

1	<b>Appendix Contents</b>	
2	<b>A Task details</b>	<b>2</b>
3	<b>B Training details</b>	<b>3</b>
4	<b>C Task performance of trained networks</b>	<b>4</b>
5	<b>D Memory demand of each task</b>	<b>5</b>
6	<b>E Solution degeneracy in chaotic RNNs</b>	<b>6</b>
7	<b>F Additional details on the degeneracy metrics</b>	<b>7</b>
8	<b>G Representational degeneracy</b>	<b>8</b>
9	<b>H Dense sweep on feature learning, network width, and regularization strength</b>	<b>10</b>
10	<b>I Detailed characterization of OOD generalization performance</b>	<b>10</b>
11	<b>J A short introduction to Maximal Update Parameterization (<math>\mu P</math>)</b>	<b>12</b>
12	<b>K Theoretical relationship between parameterizations</b>	<b>13</b>
13	<b>L Verifying larger <math>\gamma</math> reliably induces stronger feature learning in <math>\mu P</math></b>	<b>15</b>
14	<b>M Verifying <math>\mu P</math> reliably controls for feature learning across network width</b>	<b>17</b>
15	<b>N Regularization's effect on degeneracy for all tasks</b>	<b>19</b>
16	<b>O Test feature learning effect on degeneracy in standard parameterization</b>	<b>20</b>
17	<b>P Test network size effect on degeneracy in standard parameterization</b>	<b>22</b>
18	<b>Q Disclosure of compute resources</b>	<b>23</b>

## 19 A Task details

### 20 A.1 N-Bit Flip Flop

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

### 21 A.2 Delayed Discrimination

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

### 22 A.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

### 23 A.4 Path Integration

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

## 24 B Training details

### 25 B.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 ( $\mu P$ )	1

### 26 B.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 ( $\mu P$ )	0.1

### 27 B.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 ( $\mu P$ )	1

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 ( $\mu P$ )	0.1

## 29 C Task performance of trained networks

30 In all experiments, we train networks until they reach a **near-asymptotic, task-specific mean-**  
 31 **squared error (MSE) threshold** (0.001 for N-BFF, 0.01 for Delayed Discrimination, and 0.05 for  
 32 Sine-Wave Generation and Path Integration), after which we allow a patience period of 3 epochs and  
 33 stop training to measure degeneracy. This early-stopping criterion ensures that networks trained on  
 34 the same task/condition achieve comparable final losses before any degeneracy analysis.

35 To quantify the residual variation, we report the coefficient of variation (CV) of the final training loss  
 36 across seeds for each condition, expressed as % of the mean. Header labels match the x-axis levels  
 37 used in the main-text figures. Final losses cluster tightly near small values of the loss threshold, so  
 38 even a double-digit CV translates to very small absolute variation. For example, a 10% CV at an  
 39 MSE of 0.001 implies an s.d. of  $10^{-4}$ ; at 0.01 it's  $10^{-3}$ . Additionally, the networks converged well  
 40 on a global scale. Across our experiments, the mean MSE after training is under 2% of the mean  
 41 MSE at initialization, indicating that training has converged well. Individual values: 0.059% (N-  
 42 BFF), 1.6% (Delayed Discrimination), 0.32% (Sine-Wave Generation), 0.94% (Path Integration).  
 43 CV can look large when the mean is tiny (the denominator is small). For example, a 16% CV on  
 44 Sine-Wave Generation task corresponds to 0.05% of the initialization loss, which is consistent with  
 45 minor differences due to the stochastic gradients rather than under-training.

46 These variability values are also not monotonic in any factor and sometimes move opposite to the  
 47 degeneracy trends, arguing against a loss-dispersion confound to solution degeneracy.

Table 1: Coefficient of variation (CV) of the final training loss across 50 networks for each task complexity level.

Task Complexity	Level 1	Level 2	Level 3	Level 4
N-BFF	6.30%	4.60%	9.30%	3.50%
Delayed Discrim.	15.90%	8.40%	9.50%	—
Sine Wave Gen.	9.94%	9.20%	8.70%	—
Path Integr.	9.16%	2.85%	—	—

Table 2: Coefficient of variation (CV) of the final training loss across 50 networks for each feature learning strength ( $\gamma$ )

Feature Learning Strength	$\gamma_1$	$\gamma_2$	$\gamma_3$	$\gamma_4$
N-BFF	9.70%	9.10%	13.40%	11.70%
Delayed Discrim.	8.70%	12.60%	11.70%	12.30%
Sine Wave Gen.	3.50%	3.90%	10.90%	11.70%
Path Integr.	5.40%	5.20%	6.20%	—



Table 3: Coefficient of variation (CV) of the final training loss across 50 networks for each network width.

Network Width	64 units	128 units	256 units
N-BFF	3.80%	4.20%	3.50%
Delayed Discrim.	3.30%	3.00%	3.20%
Sine Wave Gen.	17.80%	16.60%	16.40%
Path Integr.	5.10%	5.40%	5.90%

Table 4: Coefficient of variation (CV) of the final training loss across 50 networks for each L1 regularization strength.

L1 Regularization	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$
N-BFF	2.10%	6.90%	1.10%	—
Delayed Discrim.	15.90%	14.50%	16.70%	14.90%
Sine Wave Gen.	10.40%	11.10%	11.10%	—
Path Integr.	9.00%	7.10%	3.00%	—

Table 5: Coefficient of variation (CV) of the final training loss across 50 networks for each rank regularization strength.

Rank Regularization	$\lambda_1$	$\lambda_2$	$\lambda_3$	$\lambda_4$
N-BFF	2.10%	7.20%	4.30%	—
Delayed Discrim.	15.90%	16.90%	13.60%	12.10%
Sine Wave Gen.	13.90%	14.30%	15.90%	—
Path Integr.	7.70%	7.90%	6.70%	—

## 48 D Memory demand of each task

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

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

51 and **train a two-layer MLP to predict the subsequent target**  $y_{t+1}$ . We then evaluate the held-  
52 out mean-squared error  $\text{MSE}(h)$ , averaged over multiple random initializations. We identify the  
53 smallest history length  $h^*$  at which the error curve plateaus or has a minimum, and take  $h^*$  as the  
54 task’s intrinsic memory demand.

