

Modular Connectivity in Neural Networks Emerges from Poisson Noise-Motivated Regularization and Promotes Robustness and Compositional Generalization

Daoyuan Qian,^{1,2,3} Qiyao Liang^{1,2,4} and Ila Fiete^{1,2,3,*}

¹*McGovern Institute for Brain Research, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*

²*K. Lisa Yang Integrative Computational Neuroscience Center in the Yang-Tan Collective, Cambridge, Massachusetts 02142, USA*

³*Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*

⁴*Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA*



(Received 7 December 2025; accepted 29 June 2026; published 1 September 2026)

Circuits in the brain commonly exhibit modular architectures that factorize complex tasks, resulting in the ability to compositionally generalize and reduce catastrophic forgetting. In contrast, artificial neural networks (ANNs) appear to mix all processing, because modular solutions are difficult to find as they are vanishing subspaces in the space of possible solutions. Here we draw inspiration from fault-tolerant computation and the Poisson-like firing of real neurons to show that activity-dependent neural noise, combined with nonlinear neural responses, drives the emergence of solutions that reflect an accurate understanding of modular tasks, corresponding to acquisition of a correct world model. We find that noise-driven modularization can be recapitulated by a deterministic regularizer that multiplicatively combines weights and activations, revealing rich phenomenology not captured in linear networks or by standard regularization methods. We show that modularized ANNs exhibit superior robustness to noise and adversarial attacks. When applied as an architectural constraint, a premodularized ANN acquires the ability to generalize and extrapolate well beyond training data, compared to ANNs without such inductive biases. Together, our work demonstrates a regularizer and architectures that could encourage modularity emergence to yield functional benefits.

DOI: [10.1103/8bps-t2vq](https://doi.org/10.1103/8bps-t2vq)

I. INTRODUCTION

The crux of biological organization is modularity: multiple specialist structures that operate on subsets of inputs and are combined in various ways to enable adaptive responses to highly diverse situations. Evolutionary theorists have proposed various reasons for the emergence of modular structure, including the need for rapid adaptation to nonstationary conditions, robustness, and evolvability [1]. In the context of brains and intelligence, modular systems hypothetically enable the generation of “infinite use from finite means” [2], allow for learning sample efficiency [3–6], avoid catastrophic forgetting by reusing rather than slowly relearning and overwriting computational primitives [7,8], and enable fault tolerance [9]. Feedforward neural networks trained on multiple tasks were also found to exhibit increased modularity in tandem with performance increases [10]. Intuitively, a modular network structure allows disentangled processing to occur with minimum interference between unrelated information channels. However, modern machine learning methods, including stochastic gradient descent applied to large neural networks,

tend not to find modular solutions even for inherently modular tasks. Furthermore, when supplied with modular representations in the form of independent task factors, artificial neural networks (ANNs) usually do not then learn to compose them [11–13].

The functional benefits of modular networks call for efforts to discover possible drivers of modularization and to find the best ways to incorporate these modularity emergence mechanisms in ANNs. In particular, how might an ANN automatically identify the true modular structure of a task and reorganize its connections accordingly? We use a strict definition of modular discovery: It is not sufficient for the network to merely correctly perform the modular task, but its internal (latent) representations must consistently reflect the inherently modular factorized structure of the task. For instance, a task involving two inputs and two outputs, with each desired output a function of one input, can by our definition be solved in modular and nonmodular ways: A modular learner would not mix the two inputs in any of its latent representations, while a nonmodular learner may mix and then demix the variables within its latents, effectively solving the task but not reflecting an understanding of the factorizability of the underlying task.

In a study involving Boolean networks that implement digital functions, it was found that they evolve towards the vanishingly small space (in the space of all solutions) of modular solutions when they are forced to perform fault-tolerant computation in the presence of bit-flip noise [9]. Separately, it was shown that penalizing large neuron activation values under a non-negative activation constraint, a form of activity

*Contact author: fiete@mit.edu

Published by the American Physical Society under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

regularization, can promote the emergence of factorized representations [14]. Can noise-based modularization from the former work be similarly induced in ANNs, using neuron noise? In addition, can the observation that noise is related to regularization [15,16] be leveraged to define a new regularizer that serves as a generic bias towards the emergence of modular ANN solutions?

Here we investigate the potential of neural noise to drive modularity emergence in ANNs over a range of noise characteristics and explore how modularity emergence can affect the robustness and generalization ability of networks in downstream tasks. Neural noise is typically modeled by additive Gaussian noise in artificial networks. Characterization of biological neurons, however, showed that the variance in firing rate changes with the mean firing rate, consistent with Poisson statistics in some cases [17], and more generally can be fitted to a power law [18]. Motivated by these observations, we consider noises that scale with neuron activation in ANNs. Denoting the activity output of a single neuron by h and the corresponding noise by η , we draw noise values from a zero-mean Gaussian distribution with a variance dependent on h with an exponent α as $\langle \eta^2 \rangle_n \propto |h|^\alpha$, where $\langle \dots \rangle_n$ denotes averaging over noise realizations [Fig. 1(a)]. We find that networks trained on modular tasks find nonmodular solutions when the neurons are noise-free or the noise is additive ($\alpha = 0$). Modular connectivity emerges in nonlinear networks for $\alpha > 0$, for instance, when the noise is Poisson-like ($\alpha = 1$) or multiplicative ($\alpha = 2$).

Motivated by this observation, we next derive how the modularization effects of training with stochastic neural activations can be recapitulated by a specific nonstochastic regularization in the loss function. This regularizer, which we call the weighted-activity (WA) regularizer, is the product of the outgoing squared weight norm with the presynaptic neural activation norm. We demonstrate empirically that WA regularization drives the emergence of input-selective neural responses that reduce overall activation (similar to [14]), but further it drives the network to modular solutions similar to the effects of multiplicative noise injection. We characterize the WA regularizer under different neuron activation functions, task traits, and training data density. We show that rectified linear unit (ReLU) activation is important in promoting modularization through its nonsaturating behavior, in contrast to hyperbolic tangent and sigmoid activations. When the input factors are entangled, the regularizer also finds the optimal way of disentangling them in the appropriate depth of a deep nonlinear network. We find that modularization requires sufficient training data, meaning that the WA regularizer does not solve the problem of sample efficiency in module discovery. Perhaps consistent with this finding, modular organization of the brain in nature often occurs at early stages of development, guided through processes acquired over evolutionary timescales, and the functional benefits are only manifest when (re)training the already modularized network on new tasks [19].

Moreover, we explore the functional advantage of modularized networks. Networks trained with WA regularization exhibit improved robustness; starting from an already modular network, we show that modularity acts as a universal bias for compositional generalization: The independent channels

of processing enable the network to quickly learn the compositional structure of the task and allow it to perform well on out-of-distribution data while improving data efficiency to convergence.

Taken together, these insights may help explain the modular organization and competence of biological neural networks and promote greater modularity in artificial ones.

II. MODULAR SOLUTIONS EMERGE WITH NONADDITIVE NEURAL NOISE

A. Network and task setup

Directly motivated by [9], we consider the effects of noise in ANNs when training on modular problems. We first explore this question empirically, training a fully connected feedforward network (three hidden layers, 32 hidden units in each layer) to learn nonlinear function regression [Fig. 1(b)] by gradient descent on the mean square error (MSE). Each hidden unit computes the sum of its inputs and injects a noise before computing its output. Denoting inputs and outputs by vectors $\mathbf{x} = [x_1, \dots, x_4]^T$ and $\mathbf{y} = [y_1, \dots, y_4]^T$, the task is to learn the function $\mathbf{y} = \mathbf{f}(\mathbf{x})$ defined as

$$\mathbf{f}(\mathbf{x}) = [f_1(x_1), f_2(x_2), f_3(x_3), f_4(x_4)]^T, \quad (1)$$

where the f_i 's are four independent, randomly generated nonlinear functions [see Fig. 1(b), Appendix A, and [20]]. Inputs $\mathbf{x}$ are independently drawn from the uniform distribution over $(-2, 2)$ so the task has a true modular structure. Given the column vector of neural activations $\mathbf{h}_\ell$ in layer ℓ , inputs to the next layer are

$$\mathbf{h}_{\ell+1} = \phi[\mathbf{W}_\ell(\mathbf{h}_\ell + \boldsymbol{\eta}_\ell) + \mathbf{b}_\ell], \quad (2)$$

where $\mathbf{W}_\ell$ and $\mathbf{b}_\ell$ are weights and biases in layer ℓ , $\boldsymbol{\eta}_\ell$ is a zero-mean noise vector, and $\phi(x)$ is a pointwise transfer function, which we take to be the leaky ReLU:

$$\phi(x) = \begin{cases} x & \text{if } x > 0 \\ 0.1x & \text{otherwise.} \end{cases} \quad (3)$$

The small leak of 0.1 prevents vanishing gradients during training. The injected activation noise is zero mean and Gaussian, with a covariance that scales with the absolute value of the summed inputs with exponent α [Fig. 1(a)],

$$\langle \boldsymbol{\eta}_\ell \boldsymbol{\eta}_\ell^T \rangle_n = \sigma_n^2 \text{diag}(|\mathbf{h}_\ell|)^\alpha, \quad (4)$$

for a noise strength σ_n . In practice, the noise $\boldsymbol{\eta}_\ell$ is generated by first sampling a vector of Gaussian random variables with standard deviation σ_n (set to 0.1) and multiplied elementwise with $|\mathbf{h}_\ell|^{\alpha/2}$ (the exponent is calculated elementwise too). Setting $\alpha = 0, 1$, and 2 corresponds to injecting additive, Poisson-like, and multiplicative noise, respectively. The simulation code is available on a GitHub repository [20].

B. Detecting modular structure

A network that recognizes the modular task structure should exhibit modular connectivity, such that each input x_i projects without mixing with other input-driven responses in the hidden layers to the output representing $f_i(x_i)$, even though the network begins with mixing in all layers. However, there

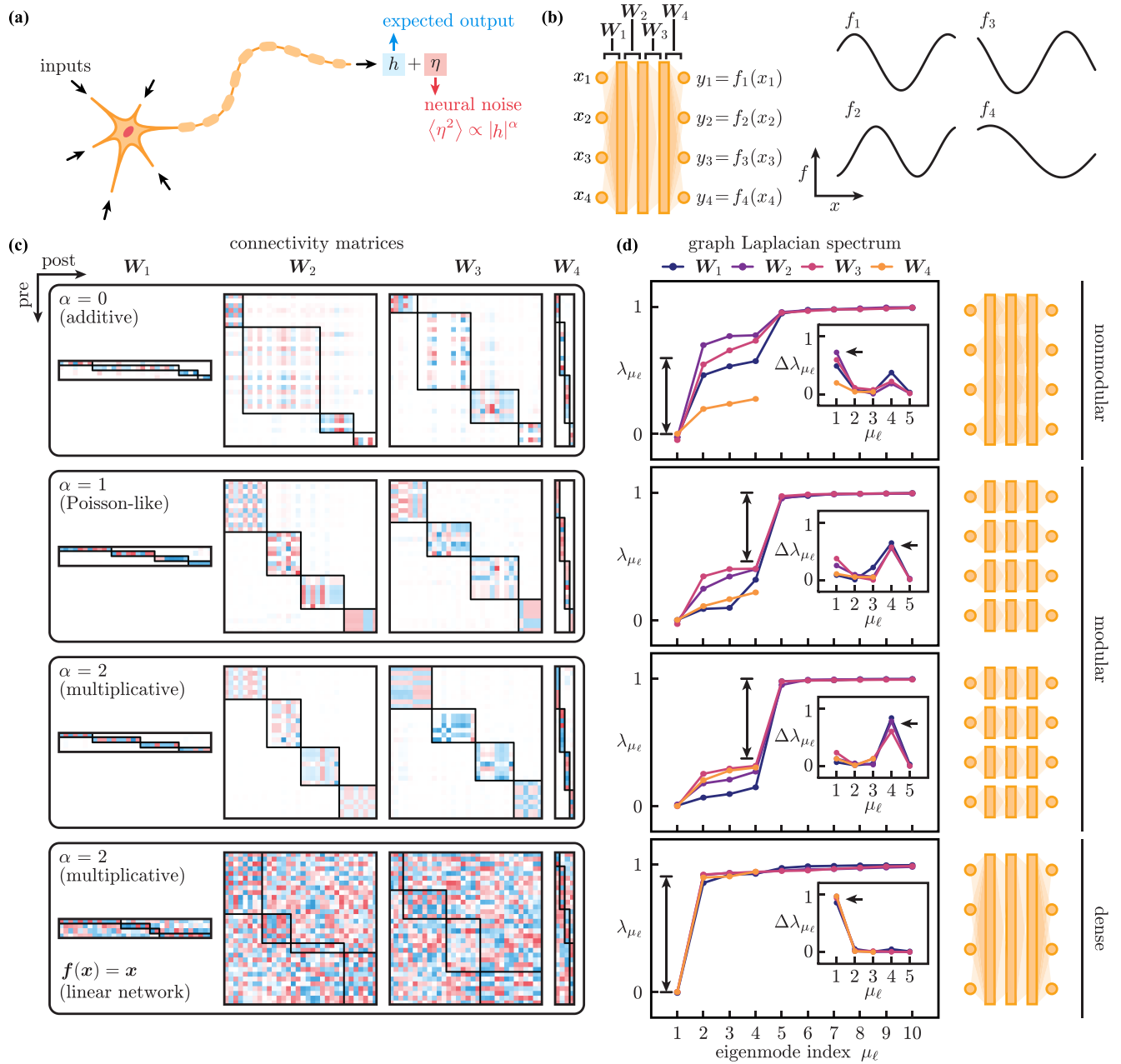


FIG. 1. Nonadditive noise drives deep nonlinear ANNs to modularize. (a) A single neuron takes in multiple inputs and produces an output h with neural noise η , the size of which scales as some power of $|h|$. (b) Modular task. A network with three hidden layers (32 units per intermediate layer) must learn an input-output mapping involving four independent nonlinear functions of the inputs. (c) Learned connectivity matrices across layers for nonlinear networks and $\alpha = 0, 1$, and 2 (top three rows). The network learns modular connectivity if $\alpha > 0$. Rows and columns of the matrices are rearranged according to detected clusters. When the task involves learning identity functions with a linear network and $\alpha = 2$, the learned connectivity is nonmodular (bottom row). (d) Network modularity can be characterized via the eigenspectra of the graph Laplacian matrices of the connectivities in (c). The μ_ℓ th eigenvalue of the ℓ th layer is denoted by λ_{μ_ℓ} (first ten modes are shown). Clusters are detected as eigenvalues near 0, while eigenvalues closer to 1 correspond to trivial structures. The number of clusters is typically identified by finding the largest difference between neighboring eigenvalues $\Delta\lambda_{\mu_\ell} \equiv \lambda_{\mu_\ell+1} - \lambda_{\mu_\ell}$ (double arrows and insets).

are many more alternative solutions to the task, in which the input-to-hidden layers mix the inputs in some arbitrary way and the hidden-to-output layers undo that transformation to demix them. After training, we determine whether the network exhibits modular connectivity structure by analyzing the connectivity matrix $W_{\ell-1}$ projecting from layer $\ell - 1$ (neurons indexed by $\mu_{\ell-1}$) to hidden layer ℓ (neurons

indexed by μ_ℓ). We compute clusters of layer ℓ neurons that share similar input sources using the random walk variant of the graph Laplacian matrix, denoted L_ℓ^{rw} . Very briefly, each row of the weight matrix $W_{\ell-1}$ represents an input weight vector to a postsynaptic neuron, and we calculate the row-wise normalized weights $\tilde{W}_{\ell-1}$ by rescaling the input vectors to have unit norm and taking the absolute value of the

entries

$$[\tilde{\mathbf{W}}_{\ell-1}]_{\mu_\ell \mu_{\ell-1}} \equiv \frac{|[\mathbf{W}_{\ell-1}]_{\mu_\ell \mu_{\ell-1}}|}{\sqrt{\sum_{\mu_{\ell-1}} |[\mathbf{W}_{\ell-1}]_{\mu_\ell \mu_{\ell-1}}|^2}}. \quad (5)$$

We take the absolute value because we are interested in whether a postsynaptic neuron receives inputs from a presynaptic neuron regardless of the sign of the connectivity. The $\tilde{\mathbf{W}}_{\ell-1}$ satisfies $\sum_{\mu_{\ell-1}} [\tilde{\mathbf{W}}_{\ell-1}]_{\mu_\ell \mu_{\ell-1}}^2 = 1$. We then calculate the (cosine) similarity matrix $\mathbf{S}_\ell$, the degree matrix $\mathbf{D}_\ell$, and the random walk Laplacian matrix $\mathbf{L}_\ell^{\text{rw}}$ as [21] (the label ℓ is used since the $\mathbf{W}_{\ell-1}$ matrix is used to cluster neurons in the postsynaptic layer ℓ)

$$\mathbf{S}_\ell \equiv \tilde{\mathbf{W}}_{\ell-1} \tilde{\mathbf{W}}_{\ell-1}^T, \quad (6a)$$

$$\mathbf{D}_\ell \equiv \text{diag}(\mathbf{S}_\ell \mathbf{1}), \quad (6b)$$

$$\mathbf{L}_\ell^{\text{rw}} \equiv \mathbf{I} - \mathbf{D}_\ell^{-1} \mathbf{S}_\ell, \quad (6c)$$

where $\mathbf{1}$ is a column vector of 1's (so matrix multiplication with $\mathbf{1}$ is a row sum) and $\mathbf{I}$ is the identity matrix. The eigenvalues λ_{μ_ℓ} of $\mathbf{L}_\ell^{\text{rw}}$ lie in the range $[0, 2]$ with eigenvalues of 0, 1, and 2 indicating clusters of connected components, treelike branches, and bipartite graphs, respectively [21]. With four inputs, a modularized network should have four eigenvalues of $\mathbf{L}_\ell^{\text{rw}}$ near 0, with the rest near 1.

