nature communications

Article

https://doi.org/10.1038/s41467-025-58276-6

# Coding schemes in neural networks learning classification tasks

Received: 27 July 2024

Accepted: 17 March 2025

Published online: 09 April 2025

Check for updates

Alexander van Meegen [1] & Haim Sompolinsky [1,2]

Neural networks posses the crucial ability to generate meaningful representations of task-dependent features. Indeed, with appropriate scaling, supervised learning in neural networks can result in strong, task-dependent feature learning. However, the nature of the emergent representations is still unclear. To understand the effect of learning on representations, we investigate fully-connected, wide neural networks learning classification tasks using the Bayesian framework where learning shapes the posterior distribution of the network weights. Consistent with previous findings, our analysis of the feature learning regime (also known as 'non-lazy' regime) shows that the networks acquire strong, data-dependent features, denoted as coding schemes, where neuronal responses to each input are dominated by its class membership. Surprisingly, the nature of the coding schemes depends crucially on the neuronal nonlinearity. In linear networks, an analog coding scheme of the task emerges; in nonlinear networks, strong spontaneous symmetry breaking leads to either redundant or sparse coding schemes. Our findings highlight how network properties such as scaling of weights and neuronal nonlinearity can profoundly influence the emergent representations.

The remarkable empirical success of deep learning stands in strong contrast to the theoretical understanding of trained neural networks. Although every single detail of a neural network is accessible and the task is known, it is still very much an open question how the neurons in the network manage to collaboratively solve the task. While deep learning provides exciting new perspectives on this problem, it is also at the heart of more than a century of research in neuroscience.

Two key aspects are representation learning and generalization. It is widely appreciated that neural networks are able to learn useful representations from training data[1–3]. But from a theoretical point of view, fundamental questions about representation learning remain wide open: Which features are extracted from the data? And how are those features represented by the neurons in the network? Furthermore, neural networks are able to generalize even if they are deeply in the overparameterized regime[4–9] where the weight space contains a subspace—the solution space—within which the network perfectly fits the training data. This raises a fundamental question about the effect of implicit and explicit regularization: Which regularization biases the solution space towards weights that generalize well?

We here investigate the properties of the solution space using the Bayesian framework where the posterior distribution of the weights determines how the solution space is sampled[10,11]. For theoretical investigations of this weight posterior, the size of the network is taken to infinity. Crucially, the scaling of the network and task parameters, as the network size is taken to infinity, has a profound impact: Neural networks can operate in different regimes depending on this scaling. Here, the relevant scales are the width of the network $N$ and the size of the training data set $P$. A widely used scaling limit is the infinite width limit where $N$ is taken to infinity while $P$ remains finite[12–19]. A scaling limit closer to empirically trained networks takes both $N$ and $P$ to infinity at a fixed ratio $\alpha = P/N$[20–26].

In addition to the scaling of $P$ and $N$, the scaling of of the final network output with $N$ is important because it has a strong effect on representation learning[27–29] (illustrated in Fig. 1b,d). If the readout scales its inputs with $1/\sqrt{N}$ the network operates in the 'lazy' regime.

[1]Center for Brain Science, Harvard University, Cambridge, MA 02138, USA. [2]Edmond and Lily Safra Center for Brain Sciences, Hebrew University, Jerusalem 9190401, Israel. e-mail: alexander.vanmeegen@epfl.ch; hsompolinsky@mcb.harvard.edu

https://doi.org/10.1038/s41467-025-58276-6

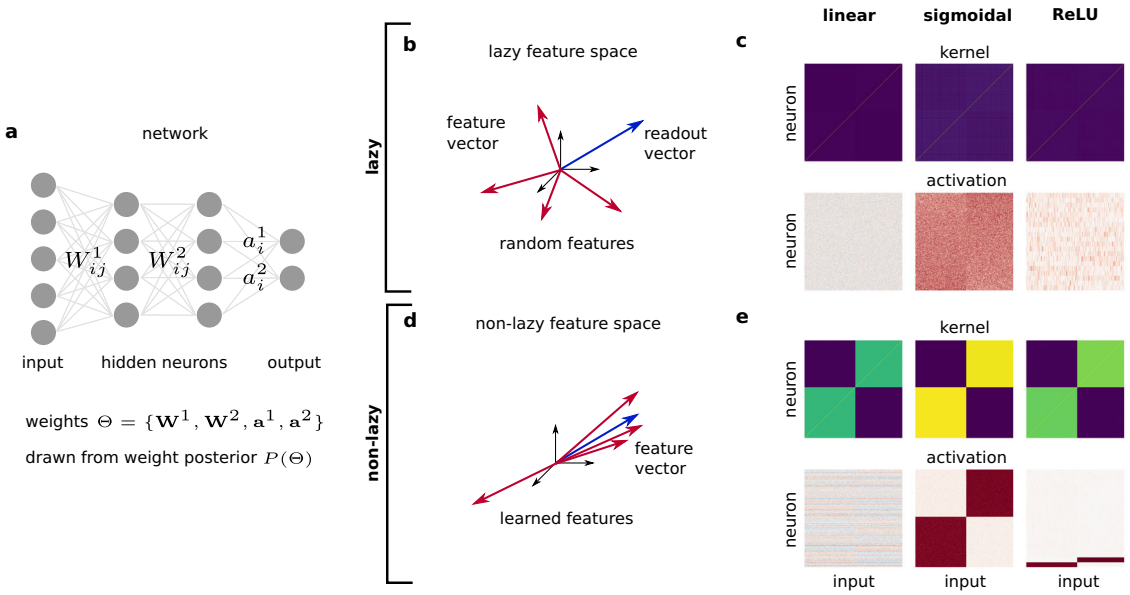


**Fig. 1 | Overview of lazy and non-lazy regimes. a** Sketch of the network with two hidden layers ($L = 2$) and two outputs ($m = 2$). **b**, **d** Last layer feature space in the lazy regime where predominantly random features are almost orthogonal to the readout weight vector (**b**) and in the non-lazy regime where learned features are aligned to the readout weight vector (**d**). **c**, **e** Kernels and activations of feature layer neurons in lazy (**c**) and non-lazy (**e**) networks for a binary classification task and three neuronal nonlinearities (linear, sigmoidal, and ReLU). For ReLU networks, only the 20 most active neurons are shown. In (**e**), all three kernels show a similar, pronounced task-learned structure. However, the patterns of activation are strikingly different.

Lazy networks rely predominantly on random features, and representation learning is only a $1/N$ correction both for finite $P$[18,30–33] and fixed $\alpha$[20] (illustrated in Fig. 1c); in the latter case, it nonetheless affects the predictor variance[20,24,25]. If the readout scales its inputs with $1/N$ the network operates in the 'non-lazy' (also called mean-field or rich) regime and learns strong, task-dependent representations[34–40] (illustrated in Fig. 1e). However, the nature of the solution, in particular how the learned features are represented by the neurons, remains unclear. Work-based on a partial differential equation for the weights[34–37] only provides insight in effectively low dimensional settings; investigations based on kernel methods[38–41] average out important structures at the single neuron level.

In this paper, we develop a theory for the weight posterior of non-lazy networks in the limit $N, P \to \infty$. We show that the representations in non-lazy networks trained for classification tasks exhibit a remarkable structure in the form of coding schemes, where distinct groups of neurons are characterized by the subset of classes that activate them. Another central result is that the nature of the learned representation strongly depends on the type of neuronal nonlinearity (illustrated in Fig. 1e). We consider three nonlinearities: Linear, sigmoidal, and ReLU leading to analog, redundant, and sparse coding schemes, respectively. We start the manuscript with the learned representations on training inputs in the simple setting of a toy task. Next, we analyze the dynamics in weight space, focusing on spontaneous symmetry breaking. Moving beyond the toy task, we investigate learned representations on training and test inputs and generalization on MNIST and CIFAR10. The paper concludes with a brief summary and discussion of our results.

## Results

### Non-lazy networks

We consider fully-connected feedforward networks with $L$ hidden layers and $m$ outputs (Fig. 1a)

$$f_r(\boldsymbol{x};\Theta) = \frac{1}{N}\sum_{i=1}^{N} a_i^r \phi\left[z_i^L(\boldsymbol{x})\right], \quad r = 1, \ldots, m, \tag{1}$$

where $\phi(z)$ denotes the neuronal nonlinearity, $z_i^L(\boldsymbol{x})$ the last layer preactivation, and $\boldsymbol{x}$ is an arbitrary $N_0$-dimensional input. The preactivations are $z_i^\ell(\boldsymbol{x}) = \frac{1}{\sqrt{N}}\sum_{j=1}^{N} W_{ij}^\ell \phi[z_j^{\ell-1}(\boldsymbol{x})]$ in the hidden layers $\ell = 2, \ldots, L$ and $z_i^1(\boldsymbol{x}) = \frac{1}{\sqrt{N_0}}\sum_{j=1}^{N_0} W_{ij}^1 x_j$ in the first layer. The activations are $\phi[z_i^\ell(\boldsymbol{x})]$ and we assume for simplicity that the width of all hidden layers is $N$. We call the last hidden layer $\ell = L$ the feature layer.

The trainable parameters of the networks are the readout and hidden weights, which we collectively denote by $\Theta$. The networks are trained using empirical training data $\mathcal{D} = \{(\boldsymbol{x}_\mu, \boldsymbol{y}_\mu)\}_{\mu=1}^{P}$ containing $P$ inputs $\boldsymbol{x}_\mu$ of dimension $N_0$ and $P$ targets $\boldsymbol{y}_\mu$ of dimension $m$ (for classification $m$ is the number of classes). The $N_0 \times P$ matrix containing all training inputs is denoted by $\boldsymbol{X}$; the $P \times m$ matrix containing all training targets is denoted by $\boldsymbol{Y}$.

Importantly, the multiplication with $1/N$ in (1) puts the network in the non-lazy regime; a multiplication with $1/\sqrt{N}$ corresponds to the lazy regime. The importance of the multiplicative factor can be understood geometrically: In the lazy regime (Fig. 1b), the feature layer activations (called the feature vectors) $\phi[z_i^L(\boldsymbol{x}_\mu)]$, $i = 1, \ldots, N$, are predominantly random resulting in a small overlap of $O(\sqrt{N})$ with the readout vector which requires a relatively large readout normalization, $1/\sqrt{N}$. In the non-lazy regime (Fig. 1d), the feature layer activations exhibit a strong learned component, such that they have a large overlap of $O(N)$ with the readout vector, which makes the normalization with $1/N$ in Eq. (1) appropriate. Put differently, for non-lazy networks, the learned feature vectors and the readout vector are aligned.

To study the properties of the network after learning, we employ a Bayesian framework where, following learning, the parameters $\Theta$ are drawn from a posterior distribution $P(\Theta) = \frac{1}{Z} P_0(\Theta) \exp[-\beta \mathcal{L}(\Theta)]$ where $P_0(\Theta)$ is a Gaussian prior, $\mathcal{L}(\Theta)$ a mean squared error loss, and $Z$ the normalization constant (see Methods for details). We denote expectations w.r.t. the weight posterior by $\langle\cdot\rangle_\Theta$. In the "zero temperature limit" $\beta \to \infty$ any set of parameters $\Theta$ drawn from the posterior corresponds to a network, which perfectly solves the training task.

**Table 1 | Glossary of neural coding terminology**

| Term | Definition |
|---|---|
| code | The subset of classes that activate a neuron. |
| coding scheme | The collection of codes implemented by a neuronal population. |
| sparse coding | Only a small subset of neurons exhibit codes. |
| redundant coding | All the codes in the scheme are shared by a large subset of neurons. |
| analog coding | All neurons respond to all classes but with different strengths. |

The geometrical intuition given above about the effect of non-lazy scaling of the output implicitly assumed that the individual components of both the readout vectors as well as the feature vectors are of $O(1)$, hence their norms are of $O(\sqrt{N})$. However, learning could change the order of magnitude of these norms, for example, during learning the readout weights could grow by a factor $\sqrt{N}$ which would trivially undo the non-lazy scaling and result in a network operating in the lazy regime, despite the $O(1/N)$ scaling of the output. To control the properties of the network after learning within the Bayesian framework, we adjust in particular the prior variance of the readout weights $\sigma_a^2$ such that all preactivation norms as well as the norm of the readout weights are $O(\sqrt{N})$. In summary, we consider networks drawn from a weight posterior such that they are perfectly trained and such that the non-lazy scaling is not trivially undone by learning, allowing us to study the salient properties of learning in non-lazy networks.

To understand the properties of the emergent representations, we develop a mean-field theory of the weight posterior. To this end, we consider the limit $N, N_0, P \to \infty$ while the number of readout units, as well as the number of layers remain finite, i.e., $m, L = O(1)$. The relation between $P$ and the size parameters will be discussed below.

## Kernel and coding schemes

The properties of learned representations are frequently investigated using the kernel[20,39,42,43], which measures the overlap between the neurons' activations at each layer induced by pairs of inputs:

$$\mathcal{K}_\ell(\boldsymbol{x}_1, \boldsymbol{x}_2; \Theta) = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\phi}\left[z_i^\ell(\boldsymbol{x}_1)\right]\boldsymbol{\phi}\left[z_i^\ell(\boldsymbol{x}_2)\right] \tag{2}$$

where the inputs $\boldsymbol{x}_1, \boldsymbol{x}_2$ can be either from the training or the test set. We denote the posterior averaged kernel by $\mathcal{K}_\ell(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle \mathcal{K}_\ell(\boldsymbol{x}_1, \boldsymbol{x}_2; \Theta)\rangle_\Theta$; the posterior averaged $P \times P$ kernel matrix on all training inputs is denoted by $\boldsymbol{K}_\ell$. In addition to capturing the learned representations, the kernel is a central observable because it determines the statistics of the output function in wide networks[12–15].