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

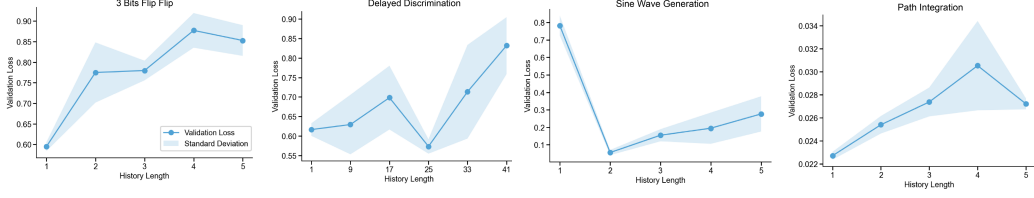


Figure 1: **Memory demand of each task.** The held-out mean-squared error  $MSE(h)$  of a two-layer MLP predictor is plotted against history length  $h$ . The intrinsic memory demand  $h^*$ , defined by the plateau or minimum of each curve, is 1 for the N-Bits Flip-Flop and Path Integration tasks, 2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

## 64 E Solution degeneracy in chaotic RNNs

65 Our original task suite comprises neuroscience-motivated tasks that produce stable-attractors: fixed-  
 66 point (N-Bit Flip Flop, Delayed Discrimination), limit cycle (Sine Wave Generation), and attractor  
 67 manifold (Path Integration). To further demonstrate that the observed effect of the four factors on  
 68 degeneracy extend to RNNs with chaotic activity, here we add a chaotic attractor task and verified  
 69 that the effects of all four factors on dynamical and weight degeneracy are consistent with Table 1.

70 **Lorenz 96 Attractor Dataset** We simulated trajectories from the Lorenz 96 dynamical system  
 71 [Lorenz, 1996], defined by

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \quad i = 1, \dots, N,$$

72 with cyclic boundary conditions  $x_{-1} = x_{N-1}$ ,  $x_0 = x_N$ ,  $x_{N+1} = x_1$ . The external forcing  
 73 parameter was set to  $F = 8.0$ , a standard choice that induces chaotic dynamics.

74 To generate the dataset, we numerically integrated the system for  $N = 16, 24$ , and 32 dimensions.  
 75 Each simulation used a time step of  $\Delta t = 0.01$  and produced 15000 time points after discarding an  
 76 initial transient of 1000 steps to remove non-stationary behavior. Initial conditions were sampled as  
 77 small random perturbations around the fixed point  $x_i = F$ :

$$x_i(0) = F + 0.1 \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, 1).$$

78 For each condition, we trained 50 RNNs on next-step prediction until the networks achieve a near-  
 79 asymptotic MSE loss at 0.0005. After training, the average Lyapunov exponent of RNNs trained on  
 80 the Lorenz 96 attractor with 16 dimensions is  $12.58 \pm 0.74$ , indicating chaotic neural dynamics.

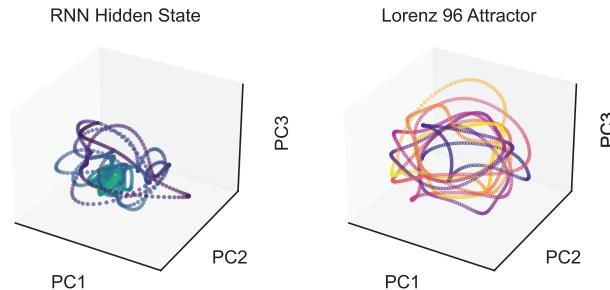


Figure 2: RNN recurrent activities and Lorenz 96 attractor ( $N = 16$ ) trajectories projected onto their respective top 3 principle components.

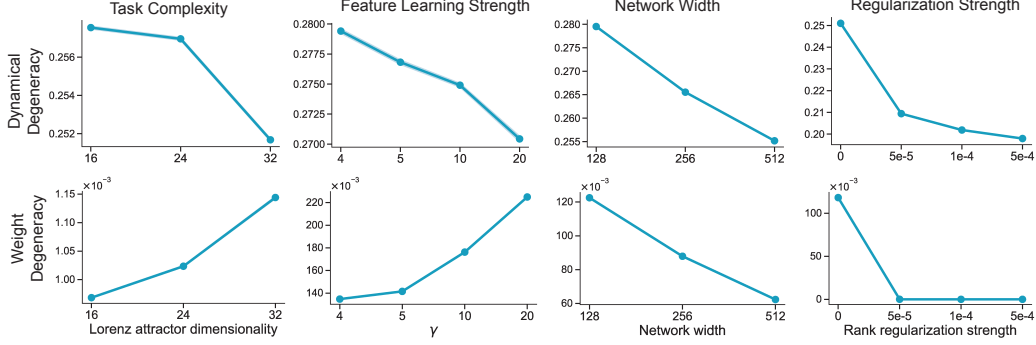


Figure 3: Varying the four factors on Lorenz 96 next-step prediction task changes solution degeneracy across the dynamical and weight level in a way that is consistent with Table 1.

## 81 F Additional details on the degeneracy metrics

### 82 F.1 Dynamical Degeneracy

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

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

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

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

### 101 F.2 Weight degeneracy

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

### 105 F.3 Establishing a null distribution for dynamical and weight degeneracy

106 The DSA scores that we used to define the dynamical degeneracy are inherently context-dependent.  
 107 Specifically, the absolute scale of DSA distances can vary with hyperparameters, particularly the  
 108 delay embedding dimension and the rank used in DSA, because the underlying Procrustes anal-  
 109 ysis between two dynamics matrices relies on the Frobenius norm, which in turn depends on the  
 110 dimension of the dynamic operator being compared. Following the procedure described in the orig-  
 111 inal DSA paper, we fixed these hyperparameters across all groups within each task to ensure fair  
 112 comparison.

113 To further validate the interpretation of DSA values, we computed null distributions of the DSA  
 114 scores, i.e. the distribution of DSA scores when sampled neural activities come from *identical* net-  
 115 works. For each of the 50 networks analyzed in Figure 3B, we randomly split the sampled neural  
 116 trajectories from the same network into two subsets and computed DSA distances between them.  
 117 This procedure yields a distribution of DSA scores expected from *identical dynamical systems*,  
 118 which serves as a reference noise floor. The 95% confidence intervals (CIs) for these null distribu-  
 119 tions are reported below (header labels such as "Level 1" correspond to the task-complexity levels  
 120 shown in the main-text figures). These CIs are, on average, an order of magnitude smaller than the  
 121 computed dynamical degeneracy, indicating that the observed differences between networks trained  
 from different initializations are statistically significant.

Table 6: Establishing a null distribution for dynamical degeneracy: 95% confidence intervals of null DSA scores computed by comparing trajectories from the same network. CIs are on average an order of magnitude smaller than across-network distances.

Task Complexity	Level 1	Level 2	Level 3	Level 4
N-BFF	[0.011, 0.013]	[0.009, 0.016]	[0.008, 0.013]	[0.006, 0.009]
Delayed Discrimination	[0.039, 0.064]	[0.014, 0.076]	[0.025, 0.032]	—
Sine Wave Generation	[0.057, 0.102]	[0.054, 0.081]	[0.048, 0.073]	—
Path Integration	[0.023, 0.037]	[0.010, 0.018]	—	—

122

123 For the PIF distance we used to define weight degeneracy, we similarly established a noise floor  
 124 by *randomly permuting* each trained network’s recurrent weight matrix and computing the distance  
 125 between the permuted and original matrices. The PIF metric reliably recovers a PIF distance of 0  
 126 under this null setting, confirming its robustness to noise and the meaningfulness of the reported  
 127 cross-network PIF differences.

