Appendix Contents

A Task descriptions

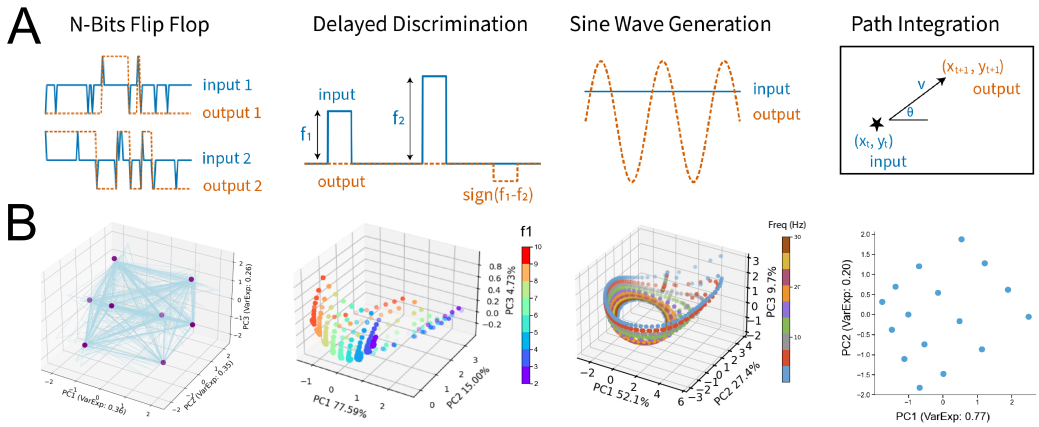

Figure 1: **Our task suite spans memory, integration, pattern generation, and decision-making.** Each task is designed to place distinct demands on the network's dynamics. **N-Bit Flip-Flop**: The network must remember the last nonzero input on each of N independent channels. **Delayed Discrimination**: The network compares the magnitude of two pulses, separated by a variable delay, and outputs their sign difference. **Sine Wave Generation**: A static input specifies a target frequency, and the network generates the corresponding sine wave over time. **Path Integration**: The network integrates velocity inputs to track position in a bounded 2D or 3D arena (schematic shows 2D case).

N-Bit Flip-Flop Task Each RNN receives N independent input channels taking values in $\{-1, 0, +1\}$, which switch with probability p_{switch}. The network has N output channels that must retain the most recent nonzero input on their respective channels. The network dynamics form 2^N fixed points, corresponding to all binary combinations of $\{-1, +1\}^N$.

Delayed Discrimination Task The network receives two pulses of amplitudes $f_1, f_2 \in [2, 10]$, separated by a variable delay $t \in [5, 20]$ time steps, and must output $\text{sign}(f_2 - f_1)$. In the N-channel variant, comparisons are made independently across channels. The network forms task-relevant fixed points to retain the amplitude of f_1 during the delay period.

Sine Wave Generation The network receives a static input specifying a target frequency $f \in [1, 30]$ and must generate the corresponding sine wave $\sin(2\pi f t)$ over time. We define N_{freq} target frequencies, evenly spaced within the range $[1, 30]$, and use them during training. In the N-channel variant, each input channel specifies a frequency, and the corresponding output channel generates a sine wave at that frequency. For each frequency, the network dynamics form and traverse a limit cycle that produces the corresponding sine wave.

Path Integration Task Starting from a random position in 2D, the network receives angular direction θ and speed v at each time step and updates its position estimate. In the 3D variant, the network takes as input azimuth θ, elevation ϕ, and speed v, and outputs updated (x, y, z) position. The network performs path integration by accumulating velocity vectors based on the input directions and speeds. After training, the network forms a Euclidean map of the environment in its internal state space.

B Additional axes of task complexity

In the main text, we controlled task complexity by varying the number of independent input–output channels, effectively duplicating the task across dimensions. Here, we explore two alternative approaches: increasing the task's memory demand and adding auxiliary objectives.

Changing memory demand. Of the four tasks, only Delayed Discrimination requires extended memory, as its performance depends on maintaining the first stimulus across a variable delay. See Appendix H for a quantification of each task's memory demand. We increased the memory load in Delayed Discrimination by lengthening the delay period. This manipulation reduced degeneracy

at the dynamical and behavioral levels but increased it at the weight level, mirroring the effect of increasing task dimensionality (Figure 2A).

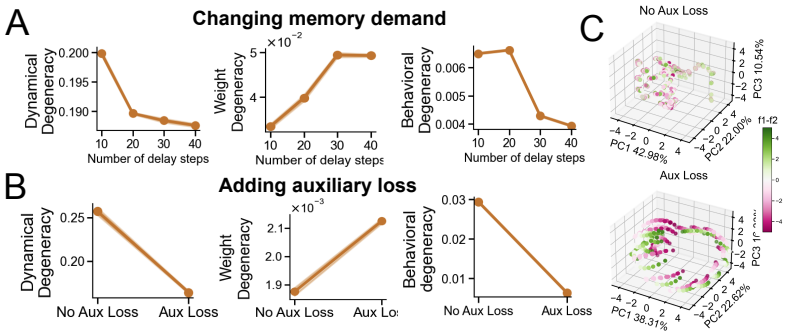

Figure 2: **Memory demand and auxiliary loss modulate degeneracy in distinct ways.** In the Delayed Discrimination task, both manipulations reduce dynamical and behavioral degeneracy while increasing weight degeneracy. The auxiliary loss also induces additional line attractors in the network's dynamics, as shown in (C).

Adding auxiliary loss. We next examined how adding an auxiliary loss affects solution degeneracy in the Delayed Discrimination task. Specifically, the network outputs both the sign and the magnitude of the difference between two stimulus values ($f_2 - f_1$), using separate output channels for each. This manipulation added a second output channel and increased memory demand by requiring the network to track the magnitude of the difference between incoming stimuli. Consistent with our hypothesis, this manipulation reduced dynamical and behavioral degeneracy while increasing weight degeneracy (Figure 2B). Crucially, the auxiliary loss induced additional line attractors in the network dynamics, further structuring internal trajectories and aligning neural responses across networks (Figure 2C). While the auxiliary loss increases both output dimensionality and temporal memory demand, we interpret its effect holistically as a structured increase in task complexity.

C Higher task complexity induces more feature learning

We hypothesize that the increased weight degeneracy observed in harder tasks reflects stronger feature learning. Specifically, harder tasks may force network weights to travel farther from their initialization. If more complex task variants, like those in Section **??**, truly induce greater feature learning, then networks should traverse a greater distance in weight space, resulting in more dispersed final weights. To test this idea, we measured feature learning strength in networks trained on different task variants using two complementary metrics [Liu et al., 2023, George et al., 2022]: **Weight-change norm:** $\|\mathbf{W}_T - \mathbf{W}_0\|_F$, where larger values indicate stronger feature learning. **Kernel alignment (KA):** measures the directional change of the neural tangent kernel (NTK) before and after training: $\mathrm{KA}\left(K^{(f)}, K^{(0)}\right) = \dfrac{\mathrm{Tr}\left(K^{(f)}K^{(0)}\right)}{\left\|K^{(f)}\right\|_F \left\|K^{(0)}\right\|_F}$, where $K = \nabla_W \hat{y}^\top \nabla_W \hat{y}$. Lower KA indicates greater NTK rotation and thus stronger feature learning.

More complex tasks consistently drive stronger feature learning and greater dispersion in weight space, as reflected by increasing weight-change norm and decreasing kernel alignment across all tasks (Figure 3).

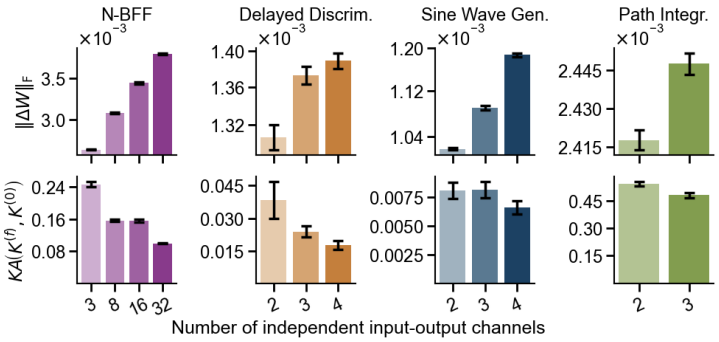

Figure 3: **More complex tasks drive stronger feature learning in RNNs.** Increased input–output dimensionality leads to higher weight-change norms and lower kernel alignment. Error bars indicate ± 1 standard error.