Crucially, the kernels are insensitive to how the learned features are represented by the neuronal activations. Consider, for example, binary classification in two extreme cases: (1) Each class activates a single (different) neuron and all remaining neurons are inactive; (2) each class activates half of the neurons and the remaining half of the neurons are inactive. In both scenarios the kernels agree up to an overall scaling factor despite the drastic difference in the underlying representations (illustrated in Fig. 1e).

The central goal of our work is to understand the nature of the representations using a theory that goes beyond kernel properties. To this end, we use the notion of a neural code, which we define as the subset of classes that activate a given neuron (see Table 1). At the population level, this leads to a "coding scheme": The collection of codes implemented by the neurons in the population. As we will show the coding schemes encountered in our theory are of three types (illustrated in Fig. 1e): (1) Sparse coding where only a small subset of neurons exhibit codes (as in the first scenario above). (2) Redundant coding, where all the codes in the scheme are shared by a large subset of neurons (as in the second scenario above). (3) Analog coding where all neurons respond to all classes but with different strengths.

Surprisingly, the nature of the learned representations varies drastically with the choice of the activation function of the hidden layer, $\phi(z)$, although the kernel matrix is similar in all cases (Fig. 1e). Specifically, in the linear case, $\phi(z) = z$, the coding is analog, in the sigmoidal case, $\phi(z) = \frac{1}{2}[1 + \mathrm{erf}(\sqrt{\pi}z/2)]$, the coding is redundant, and in the case of ReLU, $\phi(z) = \max(0, z)$, the coding is sparse.

## Main theoretical result

To explain the emergence of the coding schemes, we have derived a mean-field theory for the kernels, the learned representations, the input-output function (predictor statistics) as well as the performance of the networks, which applies to general distributions of inputs and outputs (see supplements A-C). To illustrate the theoretical predictions regarding the learned representations, we first employ a toy classification task where all inputs $\boldsymbol{x}_\mu$ are mutually orthogonal, $\frac{1}{N_0}\boldsymbol{x}_\mu^\top \boldsymbol{x}_\nu = \delta_{\mu\nu}$, and the class memberships are randomly assigned. The corresponding targets $\boldsymbol{y}_\mu$ are one-hot encoded: If $\boldsymbol{x}_\mu$ belongs to the $r$-th class, the $r$-th element of $\boldsymbol{y}_\mu$ is $y_+$, and all other elements are $y_-$. The simplicity of the data allows for extensive numerical evaluations across parameters. Later, we will show results for MNIST[44] and CIFAR10[45] image classification, where representations of both training and test inputs are shown, and the consequences for the generalization performance are investigated.

A general, remarkable outcome of the theory is that the joint posterior of the readout weights and the activations factorizes across neurons and layers,

$$\prod_{i=1}^{N}[P(\boldsymbol{a}_i)P(\boldsymbol{z}_i^L \mid \boldsymbol{a}_i)]\prod_{\ell=1}^{L-1}\left[\prod_{j=1}^{N} P(\boldsymbol{z}_j^\ell)\right], \tag{3}$$

where $\boldsymbol{a}_i$ denotes the $m$-dimensional vector containing the readout weights for neuron $i$ and $\boldsymbol{z}_i^\ell$ denotes the $P$-dimensional vector containing the preactivations of neuron $i$ on all training inputs. Importantly, while the feature layer representations $\boldsymbol{z}_i^L$ depend on the corresponding readout vector, $\boldsymbol{a}_i$, the distribution of the activations of previous layers are completely decoupled, so that even when the readout and top layers break permutation symmetry, this broken symmetry does not affect lower layers as we will see below. The form of the posteriors of both readout weights and preactivations depends on the nature of the neuronal activation functions. Below we present the key results of the theory for the different activation functions (further details in supplements A-C for linear, sigmoidal, and ReLU, respectively).

## Analog coding scheme

We begin with linear networks $\phi(z) = z$. In this case, the single neuron readout posterior $P(\boldsymbol{a}_i)$ is Gaussian, $P(\boldsymbol{a}_i) = \mathcal{N}(\boldsymbol{a}_i \mid \boldsymbol{0}, \boldsymbol{U})$, where $\boldsymbol{U}$ is a $m \times m$ matrix determined by $\boldsymbol{U}^{L+1} = \sigma_a^{2L}\boldsymbol{Y}^\top \boldsymbol{K}_0^{-1}\boldsymbol{Y}$ with the input kernel $\boldsymbol{K}_0 = \frac{1}{N_0}\boldsymbol{X}^\top\boldsymbol{X}$. Because $\boldsymbol{Y}^\top\boldsymbol{K}_0^{-1}\boldsymbol{Y} = O(P)$, where $P$ is the size of the training set, the predicted norm-squared of the posterior readout weights grows with $\sigma_a^{2L}P$ where $\sigma_a^2$ is the readout weights variance of the prior and $L$ is the network depth. Hence, to prevent growth of the norm of the readout weights with $P$ we set $\sigma_a^2 = 1/P^{1/L}$. Furthermore, the posterior mean of the feature layer activations $\boldsymbol{z}_i^L$, conditioned on $\boldsymbol{a}_i$, is $\boldsymbol{Y}\boldsymbol{U}^{-1}\boldsymbol{a}_i$, where $\boldsymbol{Y}$ is the training label matrix, yielding an analog coding of the task where the strength of the code is determined by the strength of the Gaussian distributed readout weight $\boldsymbol{a}_i$.

For linear networks, the single neuron posterior of the activations in the lower layers $\ell < L$ is a multivariate Gaussian: $P(\boldsymbol{z}_i^\ell) = \mathcal{N}(\boldsymbol{z}_i^\ell \mid \boldsymbol{0}, \boldsymbol{K}_\ell)$ for $\ell = 1, \ldots, L-1$, where the covariance is given by the kernel matrix $\boldsymbol{K}_\ell = \boldsymbol{K}_0 + \sigma_a^{2(L-\ell)}\boldsymbol{Y}\boldsymbol{U}^{-(L-\ell+1)}\boldsymbol{Y}^\top$ consisting of a contribution $\boldsymbol{K}_0$ due to the prior and a learned low-rank contribution $\sigma_a^{2(L-\ell)}\boldsymbol{Y}\boldsymbol{U}^{-(L-\ell+1)}\boldsymbol{Y}^\top$. The strength of the learned part increases across layers, $\sigma_a^{2(L-\ell)} = 1/P^{1-\frac{\ell}{L}}$. In

https://doi.org/10.1038/s41467-025-58276-6

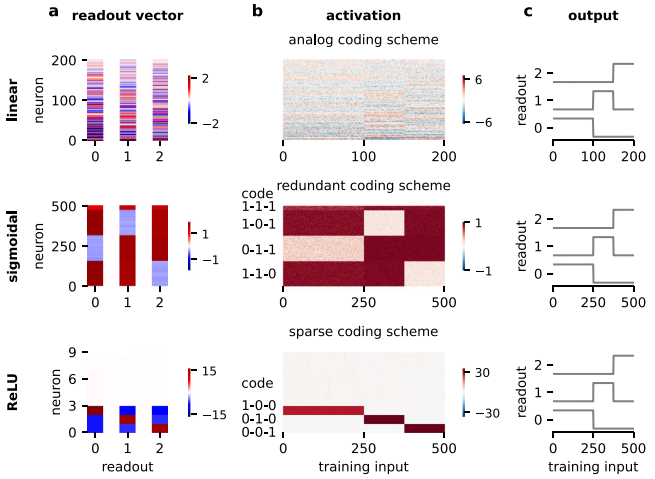


**Fig. 2 | Coding schemes in the feature layer for linear (top), sigmoidal (middle), and ReLU (bottom) nonlinearity. a** Sample of the readout weight vectors of all three classes. For ReLU, only readout weights of the nine most active neurons are shown. **b** Activations on all training inputs for a given weight sample. For ReLU, only the nine most active neurons are shown. **c** Network output for the weighted sample shown in (**a**, **b**). The output perfectly matches the one-hot coding of the task where the first half of the inputs belong to the first class, and the remaining inputs belong to the two remaining classes with equal proportions.

the last layer, it is $O(1)$, indicating that even in the linear case the network learns strong features with a strength that increases across layers, culminating in an $O(1)$ contribution in the feature layer. This is in strong contrast to the lazy case where the learned contribution to the kernels is suppressed by $1/N$ in all layers[20].

A sample of the analog coding scheme is shown in Fig. 2b for the single-hidden-layer network on the toy classification task. For the task, we use three classes with unequal proportions: half of the training inputs belong to the first class, and the remaining inputs belong to the other two classes with equal ratios. The corresponding structure is clearly recognizable in the activations. Furthermore, the activations are sorted by their mean squared activity, which shows the analog nature of the code: There is a continuous gradient from weakly to strongly coding neurons. In line with the Gaussian readout posterior, the readout vectors display no clear structure (Fig. 2a), but in combination with the activations, they are perfectly adjusted to solve the task (Fig. 2c).

We show a sample of the lower layer activations in Fig. 3 for a three-hidden-layer network and the toy task with three classes. In contrast to the feature layer activations (Fig. 3c), the learned structure is hardly apparent in the first layers (Fig. 3a, b). Corresponding to the theory, the learned low-rank contribution to the kernel increases in strength across layers (Fig. 3d–f) until the kernel perfectly represents the task in the last layer.

## Redundant coding scheme

We now consider nonnegative sigmoidal networks, $\phi(z) = \frac{1}{2}[1 + \mathrm{erf}(\sqrt{\pi}z/2)]$. To avoid growth of the readout weights with $P$ we must choose $\sigma_a^2 = 1/P$ for arbitrary depth (see supplement B).

With sigmoidal nonlinearity, the readout posterior is drastically different from Gaussian: It is a weighted sum of Dirac deltas $P(\boldsymbol{a}_i) = \sum_{\gamma=1}^{n} P_\gamma \delta(\boldsymbol{a}_i - \boldsymbol{a}_\gamma)$ where the weights $P_\gamma$ and the $m$-dimensional vectors $\boldsymbol{a}_\gamma$ are determined jointly for all $\gamma = 1, \ldots, n$ by a set of self-consistency equations which depend on the training set (see supplement B.1). This means that the vector of readout weights is redundant: Each of the $N$ neurons chooses one of $n$ possible readout weights $\boldsymbol{a}_\gamma$ with a probability $P_\gamma$. The number of distinct readout weights $n$ determines the number of codes employed by the network.

The readout weights $\boldsymbol{a}_i$ determine the last layer preactivation $\boldsymbol{z}_i^L$ through the single neuron conditional distribution $P(\boldsymbol{z}_i^L \mid \boldsymbol{a}_i)$. Due to the redundant structure of the readout posterior, a redundant coding scheme emerges in which, on average, a fraction $P_\gamma$ of the neurons exhibit identical preactivation posteriors $P(\boldsymbol{z}_i^L \mid \boldsymbol{a}_\gamma)$. These posteriors have pronounced means that reflect the structure of the task and the code but also significant fluctuations around the mean which do not vanish in the limit of large $N$ and $P$ (see Fig. 4c).

For the single-hidden-layer network on the toy task with three classes, the redundant coding of the task is immediately apparent (Fig. 2b) but also the remaining fluctuations across inputs from the same class are visible. In this example, there are four codes: 1-1-0, 0-1-1, 1-0-1, and 1-1-1. Note that the presence of code 1-1-1, which is implemented only by a small fraction of neurons, is accurately captured by the theory (Fig. 4a middle peak). The relation between the neurons' activations and their readout weights is straightforward: For neurons with code 1-1-0, the readout weights are positive for classes 0 and 1 and negative for class 2 (Fig. 2a), and vice versa for the other codes, leading to a perfect solution of the task (Fig. 2c).

Clearly, this coding scheme with four codes is not a unique solution. Indeed, the theory admits other coding schemes which are, however, unlikely to occur (see supplement B.1). Interestingly, the details of the coding scheme depend on the choice of values for the targets $y_\mu^r$; for example, increasing $y_-$ from 0.1 to 0.5 in the above toy task leads to successive transitions between four different coding schemes (see supplement E).

The posterior of the preactivations $P(\boldsymbol{z}_i^\ell)$ in the lower layers $\ell < L$ is multimodal with each mode corresponding to a code (see supplement B.1). Since all neurons sample independently from $P(\boldsymbol{z}_i^\ell)$, a redundant coding scheme emerges where all neurons coming from the same mode of $P(\boldsymbol{z}_i^\ell)$ share their code. Due to the appearance of a coding scheme in the activations, the corresponding kernels posses a strong low-rank component reflecting the task. We use the toy task and a three-hidden-layer network to show the redundant coding schemes across layers (Fig. 3a–c) and the corresponding kernels (Fig. 3d–f). For the feature layer representations, the main impact of the increased depth is a reduction of fluctuations across inputs from the same class (Fig. 2b vs. Fig. 3c)—the coding scheme "sharpens". Across layers, the coding scheme sharpens from the input to the feature layer (Fig. 3a–c). Furthermore, the coding scheme can vary across layers, e.g., there are five codes in the first layer (Fig. 3a) and six codes in the second and third layers (Fig. 3b, c).