To visualize the modular connectivity structure, we reorder neuron indices μ_ℓ by performing a k -means clustering of the four eigenvectors with the smallest eigenvalues, to group postsynaptic neurons into four sets. Plotting the network connectivity matrix reveals increasingly modular organization as α increases from 0 to 2 [Fig. 1(c), top three panels]. After computing and sorting the eigenspectra $\{\lambda_{\mu_\ell}\}$ in increasing order, the largest spectral gap $\Delta\lambda_{\mu_\ell} \equiv \lambda_{\mu_\ell+1} - \lambda_{\mu_\ell}$ reflects the most drastic change in network structure across different eigenmodes. Consistent with standard network literature [21], we plot $\Delta\lambda_{\mu_\ell}$ versus μ_ℓ to identify the number of network modules as $\mu_\ell^* = \arg \max_{\mu_\ell} (\Delta\lambda_{\mu_\ell})$ [Fig. 1(d), double arrows]. For a modular network of four clusters, a large eigenvalue difference is expected between the fifth (near 1) and fourth (near 0) eigenvalues. A pronounced four-cluster connectivity structure is present only for $\alpha = 1$ and 2, but not for $\alpha = 0$ [Fig. 1(d), top three panels]. As a control, we train a linear network without bias to learn four identity functions $y = x$ with $\alpha = 2$. The resulting connectivities are dense without modular structure [Figs. 1(c) and 1(d), bottom panels]. This shows the nonlinearity and the bias are likely essential for the emergence of ANN modularity.

As a separate measure of modularity, we also consider the response of each hidden neuron to individual inputs. This unit-level modularity is defined and computed for the networks in Fig. 1(c), and we find that similar conclusions on the efficacy of Poisson-like and multiplicative noises in promoting modularity can be drawn (see Fig. 12 in Appendix B). Intuitively, modularity across the unit and network levels translates to blocky connectivity matrices, which is reliably captured by the graph-theory-based measure outlined above; thus we will use the graph Laplacian as the main detector of modularity in what follows.

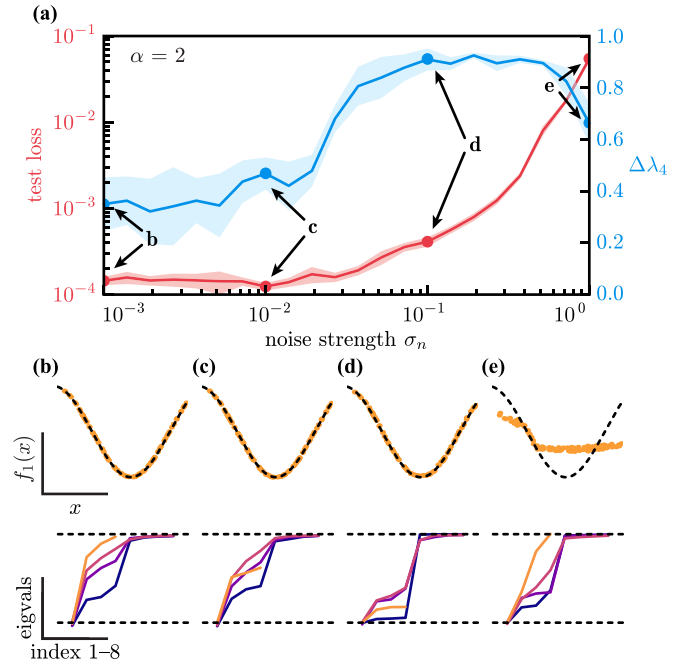


FIG. 2. Noise strength leads to a loss-modularity tradeoff. (a) In the underparametrized small-network regime, we find that increasing the noise strength (here using multiplicative noise $\alpha = 2$) adversely affects model performance (red loss curve) while encouraging modular connectivity (blue eigengap curve). Shaded regions are standard deviation from five independent runs. (b)–(e) Shown on top are exemplar plots for model prediction (orange scatters) and ground truth (black dashed lines) of the first learned function. Shown on the bottom are spectra of the Laplacian matrices, computed in the same way as in Fig. 1. Dashed horizontal lines are eigenvalues of 0 and 1, respectively.

C. Effect of noise strength

For modularization-promoting noise, there is a tradeoff between module emergence and test loss as the noise strength σ_n increases. At larger σ_n , the noise perturbs the gradient descent process and the test loss (mean square error loss with noise set to 0) increases [Fig. 2(a), red]. Meanwhile, the modularizing effect gets stronger with σ_n , evident from plotting the fourth eigengap $\Delta\lambda_4$ of the first layer ($\ell = 1$) [Fig. 2(a), blue]. Inspecting the learned function for the first input $f_1(x_1)$ and full eigenspectra at various σ_n confirms the tradeoff between test loss and modularity [Figs. 2(b)–2(e)]. The simulations of Fig. 1 were performed near the balance point ($\sigma_n = 0.1$) in the two competing effects of noise, module emergence and task performance deterioration. To better determine whether injecting activity-dependent noise jeopardizes network performance, we perform the same simulations for $\alpha = 0$ and $\alpha = 1$. We find that the test losses increase with σ_n following the same trends for $\alpha = 0, 1$, and 2, while pronounced modularization is seen only for $\alpha = 1$ and 2 (see Fig. 13 in Appendix C). This shows that injecting activity-dependent noise has a similar effect on network performance as the traditional additive noise, but with the added effect of modularizing the network connectivity.

III. NOVEL EFFECTIVE REGULARIZERS FROM NOISE

To better understand why specific activity-dependent noise promotes modular connectivity, we next characterize its effects analytically. We show that the effects can be viewed as adding novel regularization terms to the loss function, allowing us to obtain nonstochastic inducers for modularity emergence. Deriving deterministic regularizers can recapitulate the beneficial effects of noise without the problems of slow learning speed and loss of performance from excess noise that are introduced by learning with noise.

A. Single-layer networks

In a network with one hidden layer, the input ($\mathbf{h}_1 = \mathbf{x}$), hidden ($\mathbf{h}_2$), and output ($\mathbf{h}_3 = \mathbf{y}$) layers are related via Eq. (2). We follow [22] to incorporate the effects of the nonlinearity $\phi(x)$ by defining a diagonal matrix $\mathbf{Q}_\ell$ for each hidden layer ℓ such that the state updates follow a (manifestly) linear form

$$\mathbf{h}_1 = \mathbf{x}, \quad (7a)$$

$$\mathbf{h}_2 = \mathbf{Q}_2(\mathbf{W}_1\mathbf{h}_1 + \mathbf{b}_1), \quad (7b)$$

$$\mathbf{h}_3 = \mathbf{W}_2(\mathbf{h}_2 + \boldsymbol{\eta}_2) + \mathbf{b}_2, \quad (7c)$$

where the μ_ℓ th entry $[\mathbf{Q}_\ell]_{\mu_\ell\mu_\ell}$ is given by inputs from the previous layer; $\mathbf{Q}_\ell$ is different for each sample and depends on previous weights and biases. There is no $\mathbf{Q}_3$ since the output layer is linear, and there is no $\boldsymbol{\eta}_1$ because that would correspond to input noise. As before, the noise $\boldsymbol{\eta}$ can be additive, Poisson-like, or multiplicative ($\alpha = \{0, 1, 2\}$). For $\mathbf{Q}_2$, the entries are

$$[\mathbf{Q}_2]_{\mu_2\mu_2} = \begin{cases} 1 & \text{if } [\mathbf{W}_1(\mathbf{h}_1 + \boldsymbol{\eta}_1) + \mathbf{b}_1]_{\mu_2} > 0 \\ 0.1 & \text{otherwise.} \end{cases} \quad (8)$$

The expected excess squared loss ΔL due to injected noise, based on the difference in value of the network output with and without noise, is given by

$$\begin{aligned} \Delta L &\equiv \langle \Delta \mathbf{h}_3^T \Delta \mathbf{h}_3 \rangle_n = \sigma_n^2 \text{tr}[\text{diag}(|\mathbf{h}_2|^\alpha) \mathbf{W}_2^T \mathbf{W}_2] \\ &= \sigma_n^2 \sum_{\mu_3, \mu_2} [\mathbf{W}_2]_{\mu_3\mu_2}^2 |\mathbf{h}_2|_{\mu_2}^\alpha, \end{aligned} \quad (9)$$

where $\langle \cdots \rangle_n$ designates the expectation over the injected noise $\boldsymbol{\eta}$, μ_3 and μ_2 label the output and hidden neurons, respectively, and we have used the noise covariance (4). This form of the excess error and the empirical results of modularity with $\alpha > 0$ suggests that minimization of the products of weights and activations, activity dependent and weighted by outgoing connectivity, is important. Therefore, we call these weighted-activity regularizers. Below we examine the form of the WA- α regularizers for different α , neural nonlinearities, and the role of the data distribution; after that, we will examine the effects of these regularizers.

1. Additive noise ($\alpha = 0$)

The expression in Eq. (9) is not analytically tractable for general α , but we can study special cases. For $\alpha = 0$, the squared loss becomes an L_2 -norm of $\mathbf{W}_2$,

$$\Delta L|_{\alpha=0} = \sigma_n^2 \text{tr}(\mathbf{W}_2^T \mathbf{W}_2). \quad (10)$$

Thus, applying additive Gaussian noise during training is equivalent to an L_2 weight regularization in the loss, which is in turn equivalent to weight decay in the weight updates [23]. We saw in simulations [Fig. 1(c)] that additive Gaussian noise does not promote modularization. Because $\text{tr}(\mathbf{W}_2^T \mathbf{W}_2)$ is rotationally symmetric with respect to both pre- and postsynaptic neurons, mixing information channels does not induce an extra cost. This result holds for both linear and nonlinear activation functions. Thus, we surmise that even in arbitrary networks, L_2 weight regularization will not effectively drive modularization.

2. Multiplicative noise ($\alpha = 2$) and a linear network

For $\alpha = 2$ the excess loss involves $\text{diag}(|\mathbf{h}_2|)^2$, corresponding to diagonal entries of the matrix $\mathbf{h}_2 \mathbf{h}_2^T$. We attempt to compute this by treating the network inputs as independent Gaussian variables, with $\langle \mathbf{h}_1 \mathbf{h}_1^T \rangle_d = \sigma_d^2 \mathbf{I}$, where σ_d is the input standard deviation and $\langle \cdots \rangle_d$ designates the expectation. However, taking the expectation over $\mathbf{h}_2 \mathbf{h}_2^T$ involves the matrices $\mathbf{Q}_2$, which depend nonlinearly on $\mathbf{h}_1$, making an explicit general expression intractable. For a linear network, $\mathbf{Q}_2 = \mathbf{I}$, so we obtain

$$\langle \mathbf{h}_2 \mathbf{h}_2^T \rangle_d|_{\text{linear}} = \sigma_d^2 \mathbf{W}_1 \mathbf{W}_1^T. \quad (11)$$

Because we assumed the inputs $\mathbf{h}_1$ have zero mean, the biases have dropped out in Eq. (11). Combining Eqs. (9) and (11) and writing out the indices explicitly, we have

$$\langle \Delta L \rangle_d|_{\alpha=2, \text{linear}} = \sigma_n^2 \sigma_d^2 \sum_{\mu_3, \mu_2, \mu_1} [\mathbf{W}_2]_{\mu_3\mu_2}^2 [\mathbf{W}_1]_{\mu_2\mu_1}^2, \quad (12)$$

where expectations were taken over both the noise and data. For each hidden neuron, the loss is the product of the L_2 -norms of its input and output weights. From simulations [Figs. 1(c) and 1(d), bottom row], we know that $\alpha = 2$ noise does not promote modularity in a linear network; thus the corresponding regularization, given by the product of the squared norms of the incoming and outgoing weights at each neuron, will not lead to modularity in linear networks.

3. Multiplicative noise ($\alpha = 2$) and a nonlinear network

In a nonlinear network with $\alpha = 2$, $\mathbf{Q}_2$ changes across input samples, which complicates analytical derivation. Nonetheless, we can gain some qualitative insight by approximating $\mathbf{h}_1 \mathbf{h}_1^T \approx \sigma_d^2 \mathbf{I}$ for each data sample. The per-sample loss of Eq. (9) becomes

$$\Delta L|_{\alpha=2} \approx \sigma_n^2 \sigma_d^2 \sum_{\mu_3, \mu_2, \mu_1} [\mathbf{W}_2]_{\mu_3\mu_2}^2 [\mathbf{Q}_2]_{\mu_2\mu_2}^2 [\mathbf{W}_1]_{\mu_2\mu_1}^2. \quad (13)$$

Through $\mathbf{Q}_2$, we may view the nonlinear activations as a modulating factor in the weight normalization; its role will be suppressed if the nonlinearity $\phi(x)$ plateaus at large x , which corresponds to a small $\phi'(x)$ and thus smaller entries in $\mathbf{Q}_2$. This happens for hyperbolic tangent and sigmoid activations, where $\phi_{\tanh}(x) = \tanh(x)$ and $\phi_{\text{sigmoid}}(x) = 1/(1 + e^{-x})$. The effect of this suppression for plateauing activation functions is that the network could prefer both small and large neuron activity values. In [14] and as we will show below, modularization is driven partly by penalties on large activity values,

so we expect that modularization behavior will be suppressed for these activation functions.

4. Effect of data distribution

The data-dependent nature of ΔL in Eq. (9) means that special data distributions may affect the training phenomenology. We briefly explore this here. Consider Eq. (9) for $\alpha = 1$ (Poisson-like noise) in a linear network without bias:

$$\begin{aligned} \Delta L|_{\alpha=1, \text{linear}} &= \sigma_n^2 \sum_{\mu_3, \mu_2} [W_2]_{\mu_3 \mu_2}^2 |h_2]_{\mu_2}| \\ &= \sigma_n^2 \sum_{\mu_3, \mu_2} [W_2]_{\mu_3 \mu_2}^2 \left| \sum_{\mu_1} [W_1]_{\mu_2 \mu_1} [h_1]_{\mu_1} \right|. \end{aligned} \quad (14)$$

If the input entries are sparse, for instance, if only one entry of h_1 is nonzero in each input sample (one-hot), summing Eq. (14) over all different samples reduces the excess loss to a product of L_1 and L_2 weight norms, which we call L_{12} regularization:

$$L_{12} \equiv \sum_{\mu_2} \left(\sum_{\mu_3} [W_2]_{\mu_3 \mu_2}^2 \right) \left(\sum_{\mu_1} |[W_1]_{\mu_2 \mu_1}| \right). \quad (15)$$

Applying L_{12} encourages sparsity for inputs to the hidden neurons μ_2 , where the input entries are $[W_1]_{\mu_2 \mu_1}$, by virtue of the L_1 regularization on $[W_1]_{\mu_2 \mu_1}$. The connectivity does not become sparse, because if the neuron has no outgoing weights $[W_2]_{\mu_3 \mu_2}$, the L_1 regularization vanishes. As a result, modularization should arise even in a linear network in this special case. We also observe from simulations that a regularizer similar to L_{12} but with the input and output weight matrices swapped still leads to modularization [20].

The effective additional losses such as Eqs. (12) and (15) resemble group-norm regularizers, which are related to measures of weight magnitudes in deep networks [24]. Ensuring the weight magnitudes to be of similar orders is important for the stability of training, and preservation of rescaling symmetry in ReLU networks is useful, as considered in Path-SGD [25]. Since the regularizers that we consider break rescaling symmetry, a generic L_2 normalization is used in simulations to prevent exploding weights.

We numerically test the effect of Poisson-like noise injection with sparse training data (only one entry of x is nonzero in each example) by training a linear network with a single hidden layer of 32 neurons and four inputs and outputs to learn the linear map: $f(x) = x$. At convergence, the two weight matrices satisfy $W_2 W_1 = I$, where I is the identity matrix, so we focus on the structure of W_1 alone. Inspection of W_1 shows a modular connectivity structure (Fig. 3, top panel). The same structure emerges when using the L_{12} regularizer instead of Poisson-like noise, even if the inputs are not sparse (Fig. 3, second panel). Under the same conditions (nonsparse input), training with L_1 or L_2 regularizers does not lead to modularization (Fig. 3, bottom panels), suggesting that L_{12} , though it was derived under the special condition of very sparse (one-hot) inputs in single-layer linear networks, may be an interesting regularizer, but its behavior in deeper networks would require further exploration (see Fig. 14 in Appendix D).