## 128 G Representational degeneracy

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

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

139 We found that as we vary the four factors that robustly control the dynamical degeneracy across task-  
 140 trained RNNs, the representational-level degeneracy isn’t necessarily constrained by those same  
 141 factors in the same way. In RNNs, task-relevant computations are implemented at the level of net-  
 142 work’s dynamics instead of static representations, and RNNs that implement similar temporal dy-  
 143 namics can have disparate representational geometry. Therefore, it is expected that task complexity,  
 144 learning regime, and network size change the task-relevant computations learned by the networks  
 145 by affecting their neural dynamics instead of representations. DSA captures the dynamical aspect  
 146 of the neural computation by fitting a forward operator matrix  $A$  that maps the network’s activity at  
 147 one time step to the next, therefore directly capturing the temporal evolution of neural activities. By  
 148 contrast, SVCCA aligns the principal subspaces of activation vectors at each time point but treats  
 149 those vectors as independent samples—it never examines how one state evolves into the next. As a  
 150 result, SVCCA measures only static representational similarity and cannot account for the temporal  
 151 dependencies that underlie RNN computations. Nonetheless, we expect SVCCA might be more  
 152 helpful in measuring the solution degeneracy in feedforward networks.

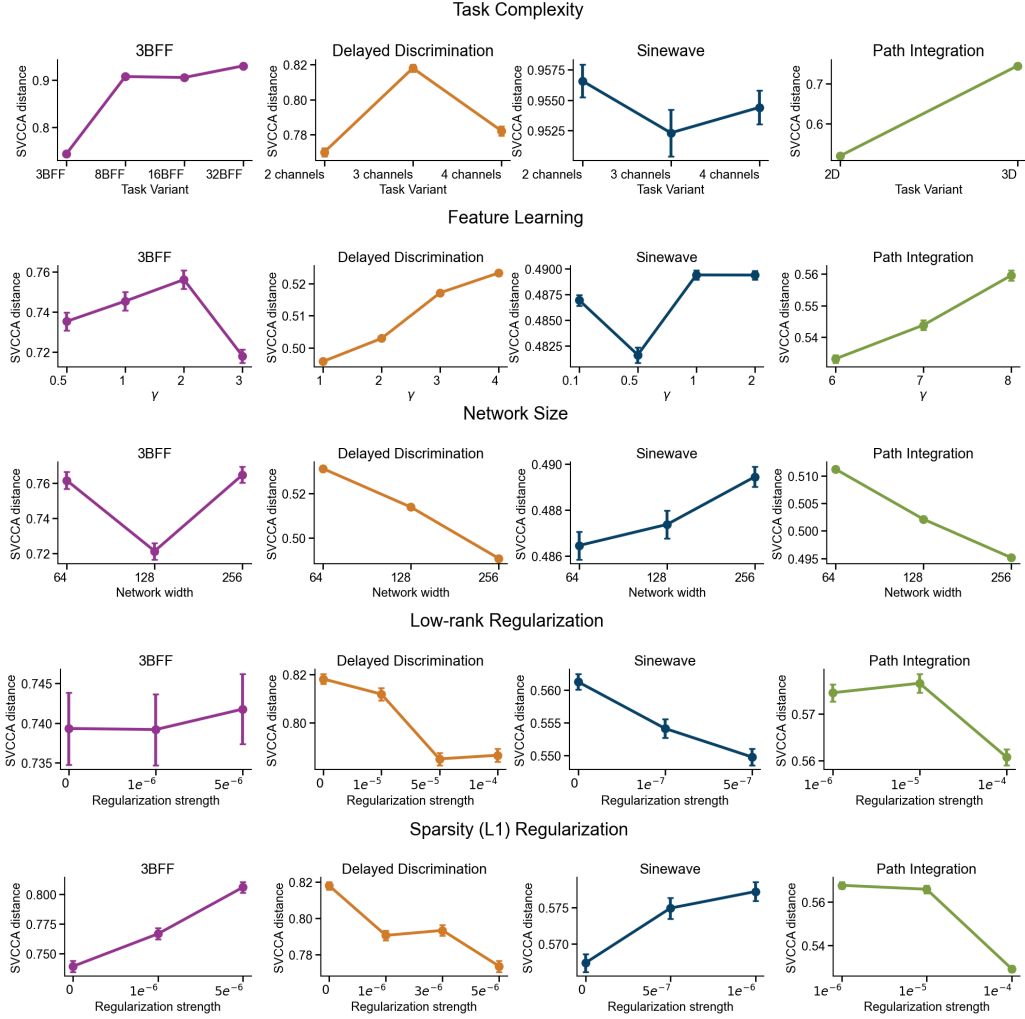


Figure 4: Representational degeneracy, as measured by the average SVCCA distance between networks, does not necessarily change uniformly as we vary task complexity, feature learning strength, network size, and regularization strength.

## H Dense sweep on feature learning, network width, and regularization strength

it is important to know whether the degeneracy trends generalize to intermediate values and beyond the ranges reported in the main paper. To test this, we used 3-BFF as an example and ran a dense sweep both interpolating within and extrapolating beyond the ranges shown in Figs. 3 and 6–8. We demonstrate that cross dynamical and weight levels, the degeneracy trends remain consistent and interpolate smoothly across these intermediate values.

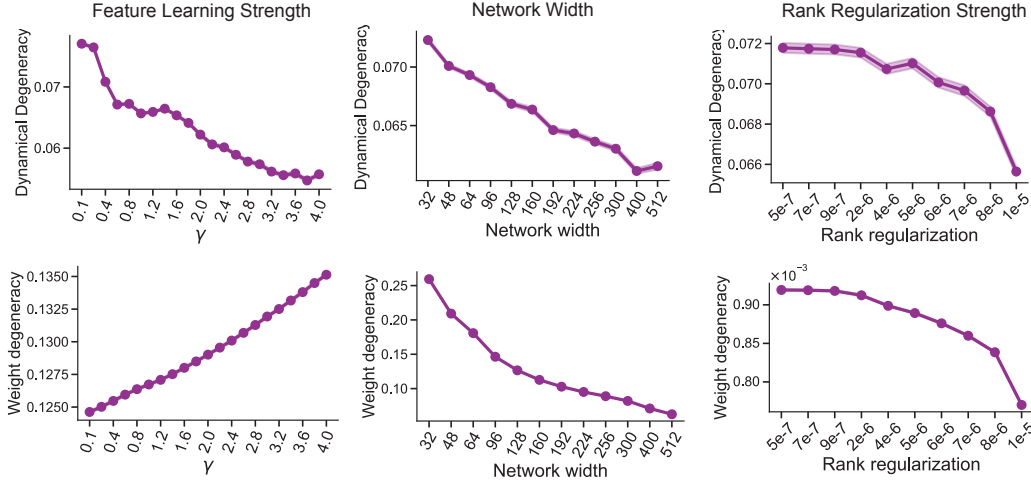


Figure 5: Feature learning, network width, and regularization strength’s effect on degeneracy over a denser sweep of conditions on the 3-Bits Flip Flop task.