## Sparse coding scheme

Last, we consider ReLU networks, $\phi(z) = \max(0, z)$. Here we set $\sigma_a^2 = 1/P^{1/L}$ as in the linear case owing to the homogeneity of ReLU. For the theory we consider only single-hidden-layer networks $L = 1$.

The nature of the readout posterior is fundamentally different compared to the previous two cases: It factorizes into a small number $n = O(1)$ of outlier neurons and a remaining bulk of neurons, $\prod_{i=1}^{n}[P_i(\boldsymbol{a}_i)] \prod_{i=n+1}^{N}[P(\boldsymbol{a}_i)]$. The readout posteriors of the outlier neurons $i = 1, \ldots, n$ are Dirac deltas at $O(\sqrt{N})$ values, $P_i(\boldsymbol{a}_i) = \delta(\boldsymbol{a}_i - \sqrt{N}\bar{\boldsymbol{a}}_i)$. For the bulk neurons $i = n + 1, \ldots, N$, the readout weight posterior is a single Dirac delta with a probability mass of $O(1)$, $P(\boldsymbol{a}_i) = \delta(\boldsymbol{a}_i - \boldsymbol{a}_0)$. The values of $\boldsymbol{a}_0$ and $\bar{\boldsymbol{a}}_i$ are determined self-consistently (see supplement C.1).

The feature layer representations are determined by the conditional distribution of the preactivations. For the outliers, the preactivation posterior is a Dirac delta at $O(\sqrt{N})$ values, $P(\boldsymbol{z}_i \mid \sqrt{N}\bar{\boldsymbol{a}}_i) = \delta(\boldsymbol{z}_i - \sqrt{N}\bar{\boldsymbol{z}}_i)$. Thus, for the $n = O(1)$ outlier neurons the $O(\sqrt{N})$ preactivations and readout weights exploit the homogeneity of the activation function to jointly overcome the non-lazy scaling. All $N - n$ neurons of the bulk are identically distributed according to a non-Gaussian posterior $P(\boldsymbol{z}_i \mid \boldsymbol{a}_0)$, which means that the neurons from the bulk share the same code.

In total, the bulk and outliers provide $n + 1$ codes. Since $m$ codes are necessary to solve a task, the minimal number of outliers is

https://doi.org/10.1038/s41467-025-58276-6

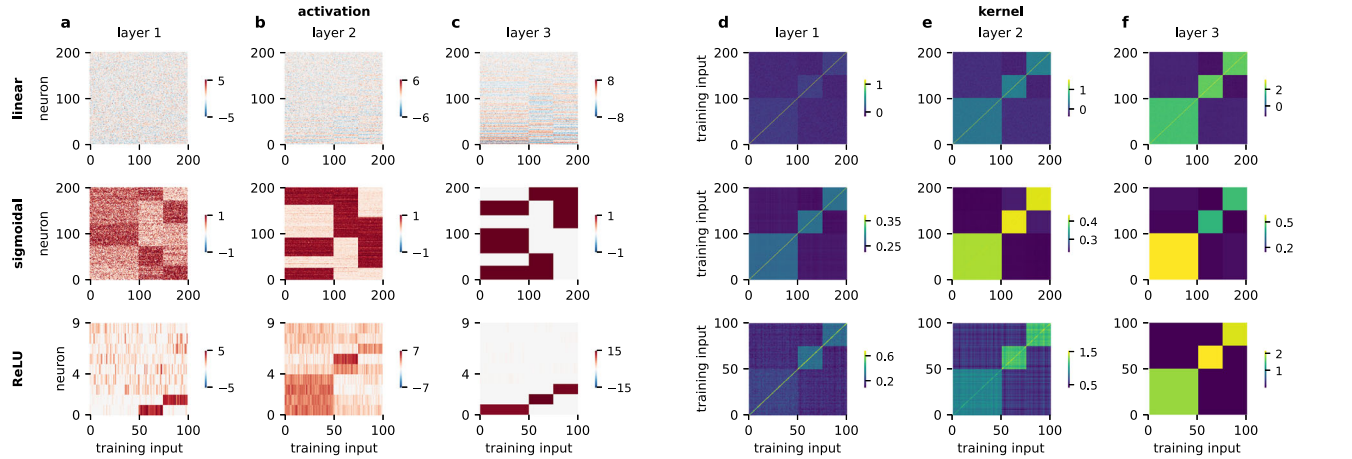


**Fig. 3 | Lower layer representations in linear (top), sigmoidal (middle), and ReLU (bottom) networks with three hidden layers. a–c** Activations on all training inputs for a given weight sample across layers. For ReLU, only the nine most active neurons are shown. **d–f** Posterior-averaged kernels on training inputs across layers.

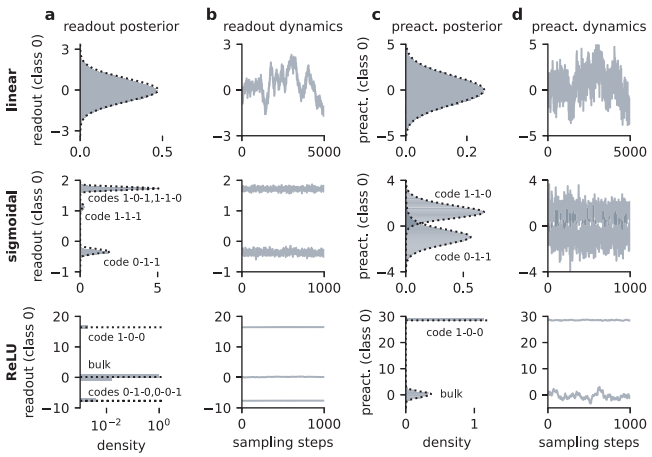


**Fig. 4 | Feature layer dynamics for linear (top), sigmoidal (middle), and ReLU (bottom) nonlinearity. a** Readout weight posterior on first class; theoretical distribution as black dashed line (for sigmoidal networks with finite $P$ correction, see supplement B.1). **b** Readout weight dynamics for selected neurons during sampling. **c** Last-layer-preactivation posterior on inputs from the first class; theoretical distribution as black dashed line. For sigmoidal and ReLU networks distributions conditioned on selected codes are shown. **d** Last-layer-preactivation dynamics during sampling. Neurons selected corresponding to the distributions shown in **c**. preact., preactivation.

$n = m - 1$; if the bulk is task agnostic the minimal number of outliers increases to $n = m$. On the toy task with three classes, the latter occurs: There are three outliers coding for a single class each while the bulk is largely task agnostic (Fig. 2b). The relation between outlier readout weights and activations is straightforward: Readout weights for the outlier neurons are positive for the coded class and negative otherwise (Fig. 2a) leading in combination to a perfect solution of the task (Fig. 2c).

We investigate deeper ReLU networks only empirically. For a three-hidden-layer network on the toy task, all layers display task-dependent representations (Fig. 3a–c): The first layer displays a sparse coding scheme with two outlier neurons coding for the second and third class and a bulk which is active on all classes, the second layer a coding scheme with redundant outliers coding for the first and second class, and the feature layer a sparse coding scheme with three outlier neurons coding for one class each as in the single-hidden-layer case. Corresponding to the task-dependent representations, the kernels posses low-rank components in all layers, which increase in strength towards the feature layer (Fig. 3d–f). For the feature layer representations, increasing the depth has no effect: Already for a single hidden layer there are no fluctuations across inputs from the same class, thus the sparse coding scheme cannot sharpen, unlike the redundant coding scheme in sigmoidal networks.

## Dynamics

So far, we considered a single sample drawn from the weight posterior to reveal the structure of a typical solution. Next, we investigate the impact of this structure on the dynamics of readout weights and representations during sampling from the posterior.

For linear networks, the posteriors of both readout weights (Fig. 4a) and activations (Fig. 4c) are Gaussian and thus unimodal. During sampling, their single peak is fully explored by any given readout weight (Fig. 4b) and activation (Fig. 4d).

In sigmoidal networks the readout posterior splits into disconnected branches (Fig. 4a). This leads to a fundamental consequence of the theory: For any given neuron sampling is restricted to one of the branches of the posterior (Fig. 4b), i.e., ergodicity is broken. Since the readout weights determine the code through the conditional distribution $P(\boldsymbol{z}_i^L \mid \boldsymbol{a}_\gamma)$ (Fig. 4c), fixing the readout weights implies fixing the code, despite the non-vanishing fluctuations of preactivations (Fig. 4d). This has an important consequence for the permutation symmetry of fully connected networks: The permutation symmetry cannot be restored dynamically. In the linear case, all permutation symmetric solutions are visited during sampling, i.e., permutation symmetry is restored through the dynamics, while in the sigmoidal case, the neurons cannot permute across codes, and the permutation symmetry remains broken.

Remarkably, there is a fundamental difference between the lower layers and the feature layer in sigmoidal networks: In the feature layer, a neuron's code is fixed, but not in the lower layers (Fig. 5). This can be understood from the factorization of the posterior across layers, Eq. (3), where the lower layers are decoupled from the feature layer and thus not affected by the permutation symmetry breaking in the feature layer. Although the individual neurons change their code during sampling, the global structure of the code is preserved: For each weight sample, the neurons can be reordered such that the coding scheme is apparent (Fig. 5c, d).

Similar to the sigmoidal case, the readout posterior of ReLU networks splits into disconnected branches which correspond to the bulk and the outliers (Fig. 4a). Again, this implies ergodicity breaking: During sampling the readout weights are restricted to their branch of the posterior (Fig. 4b). Corresponding to the fixed readout weights

https://doi.org/10.1038/s41467-025-58276-6

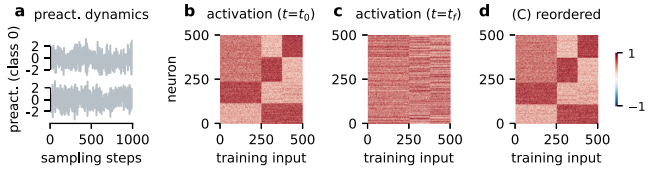


**Fig. 5 | Dynamics in the first layer of a two-hidden-layer sigmoidal network.** **a** Preactivation dynamics of selected first layer neurons on the first training input during sampling. **b**, **c** Activations of all neurons using first (**b**) and last (**c**) weight sample. **d** Reordered activations shown in **c**. preact., preactivation.

also the code is fixed: The outlier neurons do not permute, and the bulk neurons remain in the bulk during sampling (Fig. 4d). In contrast to the sigmoidal case and the bulk neurons, the activations of outlier neurons freeze as well; otherwise fluctuations cannot average out across neurons in the final readout. While the details between sigmoidal and ReLU networks differ, in both cases, permutation symmetry is broken.

### Generalization

Going beyond the training data, we investigate generalization to unseen test inputs. We consider both the representations of the test inputs, quantified by the posterior-averaged kernel $\mathcal{K}_\ell(\boldsymbol{x}_1,\boldsymbol{x}_2) = \langle \mathcal{K}_\ell(\boldsymbol{x}_1,\boldsymbol{x}_2;\Theta)\rangle_\Theta$, as well as the generalization performance of the mean predictor $f_r(\boldsymbol{x}) = \langle f_r(\boldsymbol{x};\Theta)\rangle_\Theta$.

To extend the theory to account for test inputs, we include the $P_*$ dimensional vector of test activations $\boldsymbol{z}_i^{\ell,*}$ with elements $z_i^\ell(\boldsymbol{x}_\mu^*)$ on a set of test inputs $\boldsymbol{x}_\mu^*$, $\mu = 1, \ldots, P_*$, in the joint posterior. The joint posterior still factorizes across neurons and layers into single neuron posteriors $P(\boldsymbol{z}_i^{L,*}|\boldsymbol{z}_i^L)P(\boldsymbol{z}_i^L|\boldsymbol{a}_i)P(\boldsymbol{a}_i)$ in the last layer and $P(\boldsymbol{z}_i^{\ell,*}|\boldsymbol{z}_i^\ell)P(\boldsymbol{z}_i^\ell)$ in the previous layers $\ell < L$. For all nonlinearities, the distribution of the test preactivations conditioned on the training preactivations, $P(\boldsymbol{z}_i^{\ell,*}|\boldsymbol{z}_i^\ell)$, is Gaussian in all layers $\ell = 1, \ldots, L$ (see supplements A.2, B.2, C). This allows to compute the mean predictor as well as the kernel on arbitrary inputs.

An important result of the theory is that the fluctuations of the predictor $\langle \delta f_r(\boldsymbol{x};\Theta)^2\rangle_\Theta$ can be neglected, unlike in lazy networks[20]. This implies that the (non-negative) variance contribution from the bias-variance decomposition $\langle [y_\mu^r - f_r(\boldsymbol{x}_\mu;\Theta)]^2\rangle_\Theta = [y_\mu^r - f_r(\boldsymbol{x}_\mu)]^2 + \langle \delta f_r(\boldsymbol{x}_\mu;\Theta)^2\rangle_\Theta$ is absent in the generalization error $\varepsilon_r = \frac{1}{P_*}\sum_{\mu=1}^{P_*}\langle [y_\mu^{r,*} - f_r(\boldsymbol{x}_\mu^*;\Theta)]^2\rangle_\Theta$. The remaining generalization error $\varepsilon_r = \frac{1}{P_*}\sum_{\mu=1}^{P_*}[y_\mu^r - f_r(\boldsymbol{x}_\mu)]^2$ is, therefore, reduced compared to the lazy case.

