CHARTING AND NAVIGATING THE SPACE OF SOLUTIONS FOR RECURRENT NEURAL NETWORKS

Anonymous Author(s)
Affiliation
Address
email

Abstract

In recent years Recurrent Neural Networks (RNNs) were successfully used to model the way neural activity drives task-related behavior in animals, operating under the implicit assumption that the obtained solutions are universal. Observations in both neuroscience and in machine learning challenge this assumption. Animals can approach a given task with a variety of strategies, and training machine learning algorithms introduces the phenomenon of underspecification. These observations imply that every task is associated with a space of solutions. To date, the structure of this space is not understood, limiting the approach of comparing RNNs with neural data. Here, we characterize the space of solutions associated with a given task, and how hyperparameters bias networks within this space. We first study a simple two-neuron network on a task that leads to multiple solutions. We trace the nature of the final solution back to the network's initial connectivity, and identify discrete dynamical regimes that underly this diversity. We then examine the neuroscience-inspired Ready-Set-Go timing task. We find a rich set of solutions, even under identical hyperparameters. We uncover this variety by testing the trained networks' ability to extrapolate, as a perturbation to a system often reveals hidden structure. Furthermore, we relate extrapolation patterns to specific dynamical objects and to effective algorithms found by the networks. Using a set of features, we define a space of solutions. We find that most networks stay within the vicinity of the solution defined by their initial connectivity. Taken together, our results shed light on the concept of the space of solutions and its uses both in Machine learning and in Neuroscience.

1 Introduction

Modern machine learning operates in an over-parameterized regime, implying that many different parameter-sets can achieve low error on a given training set [1]. This observation implies that for every task, there exists a space of solutions that can implement it. What are the properties of such a solution space? What biases networks to prefer one solution over the other? These questions remain largely unanswered. A parallel phenomenon in neuroscience is the variety of strategies with which animals can approach a given task [2]. In light of these observations, the recent success of matching the activity of trained artificial networks to that of experimental recording seems puzzling [3, 4, 5].

To facilitate the use of artificial networks as neuroscience models, it is necessary to chart the space of solutions that arises when optimizing networks for a given task, as well as understand how different modeling choices bias networks to different areas in this space. Here, we first apply this approach to a simple two-neuron network and demonstrate how distinct solutions arise. We then we study a timing task [6] inspired by the neuroscience literature. We show that different networks find

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

qualitatively different solutions, even when using identical hyper-parameters. To expose the variety of solutions, we challenge the networks with inputs that are outside the distribution of the training set, and systematically characterize the responses. Furthermore, we are able to show that the diversity revealed with these challenging inputs correspond to qualitatively different computations performed by the network within the training set. In both cases, we show that network architecture can bias the solutions, but that initial weights of networks largely determine their final solution type within a given architecture.

2 Related work

Recent work, [7] studied the effects of modeling choices over the dynamics of the trained solutions, in a few canonical tasks. They trained thousands of RNNs, while systematically controlling the hyperparameters, and analyzed the geometrical and topological aspects of each solution. They concluded that while the geometry of different solutions can vary significantly across different architectures, the underlying computation and dynamical objects are widely universal. Our results are superficially opposite to theirs. This is probably due both to the choice of tasks, and on challenging networks with unexpected inputs.

[8] show that deep network trained on vision tasks develop different internal representations, especially in the higher layers. While reaching similar conclusions on the need to use populations of networks for neuroscience comparisons, the focus there is on representations rather than dynamics, and the space of solution is not clustered into distinct qualitatively different solutions. While writing this manuscript, we were made aware of a recent preprint by [9] that takes a very similar approach to ours. Apart from the different task, architecture and learning rule studied there, our analysis of the two-neuron network also provides a tangible example of discrete basins of attraction in solution space. An underspecified two-dimensional epidemiological model was studied by [1]. The variability in solutions there, however, was continuous. Specifically, all solutions were connected on a single manifold and the dynamics of the system did not undergo bifurcations along this manifold.

Underspecification was also highlighted in neuroscience models that are not based on artificial neural networks ([10]) and is observed more generally in complex systems ([11]).

3 Space of solutions in a 2D RNN task

We first study the space of solutions in an extremely simple scenario that nevertheless shows qualitatively different solutions. We consider a 2-neuron continuous-time network (Figure 1A), in which the state of the network $\boldsymbol{x} \in \mathbb{R}^2$ evolves according to:

$$\dot{\boldsymbol{x}} = -\boldsymbol{x} + W\phi(\boldsymbol{x}),\ \boldsymbol{x}(0) = \begin{bmatrix}1\\0\end{bmatrix} \tag{1}$$