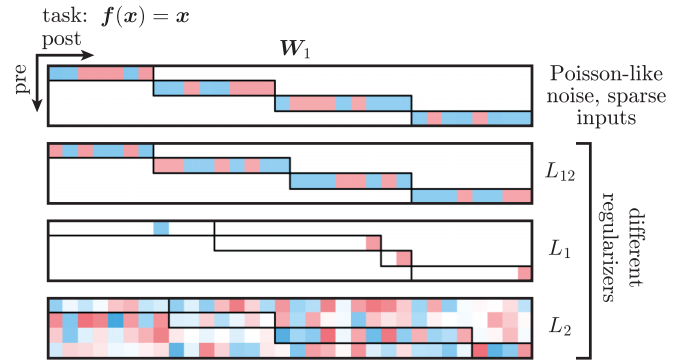


FIG. 3. Mixed (L_{12}) regularization in shallow networks can lead to modular solutions. We train linear networks with a single hidden layer on the trivial task of learning the identity function $f(x) = x$ under different conditions or regularizers. Training with explicit Poisson-like noise and sparse inputs (top panel) or with the associated L_{12} regularizer without Poisson-like noise and sparse inputs, Eq. (15), (second panel) results in modularized connectivity that partitions hidden nodes across inputs. Within each partition the connectivity is dense/all neurons are active. By contrast, L_1 or L_2 regularizers (bottom two panels) generate sparse or dense nonmodular connectivities, respectively.

Nevertheless, to derive a regularizer under more general conditions, we turn to analyzing the scenario of deeper networks and nonsparse activity.

B. Multilayer networks

In multilayer networks, computing the excess loss from noise is complicated because the noise propagates through all the layers. Even if possible to compute, this would result in a complex form of the corresponding deterministic regularizer. Motivated by the observation that modest noise sizes ($\sigma_n = 0.1$) are sufficient to drive modularization and that the effect of noise in a layer will be damped when propagated across layers, we consider an approximate excess loss, based on summing the quadratic errors from each pair of layers. Let

$$q_{\ell+1} \equiv W_{\ell} (h_{\ell} + \eta_{\ell}) + b_{\ell} \quad (16)$$

denote the input at layer $\ell + 1$, before applying the nonlinearity. The quadratic error in $q_{\ell+1}$ due to η_{ℓ} is similar to the case of a single hidden layer [cf. Eq. (9)]:

$$\langle \Delta q_{\ell+1}^T \Delta q_{\ell+1} \rangle_n = \sigma_n^2 \sum_{\mu_{\ell+1}, \mu_{\ell}} [W_{\ell}]_{\mu_{\ell+1} \mu_{\ell}}^2 |h_{\ell}]_{\mu_{\ell}}|^{\alpha}. \quad (17)$$

We thus define the multilayer network WA loss $L_{\text{WA}}^{(\alpha)}$ as

$$L_{\text{WA}}^{(\alpha)} \equiv \sum_{\ell} \sum_{\mu_{\ell+1}, \mu_{\ell}} [W_{\ell}]_{\mu_{\ell+1} \mu_{\ell}}^2 |h_{\ell}]_{\mu_{\ell}}|^{\alpha}, \quad (18)$$

which involves the product of the outgoing squared weights with the squared presynaptic neural activations. We add Eq. (18) as a regularizer while training a deep nonlinear network on the same task setting as in Fig. 1 and find through the Laplacian eigenspectra and visualization of the connectivity matrices that it drives modularity emergence without noise injection for both $\alpha = 1$ (Poisson-like noise; see Fig. 14 in Appendix D) and $\alpha = 2$ (multiplicative noise) [Fig. 4(a)].

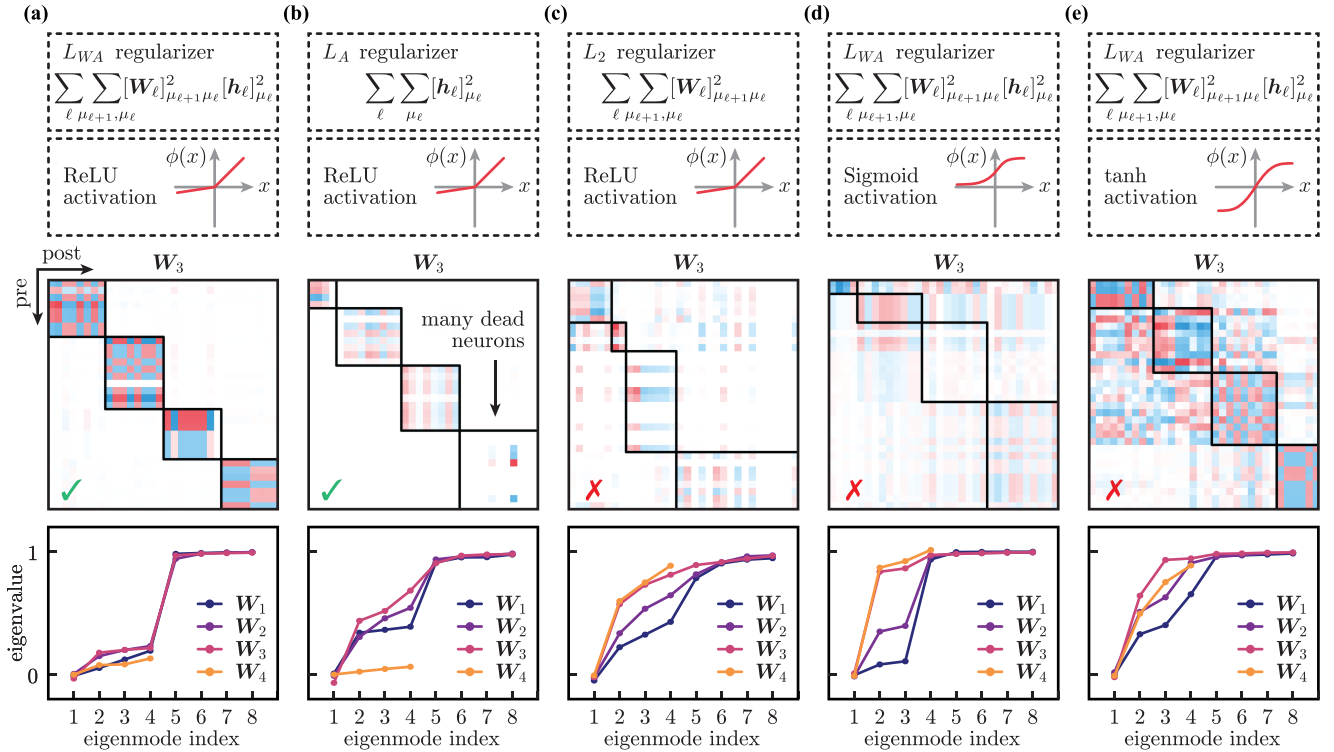


FIG. 4. The WA regularizer L_{WA} drives modularization in deep nonlinear networks. (a) The L_{WA} regularizer combines the L_2 -norm and activations and recapitulates the ANN modularization behavior seen with noise injection. (b) Compared with an activation regularizer, the weighted version avoids dead neurons as no penalty is imposed on neurons that have small input weights. (c) An L_2 regularization does not promote modular connectivity structure. (d) The error analysis in Eq. (13) suggests the neuron activation function acts as a switch on the weight regularizers, and a sigmoid activation function should have a weaker effect compared to a ReLU. (e) Similarly, the plateauing behavior of the tanh activation does not allow modularization to emerge either.

For brevity, we set $\alpha = 2$ in the following discussion and write $L_{WA} \equiv L_{WA}^{(2)}$. The WA regularizer promotes modularity in multilayer nonlinear networks even though we did not fully propagate the effective noise errors through the network when deriving the regularizer.

We compare L_{WA} with other regularizers, including the activity loss

$$L_A \equiv \sum_{\ell} \sum_{\mu_{\ell}} [h_{\ell}]_{\mu_{\ell}}^2, \quad (19)$$

and test the effects of different activation functions. Activity regularization L_A can also lead to discovery of a modular solution [Fig. 4(b)], consistent with earlier findings that penalizing neural activity can make ReLU neurons respond to specific task factors [14]. However, it results in so-called dead neurons that remain inactive and generates numerical destabilization of the graph Laplacian [Fig. 4(b)]. With L_{WA} , neuron activity regularization becomes stronger when its outgoing connection weights are larger, and vice versa. This tuning of neuron-level regularization strength by the readout weight matrix avoids dead neurons since neurons with near-zero weights experience almost no activation penalty and remain in a position to become active again.

The normal L_2 weight regularization

$$L_2 \equiv \sum_{\ell} \sum_{\mu_{\ell+1}, \mu_{\ell}} [W_{\ell}]_{\mu_{\ell+1}, \mu_{\ell}}^2 \quad (20)$$

results in no modularization [Fig. 4(c)], as predicted in the preceding section. To test our prediction that saturating nonlinearities should thwart modularity, we simulate networks with tanh and sigmoid nonlinearities while maintaining the L_{WA} regularization. Both activations lead to less modularized connectivities than ReLU [Figs. 4(d) and 4(e)]. There is slightly more modular structure with sigmoid than tanh neurons, attributable to the fact that tanh reduces to a linear form $\tanh(x) \approx x$ for small x , and in the preceding section we derived that linear responses do not promote modularity.

We additionally show training results with $L_{WA}^{(1)}$, L_1 , and L_{12} regularizers, or with dropout noise, in Fig. 14 in Appendix D. The L_1 regularizer promotes high sparsity, leading to unstable graph Laplacian spectra. Applying the L_{12} regularizer from Eq. (15) layerwise does promote modular connectivity, but also suffers from a moderate degree of dead neurons. We hypothesize that dead neurons appear despite the weighted nature of the L_1 -like suppression in Eq. (15) because only neurons already contributing to functional performance can increase in activity. If the connection weights to a neuron drop to 0 too early in training, the vanishing gradient prevents their reactivation even if that would benefit performance. Finally, dropout noise, where the activation of a neuron is randomly set to 0 in training (probability 0.05 in our simulations), can be treated as an injected noise whose variance scales quadratically with the neuron activation. This is because for an activation h , the noise η due to dropout is $-h$ with probability $p = 0.05$ and 0 otherwise, leading to $\langle \eta^2 \rangle = (-h)^2 p + 0(1 -$

$p) = h^2 p$, so it also promotes modular connectivity, similar to multiplicative noise.

C. Simultaneous discovery of modular basis and modular solution

The WA regularization discussed above promotes the emergence of fully modular connectivity in tasks that involve modular functions of the inputs, where the inputs are already disentangled; they are already the correct basis in which the tasks are modular. However, if the task requires a step that mixes across the modular streams, can the regularizer L_{WA} still find an appropriate modular solution? Relative to the required mixed inputs, the unmixed network inputs are effectively entangled; thus the network has to simultaneously find the correct mixing/demixing of the inputs to find the basis in which there is a modular solution while finding the modular solution.

Therefore, we consider tasks in which as before the target outputs involve modular functions $\mathbf{f} = [f_1(x_1), f_2(x_2), f_3(x_3), f_4(x_4)]^T$, but also involve a mixing step

$$\mathbf{y}_{\text{early}} \equiv \mathbf{f}(\mathbf{R}\mathbf{x}), \quad (21a)$$

$$\mathbf{y}_{\text{late}} \equiv \mathbf{R}\mathbf{f}(\mathbf{x}), \quad (21b)$$

where the matrix $\mathbf{R}$ mixes two of the independent factors together. The early mixing target requires inputs to be mixed before application of the independent functions [Fig. 5(a), top]; the late mixing target requires computation of the modular functions followed by mixing their outputs [Fig. 5(a), bottom].

In simulations we picked a mixing matrix that rotates factors 2 and 3 by angle $\gamma = \pi/4$,

$$\mathbf{R} \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \gamma & \sin \gamma & 0 \\ 0 & -\sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (22)$$

and trained identical fully connected ANNs with three hidden layers and the WA regularizer L_{WA} on these two tasks. Strikingly, all layers of the trained networks became modular in both cases, with most layers exhibiting a four-module structure, suggesting they have discovered that four independent nonlinear factors [Fig. 5(b)] comprise the task. For the early mixing task, the first hidden layer $\mathbf{W}_1$ acquires a three-module connectivity [three near-0 eigenvalues and the fourth eigenvalue closer to 1; Figs. 5(b)–5(d), top]. Subsequent layers then reflect the fully modular downstream aspect of the early mixing task. In other words, the network for early mixing task mixes as required the independent factors, doing so at the earliest possible point and only at that point, and then all subsequent layers develop a fully modular unmixed structure.

In contrast, for the late mixing task, the network creates a fully modular propagation and function computation on the four independent inputs through its layers but the last and performs the required mixing of the functions in only the last layer [Figs. 5(b)–5(d), bottom]. If we consider the regularizer as mimicking the effects of noise injection, then the network behavior we found here indicates that the mixing of variables incurs excess noise error so that the network works to max-

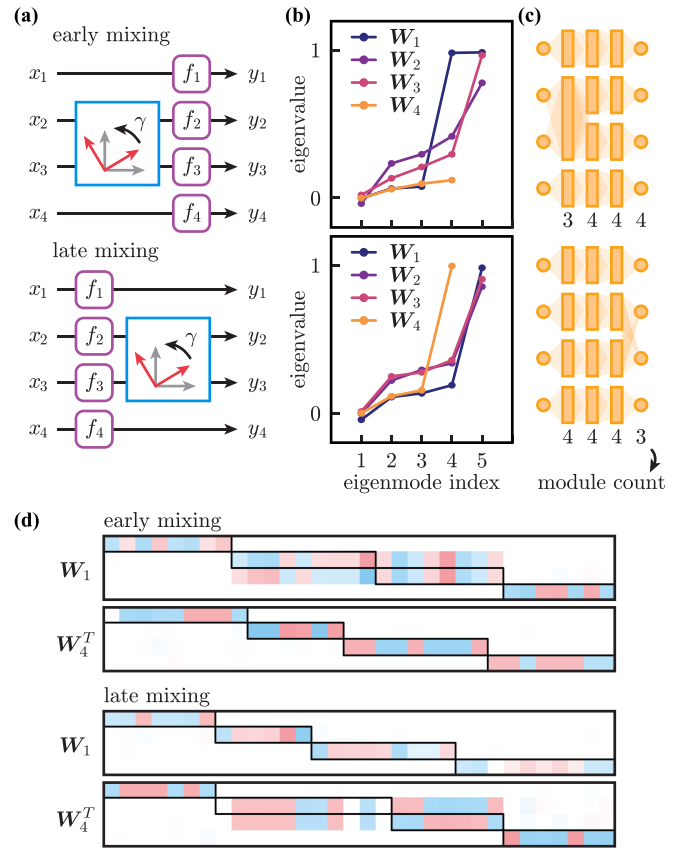


FIG. 5. The L_{WA} drives ANNs to discover demixed independent task factors within the fewest possible layers. (a) We compare two learning objectives under the WA regularization (18): early mixing of inputs according to Eq. (21a) (top) and late mixing of outputs from nonlinear functions according to Eq. (21b) (bottom). The mixing function is a linear operator represented by the blue square, and nonlinear functions are in purple. (b) Examining the Laplacian matrix of network connectivities reveals that in the early mixing case (top), the first layer has a three-module structure and later layers have four modules, indicating the first layer decomposes the two middle inputs and the later layers then compute the nonlinear functions $f_i(x)$. In contrast, for late mixing (bottom), the first three layers have four modules that compute the $f_i(x)$'s on their own, and the outputs from the middle channels are mixed only at the final layer. (c) Schematic of the distinct modularization behavior, with the number of modules counted from the Laplacian spectra stated at the bottom. (d) Connectivity matrices of the first and last layers of these networks, showing channel crosstalk in the first layer for early mixing (top) and the last layer for late mixing (bottom). Other intermediate layers all have a four-cluster blocky structure.

imally disentangle and keep independent task factors across its layers, mixing only when necessary and doing so within the fewest layers. Conversely, one could use this observation to infer, in general tasks, the degree of nonlinear processing in the training data by examining how an ANN with L_{WA} regularization demixes task factors.

