in both cases. The initial-condition prior was a fixed isotropic Gaussian ($\mu = 0$, $\sigma^2 = 0.1$). Regularization was applied via KL divergence on both IC and controller-output posteriors (scale 10^{-7} each), with both KL and L2 penalties linearly ramped in from epoch 0 over 80 epochs. The model was trained with the AdamW optimizer (learning rate 4×10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$) for up to 200 epochs with a batch size of 64 and dropout rate of 0.02 applied to encoder inputs and generator states.

B.3 CEBRA

We benchmarked the quality of JEDI's embeddings against CEBRA, a self-supervised contrastive learning framework, instantiated by a 1D convolutional encoder(offset10-model) with a receptive field spanning 10 time steps (5 left, 5 right), comprising an initial Conv1d projection from the input neuron dimension to 32 hidden units, three residual skip blocks each with a Conv1d layer of kernel size 3 and GELU activation, and a final Conv1d projection to the output embedding dimension, with L2 normalization applied to project embeddings onto the unit hypersphere. The model was trained using fixed-temperature InfoNCE objective with cosine similarity as the scoring function at temperature $\tau = 1.0$, where positive pairs were drawn using the time delta conditional distribution constructing an empirical distribution of behavioral variable changes over a time offset of 10 steps such that two time points form a positive pair when their behavioral labels fall within this empirical neighborhood — and negative pairs were sampled uniformly at random from the remaining batch; training used the Adam optimizer with learning rate 3×10^{-4} and mini-batch size 512 for 10,000 gradient steps. The encoder was applied to the full flattened input to produce per-timestep embeddings, which were then reshaped to $(N_{\text{trials}} \times T \times d)$ and mean-pooled across the time axis to yield a single d-dimensional trial-level representation. These mean-pooled embeddings were used for all downstream decoding, with ridge regression reporting R^2 on held-out test trials and logistic regression with 20-fold cross-validation reporting multi-class condition accuracy. As a control, a CEBRA-Shuffled variant with randomly permuted behavioral labels was trained under identical settings to confirm that structured embeddings reflect genuine behavior-activity co-variation rather than the contrastive objective alone.

B.4 Meta-Dynamical State Space Model

We compared against the Meta-Dynamical State Space Model of Vermani et al. (2024), the closest architectural relative to JEDI among methods fit to neural recordings, since it also conditions a hypernetwork on a learned embedding. A per-task read-in network maps observations to a 128-dimensional representation, followed by a shared read-in to a 64-dimensional space; the shared read-out is the identity and observations are modelled with a Gaussian likelihood. An embedding encoder produces a 16-dimensional per-task embedding from this shared representation through a hidden layer of width 128, and a LoRA hypernetwork maps that embedding to low-rank ($r = 2$) additive updates of the first two layers of the shared latent dynamics. The dynamics are an MLP over 16 latents with two hidden layers of width 128, and inference uses a deep Kalman filter latent encoder (16 latents, hidden width 128) that is conditioned on the embedding by concatenation, with a common initial condition across tasks and no dropout. Training minimised the negative ELBO plus a penalty $\alpha\|\Delta\|$ on the magnitude of the hypernetwork's parameter updates with $\alpha = 0.1$, using Adam at learning rate 10^{-3}, batch size 8, for 3000 steps with random masking of 30% of observations, seed 42.

For the fixed-point comparison of Table 1 we fit the model to the same GRU MemoryPro activations used for JEDI, matching its embedding dimensionality to JEDI's (16). Each of the four task phases was treated separately, and for each of the 36 target angles we encoded that phase's trials to obtain the per-task embedding, hypernetwork updates and posterior latent states. Fixed points were located in the 16-dimensional latent space by minimising kinetic energy from 256 initialisations per task with learning rate 0.01, tolerance $q < 10^{-6}$ and a uniqueness tolerance of 0.1, seeded from the encoded latent states. For each angle we retained the stable fixed point of lowest kinetic energy, giving one ring of 36 points per phase for the Procrustes alignment of Appendix C.1.