We choose $\phi := \tanh$. For the task, we simply require that $\boldsymbol{x}(T) = \begin{bmatrix}0\\1\end{bmatrix}$ at a time $T = 10$ (red dots in Figure 1B,C). Accordingly we define a loss:

$$\mathcal{L} = \frac{1}{2}\left\|\boldsymbol{x}(T) - \begin{bmatrix}0\\1\end{bmatrix}\right\|^2. \tag{2}$$

The system has four parameters, given by the elements of $W \in \mathbb{R}^{2\times 2}$. Despite the simplicity of the system we discover two qualitatively different classes of solutions (Figure 1B,C). In both cases, the origin becomes unstable – either as a saddle point, corresponding to a fixed point near the target, or as an unstable focus, corresponding to a limit cycle that passes through the target. The initial random connectivity determines which solution is chosen. Because there are four parameters, we project them into two different 2D spaces. Figure 1D shows the trace and determinant of the Jacobian at the origin namely $J := -I + W$ where I is the 2×2 identity matrix, showing that its initial nature (complex or real) largely determines the final outcome. A more detailed view of the loss

landscape can be obtained by charting learning dynamics on the first two principal components of the parameters (Figure 1E). This representation clearly shows distinct islands of low loss, corresponding to the different solutions. Note that there are multiple discrete limit cycle solutions, corresponding to integer multiples of the target time T.

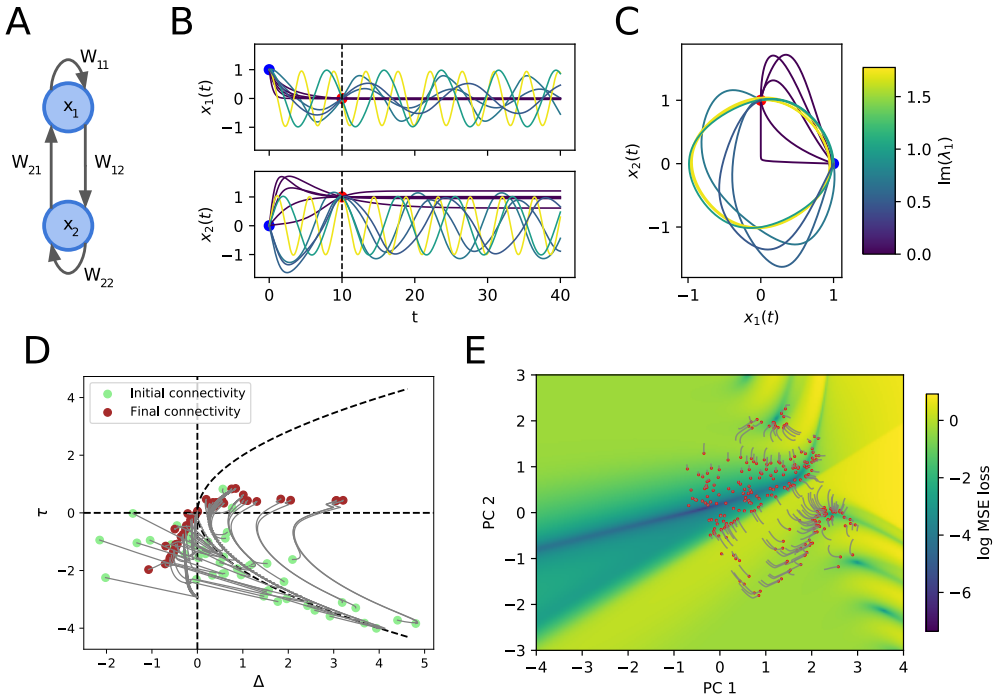

Figure 1: Two-neuron network. **A** Network architecture, with the four trainable parameters. **B** Networks trained to go from an initial state (blue) to a final state (red) in time $T = 10$. Each trajectory is from a different network, colored according to imaginary part of eigenvalue of connectivity. **C** Trajectories in B, shown in phase space. **D** Trace-determinants of gradient descent trajectories. Parameters are initialised randomly from $\mathcal{U}(-1.5, 1.5)$. **E** Loss landscape and trajectories of gradient descent along a slice of parameter space given by first two principal components of the solutions. Parameters are initialised randomly and flattened onto PC 1 and 2 before training.

We thus conclude that the network converges to qualitatively different solutions. These can be described as discrete basins of attraction in parameter space. The initial connectivity of the network determines which solution is found, and different hyperparameters can bias this choice (See supplementary).

4 Timing task

4.1 Task and architecture

Guided by the results from the two-neuron network, we examine a more complex, neuroscience-inspired, task. A priori, it is not clear whether such a task will exhibit more or less variable solutions. On the one hand, a complex task might lend itself to multiple algorithmic solutions. On the other hand, a complex task represents more constraints on the network, and hence might lead to convergence to a unique solution. We thus study the *Ready-Set-Go* timing task [6]. In our implementation of the task (Figure 2A), a recurrent neural network receives two input pulses – Ready and Set – separated by t_s time steps. The task is to generate an output pulse $t_t = t_s$ time steps after the Set pulse. The training intervals were drawn from a uniform distribution $t_s \in [30, 120]$. To increase our chances of exposing diverse solutions, we trained networks with three different RNN architectures: Vanilla, GRU, and LSTM (400 realizations each). See the Supplementary material for more details regarding

the training process. To visualize the entire training set, we stack all trials aligned to the *Ready* pulse (Figure 2B, red). This results in the *Set* pulse (yellow) forming a line with slope 1.0 and the *Go* pulse (green) a line with slope 0.5. Figure 2C shows the output of a trained network matching this pattern.