IV. SAMPLE COMPLEXITY FOR DISCOVERY OF MODULAR STRUCTURE

We have established that the WA regularizer drives the discovery of rare solutions with modular network

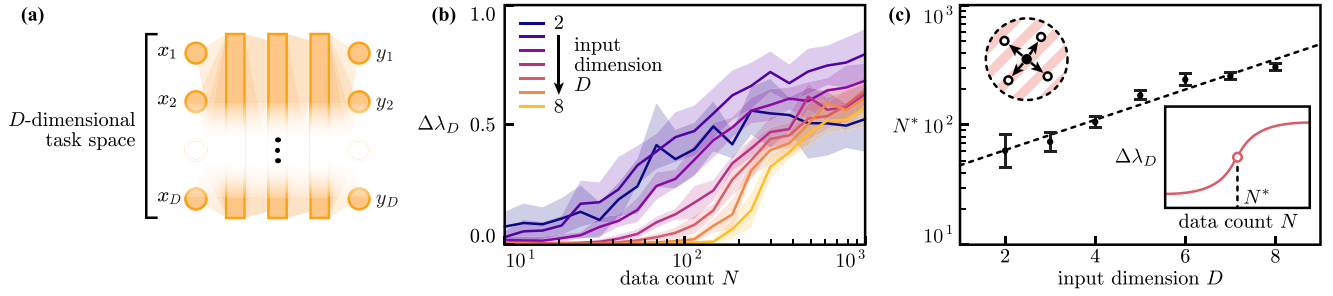


FIG. 6. Sample complexity of modularity emergence through WA regularization. (a) To study the data complexity of network modularization under L_{WA} regularization, we train networks to learn D -dimensional modular functions. (b) The eigengap $\Delta\lambda_D$ corresponding to a D -module connectivity shows a sigmoidal growth trend with N (number of unique data samples per epoch) on a logarithmic scale. (c) For each curve, we identify as N^* the critical data count required for a network to adopt a modular structure (right inset), and an exponential scaling of N^* with input dimension D is revealed. This scaling is consistent with a picture where modular connectivity emerges when training data points come within some critical distance to one another, such that the local function gradient can be estimated and a modular task structure discovered by the network (left schematic). The dashed line is an exponential scaling reference.

connectivity after training while networks without WA regularization do not. However, we do not yet know the sample complexity of this discovery process. We next ask how rapidly modular structure is discovered and acquired, by investigating the sample complexity of modularity emergence with the WA regularizer as a function of D , the number of input dimensions (so far we used $D = 4$) [Fig. 6(a)].

We train networks with varying D (the hidden layers scale linearly in size with D) and as before quantify network modularity by the D th eigengap $\Delta\lambda_D$. We plot the value of this gap as a function of the number of training datapoints N [Fig. 6(b)]. Modularity emerges later for larger D [Fig. 6(b)]; we fit $\Delta\lambda_D$ to the sigmoidal function $\Delta\lambda_D = \Delta\lambda_D^{\max} / \{1 + \exp[-v(N - N^*)]\}$ with steepness v and transition point $N \sim N^*$ [Fig. 6(c), right inset]. Plotting the resulting transition point N^* against D reveals an exponential scaling of the number of training data points N^* for modularity emergence as a function of the number of modular task dimensions [Fig. 6(c)]. When, instead of D independent input dimensions and D outputs that depend on these, the task depends on a smaller number of independent latent dimensions (e.g., if the dimensions are correlated with each other), the scaling of task complexity depends on the smaller number of independent latent dimensions (see Fig. 15 in Appendix E). The present results should thus be understood as the scaling with the dimension of the latent space, rather than the explicit dimensionality.

To understand the origins of exponential scaling in modularity emergence, consider the transition from a nonmodular network structure to a modular one in a flexible learner. The transition requires the learner to infer from the training data whether and which function gradients are zero, $\partial_j f_i \equiv \partial f_i / \partial x_j = 0$ for $i \neq j$. Around a reference training point x_i^1 (with target y_i^1), another training point x_i^a (within a hypercube of side length Δx^* of the reference, with target y_i^a) provides information on the first-order gradients via $y_i^a - y_i^1 \approx \sum_j \partial_j f_i(x^1)(x_j^a - x_j^1) \equiv \sum_j \partial_j f_i(x^1) \Delta x_j^a$, assuming that the terms Δx_j^a (and therefore Δx^*) are small enough that the higher-order terms such as $\partial_j^2 f_i$ vanish. Within this first-order approximation, local gradients can be estimated by solving the simultaneous equations $\Delta y_i^a = \sum_j \Delta x_j^a \partial_j f_i(x^1)$, which requires (pseudo)inversion of the data matrix Δx_j^a . This matrix

attains rank D only if there are at least D training points in the hypercube; in addition, higher-order terms act as a source of noise under the linear approximation, so it would require a scaled factor of $\beta > 1$ more data samples, where β increases with the effective noise and inversely with the desired squared precision of gradient estimation. Thus, we need approximately βD data points within each hypercube of length Δx^* . Suppose the full domain is a hypercube of length L , the expected number of points within the small hypercube is $N^* (\Delta x^* / L)^D$. Equating the two, we get $N^* \sim \beta D (L / \Delta x^*)^D$.

The sample complexity of modularization by a fully flexible learner (capable of learning any function) is therefore exponential in D to leading order [$O(Dm^D)$, where m denotes the number of points needed for one dimension $D = 1$]: This is the curse of dimensionality, which arises from the fundamental requirement of gradient sensing, in which a minimum number of samples is required per dimension to determine whether there is a nonzero or zero gradient in that direction [3]. Deep network learning with the WA modularity regularizer remains flexible; though it promotes discovery of the modular solution when one exists, from the vast set of functions that satisfy the task, it nevertheless remains capable of learning nonmodular solutions. The argument above and the results from Fig. 6(c) show that WA regularization is distinct from making a learner *a priori* aware of the specific modular structure of the task. Given that a fully modular task is in principle equivalent to D copies of single-input, single-output function-learning tasks, a learner fully aware of its specific modular structure could optimally learn the task with a sample complexity as small as $O(m)$. In sum, the WA regularizer enables the discovery of task modularity and finds the correct modular solution, but does not solve the curse of dimensionality in doing so.

Having explored basic properties of modular emergence, we next turn to functional benefits of modular networks.

V. FUNCTIONAL BENEFITS OF NETWORK MODULARITY

A. Noise robustness

Since the WA regularizer is derived from effective errors in the presence of activation noise, we expect L_{WA} -regularized

networks to perform better under noise injection, even though they are trained without noise, compared to networks trained without the WA regularizer. Mechanistically, each output node of a trained modular network is a function of a single variable, for instance, $y_1 = f_1(x_1)$ for the first output node. In contrast, a nonmodular network computes functions of irrelevant inputs but attains the same performance by learning that the value of the parts of the function that depend on irrelevant variables in the training data are zero: $y_1 = f_1(x_1) + g_1(x_2, x_3, x_4)$ with $g_1(x_2, x_3, x_4) = 0$. In the presence of noise during testing, the cancellation of the irrelevant inputs in g_1 becomes nonexact and the nonmodular network output acquires the extra term $\Delta g_1(x_2, x_3, x_4) > 0$, which acts as an additional source of error [Fig. 7(a)]. We characterize and compare the noise robustness of L_{WA} -regularized and L_2 -regularized networks when training over different amounts of data. The regularizers are applied during training, and activation noise of strength $\sigma_n = 0.1$ and scaling exponent $\alpha = 2$ (multiplicative noise) is injected into the network during testing. Indeed, as the number of training data points N increases and the L_{WA} -regularized network becomes more modular [Fig. 7(b), bottom panel], in tandem the test loss of the L_{WA} network begins to drop below that of the L_2 -regularized network [Fig. 7(b), top panel].

B. Prunability and functional independence

Modularity emergence with the WA regularizer drives many entries close to 0, and we find that the resulting weights are especially amenable to pruning to generate purely modular or independent input-output streams in the pruned networks.

In networks trained with the L_{WA} regularizer, the weight magnitude histograms of each layer are bimodal, unlike for networks trained with L_2 regularization [Fig. 8(a)]. We prune weights by fitting a double Gaussian distribution for weight magnitudes in each layer in logarithmic space, defining a threshold based on the mean and standard deviation of these Gaussians, and then setting to 0 all weights below the threshold. The threshold is chosen to be $(\mu_1/\sigma_1 + \mu_2/\sigma_2)/(1/\sigma_1 + 1/\sigma_2)$, where μ_1 , μ_2 , σ_1 , and σ_2 are means and standard deviations of the two fitted distributions.

We test whether the pruned networks truly implement independent functions from input to output: After the functions $f_i(x)$ are learned, we shift the input range of x_1 , going from $x_1 \in (-2, 2)$ in training to $x_1 \in (-2 + T, 2 + T)$ in testing, where T can be very large. The output y_1 is not expected to resemble its target because it was never trained in that range, but if the pruned network is implementing independent functions, the change in x_1 should not affect the correctness of y_2 , y_3 , and y_4 .

We find stark differences between L_{WA} - and L_2 -regularized networks: They begin with similar performance in the small- T regime, but as T increases in magnitude the pruned L_2 -regularized network quickly acquires large MSE losses in y_2 , y_3 , and y_4 [Figs. 8(b) and 8(c), gray]. This occurs because the L_2 -regularized network depends on appropriate cancellation of nonrelevant inputs in intermediate layers, and the learned cancellation deteriorates as one of the nonrelevant inputs ventures outside the training distribution. In contrast, the remaining outputs in the L_{WA} -regularized network remain accurate regardless of T , because it had already almost completely modularized its computation throughout all layers, and

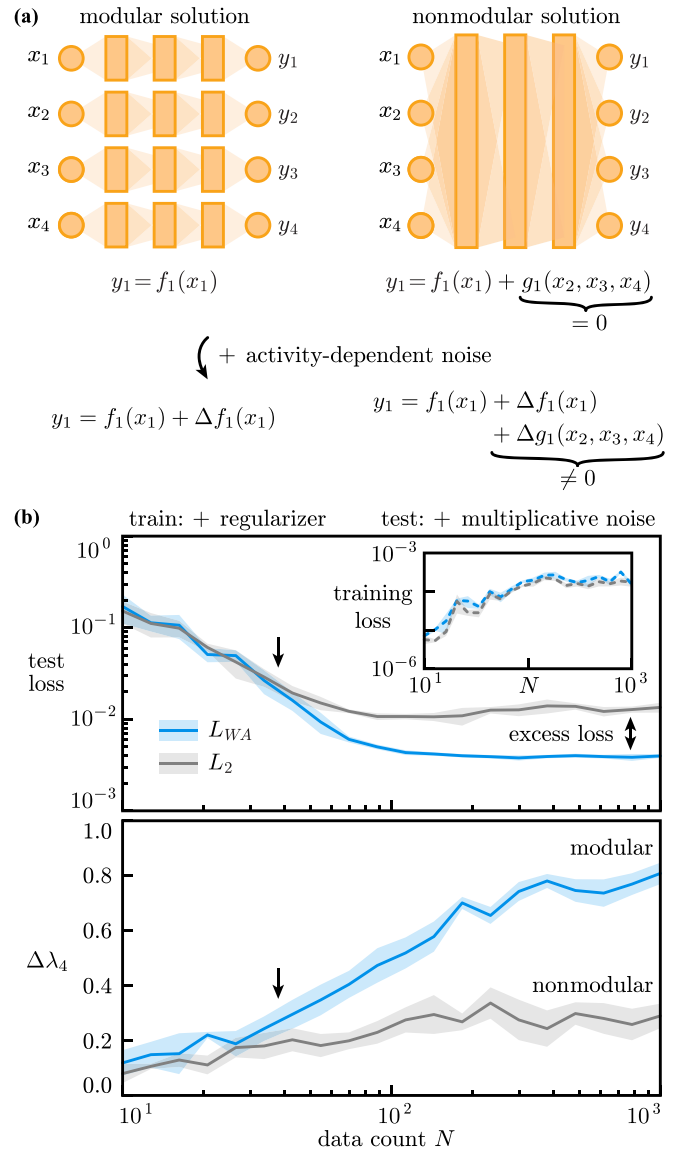


FIG. 7. Noise-robustness from modularity. (a) An output node from a modular network depends only on the relevant input, but that from a nonmodular network can have dependences on other inputs that cancel out, represented by an additional term $\epsilon_2(x_2, x_3, x_4)$ that is identically 0. However, in the presence of noise, this extra term can lead to nonzero fluctuations. (b) We train networks with WA or L_2 regularizers, with a varying number of training points N , and test the performance with explicit multiplicative noise in activation layers of strength $\sigma_n = 0.1$. WA-regularized networks outperform L_2 -regularized ones with sufficient N (top) and this occurs in tandem with the emergence of connectivity modularity, indicated by an increase in the fourth eigenvalue gap of the first connectivity weight matrix (bottom).

even small residual crosstalk was removed through pruning [Figs. 8(b) and 8(c), blue]. Thus, modular connectivity emerging from L_{WA} -regularized weights enables effective pruning to achieve exact modularity.

C. Disentangled processing and compositional generalization

We next explore the effects of modularity as a preexisting architectural bias, which we may view as established through

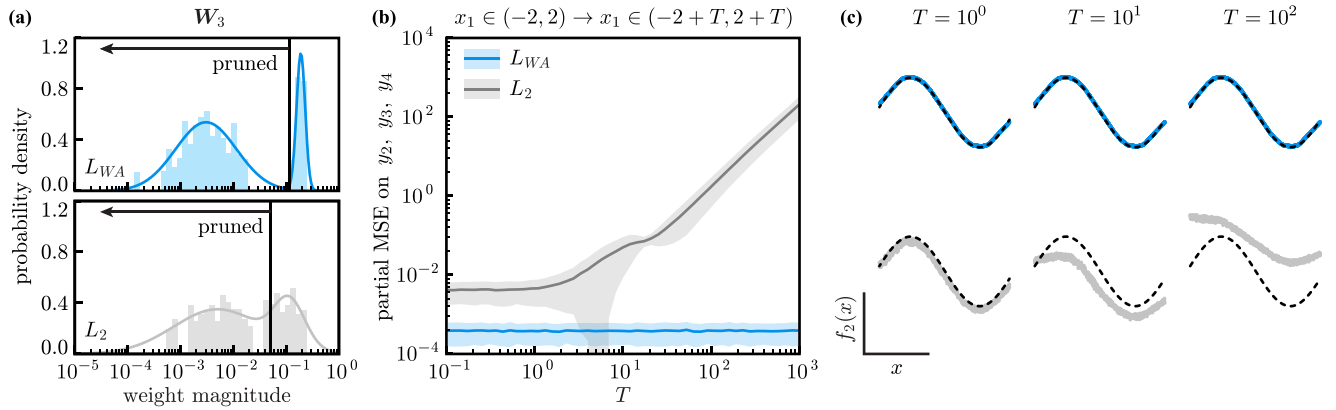


FIG. 8. Modularized networks can be effectively pruned to achieve functional independence. (a) Histograms of weight magnitudes of the third layer W_3 for models trained with L_{WA} (top) and L_2 (bottom) regularization as in Fig. 4. The modular network exhibits a distinct two-peak distribution. We fit a double Gaussian distribution to each histogram and prune the small weights by setting them to 0. (b) Functional independence of the inputs and outputs can be demonstrated by translating the range of one of the inputs x_1 by an amount T to be far from the training regime. The pruned L_{WA} -regularized network (blue) did not show perturbations on the other outputs y_2, y_3 , and y_4 . However, an L_2 -regularized network (gray) exhibits perturbations on these outputs, so the outputs are not truly independent of each other. (c) Scatter plots of the second output at a range of T values for pruned networks trained with L_{WA} (top, blue) and L_2 (bottom, gray) regularizations.

evolution or through prior learning on related tasks with a WA-type regularizer, and show that modular information processing channels enable learning with far-out-of-distribution generalization through compositionality. We consider learning of two-input, one-output functions of the form

$$f(x_1, x_2) = \chi[\psi_1(x_1) + \psi_2(x_2)], \quad (23)$$

where χ , ψ_1 , and ψ_2 are single-variable functions. As universal function approximators [26–30], any network with two inputs, one output, and reasonable size should be able to learn Eq. (23). To test qualitative effects of network architectures, we compare three different feedforward ANN architectures: modularized, fully connected (FC), and bottleneck networks [Fig. 9(a)]. Only weight decay is used as a regularization method here. The inputs in the modularized network project disjointly to the first hidden layer and then are combined in the downstream layer. The FC network has the same number of units and hidden layers but is fully connected (no architectural constraint). The bottleneck network resembles the modularized network, except that the first hidden layer is FC instead of disjoint.

We hypothesize that the modularized network is biased toward learning the true single-variable component functions χ , ψ_1 , and ψ_2 , while the FC and bottleneck networks memorize the output value given an input. If so, then when the input to the trained networks is drawn from an out-of-distribution (OOD) domain with no in-distribution (ID) support, the modularized network should be able to generalize but the other networks would not. Conceptually, consider the (very high-dimensional) space of all functions [Fig. 9(b)]. The modularized network is equivalent to the FC network with certain weights constrained to 0; thus the set of functions allowed by the modularized network is a subset of those representable by the FC network [Fig. 9(b), red and orange regions]. Now consider the data distribution. Each dataset places constraints on candidate functions. The full dataset (ID and OOD) specifies a smaller subset of functions than the ID dataset [Fig. 9(b), black and blue, respectively]. If trainable on

a specific dataset, a network with a specific architecture will converge to a function in the intersection of the architectural subset and data constraint subset. Ideally, for the modularized network these sets would intersect at a low-dimensional manifold representing the ground truth function [Fig. 9(b), star], allowing generalization to occur before the full dataset is presented to the network.