D Feature learning effect for all tasks

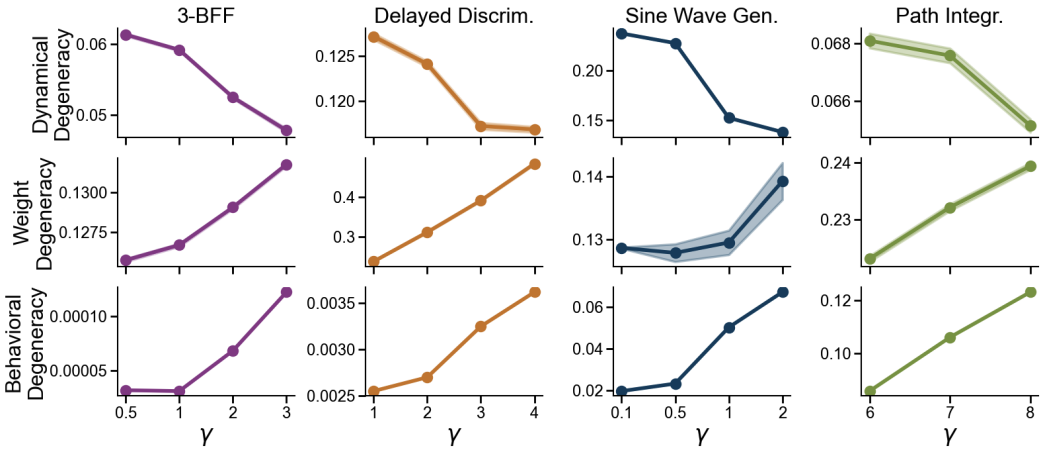

Figure 4: **Stronger feature learning reduces dynamical degeneracy but increases weight and behavioral degeneracy.** Panels show degeneracy at the dynamical, weight, and behavioral levels (top to bottom). Shaded area indicates ± 1 standard error.

E Network size effect for all tasks

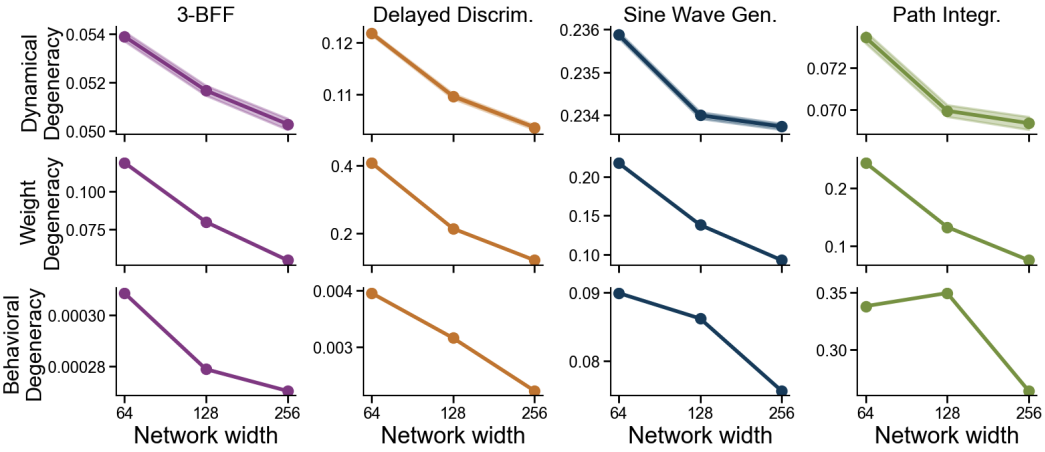

Figure 5: **Larger networks reduce degeneracy across weight, dynamics, and behavior.** Controlling for feature learning strength, wider RNNs yield more consistent solutions across all three levels of analysis. Panels show degeneracy at the dynamical, weight, and behavioral levels (top to bottom). Shaded area indicates ± 1 standard error.

F Task details

F.1 N-Bit Flip Flop

Task Parameter	Value
Probability of flip	0.3
Number of time steps	100

Task Parameter	Value
Number of time steps	60
Max delay	20
Lowest stimulus value	2
Highest stimulus value	10

F.2 Delayed Discrimination

F.3 Sine Wave Generation

Task Parameter	Value
Number of time steps	100
Time step size	0.01
Lowest frequency	1
Highest frequency	30
Number of frequencies	100

F.4 Path Integration

Task Parameter	Value
Number of time steps	100
Maximum speed (v_{max})	0.4
Direction increment std (θ_{std} / ϕ_{std})	$\pi/10$
Speed increment std	0.1
Noise std	0.0001
Mean stop duration	30
Mean go duration	50
Environment size (per side)	10

G Training details

G.1 N-Bit Flip Flop

Training Hyperparameter	Value
Optimizer	Adam
Learning rate	0.001
Learning rate scheduler	None
Max epochs	300
Steps per epoch	128
Batch size	256
Early stopping threshold	0.001
Patience	3
Time constant (μP)	1

G.2 Delayed Discrimination

Training Hyperparameter	Value
Optimizer	Adam
Learning rate	0.001
Learning rate scheduler	CosineAnnealingWarmRestarts
Max epochs	500
Steps per epoch	128
Batch size	256
Early stopping threshold	0.01
Patience	3
Time constant (μP)	0.1

G.3 Sine Wave Generation

Training Hyperparameter	Value
Optimizer	Adam
Learning rate	0.0005
Learning rate scheduler	None
Max epochs	500
Steps per epoch	128
Batch size	32
Early stopping threshold	0.05
Patience	3
Time constant (μP)	1

G.4 Path Integration

Training Hyperparameter	Value
Optimizer	Adam
Learning rate	0.001
Learning rate scheduler	ReduceLROnPlateau
Learning rate decay factor	0.5
Learning rate decay patience	40
Max epochs	1000
Steps per epoch	128
Batch size	64
Early stopping threshold	0.05
Patience	3
Time constant (μP)	0.1

H Memory demand of each task

In this section, we quantify each task's memory demand by measuring how far back in time its inputs influence the next output. Specifically, for each candidate history length h, we build feature vectors

$$\mathbf{s}_t^{(h)} = [\, x_{t-h+1}, \ldots, x_t;\ y_t \,] \ \in\ \mathbb{R}^{h\, d_{\text{in}} + d_{\text{out}}},$$

and **train a two-layer MLP to predict the subsequent target** y_{t+1}. We then evaluate the held-out mean-squared error $\mathrm{MSE}(h)$, averaged over multiple random initializations. We identify the

smallest history length h^* at which the error curve plateaus or has a minimum, and take h^* as the task's intrinsic memory demand.

From the results, we can see that the N-Bits Flip-Flop task requires only one time-step of memory—exactly what's needed to recall the most recent nonzero input in each channel. The Sine Wave Generation task demands two time-steps, reflecting the need to track both phase and direction of change. Path Integration likewise only needs one time-step, since the current position plus instantaneous velocity and heading suffice to predict the next position. Delayed Discrimination is the only memory-intensive task: our method estimates a memory demand of 25 time-steps, which happens to be the time interval between the offset of the first stimulus and the onset of the response period, during which the network needs to first keep track of the amplitude of the first stimulus and then its decision.

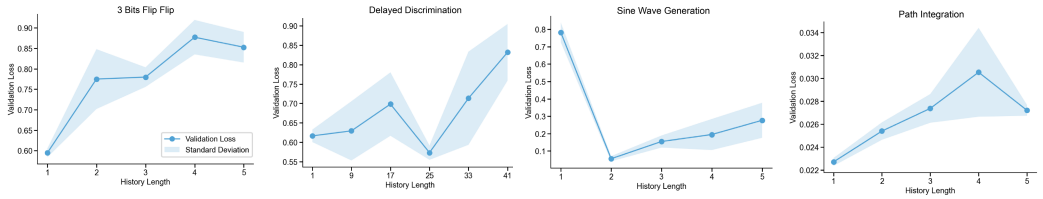

Figure 6: **Memory demand of each task**. The held-out mean-squared error $\mathrm{MSE}(h)$ of a two-layer MLP predictor is plotted against history length h. The intrinsic memory demand h^*, defined by the plateau or minimum of each curve, is 1 for the N-Bits Flip-Flop and Path Integration tasks, 2 for Sine Wave Generation, and 25 for Delayed Discrimination—matching the inter-stimulus delay interval in that task.