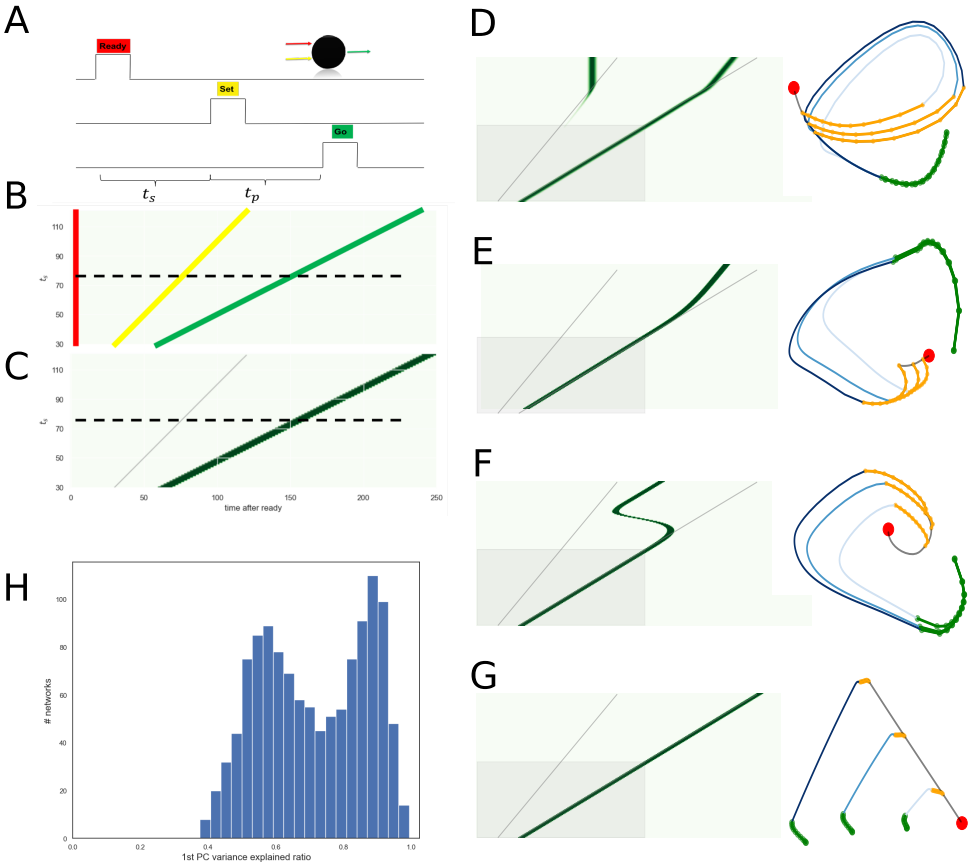

Figure 2: **A** Task. The network is presented with two pulses (Ready and Set, provided through separate input channels) separated by t_s time-steps. The desired output is a Go pulse, delayed by $t_t = t_s$ steps from the Ready pulse. **B** Desired output for all trials. The dashed line is the trial shown in (A). **C** Actual output of a trained network. **D-G** Four distinct solutions. (left): Outputs of each network to time-intervals in the range $[30, 200]$. The shaded area corresponds to the training set. (right): PCA of activity in the respective networks. The *Ready* pulse is marked as a red circle and it is followed by the activity during the *Ready-Set*, which is drawn as a black line. The orange lines correspond to the *Set* pulse given after different amounts of time t_s. The *Set-Go* trajectories are colored by t_s and evolve through time to a region in phase space (colored in green) in which the *Go* production occurs **H** A histogram of the variance explained ratio of the 1st PC of the networks entire task state space.

4.2 What constitutes a solution?

Before comparing how the different networks solve the timing task, it is worth dwelling on the concept of a solution. A normative approach might be to infer the algorithm from the network's connectivity, activity or output. For example, computing 4×5 by a multiplication table or multiple addition operations, can be thought of as two distinct algorithms. Nevertheless, the term algorithm is not even formally defined [12]. Because our motivation stems from comparing networks to biological data, we take an operative approach which relies on measures that could in principle be obtained

experimentally. We thus consider either the neural activity of the network while performing the task, or its output (behavioral) response to unexpected inputs (catch trials).