To test this, we train networks to learn a two-dimensional (2D) Gaussian function

$$f(x_1, x_2) = \exp\left(-\frac{1}{2\sigma^2}(x_1^2 + x_2^2)\right), \quad (24)$$

with $\sigma = 1/2$. The networks are trained on the ID dataset comprising all points satisfying $x_1 x_2 > 0$ within the unit circle $x_1^2 + x_2^2 < 1$, while the OOD dataset is the complementary set where $x_1 x_2 < 0$, also within the circle [Fig. 9(c)]. When we compute the mean square error loss for ID and OOD sets as the networks are trained, we observe that the OOD loss nearly matches the ID loss for the modularized network [Fig. 9(d), left], suggesting it has learned the component functions, while the FC and bottleneck architectures exhibit a large generalization gap [Fig. 9(d), middle and right]. To further verify that the modularized network has learned the component functions, we plot the intermediate outputs z_1 and z_2 from the modular blocks [Figs. 9(a) and 9(e)]. For a Gaussian function, these should be quadratic functions of the inputs x_1 and x_2 , and indeed the modular block outputs are quadratic functions [Fig. 9(e)], up to a multiplicative and additive coefficient, which represent gauge freedoms from the weights and biases of the downstream layer. In addition, we test the effect of different ID/OOD splits by parametrizing these sets as fan-shaped regions and find that the modularized network learns the true function with much less ID data than the other networks (see Fig. 16 in Appendix F). In further simulations, we also find that fully connected networks trained with the WA regularizer do not learn the modular structure of the task. This is because the explicit modular architecture is a stronger bias than the regularizer, which can overcome the lack of

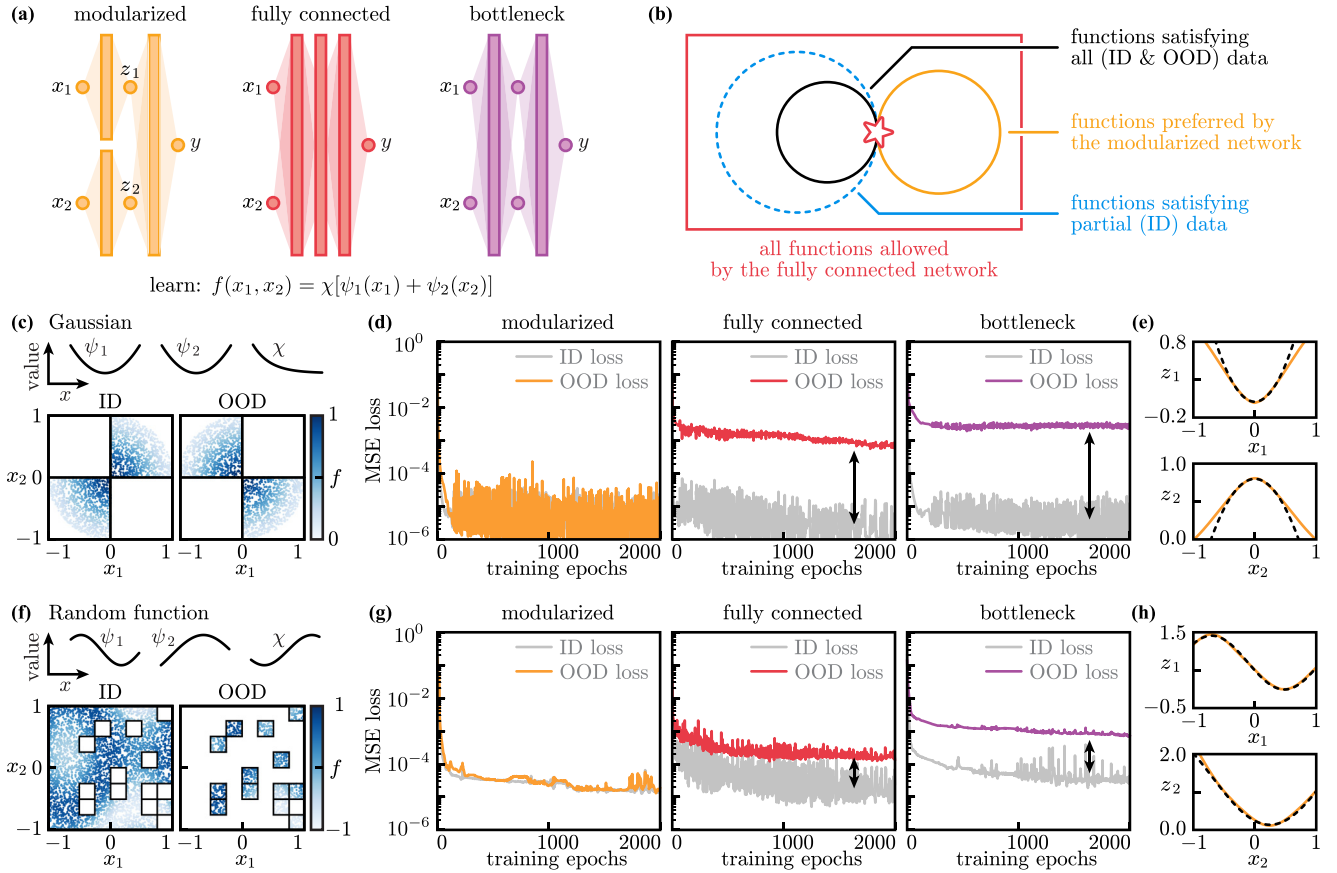


FIG. 9. Modularity-favoring architectural bias for strong out-of-distribution compositional generalization. (a) Schematics of the architectures tested. Each circle represents a single neuron unit, and the rectangles represent a group of neurons. (b) The data distribution and network architecture place constraints on the function achievable by the ANN, and generalization occurs when the constraint spaces only intersect each other on low-dimensional manifolds. (c) The Gaussian task requires an ANN to learn a two-dimensional Gaussian function, and we split the full domain into a training (ID) part and a testing (OOD) part, each comprising two quadrants within a unit circle. (d) Testing different network architectures reveals that the modularized network (left) has learned the Gaussian function across the whole domain even with partial data, since the ID and OOD loss completely overlap. In comparison, the full (middle) and bottleneck (right) networks still exhibit sizable generalization gaps. (e) The compositional generalization capability of the modularized network stems from individual modules learning quadratic functions up to additive and multiplicative gauge factors. Orange lines are the outputs of the modules, and dashed lines are quadratic functions. (f) We test the generalization bias in a more generic task setting where the single-variable functions are randomly generated. The ID and OOD regions are unions of randomly selected, nonoverlapping patches. (g) Plotting the ID and OOD losses shows that the same generalization phenomenon is recovered as in the Gaussian task. (h) Module outputs agree with the component functions ψ_1 (top) and ψ_2 (bottom), up to scaling and an additive constant.

data in the OOD region. Further comparing this observation to the early/late mixing setup of Fig. 5, it is also possible that the complexity of the functions comprising the overall learning goal also influences how the network modularizes. In Fig. 5, the final network structure correctly reflects the task setups, for a mixing function that is a linear transformation. In the Gaussian learning task, the mixing function is nonlinear (exponential), and the greater functional complexity could make the learning problem harder, therefore requiring the architectural constraint as a strong inductive bias for learning of the correct processing steps.

To test the generalizability of this result, that a modular network learns the modular structure of functions, we train these networks on generic modular function composition tasks. Each task selects component functions ψ_1 , ψ_2 , and χ from a set of sinusoidal waves with randomly sampled frequencies and phases. The ID and OOD sets here are nonoverlapping

random patches [Fig. 9(f)]. Consistent with the Gaussian result, learning this task with the modularized network results in excellent OOD generalization, while the other architectures do comparably on ID data but much worse on OOD data [Fig. 9(g)]. When we examine what functions are learned in which layers, we find that the modules in the modular layer learn the correct nonlinear functions up to scaling [Fig. 9(h)].

Having demonstrated the inductive bias of modularized networks to learn component functions, we further ask what causes the network to selectively localize modular function learning in the modular layers, as in the results of Fig. 9, rather than smearing the modular function computation across all layers. If a network has modularized blocks before the FC layers, it could in principle perform most processing in the FC layers and implement identity functions in the modular layers. We examine this question in more detail by returning to the modular function-learning task of Eq. (1) [Fig. 10(a)],

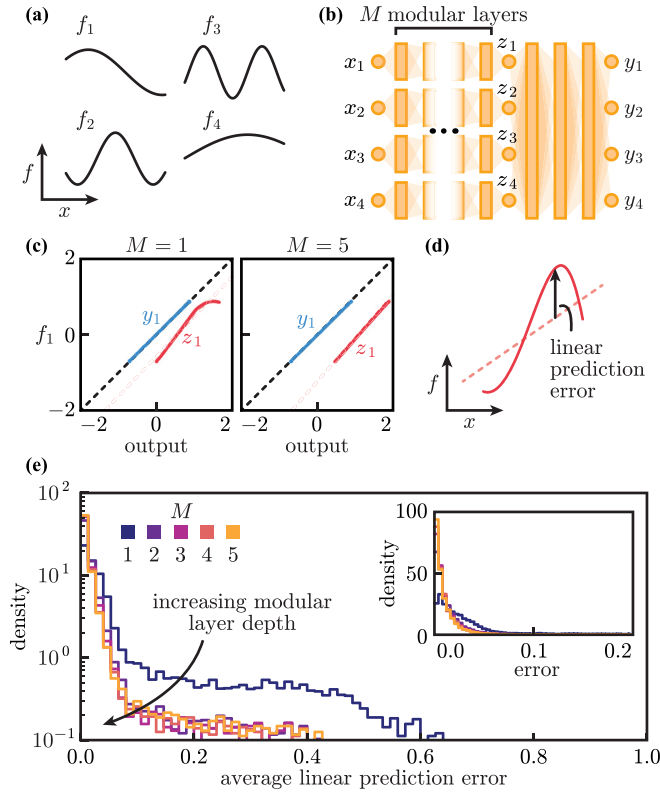


FIG. 10. Modularized blocks are predisposed to encode nonlinear transformations. (a) We revisit the task of learning four independent single-variable functions. (b) The modularized network contains M layer-deep modularized blocks with intermediate outputs $z_1, \dots, z_4$ and a fully connected block of fixed depth $H = 3$. (c) For small M the modular block outputs nonlinearly deviate from the desired outputs (deviation of the red solid curve from the red dashed curve). For larger M , the modular blocks output a response that is proportional to the final output y_1 , meaning they have learned the modular functions up to linear scaling. (d) Quantifying the deviation between the nonlinear modular target functions and the output of the modular blocks by fitting a straight line in a plot of target value and output. The deviation from this linear predictor reflects the residual error. (e) Histograms of the errors from (d) while parametrically varying M , for different functions and random seeds, showing smaller errors in the modular outputs as M increases. The inset shows the same plot in linear scale.

using a four-function task on four inputs and four modules [Fig. 10(b)], with three FC layers after the modular blocks. We study how the network partitions the computation of the individual functions across layers as we vary the depth M of the modular blocks. The total number of neurons in each modular layer is the same as the number in each FC layer.

Computing this deviation from linear for many different functions and random seeds, we observe less deviations as we parametrically increase M [see Figs. 10(e) and 17]. Interestingly, there seems to be an abrupt transition: Once the modular blocks are sufficiently expressive (i.e., with enough depth, which in this case is $M = 2$), nonlinear modular function learning localizes to them, even if the FC layers are deeper [overlapping curves for $M > 2$ in Fig. 10(e); see also Figs. 17 and 18 in Appendix G]. Furthermore, though the FC layers

learn part of the nonlinear modular target functions when the modular layers are shallow ($M < 2$) and the proportion of networks with the FC layers learning part of the nonlinear modular targets increases as the FC layers increase in depth, this stops when $M \geq 2$: Then, varying the FC layer depth does not change the proportion of networks where the FC layers compute part of the nonlinear modular targets (see Fig. 18 in Appendix G). The analytical basis of this finding can be an interesting direction for future exploration.

In sum, we have empirically found another inductive bias of the modular architectural constraint: In the presence of both modular and fully connected blocks, independent nonlinear operations are preferentially encoded in the modular blocks when those are sufficiently expressive (above a critical depth that depends on the targets but not on subsequent FC layer depth) even though in principle they could have been localized to the FC layers or spread across both.

D. Preventing adversarial attack in a naturalistic task

The synthetic tasks considered so far provide insights into the modular architecture emergence process. We next explore a more naturalistic task, on the classification of multiple images together with adversarial attack. The task involves providing D labels for a multiframe concatenation of D CIFAR10 image inputs [31]. The images are individually drawn randomly within and between classes, so the task has a modular structure. The output dimension is $10D$, where each group of ten outputs are label logits for one image in the strip of images. We use a backbone of convolutional layers from ResNet18 [32], a standard architecture for image feature extraction, followed by a fully connected feedforward classifier network with one hidden layer [Fig. 11(b)]. The L_{WA} regularizer can be applied to the hidden layer of the fully connected classifier network, similar to the synthetic tasks. Computing the graph Laplacian eigenspectra of the final layer weights for networks trained with and without WA regularization at $D = 4$ shows that four distinct clusters are formed when L_{WA} is used, indicating modular connectivity emergence. The network trained without L_{WA} remains nonmodular [Fig. 11(c)]. The data complexity of this process when varying D is similar to the function-learning analog (see Fig. 19 in Appendix H).

The modular classifier arising from WA regularization splits features extracted from each image input into disjoint groups before class labels are produced. Therefore, we hypothesize that the network will be more robust against adversarial attacks that perturb one image to change the prediction label of an unperturbed image, similar to the functional independence trait discussed in Sec. VB. In the $D = 4$ setting, we perturb the first image in the input and apply the perturbation to maximally change the class label output of the second image [Fig. 11(d)]. In a completely modular network, features extracted from the first image should not contribute to class label prediction of the second image, and the attack-driven flip rate is indeed low for the network trained with WA regularization [Fig. 11(e), orange] at a range of perturbation strengths. The perturbation strength is measured as a fraction of the maximum pixel intensity value allowed, and the nonzero flip rate stems from the fact that the induced modularity is not perfect. In contrast, a network trained without WA

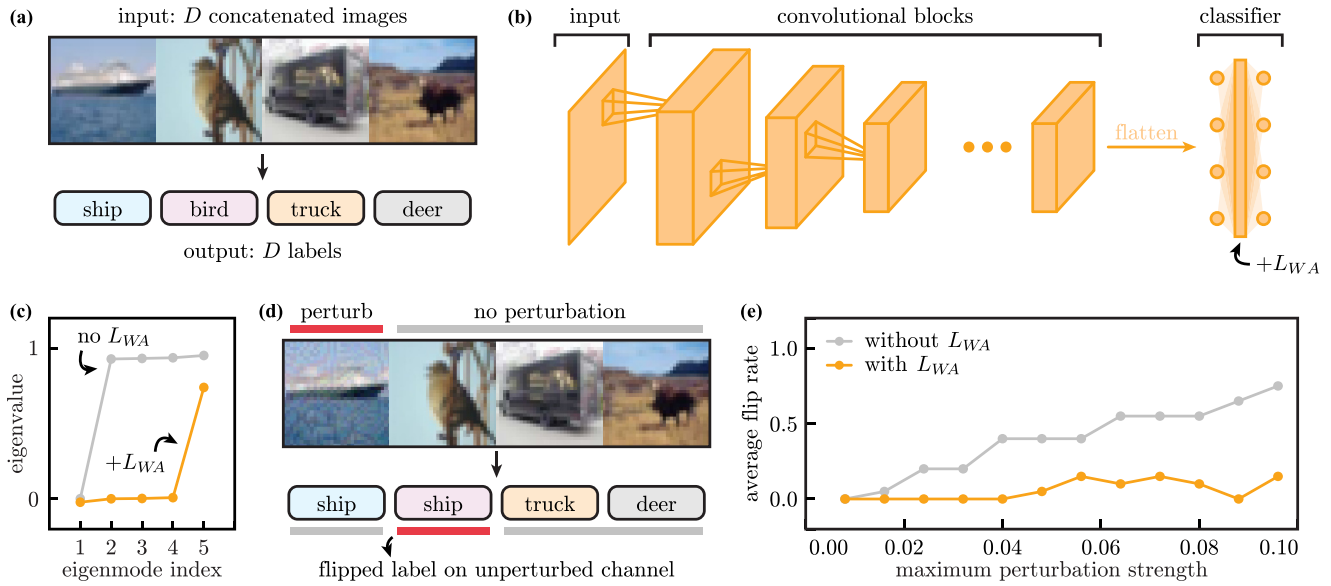


FIG. 11. Modular structure emergence robustifies against adversarial attack in image recognition. (a) The task involves providing D labels for a multiframe concatenation of D CIFAR10 image inputs. The images are individually drawn randomly within and between classes, so the task has a modular structure. (b) Model architecture. We take the convolutional blocks from ResNet18 [32] as the backbone, which outputs a list of features for the concatenated images. The features are further processed by two fully connected layers that act as the D -label classifier. The L_{WA} regularizer is optionally applied at the intermediate layer between features and output logits. (c) Training the network with $D = 4$ and computing the graph Laplacian eigenspectra of the last connectivity matrix shows a distinct four-cluster structure for the network trained with L_{WA} (orange), while this structure is absent for a network trained without the regularizer. The matrix dimension is 2048×40 (2048 hidden units and 40 logits for four groups of ten image identities), and only the first five eigenvalues are plotted. (d) To probe the functional benefit of the modular classifier, we perform a cross-channel adversarial attack on the trained model. We perturb the first image in the input, with the goal of changing the label of the second image. (e) At increasing perturbation strength (measured relative to maximum pixel intensity), the probability of a successful attack increases quickly for the nonmodular network (gray), but this probability remains low for the modular network (orange) since the information leakage from one image channel to another is diminished. Twenty attacks are performed to calculate the success rates.

regularization is much more susceptible to label flips from this type of adversarial attack [Fig. 11(e), gray]. Thus, the WA regularization has the potential to be relevant and beneficial in naturalistic tasks too.