B.5 LTRRNN

We also compared against LtrRNN (Pellegrino et al., 2023), which, like JEDI, recovers connectivity that varies across trials. Rather than generating weights from a learned code, LtrRNN parameterises the trial-by-neuron-by-neuron connectivity tensor directly as a low-tensor-rank factorisation, with factors shared across trials combined with per-trial coefficients smoothed over trial index by a Gaussian process prior, all fit end to end by gradient descent. We used 200 recurrent units at tensor rank 5, unit time constant, no recurrent noise and no bias, simulated over a duration of 2, with an orthogonal decoder and no decoder bias. The Gaussian process over trials used lengthscale $l = 15$ with $\sigma = 0.1$ and observation noise $\sigma_{\text{obs}} = 0.1$, over a continuous covariance. Inputs were modelled in the space of the recurrent units with control dimensionality 3, a control network of two hidden layers of width 150, control hidden dimension 150 and control time constant 2, fit condition-wise; controls were applied to the preparatory (context) epoch only, and the preparatory epoch itself was excluded from the reconstruction loss. Training used Adam at learning rate 10^{-3} for 8000 iterations with squared-norm regularisation of weight 0.01, a held-out fraction of 0.2 with 20% of entries masked in structured blocks (half-blocks of $5 \times 5 \times 1$ for training and $2 \times 2 \times 1$ for test), convergence monitored every 50 iterations.

For Table 1 we fit LtrRNN to the same GRU MemoryPro activations, arranged as time $\times$ trial $\times$ neuron with 300 timesteps per trial and time measured relative to stimulus onset. Trials were indexed in dataset order for the Gaussian process prior, which imposes smoothness across that ordering rather than across experimental time. Fixed points were then recovered per phase from each trial's reconstructed connectivity using the procedure of Appendix C.1.

B.6 OUR PROPOSED MODEL

We employed a hierarchical model instantiated by a hypernetwork made up of a three-layer feed-forward neural network that received 16-dimensional context vectors as input and generated the parameters of a downstream recurrent connectivity matrix. The network included a hidden layer of size 128, and ReLU activations were applied to the hidden layers. The output layer is linear and carries no nonlinearity. In the full-rank setting, the hypernetwork produced N^2 parameters, where N is the number of neurons, while in the low-rank setting, it generated $2 \times N \times R$ parameters for each trial of activity, with R denoting a user-defined rank. For synthetic data experiments, we chose $R = 5$, and for monkey reaching data, we used $R = 7$, benchmarked through validation set R^2 scores during training. The hypernetwork parameters Θ_h and the full collection of context vectors $\{c^{(j)}\}$ are optimized jointly by the same Adam optimizer, with learning rate 10^{-3}, batch size 32, for 2000 epochs. Contexts are initialized to zero and are unconstrained thereafter.

No regularization on weights or embeddings were applied other than the low-rank structure of the generated connectivity and gradient norm penalty. No weight decay, no dropout, and no distributional prior on the context vectors. Appendix E.5 reports what happens when explicit regularization is added.

C FIXED POINT FINDING AND VISUALIZATION

To evaluate the model's ability to capture ground-truth dynamical structures, we utilized a task-trained RNN as a benchmark. We trained this model on the MemoryPro task from the Computation through Dynamics repository (Versteeg et al., 2025) and simulated trials across four distinct phases: context, stimulus, memory, and response. We qualitatively assessed the true and inferred dynamics by analyzing their fixed-point structures. Fixed points represent regions in the state space where dynamics are slow enough to permit linear approximation, revealing the system's local stability and behavior. To identify these points, we located coordinates in the model's state space ($r \in \mathbb{R}^N$, the same variable as in Equation 1). that minimized the system's kinetic energy. We performed this analysis using a modified version of the fixed-point finding toolkit developed by Golub & Sussillo (2018), adapting it for autonomous RNNs. For visualization, we applied Principal Component Analysis (PCA) to the hidden state trajectories and projected the identified fixed points into this low-dimensional subspace. We used speed tolerance of 10^{-15} for the MemoryPro task and 10^{-9} for motor cortex recordings.