## I 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.

### I.1 Changing task complexity

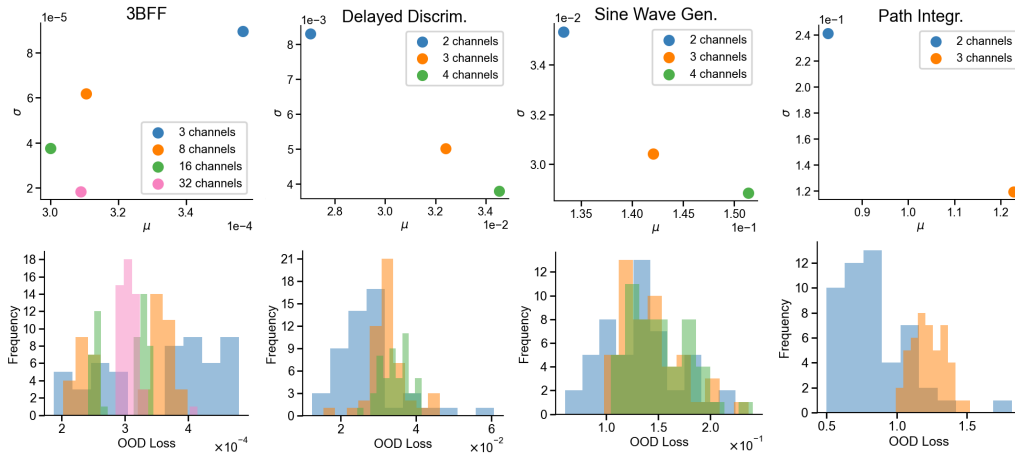


Figure 6: Detailed characterization of the OOD performance of networks while changing task complexity.

## 165 I.2 Changing feature learning strength

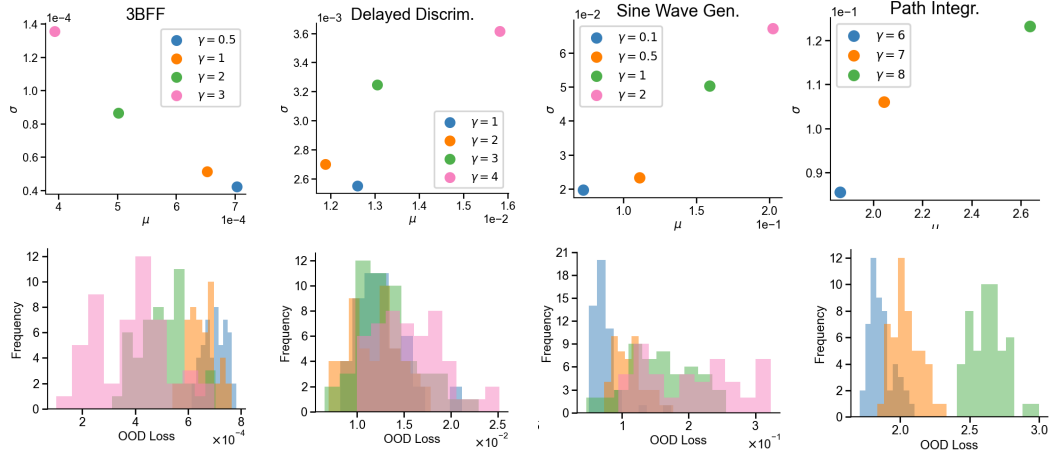


Figure 7: Detailed characterization of the OOD performance of networks while changing feature learning strength. Across Delayed Discrimination, Sine Wave Generation, and Path Integration tasks, networks trained with larger  $\gamma$  – and thus undergoing stronger feature learning – exhibit higher mean OOD generalization loss together with higher variability, potentially reflecting overfitting to the training task.

## 166 I.3 Changing network size

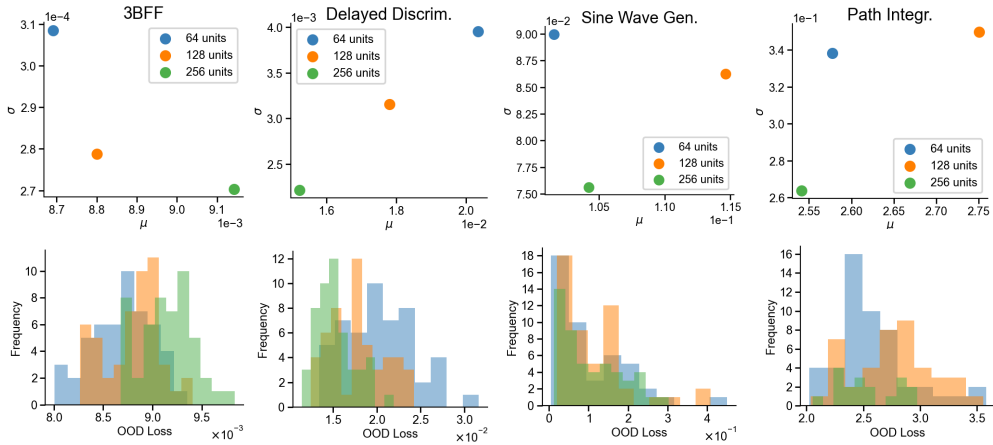


Figure 8: Detailed characterization of the OOD performance of networks while changing network size.

## 167 I.4 Changing regularization strength

### 168 I.4.1 Low-rank regularization

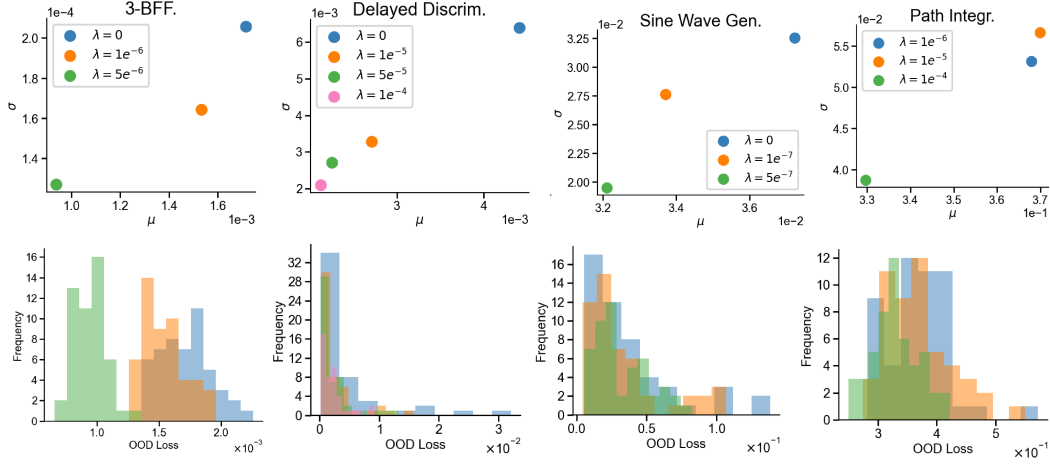


Figure 9: Detailed characterization of the OOD performance of networks while changing low-rank regularization strength.