VI. DISCUSSION

While noise injection has been viewed as a mechanism to perform gradient estimation [33–35] or to prevent overfitting [36], our work demonstrates that its influence is deeper, shaping the emergent structural properties of ANNs. Specifically, we showed that nonadditive, activity-dependent noise (e.g., Poisson-like or multiplicative) leads to the spontaneous emergence of modular structure to solve modular tasks in nonlinear networks. Characteristics of the nonlinear neuron activation functions are crucial for this process too. This result parallels longstanding insights in neuroscience, where neural noise is not simply tolerated but is hypothesized to support robustness, fault tolerance, and even computation itself [37]. Our findings suggest that similarly, in artificial systems, noise can serve as an inductive bias toward modularity, echoing prior computational work on noise-driven structure emergence [1,9].

By analytically examining the excess loss induced by noise in a single-layer network, we derived the WA regularizer as a deterministic analog mechanism to noise-driven modularity. This regularizer penalizes the product of weights and acti-

vations, capturing the modularizing effects of Poisson-like or multiplicative activation noise in a deterministic form. Unlike a standard activation regularizer, WA regularization does not suppress neurons indiscriminately; rather, it naturally avoids dead neurons by only penalizing active, highly connected units.

The WA regularizer drives the discovery of exponentially rare modular solutions when the task is modular, but allows the network to remain a fully flexible learner of nonmodular tasks as well. As a result, the discovery and emergence of a modular solution requires a number of training data points that scales exponentially with the dimensionality of the problem. We hypothesize that one potential remedy to the curse of dimensionality is by studying the covariance of the data matrix. It is known that in more realistic tasks such as image recognition, some tasks can be learned using low-order statistics of the naturalistic data, even when the apparent dimensions (number of pixels) is large [38–40], due to nontrivial covariance of the input dimensions. Similarly, for two sets of training data for the modular task (with equal number of samples), if one dataset is organized to have smaller covariance across inputs of different modules, the WA regularization might be able to identify the modular task structure more easily.

We next showed that networks trained with WA regularization possess a number of functional benefits. They are

more robust against neuron noise, are amenable to pruning, and achieve functional independence across modules. Furthermore, starting with modular architecture promotes disentangled downstream information processing, a hallmark of compositional learning. This mirrors phenomena in symbolic reasoning and generative modeling, where compositional generalization hinges not just on disentanglement but on appropriate architectural priors, showing strong out-of-distribution generalization and subexponential data scaling [12], when a modularized network architecture is trained on a modular generative task. This agreement of results suggests that constrained architectures that force inputs to be independently processed or “rendered” could potentially be a generic strategy for enhancing the generalization capabilities of neural networks. Further quantifying the data scaling behavior of our tasks as a function of task dimension would provide insight into how the model learns. Specifically, we hypothesize based on results in [12] that as the modular function learning task dimension increases, the amount of data required by modularized architectures will scale subexponentially with the number of dimensions. Thus, a modular architecture might also provide the appropriate inductive bias to solve the curse of dimensionality for modular problems. This strong performance from architecture has a flip side, which is that it reduces the expressivity of networks relative to unconstrained architectures. However, because our results did not depend on the depth of the downstream FC layers, it is possible that this benefit could be derived without a large loss in expressivity.

Our findings, on the advantages of modularity once present but slow acquisition, may offer insights into the developmental path of the brain’s neural networks. Modularized circuits for sensory, motor, and cognitive processing are widely observed in the brain [41–45]. Many of these circuits are formed *in utero* or in the early postnatal period and then their structures remain largely fixed; these modularized architectures are then reused across the lifetime of the individual as they face diverse tasks. Given our observation on the exponential sample complexity of discovery of modular solutions by a flexible learner, but the strong advantages of learning modular tasks by modular learners, it appears that biology uses evolutionary timescale information to discover the modular structure of problems and then packs and unpacks that evolutionary knowledge into genetically guided and spontaneous activity-guided unfolding of developmental programmes that create modular structures over development, to be used in new learning by the adult individual.

We have seen that modular networks effectively bias learning toward structured factorized functions when the task is modular. More broadly, the Kolmogorov-Arnold representation theorem proves that any continuous real-valued multivariate function can be factorized into a superposition of continuous single-variable functions [46,47]:

$$f(x_1, x_2, \dots, x_n) = \sum_{q=1}^{2n+1} \chi_q \left(\sum_{p=1}^n \psi_{pq}(x_p) \right). \quad (25)$$

In related work, the Kolmogorov-Arnold representation theorem prompted the search for alternative neural network architectures, known as KANs, in which edges learn uni-

variate nonlinear functions and nodes perform simple sums [48–50]. These models can perform interpretable symbolic regression with appropriate regularization and pruning. Our work may be viewed as dual to the KAN models, showing that conventional neural networks can learn solutions that match the modular structure of an actual modular problem without explicit pruning and that additionally they exhibit far greater out-of-distribution generalizability than fully connected networks when there is limited training data. The Kolmogorov-Arnold theorem further suggests that it may be possible to use modular architectures as we have done here to learn interpretable, generalizable approximations to even apparently nonmodular functions. These results lead to important questions for further exploration. What is the relationship between architectural modularity and sample complexity in learning an appropriately modular task? How can we characterize the functional classes allowed by a specific architecture? Future analytical study of how and when the function decomposition transition occurs in our networks could guide the deployment of modular architectures for more performant, generalizable, and explainable networks.

In sum, our work shows how Poisson-like or multiplicative noise, or equivalent deterministic WA regularizers, drive neural networks to discover the vanishingly small space of modular solutions out of the space of all solutions to modular problems, complementing and adding to a recent finding in Boolean networks [9]. We also offer an explanation for the prevalence of modularity in biological brains and a path toward embedding such inductive biases in artificial systems to support more robust, interpretable, and generalizable learning.

ACKNOWLEDGMENTS

D.Q. was supported by Transition Bio Ltd. and the Yang Tan Collective at MIT, K. Lisa Yang Integrative Computational Neuroscience Center at MIT. I.F. was supported in part by the Simons Foundation (SCGB Program No. 1181110), the ONR (Award No. N00014-19-1-2584), and the NSF (CISE Award No. IIS-2151077 under the Robust Intelligence program). The authors thank Dr. Liu Ziyin for helpful comments.

DATA AVAILABILITY

The data that support the findings of this article are openly available [20]; embargo periods may apply.

APPENDIX A: DETAILS ON RANDOM NONLINEAR FUNCTIONS

To generate random nonlinear functions such as those in Fig. 1(b), we write each function as a Fourier sum

$$f_i(x) \equiv \sum_{q=1}^Q A_{i,q} \sin(k_{i,q}x + \phi_{i,q}) \quad (A1)$$

and sample the coefficients $A_{i,q}$, $k_{i,q}$, and $\phi_{i,q}$ from random distributions. We set $Q = 1$ in simulations for simplicity. To choose appropriate ranges for $A_{i,q}$ and $k_{i,q}$, we draw $A_{i,q}$ from $[0.7, 1.0]$ such that the outputs are of order 1, and draw $k_{i,q}$ from $[0.4, 1.8]$ [20]. The range for $k_{i,q}$ is chosen such that the function can be well approximated by piecewise linear

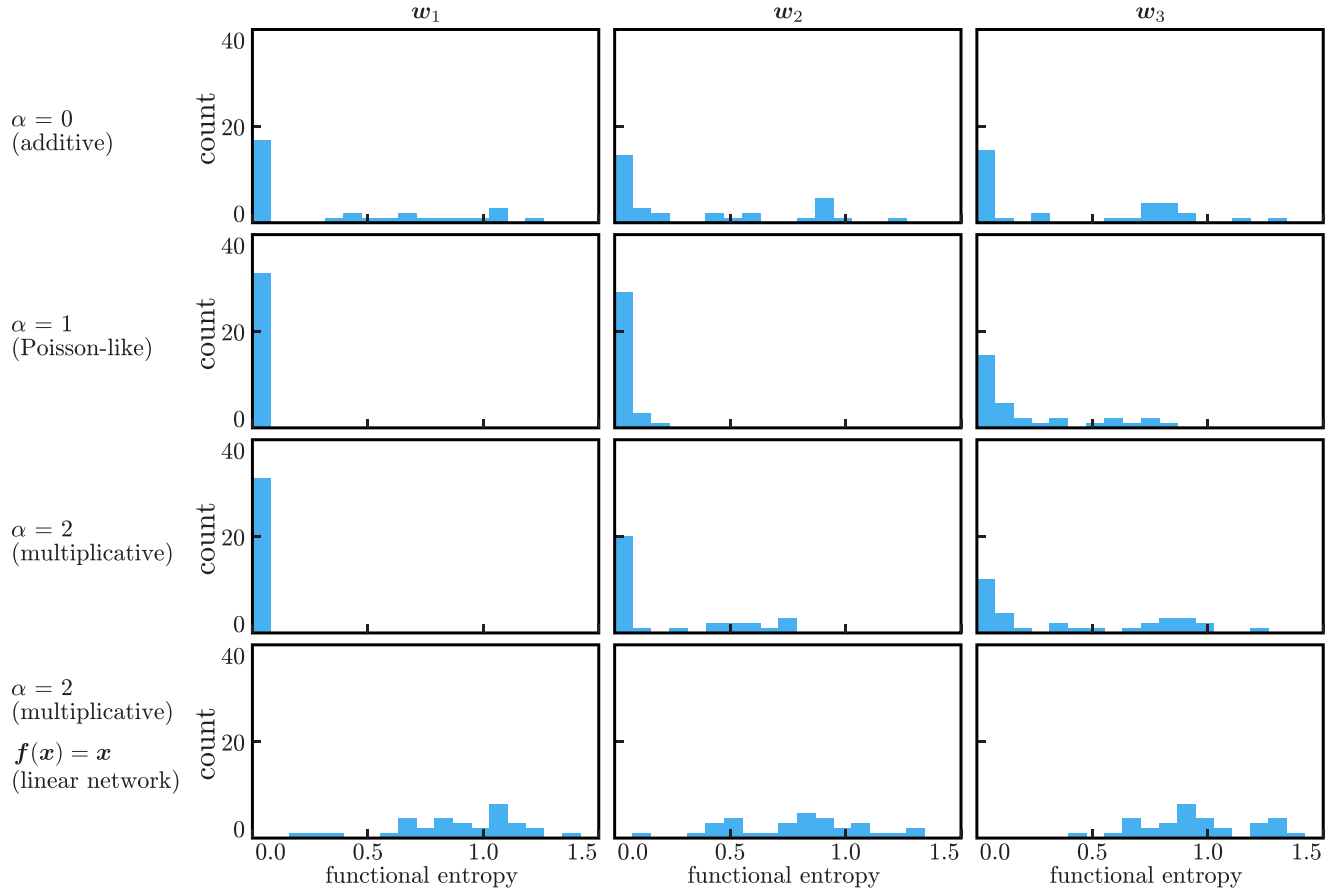


FIG. 12. Estimating modularity via neuron responses to inputs. We compute histograms of entropy of neuron responses for different hidden layers (columns) of networks in Fig. 1(c) (rows) and observe that low-entropy values dominate for more modular layers, similar to the pattern seen via computing the graph Laplacian.

functions with approximately ten segments given the input range, which ensures the architectures tested can learn these target functions.

that layer

$$\mathbf{w}_l \equiv \mathbf{W}_l \mathbf{W}_{l-1} \cdots \mathbf{W}_1. \quad (\text{B1})$$

APPENDIX B: ALTERNATIVE DEFINITION OF MODULARITY

To measure modularity based on the network function, for a hidden layer l , consider the product of weight matrices up to

In a linear network without bias, $\mathbf{w}_l$ is exactly the partial derivatives of hidden neurons in layer l with respect to each of the inputs. Entries of the matrix are $[\mathbf{w}_l]_{\mu_l, i}$, where μ_l is the neuron label and i the input label. This quantity is independent of the input values, so we use this to define a unit-level modularity measure to analyze nonlinear and linear networks

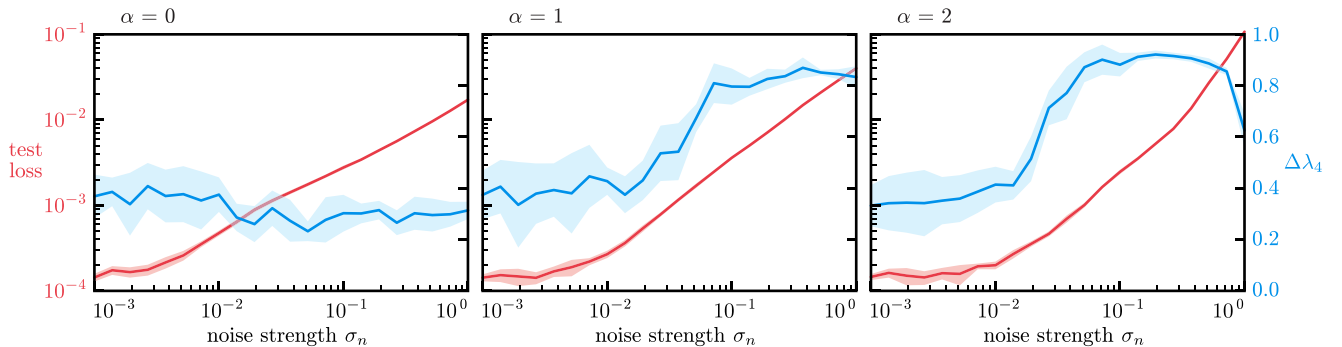


FIG. 13. Loss-modularity tradeoff for different noise characteristics. Repeating the simulations of Fig. 2 for different values of α shows that the network losses all increase with σ_n , following similar trends (red lines). The relevant eigengap $\Delta\lambda_4$, however, increases significantly for Poisson-like ($\alpha = 1$) and multiplicative ($\alpha = 2$) noises, but not additive noise ($\alpha = 0$) (blue lines).

in Fig. 1(c). We compute the entropy of neuron response for the μ_l th neuron in layer l as

$$s_{l,\mu_l} \equiv - \sum_i \left(\frac{[w_l]_{\mu_l,i}^2}{\sum_j [w_l]_{\mu_l,j}^2} \ln \frac{[w_l]_{\mu_l,i}^2}{\sum_j [w_l]_{\mu_l,j}^2} \right) \quad (\text{B2})$$

such that a neuron that responds to a single variable will have entropy close to 0 and a neuron that responds to multiple variables attains a larger value of s_{l,μ_l} .

Computing s_{l,μ_l} for the different layer responses w_l , we observe that for w_1 , networks trained with Poisson-like and multiplicative noises have entropy values close to 0, but those trained with additive noise or a linear task have many neurons responding to more input variables (Fig. 12). This is consistent

with the modular structure detected using graph Laplacians. Similar patterns are observed for w_2 and w_3 , while more high-entropy neurons appear in deeper layers, potentially because of the accumulation of small random off-target weights over the depth of the networks. In contrast, detection of modularity using weight matrices of individual layers does not suffer this effect, and the graph Laplacian eigenspectrum gives a more direct readout of modular structure formation.