C.1 Quantifying fixed-point geometric recovery

To compare recovered fixed-point geometry against ground truth, we use a Procrustes criterion. For a given MemoryPro phase we collect the fixed points recovered by a model, ordered by the stimulus angle of the trial that produced them, and the corresponding set from the ground-truth task-trained network. Both sets are projected into the leading principal components of their own hidden-state trajectories and mean-centred. We then solve for the similarity transform, rotation, isotropic scaling and translation only, with no shearing and no per-point reordering, that minimises the sum of squared distances between corresponding points, and report the residual normalised sum of squares as the disparity. A disparity of 0 denotes exact recovery up to rigid motion and uniform scale, and 1 denotes no better alignment than the centroid. Results are reported in Table 1.

D Details of Lyapunov Exponents calculation

To calculate the Lyapunov exponents, we followed algorithm proposed by Engelken & Wolf (2023) the algorithm tracks how tiny perturbations to the network's state grow or shrink over time by evolving a set of test vectors alongside the main simulation. The algorithm uses jacobian matrix to update these vectors, measuring how the network activations amplify or dampen small differences. By averaging the logarithms of the stretching factors (the values from the decomposition) over the entire simulation, we can determine the final exponents. While a positive exponent confirms that the network is chaotic, stable dynamics is characterized by negative exponent.

E Supplementary Experiments and Details

E.1 Hyperparameter Sweep

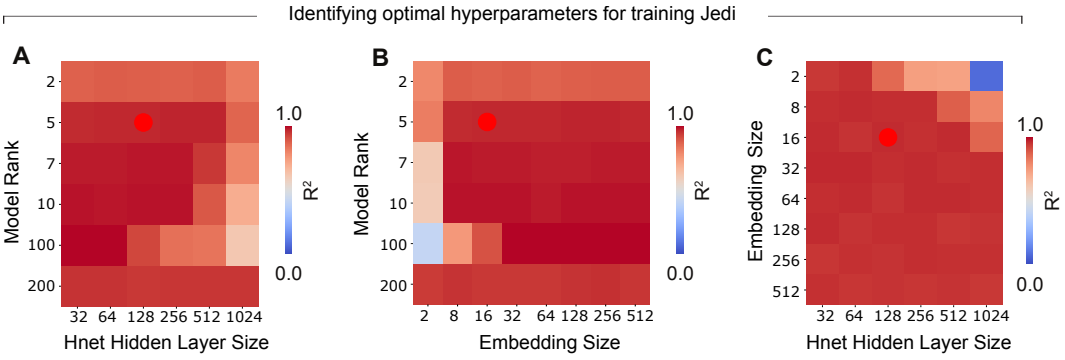

Figure S1: Jedi performance on the synthetic data generated by the teacher RNN. The heatmaps show accuracy of reconstruction as measured by the coefficient of determination R^2 score. The red circular dot highlights the hyperparameter that was chosen for experiments. A) R^2 heatmap for various model rank vs Hypernetwork hidden layer size. B) R^2 heatmap for various Model rank vs Embedding size of the hypernetworks and c) R^2 heatmap for various embedding size vs Hidden size of the hypernetwork.

E.2 Effect of data rank on performance of Jedi

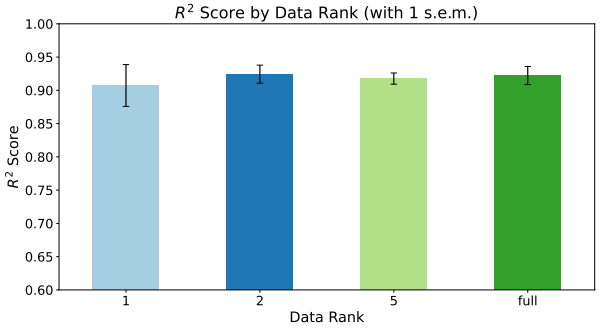

Figure S2: Performance for Jedi on the multi-task synthetic data generated by the teacher RNN with different rank of connectivity matrix J. The bar plot shows accuracy of reconstruction as measured by coefficient of determination R^2 score (with 1 s.e.m). Jedi was able to fit well on varied data ranks with the generated weights set to rank=5.

E.3 Does JEDI recover condition-specific rank?