4.3 Diverse solutions to a timing task

We begin by measuring one neural feature and one behavioral feature for each network. The neural feature is a proxy for the dimensionality of the neural trajectories. Specifically, Figure 2H shows that the variance explained by the first principal component spans a large range, possibly indicating variable mechanisms. On the behavioral side, we challenge networks with stimuli that are outside of the trained distribution. In our task, extrapolation to $t_s > 120$ is a natural choice. Figure 2D-G shows the behavior of 4 networks on $t_s \in [30, 200]$, with the shaded area depicting the trained region. The output of the networks reveals a wide range of qualitatively different behaviors – suggesting different underlying mechanisms for solving the original task.

Both of these measures could be criticized as measuring irrelevant variability. It could be that dimensionality differences reside in activity subspaces that are orthogonal to those used by the task, and hence do not reflect any algorithmic difference. Likewise, it could be that the way networks respond to long delays is so unconstrained, as to bear no relationship to what happens during the nominal task. We will show that this is not the case – neither for the example networks shown in Fig 2D-G, nor for the entire population of networks. Figure 2 D-G shows the activity of the same networks projected onto the first principal components. Crucially, only trials with $t_s \in [30, 120]$ are shown. It is quite evident that the differences in extrapolation are reflected in the network activity. Furthermore, it is possible to infer at least three distinct algorithms for solving the timing problem from these plots. Because the task requires counting time in two distinct epochs – from Ready to Set and from Set to Go – we focus on these to explain the algorithms. Network **D**, exhibiting perfect extrapolation, seems to count time linearly in one direction from the Ready to the Set pulse, and in another direction from the Set pulse to the Go output. In contrast, network **A** uses the same phase space trajectory for both epochs, effectively counting from 120 downwards upon the Ready pulse, and moving to an earlier point upon the Set pulse. Networks **B,C** are similar to network **D** in having distinct trajectories for the two epochs. The non-linear trajectories, however, correspond to imperfect extrapolation. Furthermore, the almost circular trajectory of network **C** hints at the oscillatory behavior revealed by extrapolation.

4.4 A space of solutions

The examples described above suggest that, similar to the two-neuron case, trained networks converge into distinct solutions which can be characterized both by neural signatures and by behavioral ones. In the case of the two-neuron network, the eigenvalues were a natural space to depict the various solutions. We argue that this is because the eigenvalues separated two distinct dynamical objects – fixed points and limit cycles – and advocate a similar space for the timing task. While in some cases [13, 7], Figure 2D-G, it is possible to identify such dynamical objects – there are no guarantees. Specifically, for a task that is based on measuring time, slow transients rather than fixed points are expected to play a major role. We thus characterize the dynamical objects appearing in network dynamics by leveraging the task structure. Figure 3 shows the salient locations in phase space, and how they are related to the task structure. Following a ready pulse, the network enters a slow area for the entire training interval ($t_s < 120$). After this time, some networks emit a Go pulse (panels A,C,F), some enter a limit cycle (panels A,D), and some reach a fixed point (panels B,C,F). Some LSTM networks (panel E) continue moving in the same direction without bound, due to the unbounded nature of the activation of their units [14]. We also consider the effect of a Set pulse given in an average interval ($t_s = 75$), and depict the possible locations in phase space that it can lead to. For instance, panel C represents the class of networks shown in Figure 2A. Altogether, this analysis yields six possible topologies of the dynamical landscape. These can be considered as six different algorithms, as they describe the network's response to stimuli. Nevertheless, some of the differences can only be revealed outside of the training set. We will show that experimentally accessible neural features can be used to infer these topologies, and thus give access to the space of solutions.

4.5 Inferring dynamical topology from neural activity

The topologies described above are obtained by simulating the networks for very long times, to obtain their asymptotic behavior. In a neuroscience setting, such information is not readily available. Is it