APPENDIX C: NOISE STRENGTH EFFECTS AT DIFFERENT SCALING LEVELS

We simulate training with different noise exponents α and noise strengths σ_n , and find that the test loss for all values of

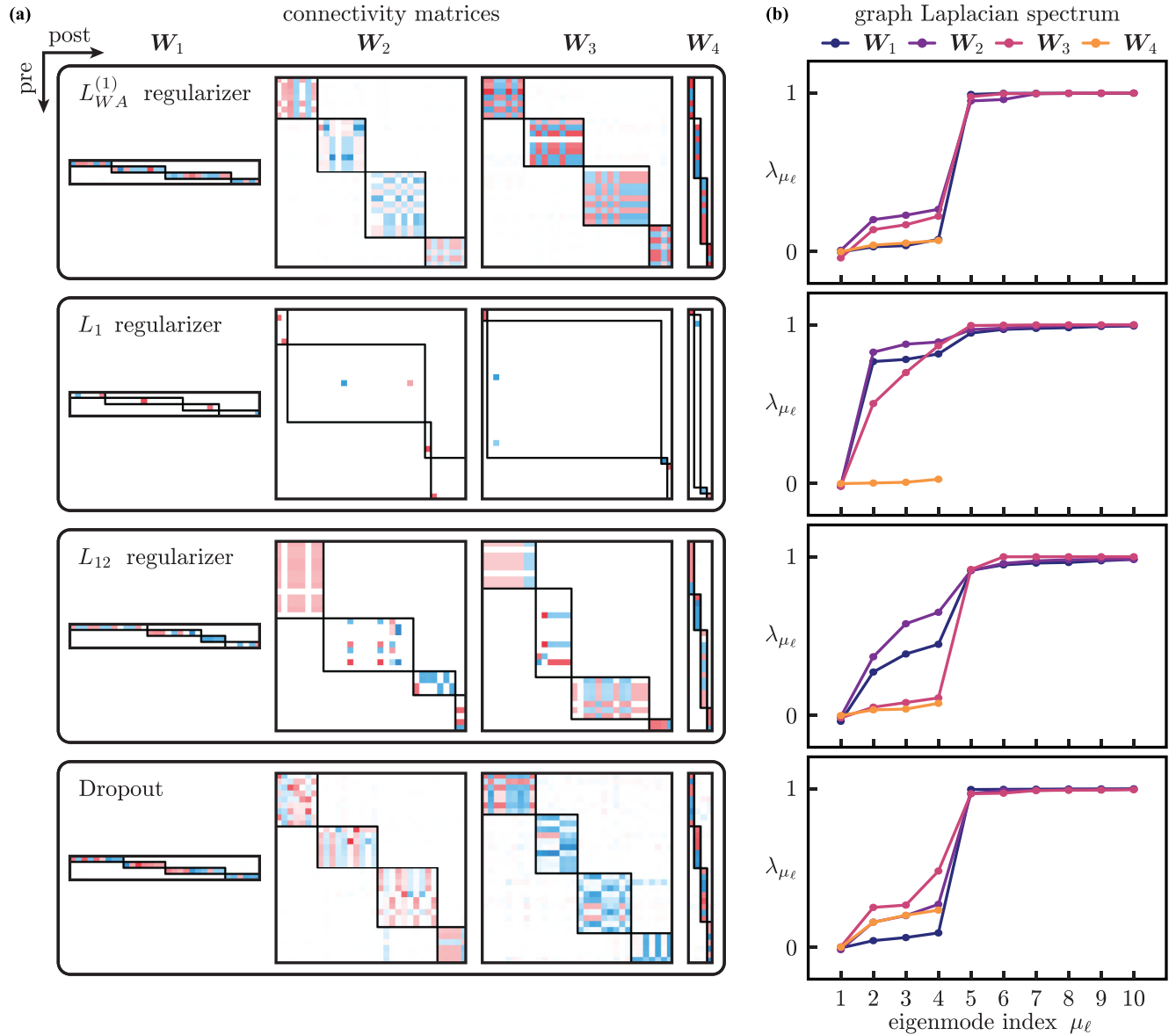


FIG. 14. Effects of other regularizers on network structure. (a) Connectivity matrices and (b) their corresponding graph Laplacian spectra for (top to bottom) $L_{WA}^{(1)}$ -regularized, L_1 -regularized, and L_{12} -regularized networks and for a network trained with dropout, where the activation of each neuron is set to 0 with probability 0.05 during training. The network connections from L_1 regularization are sparse. For L_{12} , connectivities are modular but with some dead neurons since it directly suppresses connection weights. Dropout noise, in contrast, has the same variance scaling as multiplicative noise, leading to modular connections.

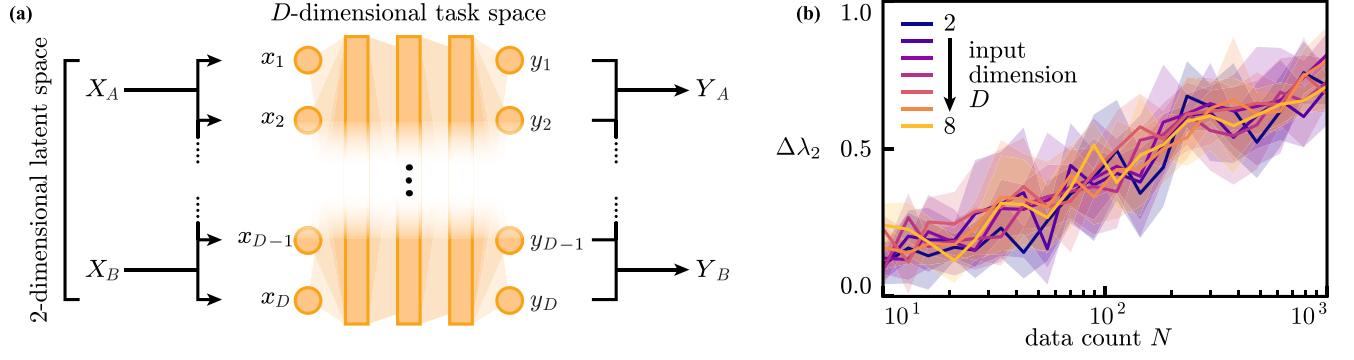


FIG. 15. Latent space dimension determines data complexity of modularization. (a) An alternative task structure where only two latent variables are present, X_A and X_B , related to outputs $Y_A = f_A(X_A)$ and $Y_B = f_B(X_B)$. The task inputs and outputs are exact copies of $X_{A,B}$ and $Y_{A,B}$. (b) The eigengap $\Delta\lambda_2$, corresponding to the formation of two modules, increases with the data count, following the same trend regardless of D .

α increases with similar trends (Fig. 13, red). However, the relevant eigengap $\Delta\lambda_4$ only increases with noise strength at $\alpha = 1$ or 2 (Fig. 13, blue), so the non-trivial scaling of noise with activation is crucial for modularity emergence.

APPENDIX D: TESTING OTHER REGULARIZERS

In addition to the ones tested in Fig. 4, we train networks with the WA regularizer at $\alpha = 1$, with L_1 regularization, with the proposed L_{12} regularizer, and dropout noise (Fig. 14). We find that the $L_{WA}^{(1)}$ regularizer behaves similarly to the $\alpha = 2$ case. The L_1 regularizer produces sparse connectivity matrices, while the L_{12} regularizer converges to a more modular structure. The dropout noise has the same scaling character-

istics as multiplicative noise, leading to modular connectivity matrices.

APPENDIX E: DATA COMPLEXITY AT FIXED LATENT DIMENSION

To study the scenario where the task space dimension D is larger than the true latent space dimension, we set the latent space to comprise two independent inputs X_A and X_B and two outputs $Y_A = f_A(X_A)$ and $Y_B = f_B(X_B)$, where f_A and f_B are two nonlinear functions. The task space inputs are

$$x_i = \begin{cases} X_A & \text{if } i \leq D/2 \\ X_B & \text{otherwise,} \end{cases} \quad (\text{E1})$$

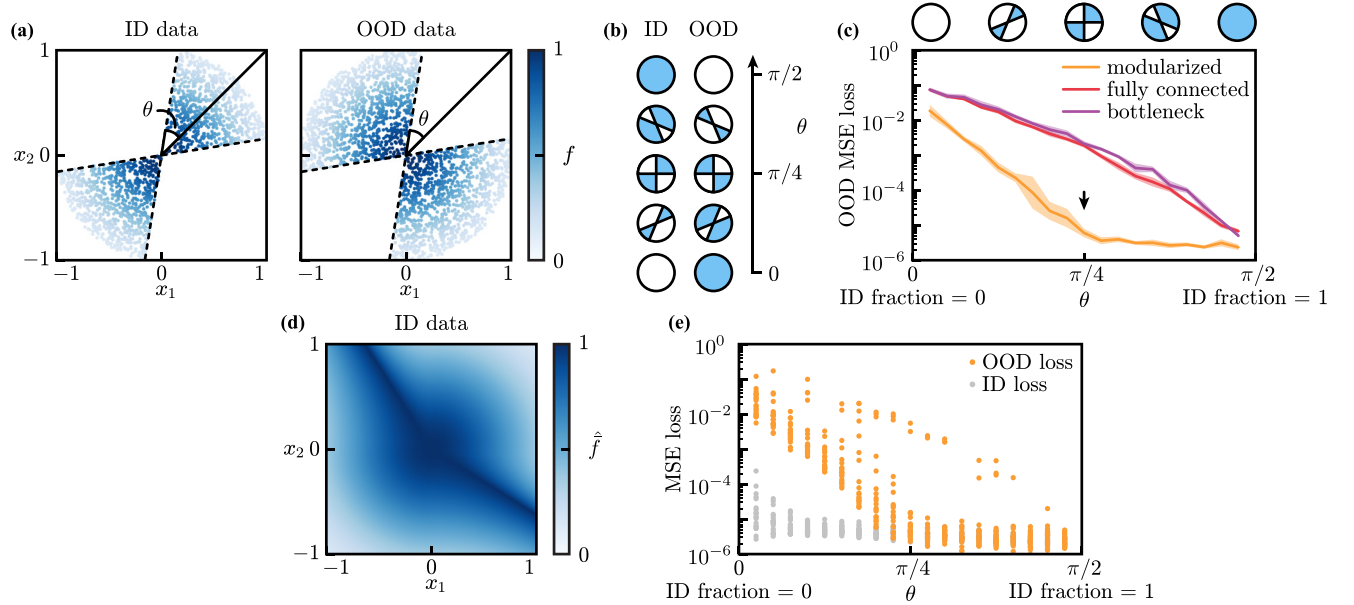


FIG. 16. Data efficiency in the Gaussian function-learning task. (a) We construct ID and OOD datasets as fan-shaped regions parametrized by the angle θ . (b) As θ varies from 0 to $\pi/2$, the input data domain grows to eventually cover the full unit disk and the OOD region shrinks. (c) Plotting the terminal OOD loss for the different network architectures shows that only the modularized network is able to generalize far into the OOD region, at $\theta = \pi/4$ (arrow). (d) Alternate solution $\hat{f}$ given by Eqs. (F5) and (F6), where it is a usual 2D Gaussian in upper right and lower left (ID) quadrants, but has a ridge in the other (OOD) quadrants. (e) Scatter plot of ID (orange) and OOD (gray) losses for all networks sampled in Fig. 9(f). Around two and three networks converge to the alternate solution out of 20. In contrast, all sampled networks for the other architectures follow the group averages in Fig. 9(f).

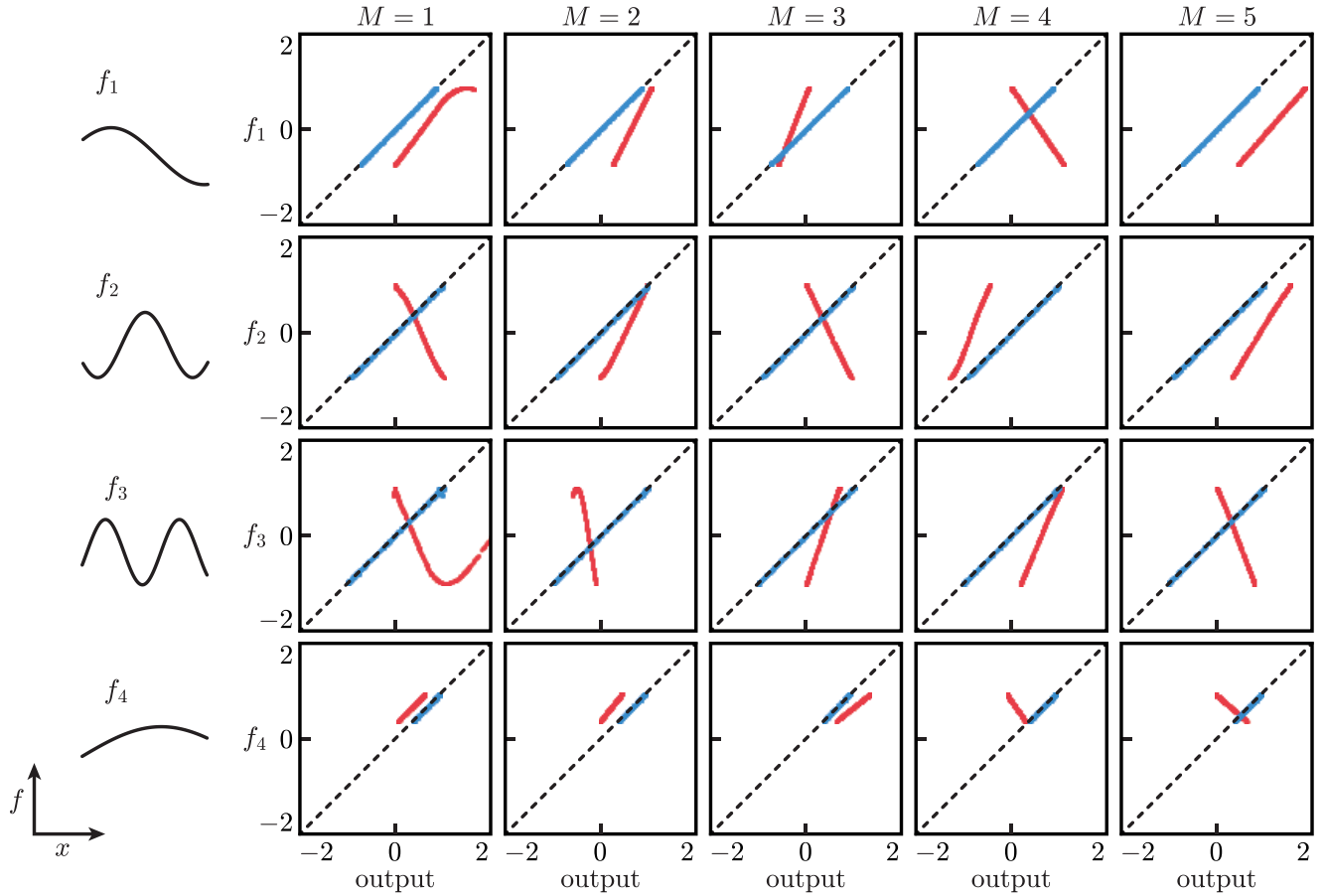


FIG. 17. Several examples of intermediate modular outputs. The scatter plots are from the simulation of Fig. 10, with three fully connected hidden layers and varying modular block depth M . Even with $M = 2$, most of the modular nonlinear functions are fully learned in the modular layers (straight red lines for most of the functions f_i).

with target outputs

$$y_i = \begin{cases} Y_A & \text{if } i \leq D/2 \\ Y_B & \text{otherwise.} \end{cases} \quad (\text{E2})$$

Training networks with varying D in the same way as in Fig. 6 and plotting the eigengap $\Delta\lambda_2$ that indicates formation of two modules as a function of data count N , we find that $\Delta\lambda_2$ increases in the same manner for all D 's tested (Fig. 15). This shows that the curse of dimensionality only applies to the dimensionality of the latent space, but not the apparent task structure.

APPENDIX F: ID/OOD SPLIT IN THE GAUSSIAN TASK

Extending the Gaussian task of Fig. 9(c), we construct ID and OOD sets as fan-shaped regions in the input domain such that the ID/OOD split can be varied systematically [Fig. 16(a)]. The ID and OOD datasets for the Gaussian task $\mathcal{D}_{\text{ID}}$ and $\mathcal{D}_{\text{OOD}}$ are defined as follows:

$$\mathcal{D}_{\text{full}} \equiv \{(x_1, x_2) | x_1^2 + x_2^2 \leq 1\}, \quad (\text{F1a})$$

$$\mathcal{D}_\theta \equiv \left\{ (x_1, x_2) \left| \tan \theta > \left| \frac{x_2 - x_1}{x_2 + x_1} \right| \right. \right\}, \quad (\text{F1b})$$

$$\mathcal{D}_{\text{ID}} \equiv \mathcal{D}_{\text{full}} \cap \mathcal{D}_\theta, \quad \mathcal{D}_{\text{OOD}} \equiv \mathcal{D}_{\text{full}} - \mathcal{D}_{\text{ID}}. \quad (\text{F1c})$$

For $\theta = 0$, there are no ID data; at $\theta = \pi/4$, exactly two quadrants are in the ID dataset; all data become ID when $\theta = \pi/2$ [Fig. 16(b)]. We quantify the terminal OOD loss for the three architectures as a function of θ by calculating the trimmed mean (25th to 75th percentile) OOD mean square error loss over 20 different instantiations of each network, as we vary θ . Comparing the trimmed mean losses, we see that the modularized network converges to the correct 2D Gaussian solution by $\theta \geq \pi/4$, while the other two networks exhibit a much more gradual convergence to the solution, only arriving near it at $\theta = \pi/2$, where there are no OOD regions in the data [Fig. 16(c)].