A natural concern with the synthetic teacher network (Appendix A.2) is that teacher and student share a low-rank form, which could make recovery easier than it would be in general, and that the rank of the generated connectivity is fixed by the user rather than inferred. We therefore built a benchmark in which the generating rank is varied and is never revealed to the model. We generated activity for the same six task types (sine, ramp, fixed-point, decay, cosine, square) using teacher RNNs whose connectivity ranks differ, and, in the harder variant, made individual trials of the same task alternate between two distinct generating ranks. JEDI is trained with a single user-specified rank cap R, and we ask whether the generating rank nevertheless leaves a signature in the learned embeddings and in the generated connectivity.

Embeddings separate by generating rank. Holding task identity fixed and changing only a trial's generating rank, the embeddings split into two groups. We quantify this with a separation ratio: for each task we split its trials into a low-rank group L and a high-rank group H by true generating rank, take each group's centroid c_L, c_H and each group's within-group spread s_L, s_H (mean Euclidean distance from a trial's embedding to its own centroid), and report $\|c_L - c_H\|/\mathrm{mean}(s_L, s_H)$. Values well above 1 indicate the two groups are cleanly separated relative to their internal scatter.

Table S1: Separation of JEDI's embeddings by the teacher's generating rank, with task identity held fixed. Separation ratio is the distance between the two rank groups' centroids divided by their mean within-group spread; values $\gg 1$ indicate clean separation.

	rank pair scaled to task		matched pair (5 vs 10)	
Task	rank pair	separation ratio	rank pair	separation ratio
sine	1 vs 2	41.3	5 vs 10	4.3
ramp	2 vs 4	54.4	5 vs 10	3.8
fp	5 vs 10	8.0	5 vs 10	5.5
decay	10 vs 20	2.9	5 vs 10	4.1
cosine	20 vs 40	1.8	5 vs 10	4.3
square	50 vs 100	1.3	5 vs 10	5.7

Separation is largest when the two ranks are small and close to the intrinsic dimensionality of the task, and decays as the baseline rank grows ($41.3\times$ at ranks 1 vs 2, falling to $1.3\times$ at 50 vs 100), which is expected: once both teachers are far above the dimensionality the task actually requires, additional rank has little effect on the generated activity and therefore little to leave a signature. Fixing a uniform comparison at rank 5 vs 10 gives consistent separation across all six tasks (3.8–$5.7\times$). The

displacement between the rank-5 and rank-10 embeddings also points in a largely shared direction across tasks (mean pairwise cosine similarity ≈ 0.8), indicating that JEDI encodes rank along a roughly common axis rather than in six task-specific subspaces.

The generated recurrent matrix tracks true rank. More directly, we computed the effective rank of each condition's own generated $J = MN^\mathsf{T}$, as the participation ratio of its singular-value spectrum, and correlated it against that condition's true teacher rank.

Table S2: Effective rank of the generated connectivity (participation ratio of its singular-value spectrum) against the true teacher rank, which is never supplied to the model.

Setup	Initialised model rank R	True Rank Range	Pearson r (true vs. effective)
Per-task rank	50	1–50	0.968
Per-trial rank	100	1–100	0.951

Across teacher ranks spanning 1–100, effective rank tracks true rank closely ($r = 0.95$–0.97) with no regularization of any kind, and it does so even when two closely spaced true ranks (5 vs 10) are used with the model's rank cap R set just above them, leaving little spare capacity in which unused dimensions could be parked. We read this as evidence that the user-specified R acts as a cap rather than as a target, and that the recovered dimensionality is driven by the data.

E.4 Embedding and initialisation protocol

Initial conditions. Equation 3 is an initial value problem and $r(0)$ is not produced by the hypernetwork. For every trial, train or test, we set $r(0) = r^{(c)}(0)$, the trial's own first recorded or simulated population state, and integrate autonomously thereafter; no further ground-truth activity enters the forward pass. Each baseline is given the same first time bin, so all methods are matched on the information available at $t = 0$.

Embeddings for held-out trials. Two procedures appear in this paper and should not be confused. (i) Generalization (Figures 2.E–F and 5.E–F): we take the mean of the training-set embeddings for a condition and evaluate that single fixed vector against held-out trials, against every condition rather than only the matching one, reporting reconstruction R^2. No gradient-based optimization is performed against the held-out trial's own activity, and the identical protocol is applied to every baseline, so the comparison is procedurally matched and confers no advantage on JEDI. (ii) *Context inference for a new trial*: the embedding is initialized at zero and optimized by gradient descent against that trial's own reconstruction loss with the hypernetwork frozen, after which a linear classifier trained on previously-seen embeddings assigns a condition. Procedure (ii) is never used to produce the results in (i).