I More details on the degeneracy metrics

I.1 Dynamical Degeneracy

Briefly, DSA proceeds as follows: Given two RNNs with hidden states $\mathbf{h}_1(t) \in \mathbb{R}^n$ and $\mathbf{h}_2(t) \in \mathbb{R}^n$, we first generate a delay-embedded matrix, $\mathbf{H}_1$ and $\mathbf{H}_2$ of the hidden states in their original state space. Next, for each delay-embedded matrix, we use Dynamic Mode Decomposition (DMD) [Schmid, 2022] to extract linear forward operators $\mathbf{A}_1$ and $\mathbf{A}_2$ of the two systems' dynamics. Finally, a Procrustes distance between the two matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ is used to quantify the dissimilarity between the two dynamical systems and provide an overall DSA score, defined as:

$$d_{\text{Procrustes}}(\mathbf{A}_1, \mathbf{A}_2) = \min_{\mathbf{Q} \in O(n)} \|\mathbf{A}_1 - \mathbf{Q}\mathbf{A}_2\mathbf{Q}^{-1}\|_F$$

where $\mathbf{Q}$ is a rotation matrix from the orthogonal group $O(n)$ and $\|\cdot\|_F$ is the Frobenius norm. This metric quantifies how dissimilar the dynamics of the two RNNs are after accounting for orthogonal transformations. We quantify Dynamical Degeneracy across many RNNs as the average pairwise distance between pairs of RNN neural-dynamics (hidden-state trajectories).

After training, we extract each network's hidden-state activations for every trial in the training set, yielding a tensor of shape (trials $\times$ time steps $\times$ neurons). We collapse the first two dimensions and yield a matrix of size (trials $\times$ time steps) $\times$ neurons. We then apply PCA to retain the components that explain 99% of the variance to remove noisy and low-variance dimensions of the hidden state trajectories. Next, we perform a grid search over candidate delay lags, with a minimum lag of 1 and a maximum lag of 30, selecting the lag that minimizes the reconstruction error of DSA on the dimensionality reduced trajectories. Finally, we fit DSA with full rank and the optimal lag to these PCA-projected trajectories and compute the pairwise DSA distances between all networks.

I.2 Weight degeneracy

We computed the pairwise distance between the recurrent matrices from different networks using Two-sided Permutation with One Transformation [Schönemann, 1966, Ding et al., 2008] function from the Procrustes Python package [Meng et al., 2022].

J Representational degeneracy

We further quantified solution degeneracy at the representational level—that is, the variability in each network's internal feature space when presented with the same input dataset—using Singular Vector Canonical Correlation Analysis (SVCCA). SVCCA works by first applying singular value decomposition (SVD) to each network's activation matrix, isolating the principal components that capture most of its variance, and then performing canonical correlation analysis (CCA) to find the maximally correlated directions between the two reduced subspaces. The resulting canonical correlations therefore measure how similarly two networks represent the same inputs: high average correlations imply low representational degeneracy (i.e., shared feature subspaces), whereas lower correlations reveal greater divergence in what the models learn. We define the representational degeneracy (labeled as the SVCCA distance below) as

$$d_{\text{repr}}(A_x, A_y) \;=\; 1 \;-\; \text{SVCCA}\big(A_x, A_y\big).$$

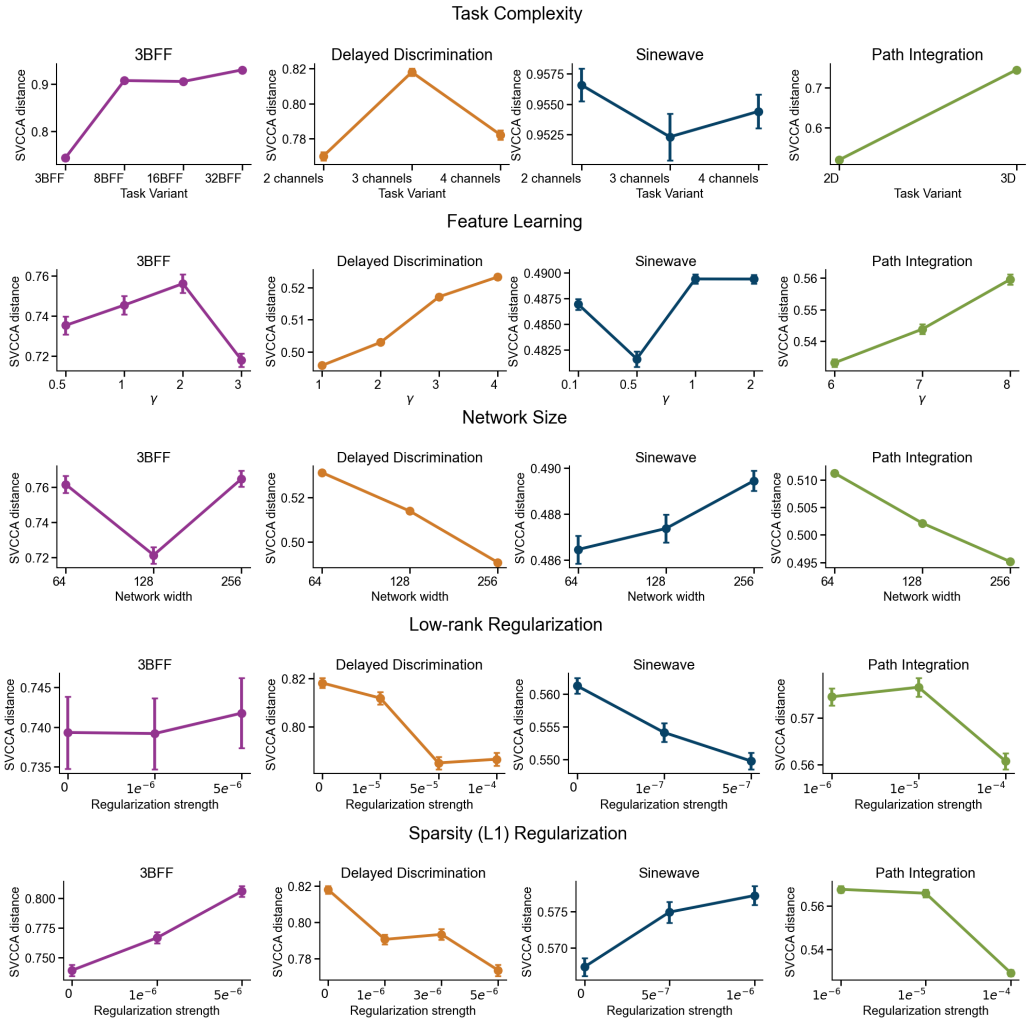

Figure 7: **Representational degeneracy, as measured by the average SVCCA distance between networks, does not necessarily change uniformly as we vary task complexity, feature learning strength, network size, and regularization strength.**

We found that as we vary the four factors that robustly control the dynamical degeneracy across task-trained RNNs, the representational-level degeneracy isn't necessarily constrained by those same

factors in the same way. In RNNs, task-relevant computations are implemented at the level of network's dynamics instead of static representations, and RNNs that implement similar temporal dynamics can have disparate representaional geometry. Therefore, it is expected that task complexity, learning regime, and network size change the task-relevant computations learned by the networks by affecting their neural dynamics instead of representations. DSA captures the dynamical aspect of the neural computation by fitting a forward operator matrix A that maps the network's activity at one time step to the next, therefore directly capturing the temporal evolution of neural activities. By contrast, SVCCA aligns the principal subspaces of activation vectors at each time point but treats those vectors as independent samples—it never examines how one state evolves into the next. As a result, SVCCA measures only static representational similarity and cannot account for the temporal dependencies that underlie RNN computations. Nonetheless, we expect SVCCA might be more helpful in measuring the solution degeneracy in feedforward networks.

K Detailed characterization of OOD generalization performance

In addition to showing the behavioral degeneracy in the main text, here we provide a more detailed characterization of the OOD behavior of networks by showing the mean versus standard deviation, and the distribution of the OOD losses.