### 169 I.4.2 Sparsity (L1) regularization

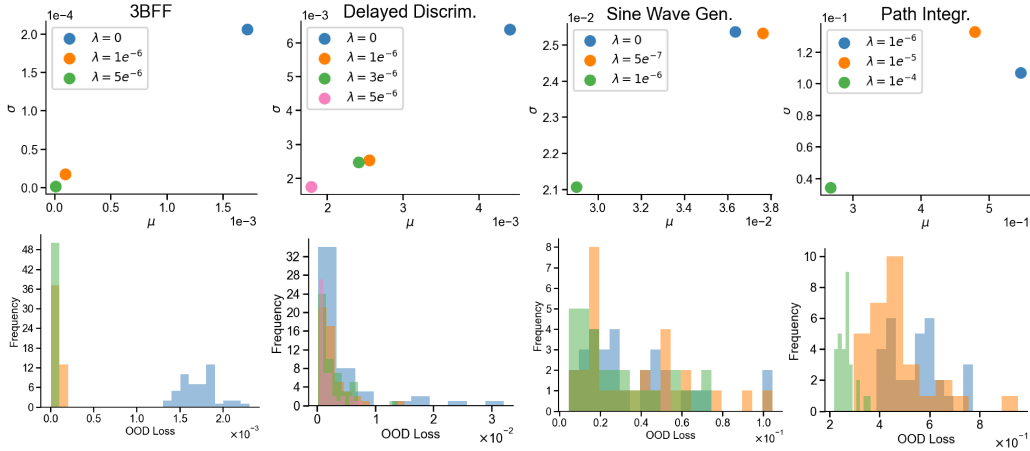


Figure 10: Detailed characterization of the OOD performance of networks while changing sparsity (L1) regularization strength.

## 170 J A short introduction to Maximal Update Parameterization ( $\mu P$ )

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

176 To preserve non-trivial feature learning at large width, the *Maximal Update Parameterization*  
 177 ( $\mu P$ ) rescales both the weight initialisation and the learning rate.  $\mu P$  keeps three quantities  
 178 *width-invariant* at every layer—(i) the norm/variance of activations (ii) the norm/variance of the



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

181 For recurrent neural networks, under Stochastic Gradient Descent (SGD), the network output, ini-  
182 tialization, and learning rates are scaled as

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

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

$$\eta_{\text{SGD}} = \eta_0 \gamma_0^2 N. \quad (3)$$

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

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

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

$$\eta_{\text{Adam}} = \eta_0 \gamma_0. \quad (6)$$

## 184 **K Theoretical relationship between parameterizations**

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

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

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

189 In  $\mu$ P RNNs, the hidden activations are updated as

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

190 When  $\tau = 1$ ,

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

191

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

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

- 194 • **Standard RNN:**  $\phi$  is applied *post-activation*, i.e. after the recurrent and input terms are  
195 linearly combined,
- 196 •  **$\mu$ P RNN:**  $\phi$  is applied *pre-activation*; i.e. before the recurrent weight matrix, so the hidden  
197 state is first non-linearized and then linearly combined

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

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

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

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

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

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

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

- 208 • Initialization scales are matched
- 209 • The learning rate is scaled appropriately with  $\gamma$
- 210 • Output weight norms are adjusted according to width

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

## 217 L Verifying larger $\gamma$ reliably induces stronger feature learning in $\mu P$

218 In  $\mu P$  parameterization, the parameter  $\gamma$  interpolates between lazy training and rich, feature-learning  
 219 dynamics, without itself being the absolute magnitude of feature learning. Here, we assess feature-  
 220 learning strength in RNNs under varying  $\gamma$  using two complementary metrics:

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

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

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

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

$$\text{KA}(K^{(f)}, K^{(0)}) = \frac{\text{Tr}(K^{(f)} K^{(0)})}{\|K^{(f)}\|_F \|K^{(0)}\|_F}, \quad K = \nabla_W \hat{y}^\top \nabla_W \hat{y}.$$

228 We demonstrate that higher  $\gamma$  indeed amplifies feature learning inside the network.

### 229 L.1 N-BFF

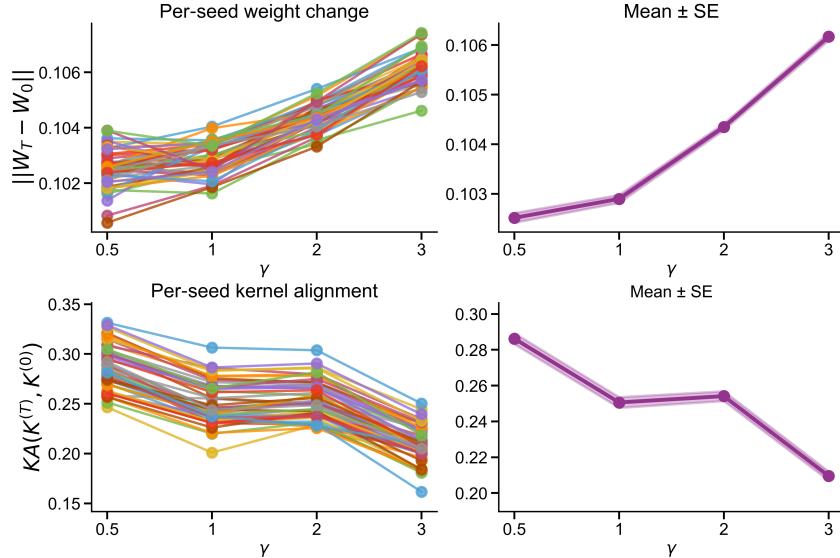


Figure 11: Weight change norm and kernel alignment for networks trained on the 3-Bits Flip Flop task as we vary  $\gamma$ . On the left panels, we show the per-seed metrics where connected dots of the same color are networks of identical initialization trained with different  $\gamma$ . On the right panels, we show the mean and standard error of the metrics across 50 networks. For larger  $\gamma$ , 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  $\gamma$ .

230 **L.2 Delayed Discrimination**

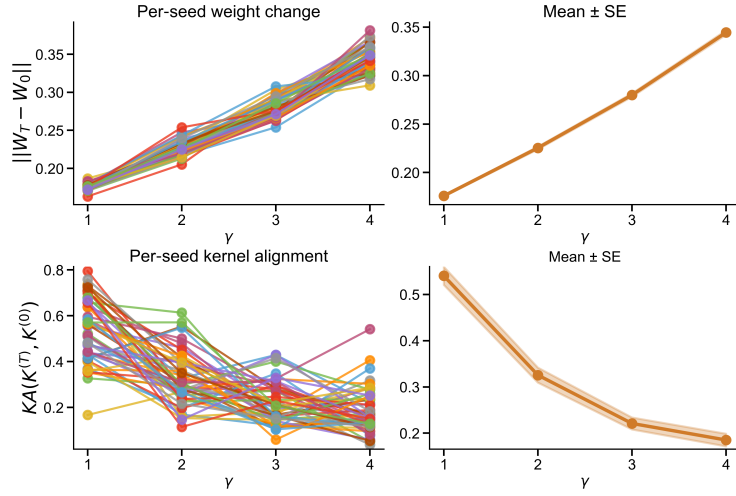


Figure 12: Stronger feature learning for networks trained under larger  $\gamma$  on the Delayed Discrimination task.