E.5 Effect of hypernetwork capacity and explicit regularization

Hypernetwork capacity. We swept the internal MLP depth (1, 3, 5 hidden layers) against width (64, 128, 256 units) on the synthetic benchmark. Reconstruction improves modestly with width (R^2 from $\approx$0.95 to $\approx$0.97–0.98), and the qualitative structure of the embedding space, six tasks forming six tight and well-separated clusters, is identical across all nine combinations, differing only by the arbitrary rotation of the PCA projection. Critically, the effective rank of the generated connectivity does not grow with capacity.

Table S3: Effective rank of the generated connectivity as a function of hypernetwork depth and width, measured against a rank cap of $R = 5$, with no explicit regularization in any run. Width compresses effective rank rather than inflating it.

Depth \ Width	64	128	256
1	2.86	2.21	1.62
3	2.62	2.03	1.89
5	2.90	2.04	1.98

Measured against a rank cap of $R = 5$ in all nine runs, effective rank never exceeds $\approx 58\%$ of the available capacity, and width *compresses* it rather than inflating it, with no explicit regularization anywhere in the sweep. Depth has a weaker, non-monotonic effect and likewise never approaches the cap. A larger hypernetwork therefore converges to a more compressed solution while fitting the data more accurately, which is the opposite of the failure mode the concern anticipates.

Regularization. We also applied L_2 weight decay directly to the hypernetwork's parameters, swept over five orders of magnitude (depth 1, width 128).

Table S4: Effect of explicit L_2 weight decay on the hypernetwork (depth 1, width 128). Decay shrinks the generator but also the context embeddings, and at 10^{-2} the embedding space collapses entirely, taking effective rank to exactly 1 for every condition.

L_2 strength	mean effective rank	std	hypernetwork weight norm
0	2.16	0.256	12854
10^{-5}	2.13	0.238	4975
10^{-4}	3.18	0.019	1748
10^{-3}	3.61	0.004	471
10^{-2}	1.000	0.000	148

Decay shrinks the hypernetwork's weights as intended, but effective rank does not compress as expected. It first rises *toward* the cap while the across-condition variance collapses toward zero, meaning every condition converges on nearly the same rank, and at 10^{-2} it collapses completely to 1.000 for every trial while the embedding space collapses with it, leaving all six tasks indistinguishable. The reason is structural rather than a matter of tuning: the per-condition embeddings are themselves parameters of the optimization and are therefore penalized alongside the generator, so this regularizer attacks the conditioning mechanism rather than the object we wish to constrain. The per-condition low-rank penalty we do use acts on the generated connectivity directly and does not have this failure mode.

Sensitivity to embedding dimension. We also crossed embedding dimension (4, 8, 16, 32) with seed , reporting reconstruction R^2.

Table S5: Reconstruction R^2 on the synthetic benchmark across embedding dimension and random seed. Performance is stable at dimensions 8 and 16 and noticeably more seed-dependent at either extreme, while the qualitative embedding geometry is consistent throughout.

Embedding dim	Seed 0	Seed 1	Seed 2
4	0.983	0.984	0.888
8	0.990	0.990	0.993
16	0.994	0.993	0.992
32	0.907	0.929	0.993

Reconstruction is stable across seeds at dimensions 8 and 16 ($R^2 \approx 0.99$ for all three seeds), but noticeably less reliable at either extreme, reaching 0.888 for one seed at dimension 4 and 0.907/0.929 at dimension 32. Despite this variance in accuracy, the qualitative geometry of the embedding space, which tasks cluster near which, is consistent across every dimension and seed we tested. Embedding dimensionality is therefore not a free parameter, and practitioners should expect genuine seed-to-seed variation at the extremes rather than assuming that larger is uniformly better.