K.1 Changing task complexity

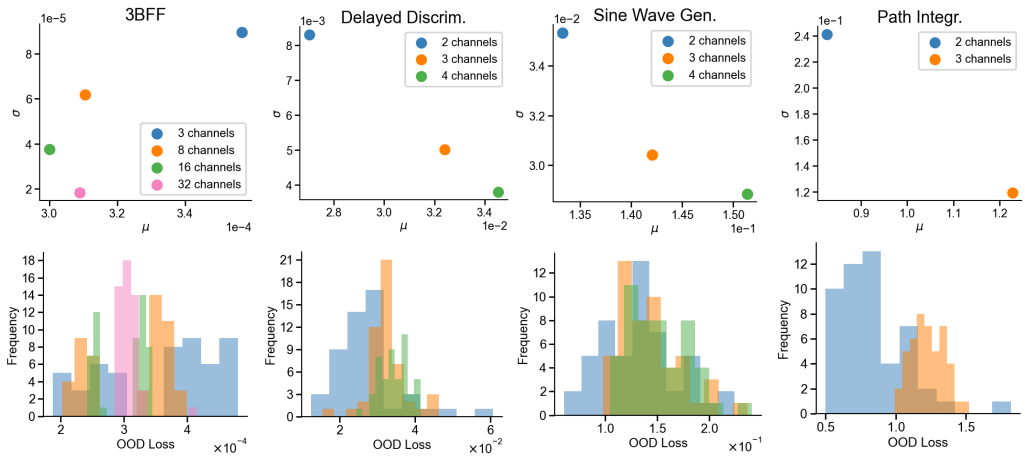

Figure 8: Detailed characterization of the OOD performance of networks while changing task complexity.

K.2 Changing feature learning strength

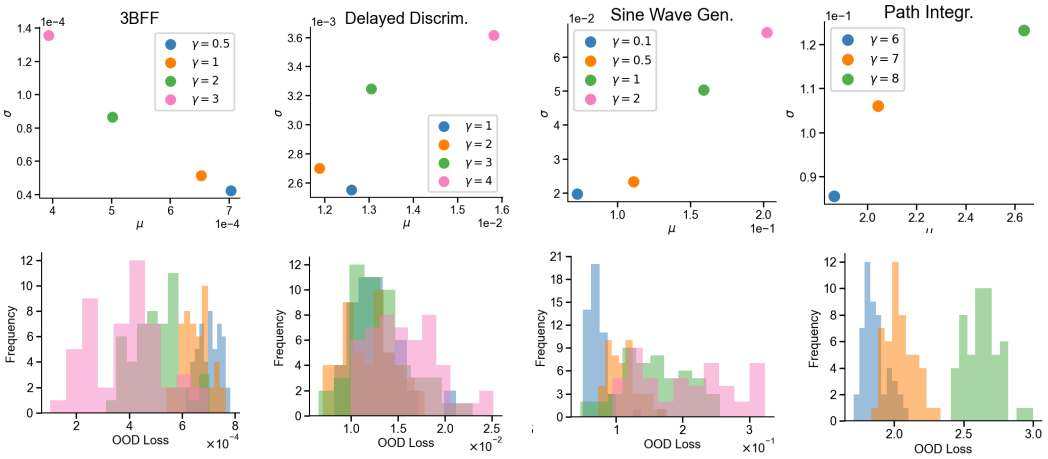

Figure 9: Detailed characterization of the OOD performance of networks while changing feature learning strength. Across Delayed Discrimination, Sine Wave Generation, and Path Integration tasks, networks trained with larger γ – and thus undergoing stronger feature learning – exhibit higher mean OOD generalization loss together with higher variability, potentially reflecting overfitting to the training task.

K.3 Changing network size

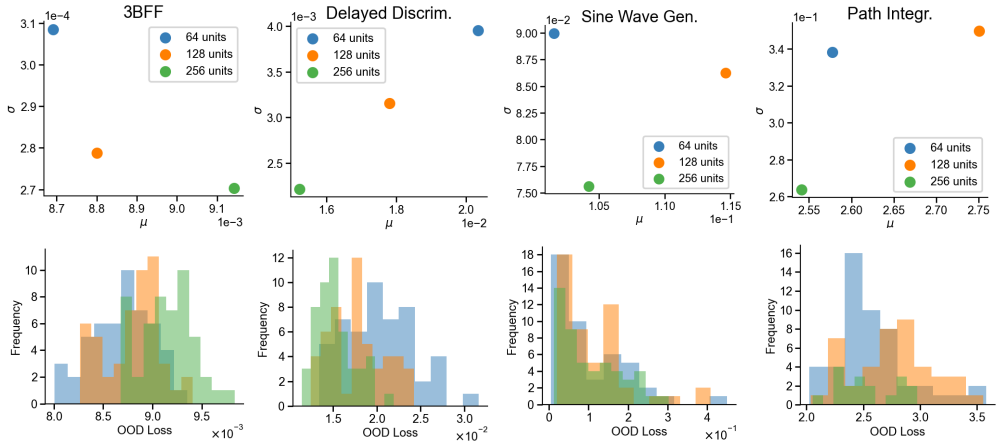

Figure 10: Detailed characterization of the OOD performance of networks while changing network size.

K.4 Changing regularization strength

K.4.1 Low-rank regularization

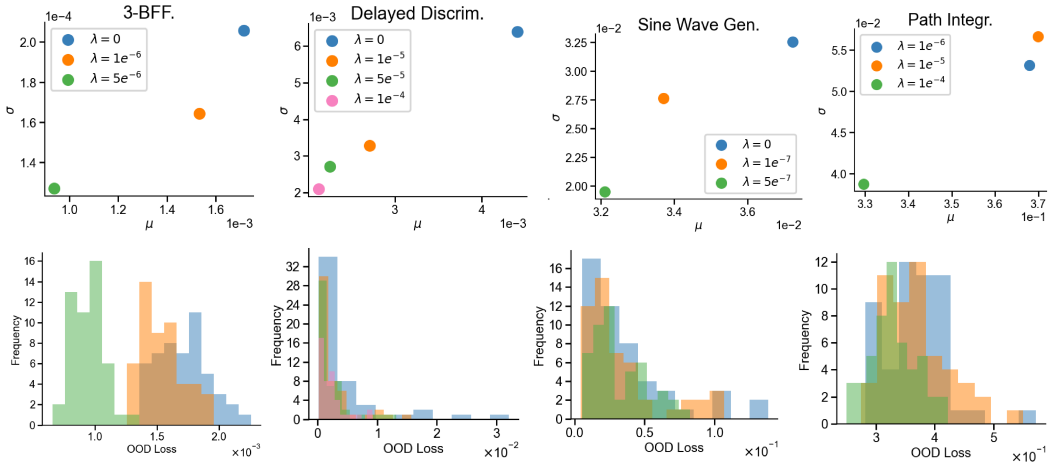

Figure 11: Detailed characterization of the OOD performance of networks while changing low-rank regularization strength.

K.4.2 Sparsity (L1) regularization

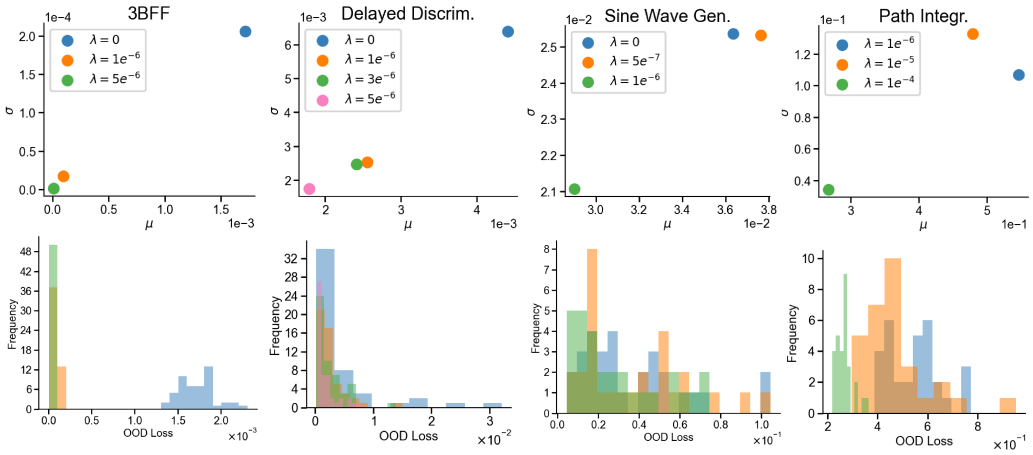

Figure 12: Detailed characterization of the OOD performance of networks while changing sparsity (L1) regularization strength.