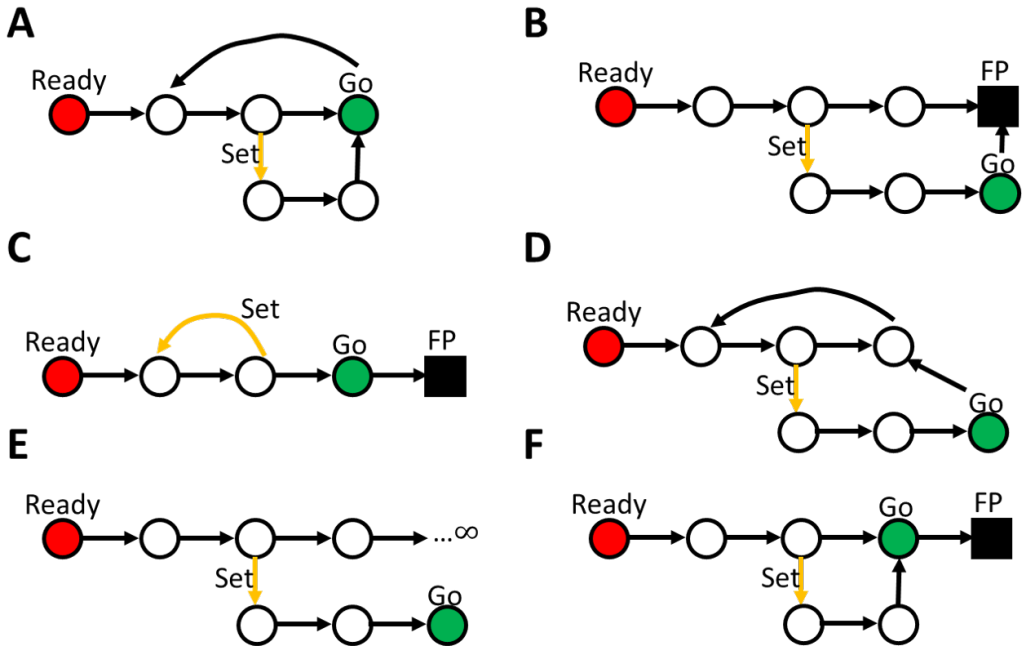

Figure 3: Six possible topologies of network dynamics. Networks differ in several qualitative features. **A,C,F** A Go pulse is emitted without a Set pulse. **A,D** Networks converge to a limit cycle. **C** The Set-Go and the Ready-Set trajectories merge. **E** Activity is unbounded, effectively implementing a counter.

possible to infer this asymptotic behavior from neural activity during the training set? To answer this question, we extracted a set of features, as summarized in Figure 4, and detailed in the supplementary material. The features mostly measure the relationship between neural activity in the different epochs of the task. Other features measure geometrical properties of a single epoch, as in the curvature of the Ready-Set trajectory, which was inspired by networks like Figure 2F. While we tried to include features that are natural descriptors of the neural activity, there is some arbitrariness in our choice. We chose to err on the side of including more features, and later rely on cross validation to avoid over fitting in our predictions [15].

We trained a decision tree to predict the topological class from the neural features. This was done separately for each architecture, and on the entire pool of networks. The classifier was trained on two thirds of the networks and tested on the remaining third. The confusion matrices are in the supplementary material, and Cohen's kappa [16] was used to summarize the prediction quality, showing an above chance performance, with Vanilla networks being the most accurately predicted Figure 4.

The neural features can also be used to predict the extrapolation performance of the various networks. We trained a simple linear regression from the neural features onto the maximum delay correctly performed by the network. Figure 5 shows that these predictions are well above chance, but do not capture the full phenomenon.

4.6 Architecture biases solution

The diversity of solutions reported here stems from both hyperparameters and random initialization. Figure 6A shows that neural features naturally segregate the different architectures. Note, however, that the variability within each architecture is comparable to that between architectures (histograms of individual features are in the supplementary material). Furthermore, Figure 6B shows an example of three networks that have very similar solutions, despite being with different architectures.

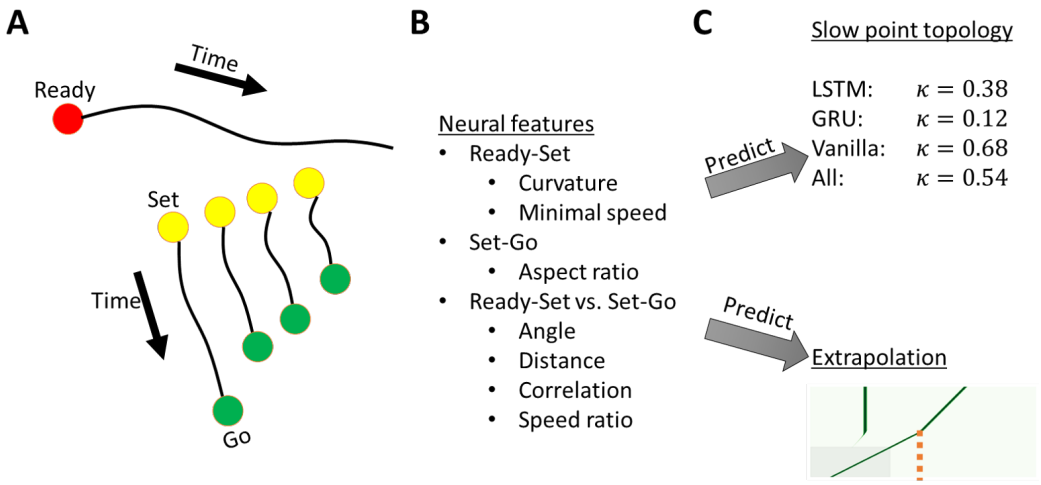

Figure 4: Using neural features to define solution-space. **A** Cartoon of network activity during training, showing the epochs from which neural features are defined. **B** List of neural features (complete definitions in the supplementary material). **C**. Neural features are used to predict both the topology of the dynamics and the extrapolation performance of networks.