231 **L.3 Sine Wave Generation**

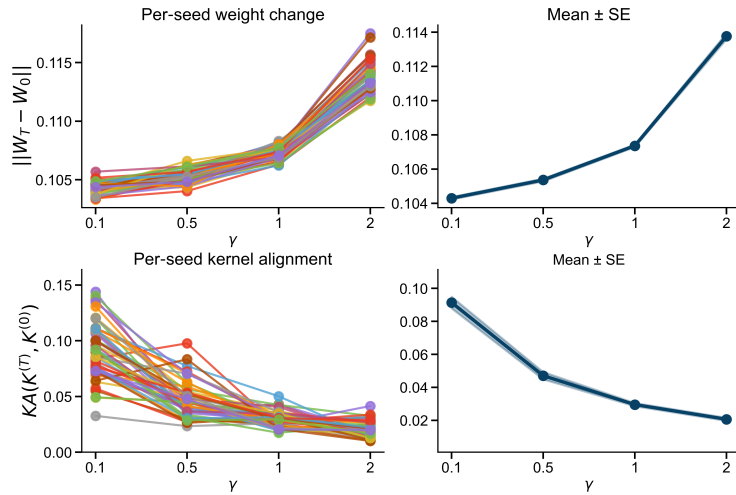


Figure 13: Stronger feature learning for networks trained under larger  $\gamma$  on the Sine Wave Generation task.

## 232 L.4 Path Integration

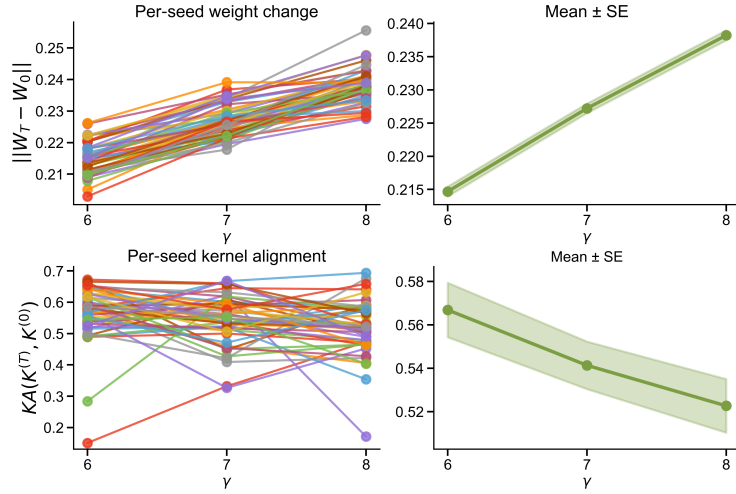


Figure 14: Stronger feature learning for networks trained under larger  $\gamma$  on the Path Integration task.

## 233 M Verifying $\mu P$ reliably controls for feature learning across network width

234 Here, we only use Kernel Alignment to assess the feature learning strength in the networks since  
 235 the unnormalized weight-change norm  $\|W_T - W_0\|_F$  scales directly with matrix size (therefore  
 236 network size) and there exists no obvious way to normalize across different dimensions. In our ear-  
 237 lier analysis where we compared weight-change norms at varying  $\gamma$ , network size remained fixed,  
 238 so those Frobenius-norm measures were directly comparable. We found that, for all tasks except  
 239 Delayed Discrimination, the change in mean KA across different network sizes remains extremely  
 240 small (less than 0.1), which demonstrates that  $\mu P$  parameterization with the same  $\gamma$  has effec-  
 241 tively controlled for feature learning strength across network sizes. On Delayed Discrimination, the  
 242 networks undergo slightly lazier learning for larger network sizes. Nevertheless, we still include  
 243 Delayed Discrimination in our analyses of solution degeneracy to ensure *our conclusions remain*  
 244 *robust even when  $\mu P$  can't perfectly equalize feature-learning strength across widths*. As shown  
 245 in the main paper, lazier learning regime generally increases dynamical degeneracy; yet, larger net-  
 246 works which exhibit lazier learning in the N-BFF task actually display lower dynamical degeneracy.  
 247 This reversed trend confirms that the changes in solution degeneracy arise from network size itself,  
 248 not from residual variation in feature learning strength.

### 249 M.1 N-BFF

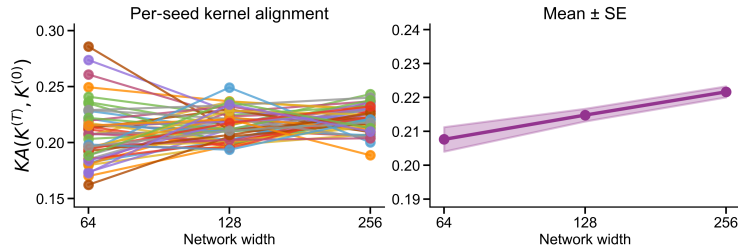


Figure 15: Kernel alignment (KA) for different network width on the 3 Bits Flip-Flop task. (Lower KA implies more feature learning.)

## 250 M.2 Delayed Discrimination

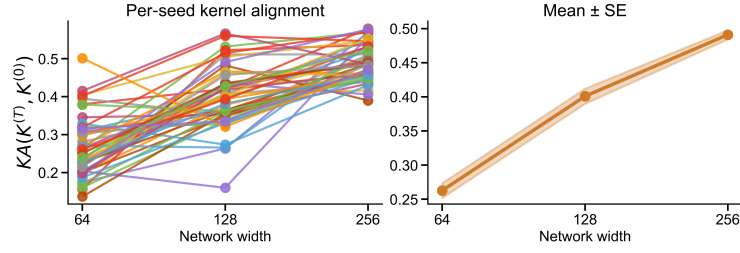


Figure 16: Kernel alignment for different network width on the Delayed Discrimination task.

## 251 M.3 Sine Wave Generation

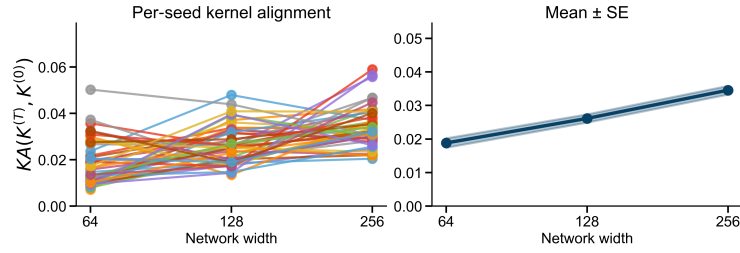


Figure 17: Kernel alignment for different network width on the Sine Wave Generation task.