L A short introduction to Maximal Update Parameterization (μP)

Under the NTK parametrization, as the network width goes to infinity, the network operates in the *lazy* regime, where its functional evolution is well-approximated by a first-order Taylor expansion around the initial parameters [Jacot et al., 2018, Lee et al., 2019, Chizat et al., 2019, Woodworth et al., 2020]. In this limit feature learning is suppressed and training dynamics are governed by the fixed Neural Tangent Kernel (NTK).

To preserve non-trivial feature learning at large width, the *Maximal Update Parametrization* (μP) rescales both the weight initialisation and the learning rate. μP keeps three quantities *width-invariant* at every layer—(i) the norm/variance of activations (ii) the norm/variance of the

gradients, and (iii) the parameter updates applied by the optimizer [Yang et al., 2022, 2023, Geiger et al., 2020, Bordelon and Pehlevan, 2022].

For recurrent neural networks, under Stochastic Gradient Descent (SGD), the network output, initialization, and learning rates are scaled as

$$f = \frac{1}{\gamma_0 N}\, \vec{w} \cdot \phi\big(h\big), \tag{1}$$

$$\partial_t h = -\, h \;+\; \frac{1}{\sqrt{N}}\; J\, \phi\big(h\big), \qquad J_{ij} \sim \mathcal{N}(0,1), \tag{2}$$

$$\eta_{\text{SGD}} = \eta_0\, \gamma_0^2\, N. \tag{3}$$

Under Adam optimizer, the network output, initialization, and learning rates are scaled as

$$f = \frac{1}{\gamma_0 N}\, \vec{w} \cdot \phi\big(h\big), \tag{4}$$

$$\partial_t h = -\, h \;+\; \frac{1}{N}\; J\, \phi\big(h\big), \qquad J_{ij} \sim \mathcal{N}(0,N), \tag{5}$$

$$\eta_{\text{Adam}} = \eta_0\, \gamma_0. \tag{6}$$

M Theoretical relationship between parameterizations

We compare two RNN formalisms used in different parts of the main manuscript: a standard discrete-time RNN trained with fixed learning rate and conventional initialization, and a μP-style RNN trained with leaky integrator dynamics and width-aware scaling.

In the standard discrete-time RNN, the hidden activations are updated as

$$h(t+1) = \phi\big(W_h h(t) + W_x x(t)\big),$$

In μP RNNs, the hidden activations are updated as

$$h(t+1) - h(t) = \tau\big(-h(t) + \frac{1}{N} J\phi(h(t)) + Ux(t)\big)$$

When $\tau = 1$,

$$h(t+1) - h(t) = -h(t) + \frac{1}{N} J\phi(h(t)) + Ux(t)$$

$$h(t+1) = \frac{1}{N} J\phi(h(t)) + Ux(t)$$

Aside from the overall scaling factor, the difference between the two parameterizations lies in the placement of the non-linearity:

- **Standard RNN:** ϕ is applied *post-activation*, i.e. after the recurrent and input terms are linearly combined,
- **μP RNN:** ϕ is applied *pre-activation*; i.e. before the recurrent weight matrix, so the hidden state is first non-linearized and then linearly combined

Miller and Fumarola [Miller and Fumarola, 2012] demonstrated that two classes of continuous-time firing-rate models which differ in their placement of the non-linearity are mathematically equivalent under a change of variables:

$$\text{v-model} \quad \tau \frac{dv}{dt} = -v + \tilde{I}(t) + W f(v)$$

$$\text{r-model:} \quad \tau \frac{dr}{dt} = -r + f(Wr + I(t))$$

with equivalence holding under the transformation $v(t) = Wr(t) + I(t)$ and $\tilde{I}(t) = I(t) + \tau\frac{dI}{dt}$, assuming matched initial conditions.

Briefly, they show that $Wr + I$ evolves according to the v-equation as follows:

$$
\begin{aligned}
v(t) &= Wr(t) + I(t) \\
\frac{dv}{dt} &= \frac{d}{dt}\big(Wr(t) + I(t)\big) \\
&= W\frac{dr}{dt} + \frac{dI}{dt} \\
&= W\left(\frac{1}{\tau}\left(-r + f(Wr + I)\right)\right) + \frac{dI}{dt} \\
\tau\frac{dv}{dt} &= -Wr + Wf(Wr + I) + \tau\frac{dI}{dt} \\
&= -(v - I) + Wf(v) + \tau\frac{dI}{dt} \\
&= -v + I + \tau\frac{dI}{dt} + Wf(v) \\
\tau\frac{dv}{dt} &= -v + \tilde{I}(t) + Wf(v)
\end{aligned}
$$

This mapping applies directly to RNNs viewed as continuous-time dynamical systems and helps relate v-type μP-style RNNs to standard discrete-time RNNs. It suggests that the μP RNN (in v-type form) and the standard RNN (in r-type form) can be treated as different parameterizations of the same underlying dynamical system when:

- Initialization scales are matched
- The learning rate is scaled appropriately with γ
- Output weight norms are adjusted according to width

In summary, while a theoretical equivalence exists, it is contingent on consistent scaling across all components of the model. In this manuscript, we use the standard discrete-time RNNs due to its practical relevance for task-driven modeling community, while switching to μP to isolate the effect of feature learning and network size. Additionally, we confirm that the feature learning and network size effects on degeneracy hold qualitatively the same in standard discrete-time RNNs, unless where altering network width induces unstable and lazier learning in larger networks (Figure Q and R).

N Verifying larger γ reliably induces stronger feature learning in μP

In μP parameterization, the parameter γ interpolates between lazy training and rich, feature-learning dynamics, without itself being the absolute magnitude of feature learning. Here, we assess feature-learning strength in RNNs under varying γ using two complementary metrics:

Weight-change norm which measures the magnitude of weight change throughout training. A larger weight change norm indicates that the network undergoes richer learning or more feature learning.

$$\frac{\|\mathbf{W}_T - \mathbf{W}_0\|_F}{N},$$

where N is the number of parameters in the weight matrices being compared.

Kernel alignment (KA), which measures the directional change of the neural tangent kernel (NTK) before and after training. A lower KA score corresponds to a larger NTK rotation and thus stronger feature learning.

$$\mathrm{KA}\big(K^{(f)}, K^{(0)}\big) \;=\; \frac{\mathrm{Tr}\big(K^{(f)}K^{(0)}\big)}{\big\|K^{(f)}\big\|_F \,\big\|K^{(0)}\big\|_F}, \qquad K \;=\; \nabla_W \hat{y}^\top \nabla_W \hat{y}.$$

We demonstrate that higher γ indeed amplifies feature learning inside the network.

N.1 N-BFF

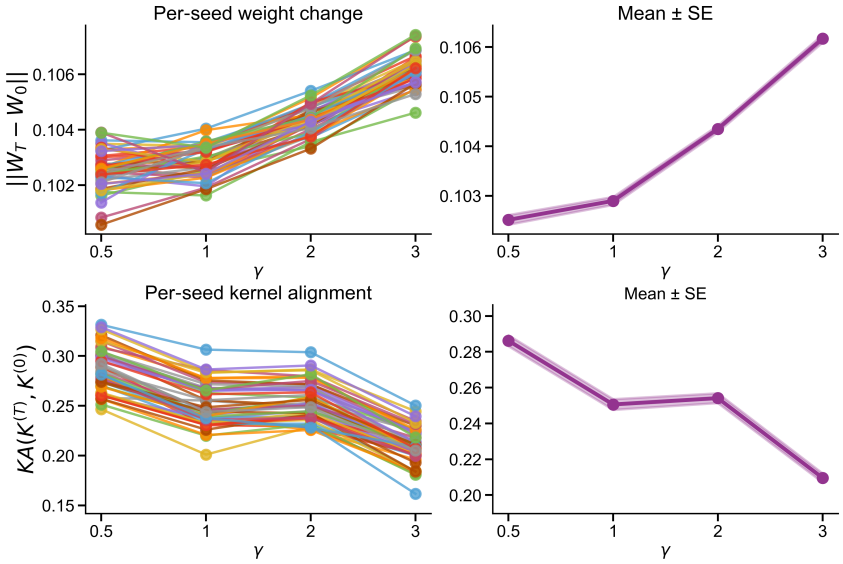

Figure 13: Weight change norm and kernel alignment for networks trained on the 3-Bits Flip Flop task as we vary γ. On the left panels, we show the per-seed metrics where connected dots of the same color are networks of identical initialization trained with different γ. On the right panels, we show the mean and standard error of the metrics across 50 networks. For larger γ, the weights move further from their initializations as shown by the larger weight change norm, and their NTK evolves more distinct from the network's NTK at initialization as shown by the reduced KA. Both indicate stronger feature learning for networks trained under larger γ.