For linear networks, the mean predictor evaluates to $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{Y}^\top \boldsymbol{K}_0^{-1}\boldsymbol{k}_0(\boldsymbol{x})$ with $\boldsymbol{k}_0(\boldsymbol{x}) = \frac{1}{N_0}\boldsymbol{X}^\top\boldsymbol{x}$ and $\boldsymbol{K}_0 = \frac{1}{N_0}\boldsymbol{X}^\top\boldsymbol{X}$ and the posterior averaged kernel is $\mathcal{K}_\ell(\boldsymbol{x}_1,\boldsymbol{x}_2) = \kappa_0(\boldsymbol{x}_1,\boldsymbol{x}_2) + \sigma_a^{2(L-\ell)}\boldsymbol{f}(\boldsymbol{x}_1)^\top \boldsymbol{U}^{-(L-\ell+1)}\boldsymbol{f}(\boldsymbol{x}_2)$ with $\kappa_0(\boldsymbol{x}_1,\boldsymbol{x}_2) = \frac{1}{N_0}\boldsymbol{x}_1^\top\boldsymbol{x}_2$ (see supplement A.2). The test kernel is identical to the training kernel except that the training targets are replaced by the predictor. As in the training kernel, the learned part is low rank and becomes more prominent across layers, reaching $O(1)$ in the last layer. Remarkably, despite the strong learned representations, the mean predictor is identical to the lazy case[20] and the Gaussian Process (GP) theory (see supplement D.3). In contrast to the lazy case, the variance of the predictor can be neglected (see supplement A.3).

We apply the theory to the classification of the first three digits of MNIST. The nature of the solution is captured by an analog coding scheme, as in the toy task, which generalizes to unseen test inputs (Fig. 6a). The generalization of the representations is confirmed by the test kernel which has a block structure corresponding to the task (Fig. 6b). Also the mean predictor generalizes well (Fig. 6c) and the class-wise generalization error is reduced compared to the GP theory (Fig. 6d). Since the mean predictors are identical for non-lazy networks and the GP theory, the reduced generalization error is exclusively an effect of the reduced variance of the predictor.

For sigmoidal networks, the mean predictor combines contributions from all $n$ codes employed by the network, $f_r(\boldsymbol{x}) = \sum_{\gamma=1}^n P_\gamma a_\gamma^r \langle\langle \phi[z^L(\boldsymbol{x})]\rangle_{z^L(\boldsymbol{x})|\boldsymbol{z}^L}\rangle_{\boldsymbol{z}^L|\boldsymbol{a}_\gamma}$ (see supplement B.2). Similarly, the feature layer kernel is composed of the contributions from all codes $\mathcal{K}_L(\boldsymbol{x}_1,\boldsymbol{x}_2) = \sum_{\gamma=1}^n P_\gamma \langle\langle \phi[z^L(\boldsymbol{x}_1)]\phi[z^L(\boldsymbol{x}_2)]\rangle_{z^L(\boldsymbol{x}_1),z^L(\boldsymbol{x}_2)|\boldsymbol{z}^L}\rangle_{\boldsymbol{z}^L|\boldsymbol{a}_\gamma}$.

We apply the theory again to the classification of the first three digits in MNIST with a single-hidden-layer network. To make the theory tractable, we neglect the fluctuations of the preactivation conditioned on the readout weights. The test activations display a clear redundant coding scheme with small deviations on certain test inputs (Fig. 6a). In the test kernel, this leads to a clear block structure (Fig. 6b). The mean predictor is close to the test targets except on particularly difficult test inputs and accurately captured by the theory (Fig. 6c); the resulting generalization error is smaller than its counterpart based on the GP theory (Fig. 6d). Similar to linear networks, reduction of the generalization error is mainly driven by the reduced variance of the predictor although in this case also the non-lazy mean predictor performs slightly better.

Using randomly projected MNIST to enable sampling in the regime $P > N_0$ leads to similar results—a redundant coding scheme on training inputs and good generalization (see supplement E). In contrast, using the first three classes of CIFAR10 instead of MNIST with $P = 100$ still leads to a redundant coding on the training inputs but the coding scheme is lost on test inputs, indicating that the representations do not generalize well, which is confirmed by a high generalization error (see supplement E). Increasing the data set size to all ten classes and the full training set with $P = 50,000$ inputs and using a two-hidden-hidden layer network with $N = 1\,000$ neurons improves the generalization to an overall accuracy of 0.45; in this case, the feature layer training activations show a redundant coding scheme on training inputs (Fig. 7b) which generalize to varying degree to test inputs (Fig. 7e).

For ReLU networks the mean predictor comprises the contributions from the bulk and the outliers, $f_r(\boldsymbol{x}) = a_0^r\langle\langle \phi[z(\boldsymbol{x})]\rangle_{z(\boldsymbol{x})|\boldsymbol{z}}\rangle_{\boldsymbol{z}|\boldsymbol{a}_0} + \sum_{i=1}^n \bar{a}_i^r \phi(\bar{z}_i(\boldsymbol{x}))$ with $\bar{z}_i(\boldsymbol{x}) = \bar{\boldsymbol{z}}_i^\top \boldsymbol{K}_0^+ \boldsymbol{k}_0(\boldsymbol{x})$ for $i = 1, \ldots, n$. As in the other cases, the predictor variance can be neglected. The test kernel is also composed of the contributions from the bulk and the outliers, $K(\boldsymbol{x}_1,\boldsymbol{x}_2) = \langle\langle \phi[z(\boldsymbol{x}_1)]\phi[z(\boldsymbol{x}_2)]\rangle_{z(\boldsymbol{x}_1),z(\boldsymbol{x}_2)|\boldsymbol{z}}\rangle_{\boldsymbol{z}|\boldsymbol{a}_0} + \sum_{\gamma=1}^n \phi[z_\gamma(\boldsymbol{x}_1)]\phi[z_\gamma(\boldsymbol{x}_2)]$. Note that for the outliers, the product of two $O(\sqrt{N})$ activations jointly overcomes the $1/N$ in the kernel, Eq. (2), similar to the predictor where the $O(\sqrt{N})$ readout weights and the $O(\sqrt{N})$ activations jointly overcome the non-lazy scaling.

On the first three digits of MNIST, the test activations exhibit three outliers which code for one class each (Fig. 6a), leading to a test kernel that displays a clear block structure due to the outliers (Fig. 6b). The mean predictor is accurately captured by a theory which only takes the outliers into account (Fig. 6c). In terms of the generalization error, the non-lazy network outperforms the GP predictor (Fig. 6d), again mainly due to the reduction of the predictor variance.

## Discussion

We developed a theory for the weight posterior of non-lazy networks in the limit of infinite width and data set size, from which we derived analytical expressions for the single neuron posteriors of the readout weights and the preactivation on training and test inputs. These single neuron posteriors revealed that the learned representations are embedded into the network using distinct coding schemes. Furthermore, we used the single neuron posteriors to derive the mean

https://doi.org/10.1038/s41467-025-58276-6

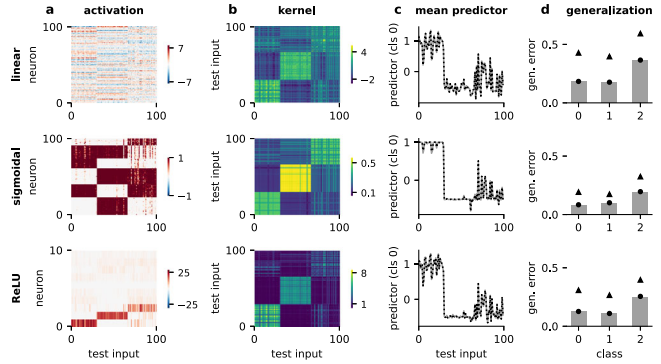


**Fig. 6 | Generalization in single-hidden-layer networks on MNIST for linear (top), sigmoidal (middle), and ReLU (bottom) nonlinearity. a** Activations of all neurons on 100 test inputs for a given weight sample; for ReLU only the ten most active neurons are shown. **b** Posterior-averaged kernel on 100 test inputs. **c** Mean predictor for class 0 from sampling (gray) and theory (black dashed). **d** Generalization error for each class averaged over 1000 test inputs from sampling (gray bars), theory (black circles), and GP theory (back triangles). gen. error, generalization error.

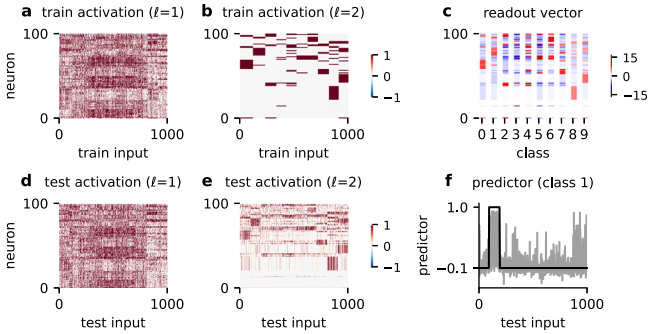


**Fig. 7 | Generalization of two-hidden-layer sigmoidal networks trained on full CIFAR10 training set. a**, **b**, **d**, **e** Activations of 100 randomly selected first (**a**, **d**) and second (**b**, **e**) layer neurons on 1000 randomly selected training (**a**, **b**) and test (**d**, **e**) inputs for a given weight sample. **c** Readout weights of all three classes for the 100 selected neurons. **f** Predictor for first class using the weighted sample used in the other panels (gray) and test target (black).

predictor and the mean kernels on training and test inputs. We applied the theory to two classification tasks: A simple toy model using orthogonal data and random labels to highlight the coding schemes and image classification using MNIST and CIFAR10 to investigate generalization. In both cases, the theoretical results are in excellent agreement with empirical samples from the weight posterior.

## Coding schemes

We show that the embedding of learned representations by the neurons exhibits a remarkable structure: A coding scheme where distinct populations of neurons are characterized by the subset of classes that activate them. The details of the coding scheme depend strongly on the nonlinearity (Fig. 2): Linear networks display an analog coding scheme where all neurons code for all classes in a graded manner, sigmoidal networks display a redundant coding scheme where large populations of neurons are active on specific combinations of classes, and ReLU networks display a sparse coding scheme in which a few individual neurons are active on specific classes while the remaining neurons are task agnostic. In networks with multiple hidden layers, the coding scheme appears in all layers but is sharpened across layers; the coding scheme in the last layer remains the same as in the single-layer case (Fig. 3).

We have shown that the analog coding scheme is a unique feature of linear networks, that a redundant coding scheme emerges for sigmoidal nonlinearities, and that the sparse coding scheme appears for rectified activation functions. The sparse coding scheme does not appear to be the result of the non-smoothness or the homogeneity of the ReLU nonlinearity. In fact, this scheme also arises for $\phi(z) = \log(1+e^z)$ and $\phi(z) = \max(0, z^2)$ (see supplement E).

Sparse representations have classically been associated with rectified nonlinearities in combination with $L1$ regularization in both neuroscience[46,47] and machine learning[48]. Moreover, it was recently shown that input noise can induce sparse representations in ReLU networks without $L1$ regularization[49]. In our case, the sparse representations emerge with $L2$ instead of $L1$ regularization on the weights and without input noise.

The fact that $L2$ regularization is frequently employed in practice raises the question: Why were the coding schemes not previously observed? Compared to standard training protocols, there are two main differences: (1) The sampling from the posterior, i.e., training for a long time with added noise, and (2) the data set size-dependent strength of the readout weight variance, which corresponds to a data set size dependent regularization using weight decay. Empirically, we see that a clear redundant coding scheme emerges also in the absence of noise and data set size-dependent regularization on the toy task (see supplement E), indicating the the coding scheme characterizes the minimum norm solution. However, in general, we expect that the minimum norm solution is separated from the initialization by barriers such that noise is necessary to overcome these barriers. Adding noise, in turn, requires the stronger regularization implemented through the scaled readout weight variance (see supplement E). Furthermore, we note that the final coding schemes only emerge after orders of magnitude more training steps (see supplement E).

From a theoretical perspective, we hypothesize that the coding schemes were elusive because they are not captured by the kernel, which is a frequently used observable in theoretical studies[26,38–41]. Furthermore, the coding schemes are controlled by the distribution of the readout weights, which have been marginalized in previous work[18,25,26,41,50]. Here, we established the readout weights themselves as crucial order parameters controlling the properties of the learned representations.

## Permutation symmetry breaking

The coding schemes determine the structure of a typical solution sampled from the weight posterior, e.g., for sigmoidal networks, a solution with a redundant coding scheme. Due to the neuron permutation symmetry of the posterior, each typical solution has permuted counterparts which are also typical solutions. Permutation symmetry breaking occurs if these permuted solutions are disconnected in solution space, i.e., if the posterior contains high barriers between the permuted solutions.

In sigmoidal and ReLU networks, permutation symmetry is broken. The theoretical signature of this symmetry breaking is the disconnected structure of the posterior of the readout weights, where the different branches are separated by high barriers. Numerically, symmetry breaking is evidenced by the fact that, at equilibrium, neurons do not change their coding identity with sampling time. We note that for the readout weights, the symmetry broken phase is also a frozen state, namely not only the coding structure is fixed, but also there are no fluctuations in the magnitude of each weight (Fig. 2). In contrast, activations in the hidden layer, which are also constant in their coding, do exhibit residual temporal fluctuations around a pronounced mean which carries the task-relevant information.