We used the trimmed mean because in rare cases, the modularized network finds an alternative solution that has a loss much bigger than the mode of the distribution. For $\theta = \pi/4$, the training dataset $\mathcal{D}_{\text{ID}}$ specifies that the learned function $\hat{f}(x, y)$ satisfies

$$\hat{f}(x_1, x_2) = \exp\left(-\frac{1}{2\sigma^2}(x_1^2 + x_2^2)\right) \quad \text{if } x_1 x_2 > 0. \quad (\text{F2})$$

The natural guess is that $\hat{f}(x_1, x_2)$ is a 2D Gaussian in the whole (x_1, x_2) plane, which can be written in the form

$$\hat{f}(x_1, x_2) = \chi[\psi_1(x_1) + \psi_2(x_2)], \quad (\text{F3})$$

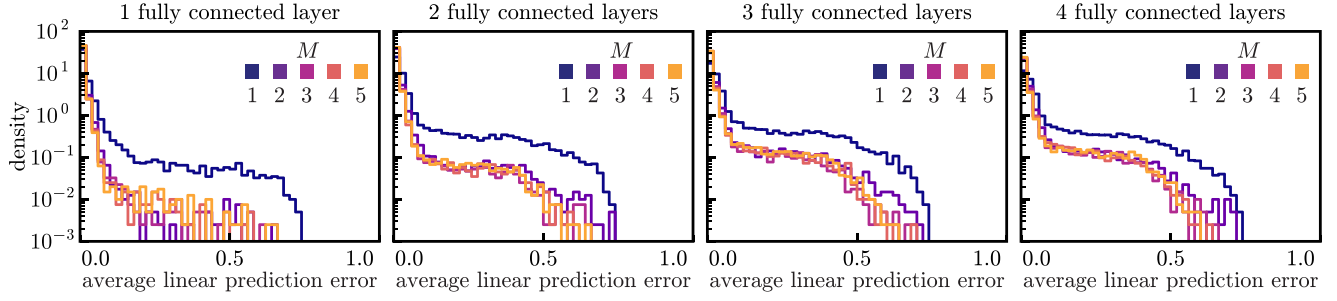


FIG. 18. Distributed expressivity to modularity transition in premodularized networks. We repeat the simulations in Fig. 10 with a varying number of fully connected layers. The number of modular layers is $M \in \{1, \dots, 5\}$. Figure 10(e) is identical to the panel with three fully connected layers. We also normalize the histograms by the fraction of simulation runs that actually converged (out of 5000), as those that did not converge are not used in computing the histograms.

where the single-input functions $\chi(x)$, $\psi_1(x)$, and $\psi_2(x)$ are

$$\chi(x) = e^{-\gamma x/2\sigma^2}, \quad (\text{F4a})$$

$$\psi_1(x) = x^2/\gamma, \quad (\text{F4b})$$

$$\psi_2(x) = x^2/\gamma \quad (\text{F4c})$$

for a gauge variable γ and up to other additive gauge variables. In some rare cases, the modularized neural network however found an alternative solution

$$\bar{\chi}(x) = e^{-|x|[\beta + \gamma \Theta(x)]/2\sigma^2}, \quad (\text{F5a})$$

$$\bar{\psi}_1(x) = \text{sgn}(x)x^2/[\beta + \gamma \Theta(x)], \quad (\text{F5b})$$

$$\bar{\psi}_2(x) = \text{sgn}(x)x^2/[\beta + \gamma \Theta(x)] \quad (\text{F5c})$$

for an additional gauge variable β , where $\Theta(x)$ is the Heaviside step function. For this alternative solution

$$\hat{f}(x_1, x_2) = \bar{\chi}[\bar{\psi}_1(x_1) + \bar{\psi}_2(x_2)], \quad (\text{F6})$$

if x_1 and x_2 are of the same sign, we recover the two-dimensional Gaussian. However, in the OOD quadrants, we take $x_1 > 0$ and $x_2 < 0$ without loss of generality, yielding

$$\bar{\psi}_1(x_1) = x_1^2/(\beta + \gamma), \quad (\text{F7a})$$

$$\bar{\psi}_2(x_2) = -x_2^2/\beta, \quad (\text{F7b})$$

and a ridge appears at a set of points (x_1^*, x_2^*) when $\bar{\psi}_1(x_1^*) + \bar{\psi}_2(x_2^*) = 0$ [Fig. 16(d)]:

$$\frac{|x_1|}{|x_2|} = \sqrt{1 + \frac{\gamma}{\beta}}. \quad (\text{F8})$$

The alternative solution defined by Eqs. (F5) is equally valid as far as the training data are concerned and reflects a major difficulty in studying artificial neural network generalization: There is no “ground truth” per se, so it is difficult to determine whether an ANN learned to generalize. We hypothesize that the modularized network structure appears to have learned to generalize because the solution space for Eqs. (F5) is a manifold disjoint from the 2D Gaussian in the function space, and which solution the network eventually finds depends on the initialization as well as the training algorithm. In Fig. 9(f), when computing the average over 20 different seeds for each θ , around two and three initializations out of 20 would find this alternative solution [Fig. 16(e)]. These have high OOD

losses that deviate very much from the majority of the networks that converge to the Gaussian, so we trim the five highest and five lowest OOD losses when computing the average over seeds.

APPENDIX G: INDUCTIVE BIASES OF MODULAR BLOCKS

When a varying number of hidden layers are used in the modular blocks, we find that increasing the number of hidden layers encourages the network to learn the task functions using the modular blocks instead of utilizing the expressivity of fully connected layers. With $M = 1$, the intermediate outputs (Fig. 17, red) require further nonlinear processing to produce the targets, while for $M \geq 2$ the intermediate outputs differ from the target by only a linear transformation (Fig. 17, blue). This transition of the locus of learning also depends on the number of fully connected layers, as increasing the number of fully connected layers promotes learning of target functions at the fully connected block (Fig. 18).

APPENDIX H: CIFAR10 DATA COMPLEXITY

To investigate the training data complexity of modularity emergence in the multiframe CIFAR10 task, we train networks with a varying number of frames and total data count. Similar to the function learning task, increasing the number of

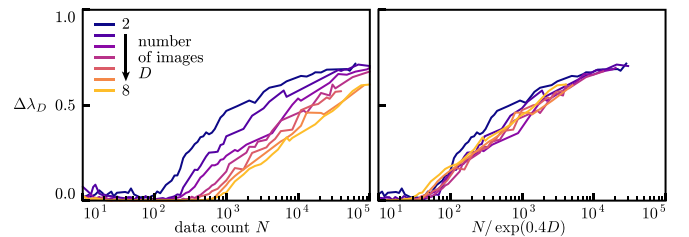


FIG. 19. Data complexity of modularization in the multiframe CIFAR10 task. We train networks with varying number of frames D and plot the relevant eigengap $\Delta\lambda_D$ as the total number of training samples N increases. Similar to the function learning task, the modular emergence slows down with D (left), and rescaling N by an amount exponential in D [here we use $N/\exp(0.4D)$] allows the curves to collapse (right). This scaling is similar to that observed in function-learning before.

frames requires more training data for modular connectivity to emerge (Fig. 19, left). The eigengap increases with data count but the functional form of this increase is no longer

sigmoidal, so we simply re-plot the eigengap against data count exponentially scaled by the frame number, and observe a collapse of the simulated curves (Fig. 19, right).

- [1] D. M. Lorenz, A. Jeng, and M. W. Deem, The emergence of modularity in biological systems, *Phys. Life Rev.* **8**, 129 (2011).
- [2] N. Chomsky, *Aspects of the Theory of Syntax* (MIT Press, Cambridge, 1965).
- [3] A. Boopathy, S. Jiang, W. Yue, J. Hwang, A. Iyer, and I. Fiete, Breaking neural network scaling laws with modularity, *Proceedings of the 42nd International Conference on Machine Learning* (ICLR, Vancouver, 2025).
- [4] T. Veniat, L. Denoyer, and M. Ranzato, Efficient continual learning with modular networks and task-driven priors, *Proceedings of the 38th International Conference on Machine Learning* (ICLR, 2021).
- [5] F. Alet, E. Weng, T. Lozano-Pérez, and L. P. Kaelbling, Neural relational inference with fast modular meta-learning, *Advances in Neural Information Processing Systems 32, Vancouver, 2019* (Curran Associates, Red Hook, 2020).
- [6] G. Parascandolo, N. Kilbertus, M. Rojas-Carulla, and B. Schölkopf, Learning independent causal mechanisms, in *Proceedings of the International Conference on Machine Learning, Stockholm, 2018*, edited by J. Dy and A. Krause (PMLR, Cambridge, 2018), Vol. 80.
- [7] K. O. Ellefsen, J.-B. Mouret, and J. Clune, Neural modularity helps organisms evolve to learn new skills without forgetting old skills, *PLoS Comput. Biol.* **11**, e1004128 (2015).
- [8] M. Hamidi, S. Khajehabdollahi, E. Giannakakis, T. Schäfer, A. Levina, and C. M. Wu, Modular growth of hierarchical networks: Efficient, general, and robust curriculum learning, *Proceedings of the ALIFE 2024: Proceedings of the 2024 Artificial Life Conference* (ASME, 2024), p. 55.
- [9] T. McCourt, I. R. Fiete, and I. L. Chuang, Noisy dynamical systems evolve error correcting codes and modularity, *PRX Life* **4**, 023002 (2026).
- [10] S. Gu, M. G. Mattar, H. Tang, and G. Pan, Emergence and re-configuration of modular structure for artificial neural networks during continual familiarity detection, *Sci. Adv.* **10**, eadm8430 (2024).
- [11] M. L. Montero, C. J. H. Ludwig, R. P. Costa, G. Malhotra, and J. S. Bowers, The role of disentanglement in generalisation, *Proceedings of the Ninth International Conference on Learning Representations, 2021* (ICLR, Appleton, 2021).
- [12] Q. Liang, D. Qian, L. Ziyin, and I. Fiete, Compositional generalization via forced rendering of disentangled latents, in *Proceedings of the 42nd International Conference on Machine Learning, Vancouver, 2025*, edited by A. Singh, M. Fazel, D. Hsu, S. Lacoste-Julien, F. Berkenkamp, T. Maharaj, K. Wagstaff, and J. Zhu (PMLR, Cambridge, 2025), Vol. 267.
- [13] Q. Liang, Z. Liu, M. Ostrow, and I. Fiete, How diffusion models learn to factorize and compose, in *Advances in Neural Information Processing Systems 37, Vancouver, 2024*, edited by A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Curran, Red Hook, 2024).
- [14] J. C. R. Whittington, W. Dorrell, S. Ganguli, and T. E. J. Behrens, Disentanglement with biological constraints: A theory of functional cell types, *Proceedings of the 11th International Conference on Learning Representations (ICLR, 2023)*.
- [15] A. Camuto and M. Willetts, Explicit regularisation in Gaussian noise injections, in *Advances in Neural Information Processing Systems 33, Vancouver, 2020*, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Curran Associates, Red Hook, 2020).
- [16] B. Poole, J. Sohl-Dickstein, and S. Ganguli, Analyzing noise in autoencoders and deep networks, [arXiv:1406.1831](https://arxiv.org/abs/1406.1831).
- [17] W. Softky and C. Koch, The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs, *J. Neurosci.* **13**, 334 (1993).
- [18] M. Gur, A. Beylin, and D. M. Snodderly, Response variability of neurons in primary visual cortex (V1) of alert monkeys, *J. Neurosci.* **17**, 2914 (1997).
- [19] M. Chen and M. W. Deem, Development of modularity in the neural activity of children's brains, *Phys. Biol.* **12**, 016009 (2015).
- [20] D. Qian, <https://github.com/dq219/NoiseMod> (GitHub, 2025).
- [21] U. von Luxburg, A tutorial on spectral clustering, *Stat. Comput.* **17**, 395 (2007).
- [22] Q. Liao, L. Ziyin, Y. Gan, B. Cheung, M. Harnett, and T. Poggio, Self-assembly of a biologically plausible learning circuit, [arXiv:2412.20018](https://arxiv.org/abs/2412.20018).
- [23] C. M. Bishop, Training with noise is equivalent to Tikhonov regularization, *Neural Comput.* **7**, 108 (1995).
- [24] B. Neyshabur, R. Tomioka, and N. Srebro, Norm-based capacity control in neural networks, in *Proceedings of the 28th Conference on Learning Theory, Paris, 2015*, edited by P. Grünwald, E. Hazan, and S. Kale (PMLR, Cambridge, 2015), Vol. 40.
- [25] B. Neyshabur, R. Salakhutdinov, and N. Srebro, Path-SGD: Path-normalized optimization in deep neural networks, in *Proceedings of the 29th International Conference on Neural Information Processing Systems, Montreal, 2015*, edited by C. Cortes, D. D. Lee, M. Sugiyama, and R. Garnett (MIT Press, Cambridge, 2015), Vol. 2, pp. 2422–2430.
- [26] C. Debaio, Degree of approximation by superpositions of a sigmoidal function, *Approx. Theory Appl.* **9**, 17 (1993).
- [27] D. X. Zhou, Universality of deep convolutional neural networks, *Appl. Comput. Harmon. Anal.* **48**, 787 (2020).
- [28] A. Pinkus, Approximation theory of the MLP model in neural networks, *Acta Numer.* **8**, 143 (1999).
- [29] M. T. Augustine, A survey on universal approximation theorems, [arXiv:2407.12895](https://arxiv.org/abs/2407.12895).
- [30] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, *Neural Networks* **6**, 861 (1993).
- [31] A. Krizhevsky, Learning multiple layers of features from tiny images, University of Toronto Report No. 0, 2009, available

- at <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf> (unpublished).
- [32] K. He, X. Zhang, S. Ren, and J. Sun, Learning multiple layers of features from tiny images, *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, 2016* (IEEE, Piscataway, 2016), pp. 770–778.
 - [33] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Mach. Learn.* **8**, 229 (1992).
 - [34] H. Seung, Learning in spiking neural networks by reinforcement of stochastic synaptic transmission, *Neuron* **40**, 1063 (2003).
 - [35] I. R. Fiete and H. S. Seung, Gradient learning in spiking neural networks by dynamic perturbation of conductances, *Phys. Rev. Lett.* **97**, 048104 (2006).
 - [36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *J. Mach. Learn. Res.* **15**, 1929 (2014).
 - [37] A. A. Faisal, L. P. J. Selen, and D. M. Wolpert, Noise in the nervous system, *Nat. Rev. Neurosci.* **9**, 292 (2008).
 - [38] H. W. Lin, M. Tegmark, and D. Rolnick, Why does deep and cheap learning work so well?, *J. Stat. Phys.* **168**, 1223 (2017).
 - [39] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review, *Int. J. Automat. Comput.* **14**, 503 (2017).
 - [40] M. Refinetti, A. Ingrosso, and S. Goldt, Neural networks trained with SGD learn distributions of increasing complexity, *J. Stat. Mech.* (2025) 024001.
 - [41] Y. Lu, J. Yin, Z. Chen, H. Gong, Y. Liu, L. Qian, X. Li, R. Liu, I. M. Andolina, and W. Wang, Revealing detail along the visual hierarchy: Neural clustering preserves acuity from V1 to V4, *Neuron* **98**, 417 (2018).
 - [42] B. K. Hulse and V. Jayaraman, Mechanisms underlying the neural computation of head direction, *Annu. Rev. Neurosci.* **43**, 31 (2020).
 - [43] C. Lyu, L. F. Abbott, and G. Maimon, Building an allocentric travelling direction signal via vector computation, *Nature (London)* **601**, 92 (2022).
 - [44] T. Hafting, M. Fyhn, S. Molden, M.-B. Moser, and E. I. Moser, Microstructure of a spatial map in the entorhinal cortex, *Nature (London)* **436**, 801 (2005).
 - [45] Y. Burak and I. R. Fiete, Accurate path integration in continuous attractor network models of grid cells, *PLoS Comput. Biol.* **5**, e1000291 (2009).
 - [46] A. N. Kolmogorov, On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition, *AMS Trans. Ser 2* **28**, 55 (1963).
 - [47] J. Schmidt-Hieber, The Kolmogorov–Arnold representation theorem revisited, *Neural Networks* **137**, 119 (2021).
 - [48] O. Dugan, R. Dangovski, A. Costa, S. Kim, P. Goyal, J. Jacobson, and M. Soljačić, OccamNet: A fast neural model for symbolic regression at scale, [arXiv:2007.10784](https://arxiv.org/abs/2007.10784).
 - [49] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, KAN: Kolmogorov-arnold networks, *Proceedings of the 13th International Conference on Learning Representations, Singapore, 2025* (ICLR, Appleton, 2025).
 - [50] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho, Discovering symbolic models from deep learning with inductive biases, *Advances in Neural Information Processing Systems 33* (Vancouver, 2020).