## 252 M.4 Path Integration

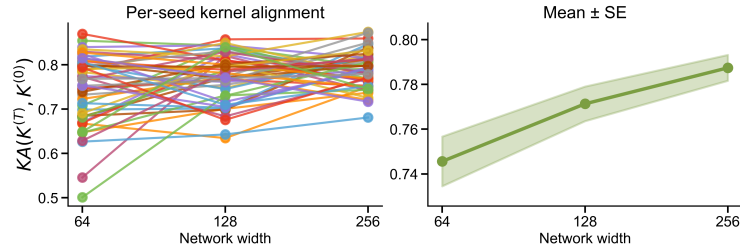


Figure 18: Kernel alignment for different network width on the Path Integration task.

## 253 N Regularization's effect on degeneracy for all tasks

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

### 258 N.1 Low-rank regularization

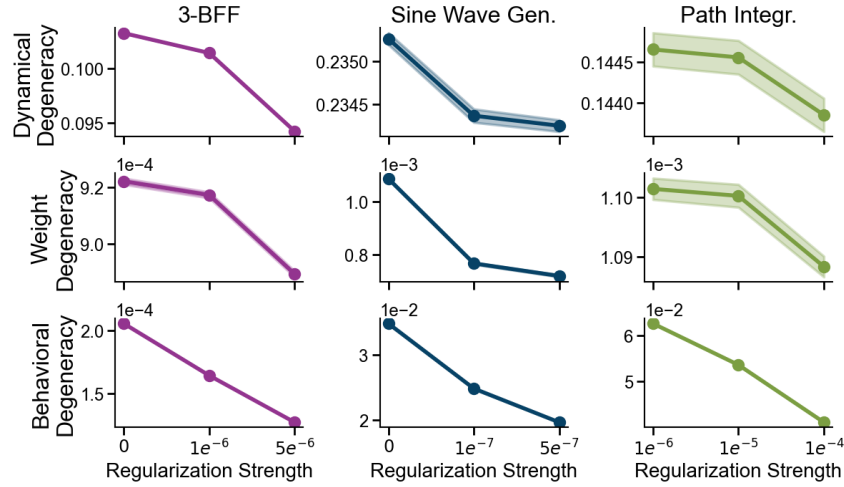


Figure 19: Low-rank regularization reduces degeneracy across neural dynamics, weight, and OOD behavior on the N-BFF, Sinewave Generation, and Path Integration task.

### 259 N.2 Sparsity regularization

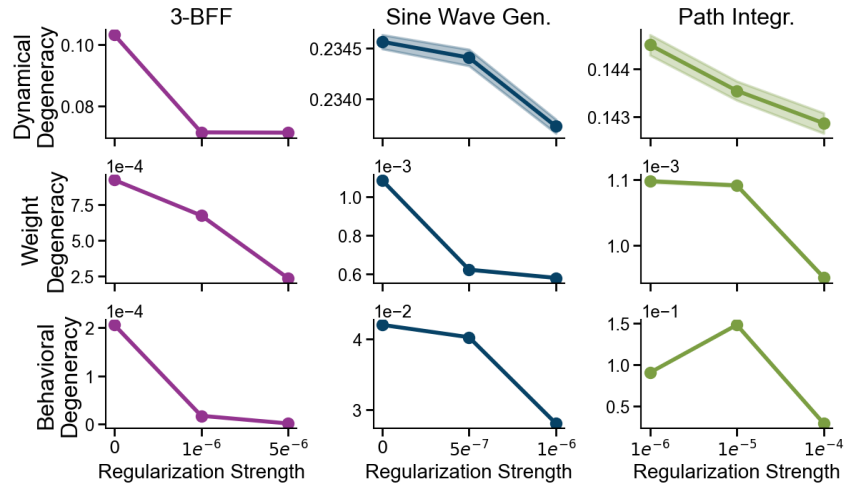


Figure 20: Sparsity regularization reduces degeneracy across neural dynamics, weight, and OOD behavior on the N-BFF, Sinewave Generation, and Path Integration task.

## 260 O Test feature learning effect on degeneracy in standard parameterization

261 While  $\mu P$  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 21).

264 To manipulate feature-learning strength in these ordinary RNNs we applied the  $\gamma$ -**trick**—scaling the network’s outputs by  $\gamma$ —and multiplied the learning rate by the same factor. With width fixed, these two operations replicate the effective changes induced by  $\mu P$ . Figure 22 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 representational dissimilarity matrix before and after training, and is defined by

$$\text{RA}(R^{(T)}, R^{(0)}) := \frac{\text{Tr}(R^{(T)} R^{(0)})}{\|R^{(T)}\| \|R^{(0)}\|}, \quad R := H^\top H,$$

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

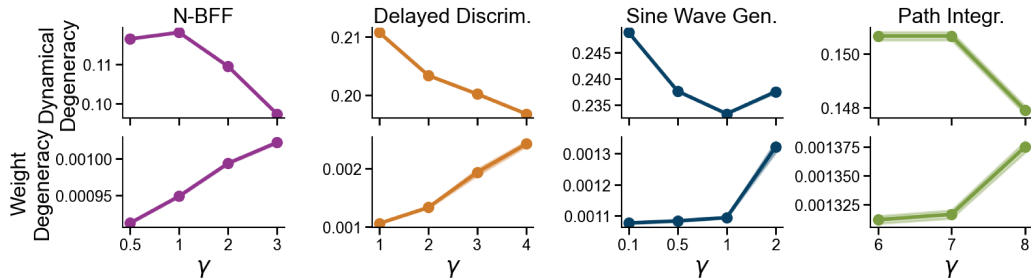


Figure 21: Stronger feature learning reliably decreases dynamical degeneracy while increasing weight degeneracy in standard discrete-time RNNs.