Interestingly, the situation is more involved in networks with two hidden layers (Fig. 4). In the first layer, permutation symmetry is broken, and the neurons' code is frozen in ReLU networks but not in sigmoidal networks. In the latter case, the typical solution has a

prominent coding scheme, but individual neurons switch their code during sampling.

Symmetry breaking has been frequently discussed in the context of learning in artificial neural networks, for example, replica symmetry breaking in perceptrons with binary weights[51–53], permutation symmetry breaking in fully connected networks[54,55] and restricted Boltzmann machines[56], or continuous symmetry breaking of the "kinetic energy" (reflecting the learning rule)[57,58]. Furthermore, the breaking of parity symmetry has been linked to feature learning[50], albeit in a different scaling limit. The role of an intact (not broken) permutation symmetry has been explored in the context of the connectedness of the loss landscape[59–61]. Here, we establish a direct link between symmetry breaking and the nature of the neural representations.

### Limitations of the theory

For our theory, we assume that the network width $N$ and the data set size $P$ are large, formally captured by the limit $N, P \to \infty$. Both limits are necessary for the coding schemes to emerge. Surprisingly, the limits are not coupled in the theory, i.e., we do not need to assume a fixed ratio $\alpha = P/N$. Correspondingly, we observe a redundant coding scheme for a $N = 1000$ sigmoidal network using the full CIFAR10 training set with $P = 50,000$. Still, we expect that our theory ceases to hold close to the network's capacity, i.e., when $P = O(N^2)$.

A second important assumption is that we work in the zero temperature limit where the posterior is restricted to the solution space. Which aspects of the theory hold at higher temperatures is an important open question. Empirically, we observe a critical temperature above which there is no longer a redundant coding scheme for sigmoidal network (see supplement E).

Our theory is valid only if the number of hidden layers and the number of outputs is small. The extension to deeper networks, in particular networks with residual connections, or to networks with a large number of outputs, as necessary, for example, for ImageNet, are interesting questions for future research.

### Neural collapse

The redundant coding scheme in sigmoidal networks and the sparse coding scheme in ReLU networks are closely related to the phenomenon of neural collapse[62]. The two main properties of collapse are (1) vanishing variability across inputs from the same class of the last layer activations and (2) last layer activations forming an equiangular tight frame (centered class means are equidistant and equiangular with maximum angle). There are two additional properties which, however, follow from the first two under minimal assumptions[62]. We note that collapse is determined only by training data in the original definition.

Formally, the first property of collapse is violated in the non-lazy networks investigated here due to the non-vanishing across-neuron variability of the activations given the readout weights. However, the mean activations conditional on the readout weights, which carry the task-relevant information, generate an equiangular tight frame in both sigmoidal and ReLU networks. This creates an interesting link to empirically trained networks where neural collapse has been shown to occur under a wide range of conditions[62,63].

Neglecting the non-vanishing variability, the main difference between neural collapse and the coding schemes is that the latter imposes a more specific structure. Technically, the equiangular tight frame characterizing neural collapse is invariant under orthogonal transformations, while the coding schemes are invariant under permutations, which is a subset of the orthogonal transformations. This additional structure makes the representations highly interpretable in terms of a neural code—conversely, applying, e.g., a rotation in neuron space to the redundant or sparse coding scheme would hide the immediately apparent structure of the solution.

### Representation learning and generalization

The case of non-lazy linear networks makes an interesting point about the interplay between feature learning and generalization: Although the networks learn strong, task-dependent representations, the mean predictor is identical to the Gaussian Process limit where the features are random. The only difference between non-lazy networks and random features is a reduction in the predictor variance. Thus, this provides an explicit example where feature learning only mildly helps generalization through the reduction of the predictor variance.

More generally, in all examples, the improved performance of non-lazy networks (Fig. 6) is mainly driven by the reduction of the predictor variance; the mean predictor does not generalize significantly better than a predictor based on random features. This shows an important limitation of our work: In order to achieve good generalization performance, it might be necessary to consider deeper nonlinear architectures or additional structures in the model. This is in line with recent empirical results which try to push multi-layer perceptrons to better performance[64].

While the learned representations might not necessarily help generalization on the trained task, they can still be useful for few-shot learning of a novel task[65–68]. Indeed, neural collapse has been shown to be helpful for transfer learning if neural collapse still holds (approximately) on inputs from the novel classes[69]. Due to the relation between coding schemes and neural collapse, this suggests that the learned representations investigated here are useful for downstream tasks—if the nature of the solution does not change on the new inputs. This remains to be systematically explored.

### Lazy vs. non-lazy regime

The definition of a lazy vs. non-lazy regime is subtle. Originally, the lazy regime was defined by the requirement that the learned changes in the weights affect the final output only linearly[70]. This implies that the learned changes in the representations are weak since changes in the hidden layer weights nonlinearly affect the final output. To overcome the weak representation learning, the non-lazy regime was defined as $O(1)$ changes of the features during the first steps of gradient descent[38,39]. This definition leads to an initialization where the readout scales its inputs with $1/N$[39]. However, the scaling might change during training—which is not captured by a definition at initialization.

We here define the non-lazy regime such that the readout scales its inputs with $1/N$ after learning, i.e., under the posterior. Importantly, we prevent the readout weights overcome the scaling by growing their norm, which leads to the requirement of a decreasing prior variance of the readout weights with increasing $P$. The definition implies that strong representations are learned: The readout weights must be aligned with the last layer's features on all training inputs, which is not the case for random features.

## Methods

### Weight posterior

The weight posterior is[20]

$$P(\Theta) = \frac{1}{Z}\exp\left[-\beta\mathcal{L}(\Theta) + \log P_0(\Theta)\right] \tag{4}$$

where $Z = \int dP_0(\Theta)\,\exp[-\beta\mathcal{L}(\Theta)]$ is the partition function, $\mathcal{L}(\Theta) = \frac{1}{2}\sum_{r=1}^{m}\sum_{\mu=1}^{P}[y_\mu^r - f_r(\boldsymbol{x}_\mu;\Theta)]^2$ a mean-squared error loss, and $P_0(\Theta)$ an i.i.d. zero-mean Gaussian prior with prior variances $\sigma_a^2$, $\sigma_\ell^2$ for readout and hidden weights, respectively. Temperature $T = 1/\beta$ controls the relative importance of the quadratic loss and the prior. In the overparameterized regime, the limit $T \to 0$ restricts the posterior to the solution space where the network interpolates the training data. For simplicity, we set $\sigma_\ell = 1$ in the main text. Throughout, we denote expectations w.r.t. the weight posterior by $\langle\cdot\rangle_\Theta$.

The interplay between loss and prior shapes the solutions found by the networks. Importantly, on the solution space the prior alone determines the posterior probability. Thus, the prior plays a central role for regularization which complements the implicit regularization due to the non-lazy scaling. The Gaussian prior used here can be interpreted as an $L2$ regularization of the weights.

Samples from the posterior can be obtained using Langevin dynamics $\frac{d}{dt}\Theta = -\nabla\mathcal{L}(\Theta) + T\nabla \log P_0(\Theta) + \sqrt{2T}\xi(t)$ where $\xi(t)$ are independent Gaussian white noise processes. These dynamics first approach the solution space on a timescale of $O(1)$ and subsequently diffuse on the solution space on a much slower time scale of $O(T^{-1})$[71]; after equilibration the dynamics provide samples from the posterior.

**Scaling limit.** We consider the limit $N, N_0, P \to \infty$ while the number of readout units as well as the number of layers remain finite, i.e., $m, L = O(1)$. Because the gradient of $\mathcal{L}(\Theta)$ is small due to the non-lazy scaling of the readout, the noise introduced by the temperature needs to be scaled down as well, requiring the scaling $T \to T/N$. In all other parts of the manuscript $T$ denotes the rescaled temperature which therefore remains of $O(1)$ as $N$ increases. In the present work, we focus on the regime where the training loss is essentially zero, and therefore consider the limit $T \to 0$ for the theory; to generate empirical samples from the weight posterior we use a small but non-vanishing rescaled temperature.

Under the posterior, the non-lazy scaling leads to large readout weights: The posterior norm per neuron grows with $P$, thereby compensating for the non-lazy scaling. To avoid this undoing of the non-lazy scaling, we scale down $\sigma_a^2$ with $P$, guaranteeing that the posterior norm of the readout weights per neuron is $O(1)$. The precise scaling depends on the depth and the type of nonlinearity.

In total there are three differences between (4) and the corresponding posterior in the lazy regime[20]: (1) The non-lazy scaling of the readout in (1) with $1/N$ instead of $1/\sqrt{N}$. (2) The scaling of the temperature with $1/N$. (3) The prior variance of the readout weights $\sigma_a^2$ needs to be scaled down with $P$.

### Parameters used in Figures

Figure 1 : single-hidden-layer networks $L = 1$ with a single output $m = 1$ and $N = P = 500$, $N_0 = 510$. For the task the classes are assigned with probability 1/2, the targets are binary $y = \pm 1$ according to the class.

Figures 2, 4: single-hidden-layer networks $L = 1$ with $m = 3$ outputs. Parameters for linear, sigmoidal, and ReLU networks, respectively: $N = P = 200, N_0 = 220$, classes assigned with fixed ratios [1/2, 1/4, 1/4], targets $y_+ = 1$ and $y_- = 0$; $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios [1/2, 1/4, 1/4], targets $y_+ = 1$ and $y_- = 1/2$; $N = P = 500, N_0 = 520$, classes assigned with fixed ratios [1/2, 1/4, 1/4], targets $y_+ = 1$ and $y_- = -1/2$.

Figure 3 : three-hidden-layer networks $L = 3$ with $m = 3$ outputs. Parameters for linear, sigmoidal, and ReLU networks, respectively: $N = P = 200, N_0 = 220$, classes assigned with fixed ratios [1/2, 1/4, 1/4], targets $y_+ = 1$ and $y_- = -1/2$; $N = P = 200, N_0 = 220$, classes assigned with fixed ratios [1/2, 1/4, 1/4], targets $y_+ = 1$ and $y_- = -1/2$; $N = P = 100$, $N_0 = 120$, classes assigned with fixed ratios [1/2, 1/4, 1/4], targets $y_+ = 1$ and $y_- = -1/2$.

Figure 5 : two-hidden-layer sigmoidal networks $L = 2$ with $m = 3$ outputs. Parameters: $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios [1/2, 1/4, 1/4], targets $y_+ = 1$ and $y_- = 1/2$.

Figure 6 : single-hidden-layer networks $L = 1$ with $m = 3$ outputs. Parameters: $N = P = 100$, $N_0 = 784$, classes 0, 1, 2 assigned randomly with probability 1/3, targets $y_+ = 1$ and $y_- = -1/2$.

Figure 7 : two-hidden-layer sigmoidal networks $L = 2$ with $m = 10$ outputs. Parameters: $N = 1000, P = 50{,}000, N_0 = 3072$, targets $y_+ = 1$ and $y_- = -1/10$.

## Data availability

The sampled weights are available on figshare with the identifier https://doi.org/10.6084/m9.figshare.26539129.

## Code availability

The code is available on figshare with the identifier https://doi.org/10.6084/m9.figshare.26539129.

## References


1. Bengio, Y., Courville, A. & Vincent, P. Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1798–1828 (2013).
2. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436 (2015).
3. Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning* (MIT Press, 2016).
4. Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. Understanding deep learning requires rethinking generalization, in https://openreview.net/forum?id=Sy8gdB9xx. *International Conference on Learning Representations* (Curran Associates, Inc., 2017).
5. Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM* **64**, 107–115 (2021).
6. Belkin, M., Hsu, D., Ma, S. & Mandal, S. Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proc. Natl Acad. Sci.* **116**, 15849 (2019).
7. Nakkiran, P. et al. Deep double descent: Where bigger models and more data hurt, in https://openreview.net/forum?id=B1g5sA4twr. *International Conference on Learning Representations* (Curran Associates, Inc., 2020).
8. Belkin, M. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numerica* **30**, 203–248 (2021).
9. Shwartz-Ziv, R. et al. Just how flexible are neural networks in practice? https://arxiv.org/abs/2406.11463 (2024).
10. MacKay, D. J. *Information Theory, Inference and Learning Algorithms* (Cambridge University Press, 2003)
11. Bahri, Y. et al. Statistical mechanics of deep learning. *Annu. Rev. Condens. Matter Phys.* **11**, 501 (2020).
12. Neal, R. M., https://doi.org/10.1007/978-1-4612-0745-0. *Bayesian Learning for Neural Networks* (Springer New York, 1996).
13. Williams, C. Computing with infinite networks, in https://proceedings.neurips.cc/paper/1996/file/ae5e3ce40e0404a45ecacaaf05e5f735-Paper.pdf. *Advances in Neural Information Processing Systems*, Vol. 9 (eds Mozer, M., Jordan, M., & Petsche, T.) (MIT Press, 1996).
14. Lee, J. et al. Deep neural networks as gaussian processes, in https://openreview.net/forum?id=B1EA-M-0Z. *International Conference on Learning Representations* (Curran Associates, Inc., 2018).
15. Matthews, A. G. D. G., Hron, J., Rowland, M., Turner, R. E. & Ghahramani, Z. Gaussian process behaviour in wide deep neural networks, in https://openreview.net/forum?id=H1-nGgWC-. *International Conference on Learning Representations* (Curran Associates, Inc., 2018).
16. Yang, G. Wide feedforward or recurrent neural networks of any architecture are gaussian processes, in *Advances in Neural Information Processing Systems*, Vol. 32 (eds Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., & Garnett, R.) (Curran Associates, Inc., 2019).
17. Naveh, G., Ben David, O., Sompolinsky, H. & Ringel, Z. Predicting the outputs of finite deep neural networks trained with noisy gradients. *Phys. Rev. E* **104**, 064301 (2021).
18. Segadlo, K. et al. Unified field theoretical approach to deep and recurrent neuronal networks. *J. Stat. Mech. Theory Exp.* **2022**, 103401 (2022).
19. Hron, J., Novak, R., Pennington, J. & Sohl-Dickstein, J. Wide Bayesian neural networks have a simple weight posterior: theory and