5 Discussion

In recent years, trained RNNs were successfully used to model biological circuits. Specifically, these networks converged to solutions that were similar to those of their biological counterparts. This observation is puzzling for any complex system and particularly regarding the brain, a complex biological system in which variability is the rule rather than the exception. Inspired by this puzzle, we study the space of solutions of RNNs for both a simple two-neuron network and a sensorimotor timing task. Through an interplay between observation, analysis, and perturbations of various forms we discovered qualitatively different solutions for the same tasks. By analyzing the neural activity of these networks, we observed that each behavioral phenotype originates from a distinct dynamical mechanism. By extracting relevant behavioral and neural features we could introduce a space of solutions, and study its regularities. Specifically, we found that unit architecture biases the solutions, with LSTM being the only architecture to implement perfect counters. Nevertheless, there was a topological overlap between the architectures, and we could locate the same dynamics and computations in all three studied architectures. We show using a 2D continuous-time RNN that qualitatively different dynamical topologies can arise in the context of single task and that learning can find different solutions depending on initialisation. In a related study, the authors analyse a low-D continuous time GRU and find a large diversity dynamical topologies [17]. They find that the 2D GRU finds topologically different solutions to variants of a working memory task depending on the input statistics. Such a modification to the task, although subtle, modifies the loss landscape and learning dynamics. In our work we emphasize that such topological diversity of solutions can arise for the same task, that is to say within the same loss landscape. Both our work and theirs identifies the implications of topological diversity for generalisation.

Our work has several limitations that should be noted. Even though the features we extracted from the data capture interesting signals, they are neither unique nor inclusive. Especially in the high-dimensional neural space, a single property can be extracted in many different ways. By letting the system evolve from various initial conditions we were able to locate dynamical objects such as fixed points, slow areas, and limit cycles, and draw a general scheme of phase space. However, we believe this search should be done systematically with tools from the theory of dynamical systems [13]. For example, minimizing the velocity function to locate all fixed point and performing the linearization around each of them can help to describe more accurately the existing transitions in phase space. In the few attempts we've done along this line we found that gradient-descent optimization methods that worked well in other tasks struggled to locate the relevant fixed point. This point touches on a broader subject, that is our entire analysis was inspired by and adjusted to the requirements of the task. That is