N.2 Delayed Discrimination

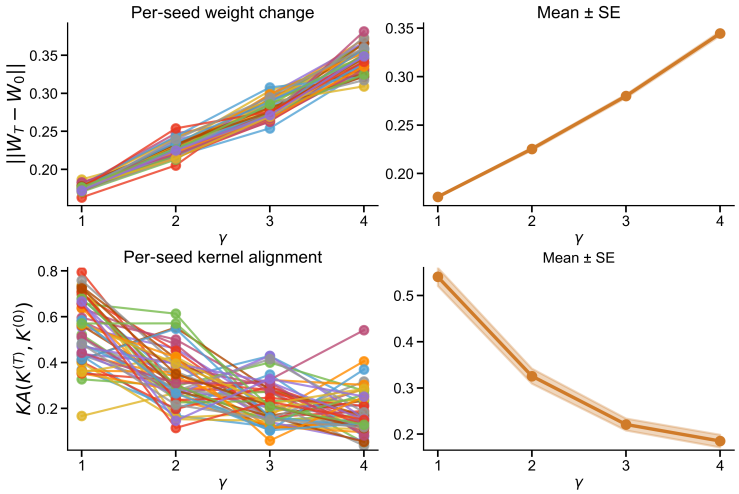

Figure 14: Stronger feature learning for networks trained under larger γ on the Delayed Discrimination task.

N.3 Sine Wave Generation

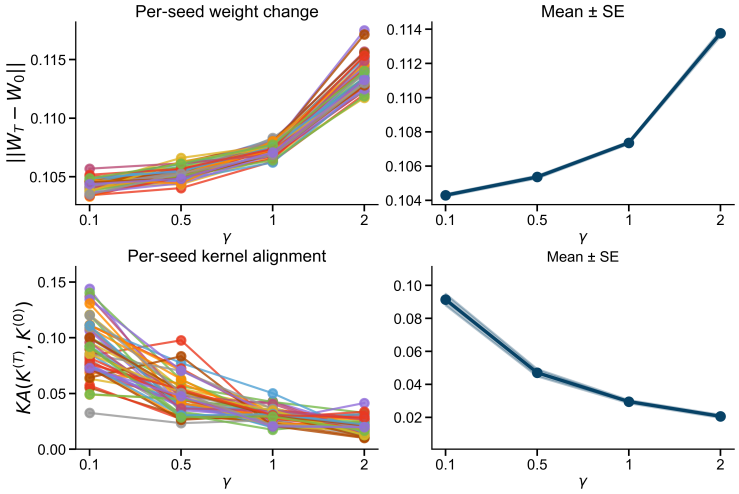

Figure 15: Stronger feature learning for networks trained under larger γ on the Sine Wave Generation task.

N.4 Path Integration

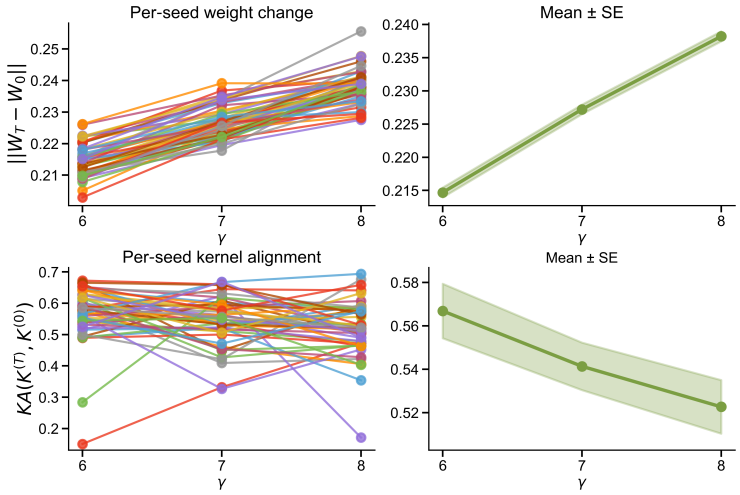

Figure 16: Stronger feature learning for networks trained under larger γ on the Path Integration task.

O Verifying μP reliably controls for feature learning across network width

Here, we only use Kernel Alignment to assess the feature learning strength in the networks since the unnormalized weight-change norm $\|\mathbf{W}_T - \mathbf{W}_0\|_F$ scales directly with matrix size (therefore network size) and there exists no obvious way to normalize across different dimensions. In our earlier analysis where we compared weight-change norms at varying γ, network size remained fixed, so those Frobenius-norm measures were directly comparable. We found that, for all tasks except Delayed Discrimination, the change in mean KA across different network sizes remains extremely small (less than 0.1), which demonstrates that μP parameterization with the same γ has effectively controlled for feature learning strength across network sizes. On Delayed Discrimination, the networks undergo slightly lazier learning for larger network sizes. Nevertheless, we still include Delayed Discrimination in our analyses of solution degeneracy to ensure *our conclusions remain robust even when μP can't perfectly equalize feature-learning strength across widths.* As shown in the main paper, lazier learning regime generally increases dynamical degeneracy; yet, larger networks which exhibit lazier learning in the N-BFF task actually display lower dynamical degeneracy. This reversed trend confirms that the changes in solution degeneracy arise from network size itself, not from residual variation in feature learning strength.

O.1 N-BFF

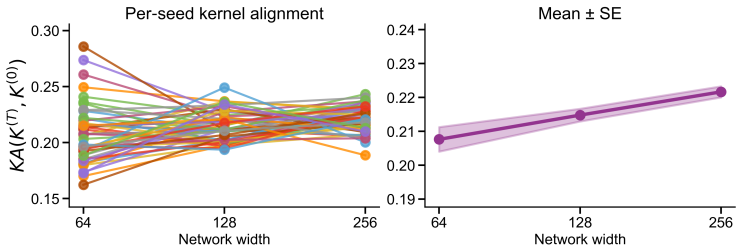

Figure 17: Kernel alignment (KA) for different network width on the 3 Bits Flip-Flop task. (Lower KA implies more feature learning.)

O.2 Delayed Discrimination

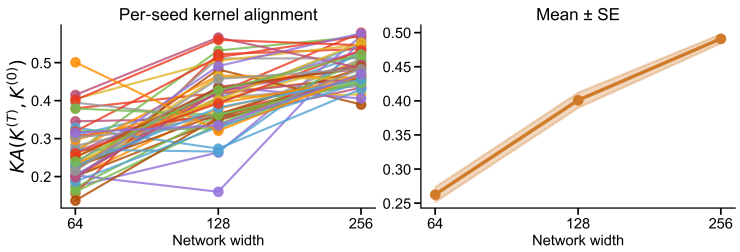

Figure 18: Kernel alignment for different network width on the Delayed Discrimination task.

O.3 Sine Wave Generation

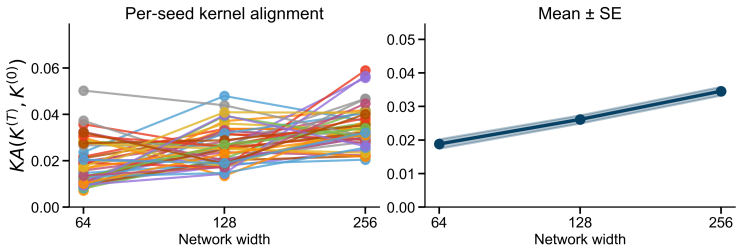

Figure 19: Kernel alignment for different network width on the Sine Wave Generation task.

O.4 Path Integration

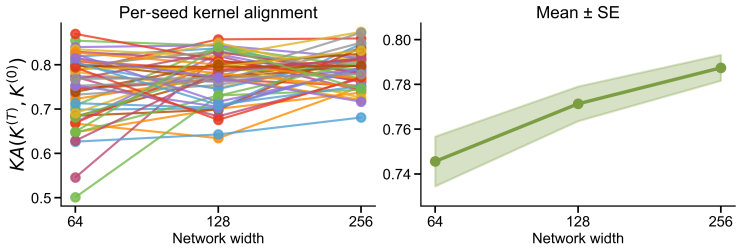

Figure 20: Kernel alignment for different network width on the Path Integration task.