accelerated sampling. in: *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 162 (eds Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., & Sabato, S.) 8926–8945 (PMLR, 2022).
20. Li, Q. & Sompolinsky, H. Statistical mechanics of deep linear neural networks: the backpropagating kernel renormalization. *Phys. Rev. X* **11**, 031059 (2021).
21. Naveh, G. & Ringel, Z. A self consistent theory of gaussian processes captures feature learning effects in finite CNNs, in https://openreview.net/forum?id=vBYwwBxVcsE. *Advances in Neural Information Processing Systems,* Vol. 34 (eds Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., & Vaughan, J. W.) (Curran Associates, Inc., 2021).
22. Zavatone-Veth, J. A., Tong, W. L. & Pehlevan, C. Contrasting random and learned features in deep bayesian linear regression. *Phys. Rev. E* **105**, 064118 (2022).
23. Cui, H., Krzakala, F. & Zdeborova, L. Bayes-optimal learning of deep random networks of extensive-width, in https://proceedings.mlr.press/v202/cui23b.html. *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 202 (eds Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., & Scarlett, J.) 6468–6521 (PMLR, 2023).
24. Hanin, B. & Zlokapa, A. Bayesian interpolation with deep linear networks. *Proc. Natl Acad. Sci.* **120**, e2301345120 (2023).
25. Pacelli, R. et al. A statistical mechanics framework for bayesian deep neural networks beyond the infinite-width limit. *Nat. Mach. Intell.* **5**, 1497–1507 (2023).
26. Fischer, K. et al. Critical feature learning in deep neural networks, in https://proceedings.mlr.press/v235/fischer24a.html. *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 235 (eds Salakhutdinov, R. et al.) 13660–13690 (PMLR, 2024).
27. Chizat, L., Oyallon, E. & Bach, F. On lazy training in differentiable programming, in https://proceedings.neurips.cc/paper_files/paper/2019/file/ae614c557843b1df326cb29c57225459-Paper.pdf. *Advances in Neural Information Processing Systems*, Vol. 32 (eds Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., & Garnett, R.) (Curran Associates, Inc., 2019).
28. Woodworth, B. et al. Kernel and rich regimes in overparametrized models, in https://proceedings.mlr.press/v125/woodworth20a.html. *Proceedings of Thirty Third Conference on Learning Theory*, Proceedings of Machine Learning Research, Vol. 125 (eds Abernethy, J. & Agarwal, S.) 3635–3673 (PMLR, 2020).
29. Geiger, M., Spigler, S., Jacot, A. & Wyart, M. Disentangling feature and lazy training in deep neural networks. *J. Stat. Mech.: Theory Exp.* **2020**, 113301 (2020).
30. Yaida, S. Non-Gaussian processes and neural networks at finite widths., in https://proceedings.mlr.press/v107/yaida20a.html. *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, Proceedings of Machine Learning Research, Vol. 107 (eds Lu, J. & Ward, R.) 165–192 (PMLR, 2020).
31. Dyer, E. & Gur-Ari, G. Asymptotics of wide networks from feynman diagrams, in https://openreview.net/forum?id=S1gFvANKDS. *International Conference on Learning Representations* (Curran Associates, Inc., 2020).
32. Zavatone-Veth, J. A., Canatar, A., Ruben, B. & Pehlevan, C. Asymptotics of representation learning in finite bayesian neural networks, in https://openreview.net/forum?id=1oRFmD0Fl-5. *Advances in Neural Information Processing Systems,* Vol. 34 (eds Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., & Vaughan, J. W.) (Curran Associates, Inc., 2021).
33. Roberts, D. A., Yaida, S. & Hanin, B. *The Principles of Deep Learning Theory* (Cambridge University Press, 2022).
34. Mei, S., Montanari, A. & Nguyen, P.-M. A mean field view of the landscape of two-layer neural networks. *Proc. Natl Acad. Sci.* **115**, E7665 (2018).
35. Chizat, L. & Bach, F. On the global convergence of gradient descent for over-parameterized models using optimal transport, in https://proceedings.neurips.cc/paper_files/paper/2018/file/a1afc58c6ca9540d057299ec3016d726-Paper.pdf. *Advances in Neural Information Processing Systems*, Vol. 31 (eds Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., & Garnett, R. (Curran Associates, Inc., 2018).
36. Rotskoff, G. & Vanden-Eijnden, E. Parameters as interacting particles: long time convergence and asymptotic error scaling of neural networks, in https://proceedings.neurips.cc/paper_files/paper/2018/file/196f5641aa9dc87067da4ff90fd81e7b-Paper.pdf. *Advances in Neural Information Processing Systems*, Vol. 31 (eds Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., & Garnett, R. (Curran Associates, Inc., 2018).
37. Sirignano, J. & Spiliopoulos, K. Mean field analysis of neural networks: a central limit theorem. *Stoch. Process. Appl.* **130**, 1820 (2020).
38. Yang, G. & Hu, E. J. Tensor programs IV: Feature learning in infinite-width neural networks, in https://proceedings.mlr.press/v139/yang21c.html. *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 139 (eds Meila, M. & Zhang, T.) 11727–11737 (PMLR, 2021).
39. Bordelon, B. & Pehlevan, C. Self-consistent dynamical field theory of kernel evolution in wide neural networks, in https://openreview.net/forum?id=sipwrPCrIS. *Advances in Neural Information Processing Systems*, Vol. 35 (eds Koyejo, S. et al.) (Curran Associates, Inc., 2022).
40. Bordelon, B. & Pehlevan, C. Dynamics of finite width kernel and prediction fluctuations in mean field neural networks, in https://openreview.net/forum?id=fKwG6grp8o. *Advances in Neural Information Processing Systems*, Vol. 36 (eds Oh, A. H. et al.) (Curran Associates, Inc., 2023).
41. Seroussi, I., Naveh, G. & Ringel, Z. Separation of scales and a thermodynamic description of feature learning in some cnns, *Nat. Commun.* **14**, https://doi.org/10.1038/s41467-023-36361-y (2023).
42. Cortes, C., Mohri, M. & Rostamizadeh, A. Algorithms for learning kernels based on centered alignment. *J. Mach. Learn. Res.* **13**, 795 (2012).
43. Kornblith, S., Norouzi, M., Lee, H. & Hinton, G. Similarity of neural network representations revisited, in https://proceedings.mlr.press/v97/kornblith19a.html. *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 97 (eds Chaudhuri, K. & Salakhutdinov, R.). 3519–3529 (PMLR, 2019).
44. LeCun, Y. The mnist database of handwritten digits, http://yann.lecun.com/exdb/mnist/ (1998).
45. Alex, K. Learning multiple layers of features from tiny images, https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (2009).
46. Olshausen, B. A. & Field, D. J. Sparse coding with an overcomplete basis set: a strategy employed by v1? *Vis. Res.* **37**, 3311 (1997).
47. Olshausen, B. A. & Field, D. J. Sparse coding of sensory inputs. *Curr. Opin. Neurobiol.* **14**, 481 (2004).
48. Glorot, X., Bordes, A. & Bengio, Y. Deep sparse rectifier neural networks, in https://proceedings.mlr.press/v15/glorot11a.html. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, Vol. 15 (eds Gordon, G., Dunson, D., & Dudík, M.) 315–323 (PMLR, 2011).
49. Bricken, T., Schaeffer, R., Olshausen, B. & Kreiman, G. Emergence of sparse representations from noise, in https://proceedings.mlr.press/v202/bricken23a.html. *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 202 (eds Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., & Scarlett, J.) 3148–3191 (PMLR, 2023).

50. Rubin, N., Seroussi, I. & Ringel, Z. Grokking as a first order phase transition in two layer networks, in https://openreview.net/forum?id=3ROGsTX3IR. *The Twelfth International Conference on Learning Representations* (Curran Associates, Inc., 2024).
51. Krauth, W. & Mézard, M. Storage capacity of memory networks with binary couplings. *J. de. Phys.* **50**, 3057–3066 (1989).
52. Seung, H. S., Sompolinsky, H. & Tishby, N. Statistical mechanics of learning from examples. *Phys. Rev. A* **45**, 6056 (1992).
53. Watkin, T. L. H., Rau, A. & Biehl, M. The statistical mechanics of learning a rule. *Rev. Mod. Phys.* **65**, 499 (1993).
54. Barkai, E., Hansel, D. & Sompolinsky, H. Broken symmetries in multilayered perceptrons. *Phys. Rev. A* **45**, 4146 (1992).
55. Engel, A., Köhler, H. M., Tschepke, F., Vollmayr, H. & Zippelius, A. Storage capacity and learning algorithms for two-layer neural networks. *Phys. Rev. A* **45**, 7590 (1992).
56. Hou, T., Wong, K. Y. M. & Huang, H. Minimal model of permutation symmetry in unsupervised learning. *J. Phys. A: Math. Theor.* **52**, 414001 (2019).
57. Kunin, D., Sagastuy-Brena, J., Ganguli, S., Yamins, D. L. & Tanaka, H. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics, in https://openreview.net/forum?id=q8qLAbQBupm. *International Conference on Learning Representations* (Curran Associates, Inc., 2021).
58. Tanaka, H. & Kunin, D. Noether's learning dynamics: Role of symmetry breaking in neural networks, in https://proceedings.neurips.cc/paper_files/paper/2021/file/d76d8deea9c19cc9aaf2237d2bf2f785-Paper.pdf. *Advances in Neural Information Processing Systems*, Vol. 34 (eds Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., & Vaughan, J. W.) 25646–25660 (Curran Associates, Inc., 2021).
59. Brea, J., Simsek, B., Illing, B. & Gerstner, W. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. https://arxiv.org/abs/1907.02911 (2019).
60. Simsek, B. et al. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances, in https://proceedings.mlr.press/v139/simsek21a.html. *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 139. 9722–9732 (eds Meila, M. and Zhang, T.) (PMLR, 2021).
61. Entezari, R., Sedghi, H., Saukh, O. & Neyshabur, B. The role of permutation invariance in linear mode connectivity of neural networks, in https://openreview.net/forum?id=dNigytemkL. *International Conference on Learning Representations* (Curran Associates, Inc., 2022).
62. Papyan, V., Han, X. Y. & Donoho, D. L. Prevalence of neural collapse during the terminal phase of deep learning training. *Proc. Natl Acad. Sci.* **117**, 24652 (2020).
63. Han, X., Papyan, V. & Donoho, D. L. Neural collapse under MSE loss: Proximity to and dynamics on the central path, in https://openreview.net/forum?id=w1UbdvWH_R3. *International Conference on Learning Representations* (Curran Associates, Inc., 2022).
64. Bachmann, G., Anagnostidis, S. & Hofmann, T. Scaling mlps: A tale of inductive bias, in https://proceedings.neurips.cc/paper_files/paper/2023/file/bf2a5ce85aea9ff40d9bf8b2c2561cae-Paper-Conference.pdf. *Advances in Neural Information Processing Systems*, Vol. 36 (eds Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., & Levine, S.) 60821–60840 (Curran Associates, Inc., 2023).
65. Fei-Fei, L., Fergus, R. & Perona, P. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**, 594 (2006).
66. Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K. & Wierstra, D. Matching networks for one shot learning, in https://proceedings.neurips.cc/paper_files/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf. *Advances in Neural Information Processing Systems*, Vol. 29 (eds Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., & Garnett, R.) (Curran Associates, Inc., 2016).
67. Snell, J., Swersky, K. & Zemel, R. Prototypical networks for few-shot learning, in https://proceedings.neurips.cc/paper_files/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf. *Advances in Neural Information Processing Systems*, Vol. 30 (eds Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R.) (Curran Associates, Inc., 2017).
68. Sorscher, B., Ganguli, S. & Sompolinsky, H. Neural representational geometry underlies few-shot concept learning. *Proc. Natl Acad. Sci.* **119**, e2200800119 (2022).
69. Galanti, T., György, A. & Hutter, M. On the role of neural collapse in transfer learning, in https://openreview.net/forum?id=SwIp410B6aQ. *International Conference on Learning Representations* (Curran Associates, Inc., 2022).
70. Jacot, A., Gabriel, F. & Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks, in https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf. *Advances in Neural Information Processing Systems*, Vol. 31 (eds Bengio, S. et al.) (Curran Associates, Inc., 2018).
71. Avidan, Y., Li, Q. & Sompolinsky, H. Connecting ntk and nngp: a unified theoretical framework for neural network learning dynamics in the kernel regime. https://arxiv.org/abs/2309.04522 (2023).