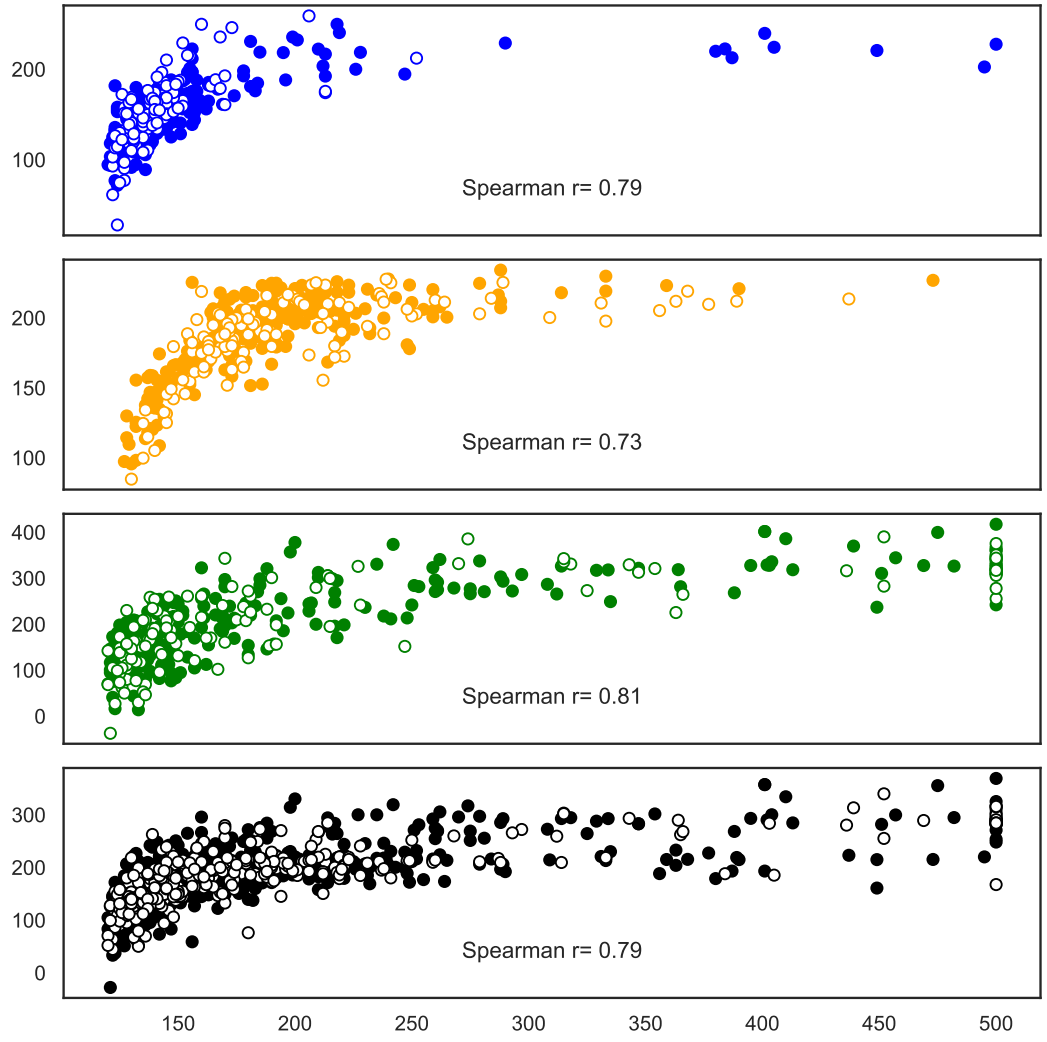

Figure 5: Predicting extrapolation performance from neural features. Linear regression was used to predict the maximum delay correctly performed by the network from the neural features (Figure 4). Filled circles are the training data, and open circles are the test data. Rows correspond to Vanilla (blue), GRU (orange), LSTM (green), and all networks (black). Spearman correlation was used to evaluate the quality of prediction.

not to say that analysis should not be tailored to the task at hand, but in order to make this procedure as systematic and impartial as possible, we should be looking for more characteristics of dynamics that are agnostic to the task [18]. Furthermore, to get a deep understanding of the algorithms and dynamics used by the networks, we focused on two tasks. Clearly, to establish the concept of the space of solutions similar analyses must be done across multiple tasks.

To conclude, we found that RNNs can produce a diverse set of solutions to the same computational task. These solutions represent distinct algorithms and are supported by corresponding dynamical objects. The solutions are isolated in parameter space, causing the initial conditions to largely determine the outcome. Furthermore, we showed that experimentally accessible tools can be used to indirectly characterize asymptotic properties of the solutions. We believe that exploring the space of solutions can advance neuroscience, machine learning, and their intersection – making more rigorous comparisons of models to data.

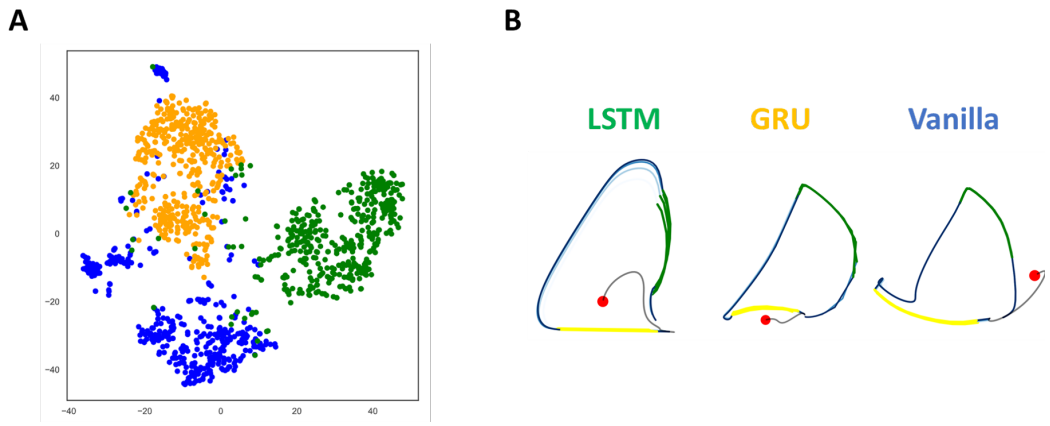

Figure 6: Architecture biases, but does not determine solution. **A** UMAP projection of the neural features, colored according to architecture. **B** Dynamics of three networks from all three architectures projected onto the principal components of neural activity.

References

[1] A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman, F. Hormozdiari, N. Houlsby, S. Hou, G. Jerfel, A. Karthikesalingam, M. Lucic, Y. Ma, C. McLean, D. Mincu, A. Mitani, A. Montanari, Z. Nado, V. Natarajan, C. Nielson, T. F. Osborne, R. Raman, K. Ramasamy, R. Sayres, J. Schrouff, M. Seneviratne, S. Sequeira, H. Suresh, V. Veitch, M. Vladymyrov, X. Wang, K. Webster, S. Yadlowsky, T. Yun, X. Zhai, and D. Sculley. Underspecification Presents Challenges for Credibility in Modern Machine Learning. [Online]. Available: http://arxiv.org/abs/2011.03395