E.6 ADDITIONAL RESULTS ON SPECTRAL ANALYSIS

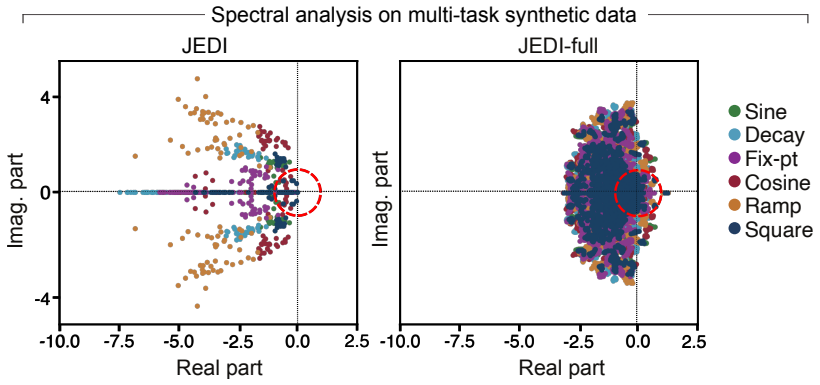

Figure S3: Eigenvalue spectra of learned recurrent weights on multi-task synthetic data. JEDI learned consistent structure across tasks, while JEDI-full exhibited a dense, isotropic cloud without task separation.

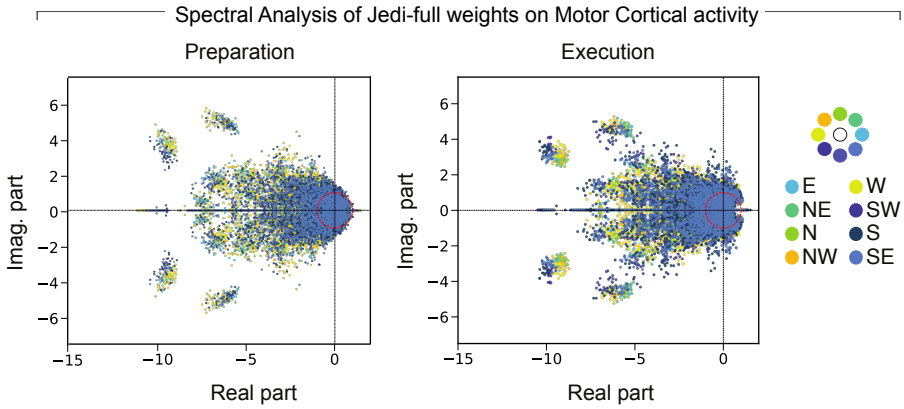

Figure S4: Eigenvalue spectra of Jedi-full weights on monkey motor cortical activity. JEDI-full exhibited a dense, isotropic cloud without task/direction separation.

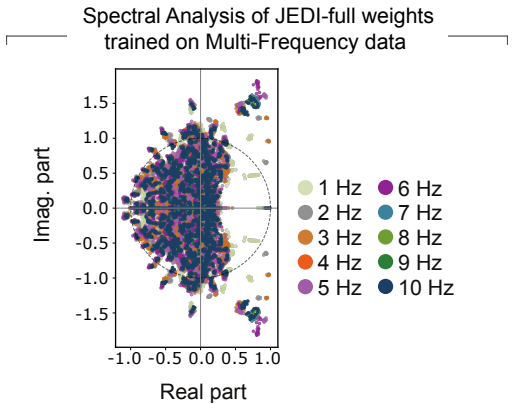

Figure S5: Eigenvalue spectra of learned recurrent weights on multiple frequency dataset. The legend indicates the specific sine frequencies used to train the models, JEDI-full consistently exhibits a dense, isotropic spectral cloud. This distribution lacks any discernible structure or cluster formation corresponding to the underlying task frequencies.

E.7 ADDITIONAL FIXED POINTS RESULTS

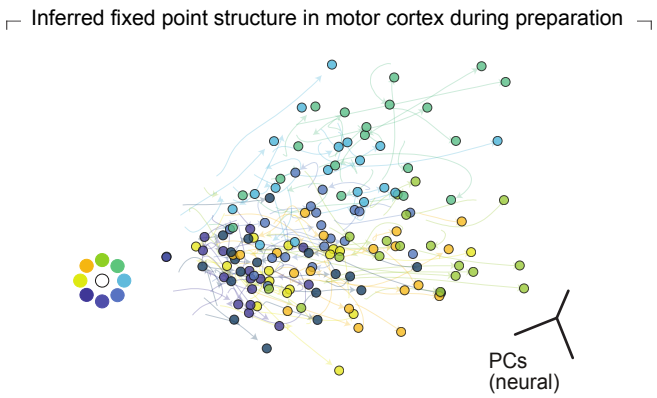

Figure S6: Stable fixed points during preparation inferred by JEDI colored by reach directions. Single-trial neural trajectories for each corresponding reach is also plotted.