P Regularization's effect on degeneracy for all tasks

In addition to showing regularization's effect on degeneracy in Delayed Discrimination task in the main paper, here we show that heavier low-rank regularization and sparsity regularization also reliably reduce solution degeneracy across neural dynamics, weights, and OOD behavior in the other three tasks.

P.1 Low-rank regularization

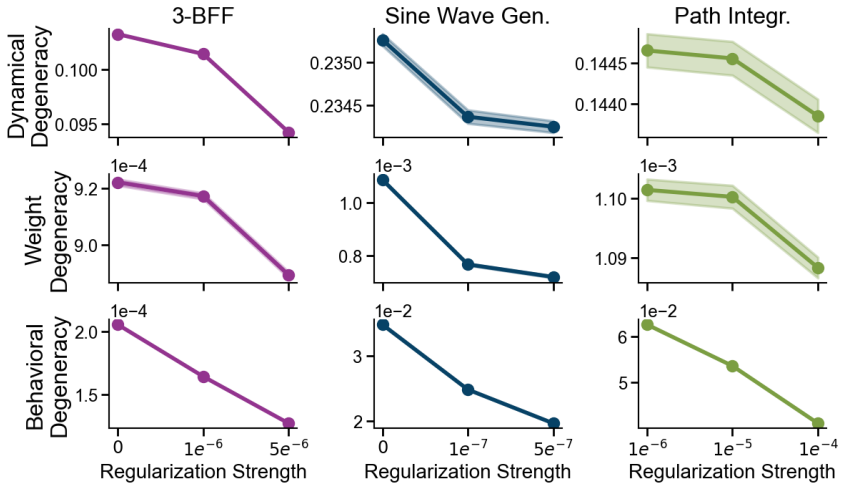

Figure 21: Low-rank regularization reduces degeneracy across neural dynamics, weight, and OOD behavior on the N-BFF, Sinewave Generation, and Path Integration task.

P.2 Sparsity regularization

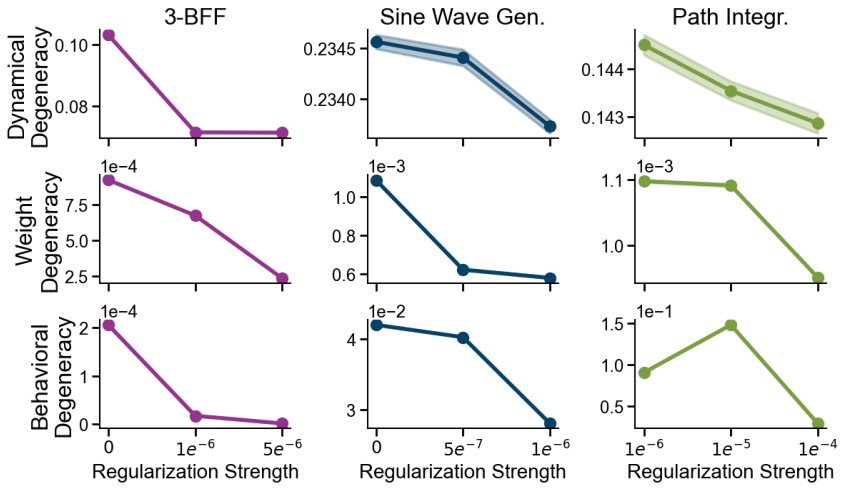

Figure 22: Sparsity regularization reduces degeneracy across neural dynamics, weight, and OOD behavior on the N-BFF, Sinewave Generation, and Path Integration task.

Q Test feature learning effect on degeneracy in standard parameterization

While μP lets us systematically vary feature-learning strength to study its impact on solution degeneracy, we confirm that the same qualitative pattern appears in *standard* discrete-time RNNs: stronger feature learning **lowers dynamical degeneracy** and **raises weight degeneracy** (Figure 23).

To manipulate feature-learning strength in these ordinary RNNs we applied the γ**-trick**—scaling the network's outputs by γ—and multiplied the learning rate by the same factor. With width fixed, these two operations replicate the effective changes induced by μP. Figure 24 shows that this combination reliably tunes feature-learning strength. Besides weight-change norm and kernel alignment, we also report **representation alignment (RA)**, giving a more fine-grained view of how much the learned features deviate from their initialization [Liu et al., 2023]. Representation alignment is the directional change of the network's represenational dissimilarity matrix before and after training, and is defined by

$$\mathrm{RA}\big(R^{(T)}, R^{(0)}\big) := \frac{\mathrm{Tr}\big(R^{(T)} R^{(0)}\big)}{\|R^{(T)}\|\,\|R^{(0)}\|}, \qquad R \;:=\; H^\top H,$$

A lower RA means more change in the network's representation of inputs before and after training, and indicates stronger feature learning.

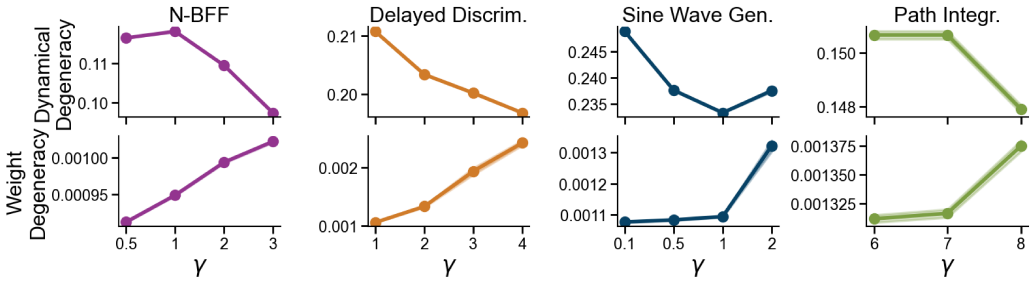

Figure 23: Stronger feature learning reliably decreases dynamical degeneracy while increasing weight degeneracy in standard discrete-time RNNs.

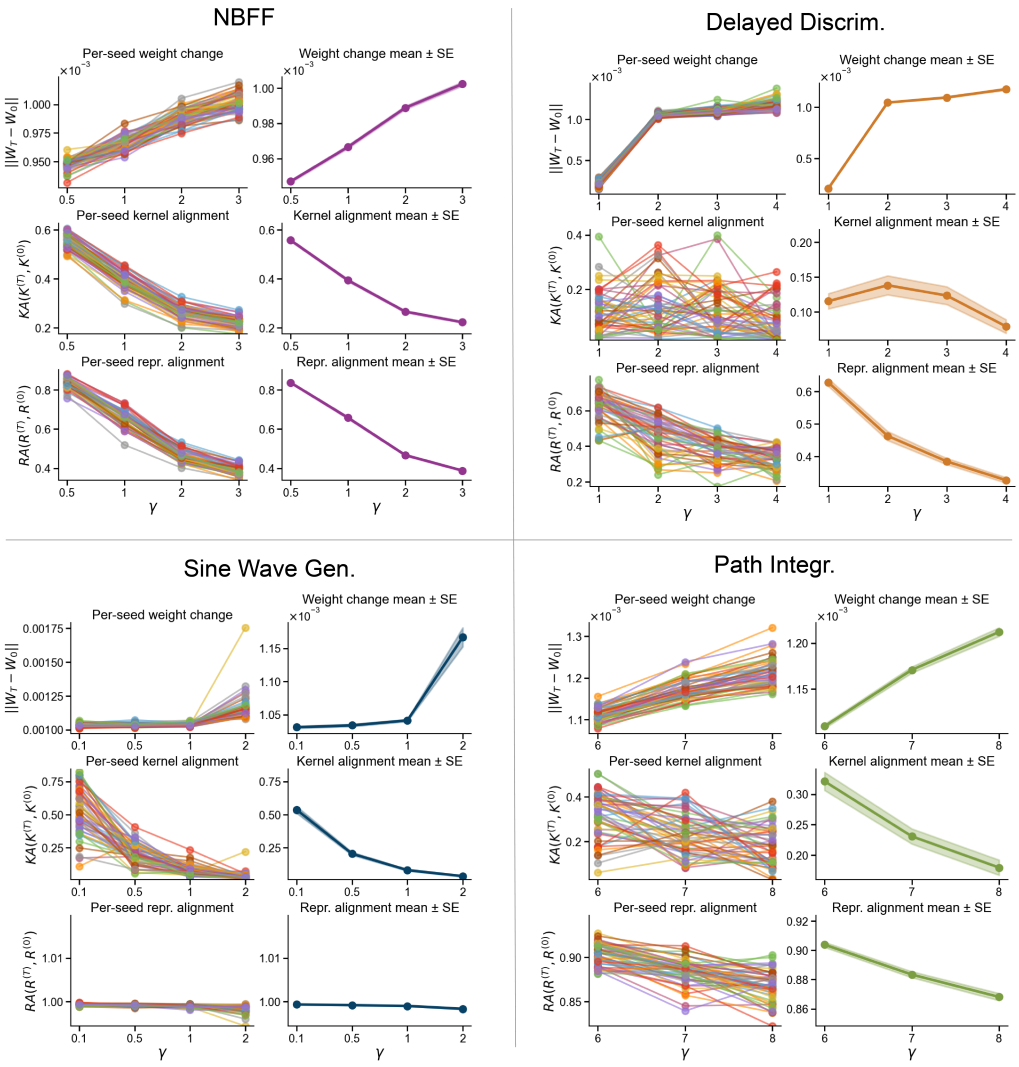

Figure 24: Larger γ reliably induces stronger feature learning in standard discrete-time RNNs.