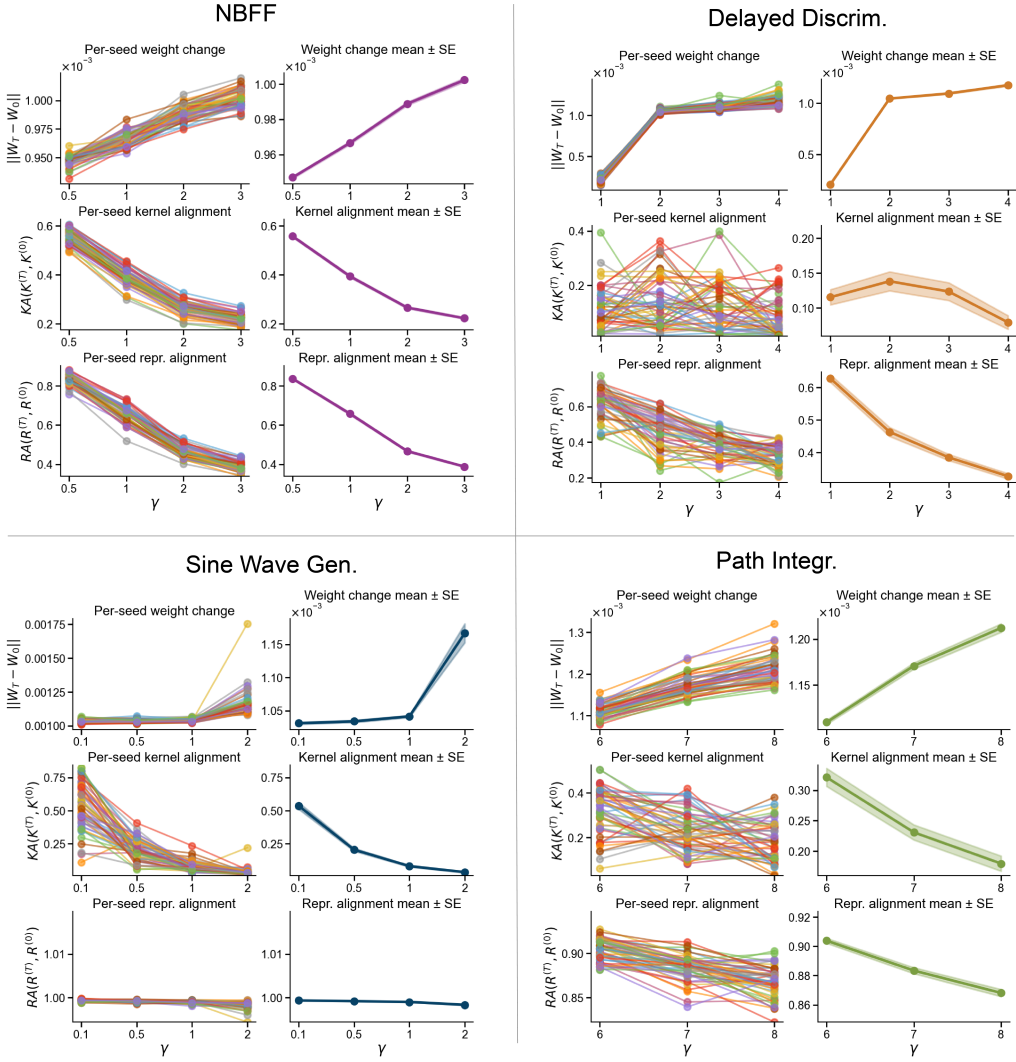


Figure 22: Larger  $\gamma$  reliably induces stronger feature learning in standard discrete-time RNNs.

## 274 P Test network size effect on degeneracy in standard parameterization

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

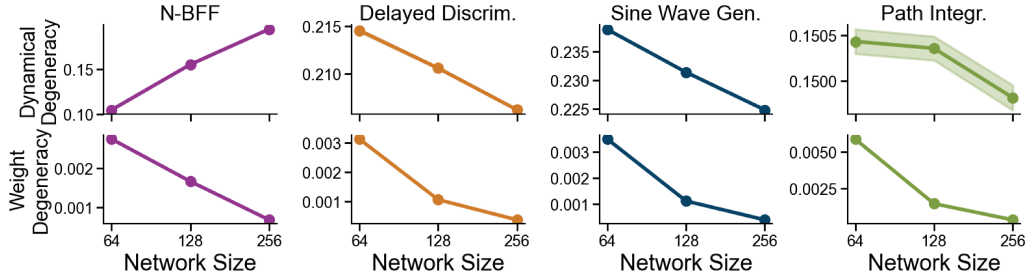


Figure 23: Larger network sizes lead to lower dynamical and weight degeneracy, except in the case where feature learning is unstable across width (in N-BFF).

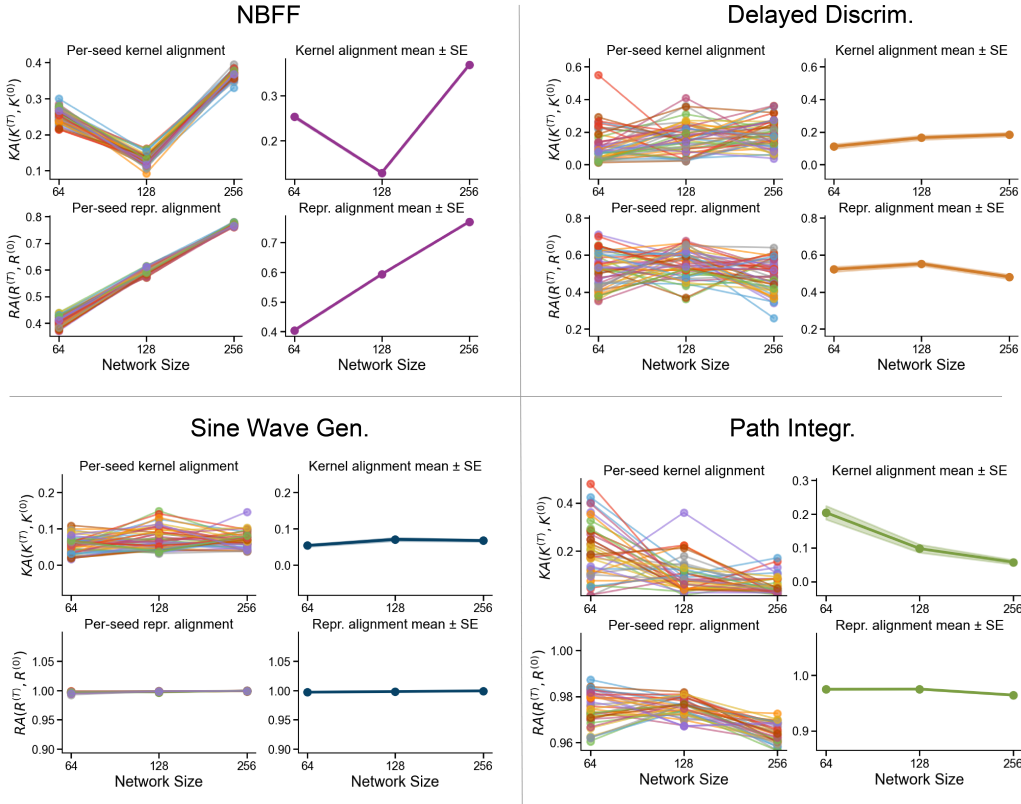


Figure 24: When changing network width in standard discrete-time RNNs, feature learning strength remains stable across width except in N-BFF, where notably lazier learning happens in the widest network.

## 281 **Q Disclosure of compute resources**

282 In this study, we conducted 50 independent training runs on each of four tasks, systematically sweep-  
283 ing four factors that modulate solution degeneracy—task complexity (15 experiments), learning  
284 regime (15 experiments), network size (12 experiments), and regularization strength (26 experi-  
285 ments), resulting in a total of 3400 networks. Each experiment was allocated 5 NVIDIA V100/A100  
286 GPUs, 32 CPU cores, 256 GB of RAM, and a 4-hour wall-clock limit, for a total compute cost of  
287 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.
- 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>.
- Edward N. Lorenz. Predictability: A problem partly solved. In *ECMWF Seminar on Predictability, 4–8 September 1995*, Reading, U.K., 1996. European Centre for Medium-Range Weather Forecasts.
- 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.