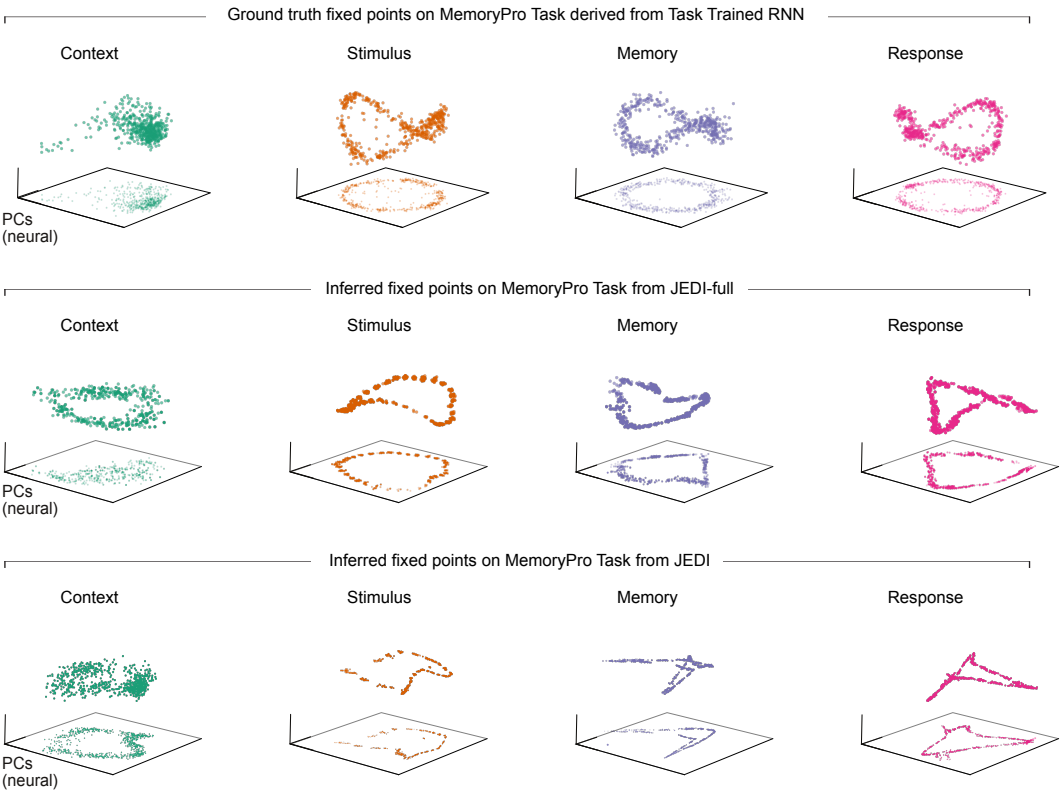

Figure S7: 3d PCA projection of fixed points from Jedi across 4 periods on MemoryPro task. The fixed points structure closely resembled the structure of task-trained RNN.

F COMPUTE RESOURCES

We list below the compute resources used per experiment:

1. Multi-Task Teacher experiment: Results were computed on external cluster equipped with Nvidia L40S GPUs. Training wall clock times was 6hrs. Inference converges in a few seconds.

2. Multi-Frequency Sine experiment: Results were computed on external cluster equipped with Nvidia L40S GPUs. Training wall clock times was 6hrs. Inference converges in a few seconds.
3. Task trained RNN experiment: Results were computed on external cluster equipped with Nvidia L40S GPUs. Training wall clock times was 1.5-2 days. Inference converges in a few seconds.
4. Monkey reaching experiment : Results were computed on external cluster equipped with Nvidia L40S GPUs. Training wall clock times was 6hrs. Inference converges in a few seconds.