## Acknowledgements

We would like to thank Qianyi Li for many helpful discussions and Lorenzo Tiberi, Chester Mantel, Kazuki Irie, and Haozhe Shan for their feedback on the manuscript. This research was supported by the Swartz Foundation, the Gatsby Charitable Foundation, the Kempner Institute for the Study of Natural and Artificial Intelligence, and Office of Naval Research (ONR) grant No. N0014-23-1-2051, and in part by grant NSF PHY-1748958 to the Kavli Institute for Theoretical Physics (KITP). The computations in this paper were run on the FASRC Cannon cluster supported by the FAS Division of Science Research Computing Group at Harvard University.

## Author contributions

A.v.M. and H.S. contributed to the development of the project. A.v.M. and H.S. developed the theory. A.v.M. performed simulations. A.v.M. and H.S. participated in the analysis and interpretation of the data. A.v.M. and H.S. wrote the paper.

## Competing interests

The authors declare no competing interests.

## Additional information

**Supplementary information** The online version contains supplementary material available at https://doi.org/10.1038/s41467-025-58276-6.

**Correspondence** and requests for materials should be addressed to Alexander van Meegen or Haim Sompolinsky.

**Peer review information** *Nature Communications* thanks Friedrich Schuessler and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. A peer review file is available.

**Reprints and permissions information** is available at http://www.nature.com/reprints

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

© The Author(s) 2025

# Coding schemes in neural networks learning classification tasks (Supplementary Information)

Alexander van Meegen[1,*] and Haim Sompolinsky[1,2,†]

[1] *Center for Brain Science, Harvard University, Cambridge, MA 02138*

[2] *Edmond and Lily Safra Center for Brain Sciences, Hebrew University, Jerusalem 9190501, Israel*

**CONTENTS**



## Appendix A: Linear Networks

### 1. Training Posterior

We start with the joint posterior of the readout weights and the training preactivations. The change of variable from the $N \times N$ (or $N \times N_0$) weights $W_\ell$ to the $N \times P$ preactivations $Z_\ell = \frac{1}{\sqrt{N}} W_\ell Z_{\ell-1}$ and $Z_1 = \frac{1}{\sqrt{N_0}} W_1 X$ replaces the prior distribution of the weights by the prior distribution of the preactivations which is $P_0(Z_\ell) = \mathcal{N}(Z_\ell \,|\, 0, I_N \otimes \sigma_\ell^2 K_{\ell-1})$ with $K_\ell = \frac{1}{N} Z_\ell^\top Z_\ell$, $\ell = 1, \ldots, L$, and $K_0 = \frac{1}{N_0} X^\top X$. Throughout, we use the matrix valued normal distribution (see, e.g.,

* alexander.vanmeegen@epfl.ch

† hsompolinsky@mcb.harvard.edu

[1]) to ease the notation, e.g., $\mathcal{N}(Z\,|\,0, I_N \otimes K)$ means $Z$ is zero mean Gaussian with correlation $\langle Z_{i\mu} Z_{j\nu}\rangle = \delta_{ij} K_{\mu\nu}$. We can write the training posterior as

$$P(A, Z_L, \ldots, Z_1) \propto \mathcal{N}\big(Y\,|\,\frac{1}{N} A^\top Z_L, \frac{T}{N} I_m \otimes I_P\big) P_0(A) \prod_{\ell=1}^{L} P_0(Z_\ell) \tag{A1}$$

where we already rescaled $T \to T/N$ and denoted the $N \times m$ matrix containing all readout weights by $A$.
Using the identity

$$\mathcal{N}\big(Y\,|\,m, u \otimes K\big) = \int dt\, e^{-i\mathrm{tr}\, t^\top Y + i\mathrm{tr}\, t^\top m - \frac{1}{2}\mathrm{tr}\, u t^\top K t}, \tag{A2}$$

where $t$ is a $P \times m$ matrix, decouples the first factor across neurons. The remaining factors $P_0(Z_\ell)$ are only coupled through the kernel $K_\ell$. Hence, we decouple these factors by introducing $K_\ell$, $\ell = 1, \ldots, L-1$, using Dirac deltas, $\delta(K_\ell - \frac{1}{N} Z_\ell^\top Z_\ell) = \int d\tilde{K}_\ell\, e^{-i\mathrm{tr}\, \tilde{K}_\ell^\top K_\ell + \frac{i}{N}\mathrm{tr}\, \tilde{K}_\ell^\top Z_\ell^\top Z_\ell}$. The training posterior factorizes across neurons into

$$\begin{aligned} P(A, Z_L, \ldots, Z_1) \propto \int dt \prod_{\ell=1}^{L-1} [dK_\ell d\tilde{K}_\ell]\, e^{\frac{T}{2} N \mathrm{tr}\, t^\top t - N \mathrm{tr}\, t^\top Y - \frac{1}{2} N \sum_{\ell=1}^{L-1} \mathrm{tr}\, \tilde{K}_\ell^\top K_\ell} \\ \times \prod_{i=1}^{N} \Big[ \underbrace{P_0(a_i) \mathcal{N}(z_i^L\,|\,0, \sigma_L^2 K_{L-1}) e^{a_i^\top t^\top z_i^L}}_{\propto P(a_i, z_i^L)} \Big] \prod_{\ell=1}^{L-1} \Big[ \prod_{j=1}^{N} \underbrace{\mathcal{N}(z_j^\ell\,|\,0, \sigma_\ell^2 K_{\ell-1}) e^{\frac{1}{2} z_j^{\ell\top} \tilde{K}_\ell z_j^\ell}}_{\propto P(z_j^\ell)} \Big] \end{aligned} \tag{A3}$$

where we rescaled $it \to Nt$ and $i\tilde{K}_\ell \to \frac{1}{2} N \tilde{K}_\ell$. The factorized single neuron posteriors are not yet normalized. Taking the normalization into account leads to

$$P(A, Z_L, \ldots, Z_1) \propto \int dt \prod_{\ell=1}^{L-1} [dK_\ell d\tilde{K}_\ell]\, e^{-NE(t, \{K_\ell, \tilde{K}_\ell\})} \prod_{i=1}^{N} [P(a_i, z_i^L)] \prod_{\ell=1}^{L-1} [\prod_{j=1}^{N} P(z_j^\ell)], \tag{A4}$$

$$\begin{aligned} E(t, \{K_\ell, \tilde{K}_\ell\}) = &- \frac{T}{2} \mathrm{tr}\, t^\top t + \mathrm{tr}\, t^\top Y + \frac{1}{2} \sum_{\ell=1}^{L-1} \mathrm{tr}\, \tilde{K}_\ell^\top K_\ell - \log \int dP_0(a) \int d\mathcal{N}(z^L\,|\,0, \sigma_L^2 K_{L-1})\, e^{a^\top t^\top z^L} \\ &- \sum_{\ell=1}^{L-1} \log \int d\mathcal{N}(z^\ell\,|\,0, \sigma_\ell^2 K_{\ell-1})\, e^{\frac{1}{2} z^{\ell\top} \tilde{K}_\ell z^\ell}. \end{aligned} \tag{A5}$$

The proportionality constant ensures $\int dt \prod_{\ell=1}^{L-1} dK_\ell d\tilde{K}_\ell\, e^{-NE(t, \{K_\ell, \tilde{K}_\ell\})} = 1$.
Using a saddle point approximation [2, 3] of the $t$, $K_\ell$, and $\tilde{K}_\ell$ integrals we arrive at

$$P(A, Z_L, \ldots, Z_1) = \prod_{i=1}^{N} [P(a_i) P(z_i^L\,|\,a_i)] \prod_{\ell=1}^{L-1} [\prod_{j=1}^{N} P(z_j^\ell)] \tag{A6}$$

where we also used Bayes rule $P(a_i, z_i^L) = P(a_i) P(z_i^L\,|\,a_i)$. The saddle point equations are

$$y_\mu^r = \langle a_r z_\mu^L \rangle_{a, z^L} + T t_\mu^r \tag{A7}$$

$$K_\ell = \langle z^\ell z^{\ell\top} \rangle_{z^\ell}, \quad \ell = 1, \ldots, L-1 \tag{A8}$$

$$\tilde{K}_{\ell-1} = \frac{1}{\sigma_\ell^2} K_{\ell-1}^{-1} \langle z^\ell z^{\ell\top} \rangle_{z^\ell} K_{\ell-1}^{-1} - K_{\ell-1}^{-1}, \quad \ell = 2, \ldots, L \tag{A9}$$

The single neuron posteriors are all Gaussian:

$$P(a) = \mathcal{N}(a\,|\,0, U) \tag{A10}$$

$$P(z^L \,|\, a) = \mathcal{N}(z^L \,|\, \sigma_L^2 K_{L-1} t a, \sigma_L^2 K_{L-1}) \tag{A11}$$

$$P(z^\ell) = \mathcal{N}(z^\ell \,|\, 0, [\sigma_\ell^{-2} K_{\ell-1}^{-1} - \tilde{K}_\ell]^{-1}), \quad \ell = 1, \dots, L-1 \tag{A12}$$

where we introduced the $m \times m$ covariance matrix of the readout weights

$$U = \left(\sigma_a^{-2} I_m - \sigma_L^2 t^\top K_{L-1} t\right)^{-1}. \tag{A13}$$

The expectations in the saddle point equations yield

$$\langle a z^{L\top} \rangle_{a, z^L} = \sigma_L^2 U t^\top K_{L-1} \tag{A14}$$

$$\langle z^\ell z^{\ell\top} \rangle_{z^\ell} = \left(\sigma_\ell^{-2} K_{\ell-1}^{-1} - \tilde{K}_\ell\right)^{-1}, \quad \ell = 1, \dots, L-1 \tag{A15}$$

We explicitly solve the saddle point equations at $T = 0$ and assume that all kernels are invertible. The first saddle point equation yields $t = \sigma_L^{-2} K_{L-1}^{-1} Y U^{-1}$. Computing the last layer kernel leads to

$$K_L = \sigma_L^2 K_{L-1} + Y U^{-1} Y^\top \tag{A16}$$

which reduces the saddle point equation for $\tilde{K}_{L-1}$ to

$$\tilde{K}_{L-1} = \sigma_L^{-2} K_{L-1}^{-1} Y U^{-1} Y^\top K_{L-1}^{-1} \tag{A17}$$

For the lower layers we combine $\tilde{K}_{\ell-1} = \frac{1}{\sigma_\ell^2} K_{\ell-1}^{-1} K_\ell K_{\ell-1}^{-1} - K_{\ell-1}^{-1}$ and $K_\ell = [\sigma_\ell^{-2} K_{\ell-1}^{-1} - \tilde{K}_\ell]^{-1}$ to $\sigma_\ell^2 K_{\ell-1} = \sigma_{\ell+1}^2 K_\ell K_{\ell+1}^{-1} K_\ell$. For $\ell = L-1$ we obtain

$$\sigma_{L-1}^2 K_{L-2} = \sigma_L^2 K_{L-1} K_L^{-1} K_{L-1} = K_{L-1} - \sigma_L^{-2} Y \left(U + \sigma_L^{-2} Y^\top K_{L-1}^{-1} Y\right)^{-1} Y^\top \tag{A18}$$

Inserting $t = \sigma_L^{-2} K_{L-1}^{-1} Y U^{-1}$ into the definition of $U$ leads to $U + \sigma_L^{-2} Y^\top K_{L-1}^{-1} Y = \sigma_a^{-2} U^2$ and thus

$$K_{L-1} = \sigma_{L-1}^2 K_{L-2} + \frac{\sigma_a^2}{\sigma_L^2} Y U^{-2} Y^\top \tag{A19}$$

Evaluating $\sigma_\ell^2 K_{\ell-1} = \sigma_{\ell+1}^2 K_\ell K_{\ell+1}^{-1} K_\ell$ at $\ell = L-2$ leads to

$$\sigma_{L-2}^2 K_{L-3} = \sigma_{L-1}^2 K_{L-2} K_{L-1}^{-1} K_{L-2} = K_{L-2} - \sigma_{L-1}^{-2} Y \left(\frac{\sigma_L^2}{\sigma_a^2} U^2 + \sigma_{L-1}^{-2} Y^\top K_{L-2}^{-1} Y\right)^{-1} Y^\top \tag{A20}$$

Inserting $K_{L-1} = \sigma_{L-1}^2 K_{L-2} + \frac{\sigma_a^2}{\sigma_L^2} Y U^{-2} Y^\top$ into $U + \sigma_L^{-2} Y^\top K_{L-1}^{-1} Y = \sigma_a^{-2} U^2$ leads to $\sigma_a^{-2} \sigma_L^2 U^2 + \sigma_{L-1}^{-2} Y^\top K_{L-2}^{-1} Y = \sigma_a^{-4} \sigma_L^2 U^3$ and thus

$$K_{L-2} = \sigma_{L-2}^2 K_{L-3} + \frac{\sigma_a^4}{\sigma_L^2 \sigma_{L-1}^2} Y U^{-3} Y^\top \tag{A21}$$

Iterating these steps yields

$$U = \sigma_a^2 I + \frac{\sigma_a^{2L}}{\prod_{\ell=1}^L \sigma_\ell^2} U^{-L} Y^\top K_0^{-1} Y \tag{A22}$$

**Readout weight scaling:** For $Y^\top K_0^{-1} Y = O(P)$ we need to scale $\sigma_a^2 = O(1/P^{1/L})$ to achieve finite norm of the readout weights, $U = O(1)$; not scaling $\sigma_a^2$ would lead to $U = O(P^{1/(L+1)})$ and hence a growing readout weight norm. In both cases the first term on the r.h.s. can be neglected at large $P$, leading to $U^{L+1} = \frac{\sigma_a^{2L}}{\prod_{\ell=1}^L \sigma_\ell^2} Y^\top K_0^{-1} Y$.