R Test network size effect on degeneracy in standard parameterization

When we vary network width, both the standard parameterization andμP parameterization display the same overall pattern: **larger networks exhibit lower dynamical and weight degeneracy**. An exception arises in the 3BFF task, where feature learning becomes unstable and collapses in the wider models. In that setting we instead see *higher* dynamical degeneracy, which we suspect because the feature learning effect (lazier learning leads to higher dynamical degeneracy) dominates the network size effect.

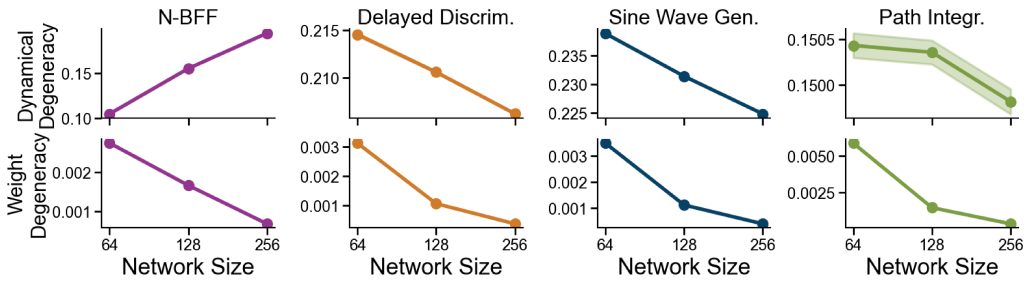

Figure 25: Larger network sizes lead to lower dynamical and weight degeneracy, except in the case where feature learning is unstable across width (in N-BFF).

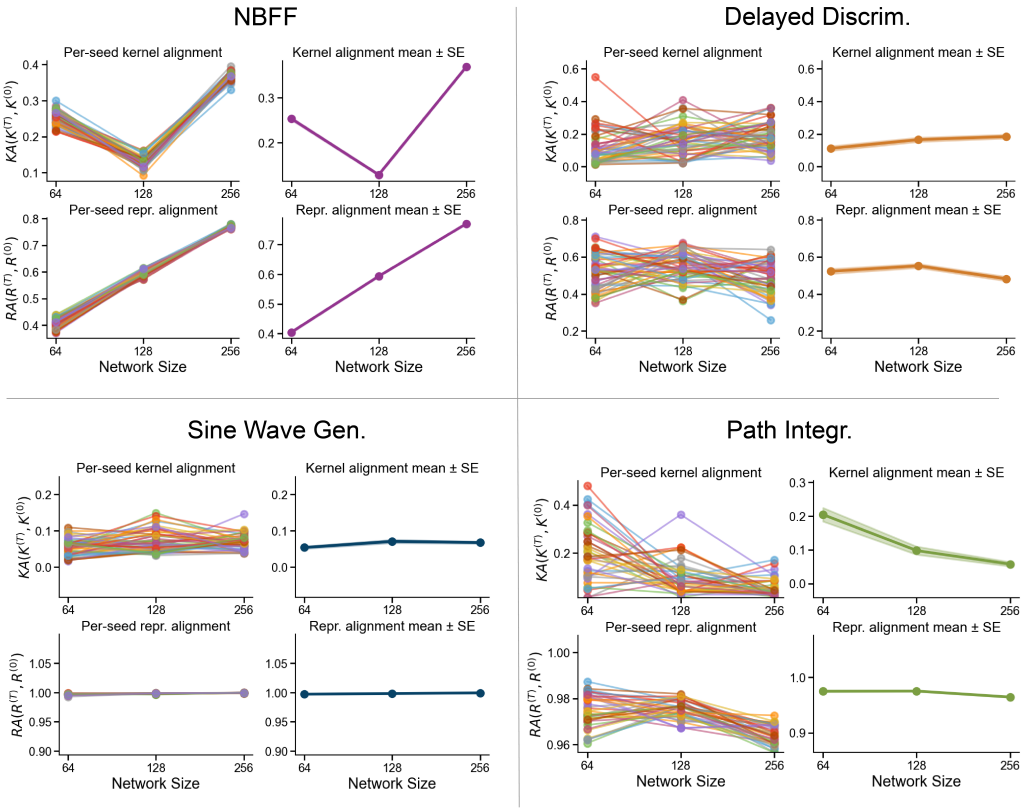

Figure 26: When changing network width in standard discrete-time RNNs, feature learning strength remains stable across width except in N-BFF, where notably lazier learning happens in the widest network.

S Disclosure of compute resources

In this study, we conducted 50 independent training runs on each of four tasks, systematically sweeping four factors that modulate solution degeneracy—task complexity (15 experiments), learning regime (15 experiments), network size (12 experiments), and regularization strength (26 experiments), resulting in a total of 3400 networks. Each experiment was allocated 5 NVIDIA V100/A100 GPUs, 32 CPU cores, 256 GB of RAM, and a 4-hour wall-clock limit, for a total compute cost of approximately 68 000 GPU-hours.

References

Blake Bordelon and Cengiz Pehlevan. Self-Consistent Dynamical Field Theory of Kernel Evolution in Wide Neural Networks, October 2022. URL http://arxiv.org/abs/2205.09653. arXiv:2205.09653 [stat].

Léon Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32:2938–2950, 2019.

Chris Ding, Tao Li, and Michael I. Jordan. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. In *Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM '08)*, pages 183–192. IEEE, 2008. doi: 10.1109/ICDM.2008.130.

Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020 (11):113301, 2020. doi: 10.1088/1742-5468/abc4de.

Thomas George, Guillaume Lajoie, and Aristide Baratin. Lazy vs hasty: Linearization in deep networks impacts learning schedule based on example difficulty. *arXiv preprint arXiv:2209.09658*, 2022. URL https://arxiv.org/abs/2209.09658.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Jaehoon Lee, Yuval Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in Neural Information Processing Systems*, volume 32, pages 8572–8583, 2019.

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. *arXiv preprint arXiv:2310.08513*, 2023. doi: 10.48550/arXiv.2310.08513. URL https://arxiv.org/abs/2310.08513.

Fanwang Meng, Michael G. Richer, Alireza Tehrani, Jonathan La, Taewon David Kim, P. W. Ayers, and Farnaz Heidar-Zadeh. Procrustes: A python library to find transformations that maximize the similarity between matrices. *Computer Physics Communications*, 276:108334, 2022. doi: 10.1016/j.cpc.2022.108334. URL https://www.sciencedirect.com/science/article/pii/S0010465522000522.

Kenneth D Miller and Francesco Fumarola. Mathematical equivalence of two common forms of firing rate models of neural networks. *Neural computation*, 24(1):25–31, 2012.

Peter J Schmid. Dynamic mode decomposition and its variants. *Annual Review of Fluid Mechanics*, 54(1):225–254, 2022.

Peter H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, Mar 1966. doi: 10.1007/BF02289451.

Bryan Woodworth, Suriya Gunasekar, Jason D. Lee, Nathan Srebro, Srinadh Bhojanapalli, Rina Khanna, Aaron Chatterji, and Martin Jaggi. Kernel and rich regimes in deep learning. *Journal of Machine Learning Research*, 21(243):1–48, 2020.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022. doi: 10.48550/arXiv.2203.03466. Accepted at NeurIPS 2021.

Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks. *arXiv preprint arXiv:2310.02244*, 2023. doi: 10.48550/arXiv.2310.02244. Accepted at ICLR 2024.