[2] S. Musall, A. E. Urai, D. Sussillo, and A. K. Churchland, "Harnessing behavioral diversity to understand neural computations for cognition," vol. 58, pp. 229–238. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0959438819300285

[3] O. Barak, D. Sussillo, R. Romo, M. Tsodyks, and L. Abbott, "From fixed points to chaos: Three models of delayed discrimination," vol. 103, pp. 214–222. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0301008213000129

[4] V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome, "Context-dependent computation by recurrent dynamics in prefrontal cortex," vol. 503, no. 7474, pp. 78–84. [Online]. Available: http://www.nature.com/articles/nature12742

[5] S. Vyas, M. D. Golub, D. Sussillo, and K. V. Shenoy, "Computation Through Neural Population Dynamics," vol. 43, no. 1, pp. 249–275. [Online]. Available: https://www.annualreviews.org/doi/10.1146/annurev-neuro-092619-094115

[6] M. Jazayeri and M. N. Shadlen, "Temporal context calibrates interval timing," vol. 13, no. 8, pp. 1020–1026. [Online]. Available: http://www.nature.com/articles/nn.2590

[7] N. Maheswaranathan, A. Williams, M. Golub, S. Ganguli, and D. Sussillo, "Universality and individuality in neural dynamics across large populations of recurrent networks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., pp. 15 629–15 641. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/5f5d472067f77b5c88f69f1bcfda1e08-Paper.pdf

[8] J. Mehrer, C. J. Spoerer, N. Kriegeskorte, and T. C. Kietzmann, "Individual differences among deep neural network models," vol. 11, no. 1, p. 5725. [Online]. Available: http://www.nature.com/articles/s41467-020-19632-w

[9] E. Ghazizadeh and S. Ching. Slow manifolds in recurrent networks encode working memory efficiently and robustly. [Online]. Available: http://arxiv.org/abs/2101.03163

[10] E. Marder and A. L. Taylor, "Multiple models to capture the variability in biological neurons and networks," vol. 14, no. 2, pp. 133–138. [Online]. Available: http://www.nature.com/articles/nn.2735

[11] B. B. Machta, R. Chachra, M. K. Transtrum, and J. P. Sethna, "Parameter Space Compression Underlies Emergent Theories and Predictive Models," vol. 342, no. 6158, pp. 604–607. [Online]. Available: https://www.sciencemag.org/lookup/doi/10.1126/science.1238723

[12] J. R. Shoenfield, "Theory of recursive functions and effective computability (hartley rogers, jr.)," *SIAM Rev.*, vol. 11, no. 3, p. 415–416, Jul. 1969. [Online]. Available: https://doi.org/10.1137/1011079

[13] D. Sussillo and O. Barak, "Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks," vol. 25, no. 3, pp. 626–649. [Online]. Available: https://www.mitpressjournals.org/doi/abs/10.1162/NECO_a_00409

[14] M. Suzgun, Y. Belinkov, S. Shieber, and S. Gehrmann, "LSTM Networks Can Perform Dynamic Counting," in *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*. Association for Computational Linguistics, pp. 44–54. [Online]. Available: https://www.aclweb.org/anthology/W19-3905

[15] R. M. Khan, C.-H. Luk, A. Flinker, A. Aggarwal, H. Lapid, R. Haddad, and N. Sobel, "Predicting Odor Pleasantness from Odorant Structure: Pleasantness as a Reflection of the Physical World," vol. 27, no. 37, pp. 10 015–10 023. [Online]. Available: https://www.jneurosci.org/lookup/doi/10.1523/JNEUROSCI.1158-07.2007

[16] M. L. McHugh, "Interrater reliability: the kappa statistic," *Biochemia Medica*, vol. 22, no. 3, pp. 276–282, Oct. 2012, publisher: Croatian Society of Medical Biochemistry and Laboratory Medicine. [Online]. Available: https://www.biochemia-medica.com/en/journal/22/10.11613/BM.2012.031

[17] I. D. Jordan, P. A. Sokol, and I. M. Park, "Gated recurrent units viewed through the lens of continuous time dynamical systems," *arXiv preprint arXiv:1906.01005*, 2019.

[18] A. A. Russo, R. Khajeh, S. R. Bittner, S. M. Perkins, J. P. Cunningham, L. Abbott, and M. M. Churchland, "Neural Trajectories in the Supplementary Motor Area and Motor Cortex Exhibit Distinct Geometries, Compatible with Different Classes of Computation," vol. 107, no. 4, pp. 745–758.e6. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0896627320303664

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] **In Discussion**
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] **This work is basic science, and does not develop any algorithm that might be used in such a manner.**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] No theoretical results
   (b) Did you include complete proofs of all theoretical results? [N/A] No theoretical results
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] **github in supplement**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] **In supplement**

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] **The effect of the random seed is part of the paper**

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [N/A]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]