For the first layer kernel, iterating the above steps leads to

$$K_1 = \sigma_1^2 K_0 + \frac{\sigma_a^{2(L-1)}}{\prod_{\ell=2}^L \sigma_\ell^2} Y U^{-L} Y^\top \tag{A23}$$

For the kernels in the later layers $2 \leq \ell \leq L-1$ we use $K_\ell = \sigma_\ell^2 K_{\ell-1} + \frac{\sigma_a^{2(L-\ell)}}{\prod_{\ell'=\ell+1}^{L}\sigma_{\ell'}^2} Y U^{-(L-\ell+1)} Y^\top$ and iterate, starting from $\ell = 2$. For each $\ell$, $K_{\ell-1}$ contains a low rank contribution of strength $\sigma_a^{2(L-\ell+1)}$ which is suppressed by $O(1/P^{1/L})$ compared to the new low rank contribution of strength $\sigma_a^{2(L-\ell)}$, hence

$$K_\ell = \prod_{\ell'=1}^{\ell}[\sigma_{\ell'}^2] K_0 + \frac{\sigma_a^{2(L-\ell)}}{\prod_{\ell'=\ell+1}^{L}\sigma_{\ell'}^2} Y U^{-(L-\ell+1)} Y^\top, \quad 2 \leq \ell \leq L-1 \tag{A24}$$

For the last layer we use $K_L = \sigma_L^2 K_{L-1} + Y U^{-1} Y^\top$ and obtain

$$K_L = \prod_{\ell=1}^{L}[\sigma_\ell^2] K_0 + Y U^{-1} Y^\top \tag{A25}$$

Setting $\sigma_\ell = 1$ recovers the results stated in the main text.

**Summary:** Posterior distributions

$$P(a) = \mathcal{N}(a \,|\, 0, U) \tag{A26}$$

$$P(z^L \,|\, a) = \mathcal{N}(z^L \,|\, Y U^{-1} a, \sigma_L^2 K_{L-1}) \tag{A27}$$

$$P(z^\ell) = \mathcal{N}(z^\ell \,|\, 0, K_\ell), \quad \ell = 1, \ldots, L-1, \tag{A28}$$

with

$$U^{L+1} = \frac{\sigma_a^{2L}}{\prod_{\ell=1}^{L}\sigma_\ell^2} Y^\top K_0^{-1} Y, \tag{A29}$$

$$K_L = \prod_{\ell=1}^{L}[\sigma_\ell^2] K_0 + Y U^{-1} Y^\top, \tag{A30}$$

$$K_\ell = \prod_{\ell'=1}^{\ell}[\sigma_{\ell'}^2] K_0 + \frac{\sigma_a^{2(L-\ell)}}{\prod_{\ell'=\ell+1}^{L}\sigma_{\ell'}^2} Y U^{-(L-\ell+1)} Y^\top, \quad \ell = 1, \ldots, L-1 \tag{A31}$$

and $K_0 = \frac{1}{N_0} X^\top X$.

### 2. Test Posterior

For test inputs, we need to include the preactivations corresponding to the set of $P_*$ test inputs $x_*$. Thus, we change variables from the $N \times N$ (or $N \times N_0$) weights $W_\ell$ to the $N \times (P + P_*)$ preactivations $\hat{Z}_\ell = \frac{1}{\sqrt{N}} W_\ell \hat{Z}_{\ell-1}$ and $\hat{Z}_1 = \frac{1}{\sqrt{N_0}} W_1 \hat{X}$ where $\hat{X}$ denotes the $N_0 \times (P + P_*)$ matrix containing the training and test inputs. Conditional on the previous layer preactivations, the prior distribution of the preactivations is $P_0(\hat{Z}_\ell) = \mathcal{N}(\hat{Z}_\ell \,|\, 0, I_N \otimes \sigma_\ell^2 \hat{K}_{\ell-1})$ with $\hat{K}_\ell = \frac{1}{N} \hat{Z}_\ell^\top \hat{Z}_\ell$, $\ell = 1, \ldots, L$, and $\hat{K}_0 = \frac{1}{N_0} \hat{X}^\top \hat{X}$. As for the training posterior, we can write the test posterior as

$$P(A, \hat{Z}_L, \ldots, \hat{Z}_1) \propto \mathcal{N}\big(Y \,|\, \frac{1}{N} A^\top Z_L, \frac{T}{N} I_m \otimes I_P\big) P_0(A) \prod_{\ell=1}^{L} P_0(\hat{Z}_\ell). \tag{A32}$$

In the first factor, only the training preactivations $Z_\ell$ appear since the loss does not depend on the test set.

The first factor decouples using (A2); for the preactivations we need to introduce the $(P+P_*)\times(P+P_*)$ kernels $\hat{K}_\ell$ for $\ell = 1,\dots,L-1$. This leads to

$$P(A,\hat{Z}_L,\dots,\hat{Z}_1) \propto \int dt \prod_{\ell=1}^{L-1}[d\hat{K}_\ell d\tilde{\hat{K}}_\ell]\, e^{\frac{T}{2}N\mathrm{tr}\, t^\top t - N\mathrm{tr}\, t^\top Y - \frac{1}{2}N\sum_{\ell=1}^{L-1}\mathrm{tr}\,\tilde{\hat{K}}_\ell^\top \hat{K}_\ell} \\ \times \prod_{i=1}^{N}\Big[\underbrace{P_0(a_i)\mathcal{N}(\hat{z}_i^L \,|\, 0, \sigma_L^2\hat{K}_{L-1})e^{a_i^\top t^\top z_i^L}}_{\propto P(a_i,\hat{z}_i^L)}\Big]\prod_{\ell=1}^{L-1}\Big[\prod_{j=1}^{N}\underbrace{\mathcal{N}(\hat{z}_j^\ell \,|\, 0, \sigma_\ell^2\hat{K}_{\ell-1})e^{\frac{1}{2}\hat{z}_j^{\ell\top}\tilde{\hat{K}}_\ell\hat{z}_j^\ell}}_{\propto P(\hat{z}_j^\ell)}\Big]. \tag{A33}$$

Note that in the last factor of $P(a_i,\hat{z}_i^L)$ only the training preactivations appear. Normalizing the single neuron posteriors yields

$$P(A,\hat{Z}_L,\dots,\hat{Z}_1) \propto \int dt \prod_{\ell=1}^{L-1}[d\hat{K}_\ell d\tilde{\hat{K}}_\ell]\, e^{-NE(t,\{\hat{K}_\ell,\tilde{\hat{K}}_\ell\})}\prod_{i=1}^{N}[P(a_i,\hat{z}_i^L)]\prod_{\ell=1}^{L-1}[\prod_{j=1}^{N}P(\hat{z}_j^\ell)], \tag{A34}$$

$$E(t,\{\hat{K}_\ell,\tilde{\hat{K}}_\ell\}) = -\frac{T}{2}\mathrm{tr}\, t^\top t + \mathrm{tr}\, t^\top Y + \frac{1}{2}\sum_{\ell=1}^{L-1}\mathrm{tr}\,\tilde{\hat{K}}_\ell^\top\hat{K}_\ell - \log\int dP_0(a)\int d\mathcal{N}(\hat{z}^L \,|\, 0,\sigma_L^2\hat{K}_{L-1})\, e^{a^\top t^\top z^L} \\ -\sum_{\ell=1}^{L-1}\log\int d\mathcal{N}(\hat{z}^\ell \,|\, 0,\sigma_\ell^2\hat{K}_{\ell-1})\, e^{\frac{1}{2}\hat{z}^{\ell\top}\tilde{\hat{K}}_\ell\hat{z}^\ell}. \tag{A35}$$

In the last layer the test preactivations can be marginalized, $\int d\mathcal{N}(\hat{z}^L \,|\, 0,\sigma_L^2\hat{K}_{L-1})\, e^{a^\top t^\top z^L} = \int d\mathcal{N}(z^L \,|\, 0,\sigma_L^2 K_{L-1})\, e^{a^\top t^\top z^L}$. Thus, the $P\times P_*$ training-test block $k_{L-1}$ and $P_*\times P_*$ test-test block $\kappa_{L-1}$ of $\hat{K}_{L-1}$ appear in the energy $E(t,\{\hat{K}_\ell,\tilde{\hat{K}}_\ell\})$ only in the term $\mathrm{tr}\,\tilde{\hat{K}}_{L-1}^\top\hat{K}_{L-1}$. Accordingly, the saddle-point equation for their conjugate kernels is $\tilde{k}_{L-1}=0$ and $\tilde{\kappa}_{L-1}=0$. This implies that the test preactivations can be marginalized in the next layer, $\int d\mathcal{N}(\hat{z}^{L-1} \,|\, 0,\sigma_{L-1}^2\hat{K}_{L-2})\, e^{\frac{1}{2}z^{L-1\top}\tilde{K}_{L-1}z^{L-1}} = \int d\mathcal{N}(z^{L-1} \,|\, 0,\sigma_{L-1}^2 K_{L-2})\, e^{\frac{1}{2}z^{L-1\top}\tilde{K}_{L-1}z^{L-1}}$ such that $k_{L-2}$ and $\kappa_{L-2}$ only appear in $\mathrm{tr}\,\tilde{\hat{K}}_{L-2}^\top\hat{K}_{L-2}$ and thus the saddle-point equation for their conjugate kernels is $\tilde{k}_{L-2}=\tilde{\kappa}_{L-2}=0$. Iterating the argument leads to the saddle-point equations $\tilde{k}_\ell=\tilde{\kappa}_\ell=0$ for $\ell=1,\dots,L-1$. For $\tilde{K}_\ell$, $\ell=1,\dots,L-1$, the saddle-point equations remain identical to the training posterior.

The saddle-point equations for the kernels are $\hat{K}_\ell = \langle \hat{z}^\ell\hat{z}^{\ell\top}\rangle_{\hat{z}^\ell}$, $\ell=1,\dots,L-1$. At $\tilde{k}_\ell=\tilde{\kappa}_\ell=0$, the expectation is taken w.r.t. the distribution $P(\hat{z}^\ell)\propto\mathcal{N}(\hat{z}^\ell \,|\, 0,\sigma_\ell^2\hat{K}_{\ell-1})\, e^{\frac{1}{2}z^{\ell\top}\tilde{K}_\ell z^\ell}$. Since the second factor does not depend on the test preactivations $z^{\ell,*}$, the conditional distribution of the test preactivations given the training preactivations is identical to the prior distribution:

$$P(z^{\ell,*} \,|\, z^\ell) = P_0(z^{\ell,*} \,|\, z^\ell) = \mathcal{N}(z^{\ell,*} \,|\, k_{\ell-1}^\top K_{\ell-1}^{-1}z^\ell, \sigma_\ell^2\kappa_{\ell-1} - \sigma_\ell^2 k_{\ell-1}^\top K_{\ell-1}^{-1}k_{\ell-1}). \tag{A36}$$

The joint distribution of training and test preactivations is $P(\hat{z}^\ell) = P_0(z^{\ell,*} \,|\, z^\ell)P(z^\ell)$ where $P(z^\ell)$ is the single neuron posterior of the training preactivations. Thus, the saddle-point equations for the training kernels $K_\ell$ remain unchanged. For the train-test and the test-test kernels, the saddle-point equations are

$$k_\ell = \langle z^\ell z^{\ell\top}\rangle_{z^\ell} K_{\ell-1}^{-1}k_{\ell-1} \tag{A37}$$

$$\kappa_\ell = k_{\ell-1}^\top K_{\ell-1}^{-1}\langle z^\ell z^{\ell\top}\rangle_{z^\ell} K_{\ell-1}^{-1}k_{\ell-1} + \sigma_\ell^2\kappa_{\ell-1} - \sigma_\ell^2 k_{\ell-1}^\top K_{\ell-1}^{-1}k_{\ell-1} \tag{A38}$$

with $K_\ell = \langle z^\ell z^{\ell\top}\rangle_{z^\ell}$. In the last layer, the conditional distribution of the test preactivations given the training preactivations is also identical to the prior and thus $P(a,\hat{z}^L) = P_0(z^{L,*} \,|\, z^L)P(a,z^L)$. In total, we arrive at

$$P(A,\hat{Z}_L,\dots,\hat{Z}_1) = \prod_{i=1}^{N}[P_0(z_i^{L,*} \,|\, z_i^L)P(z_i^L \,|\, a_i)P(a_i)]\prod_{\ell=1}^{L-1}[\prod_{j=1}^{N}[P_0(z_j^{\ell,*} \,|\, z_j^\ell)P(z_j^\ell)]] \tag{A39}$$

for the test posterior.

For the mean predictor we consider a single test input $x$ and the corresponding preactivations, leading with (A36), $P(z^L \mid a) = \mathcal{N}(z^L \mid YU^{-1}a, \sigma_L^2 K_{L-1})$, and (A10) to $f(x) = \langle a \langle \langle z^{L,*} \rangle_{z^{L,*}|z^L} \rangle_{z^L|a} \rangle_a = Y^\top K_{L-1}^{-1} k_{L-1}$. The saddle-point equation (A37) implies $K_\ell^{-1} k_\ell = K_{\ell-1}^{-1} k_{\ell-1}$ and thus by iteration

$$f_r(x) = k_0(x)^\top K_0^{-1} y_r. \tag{A40}$$

The test kernel is determined by (A38) for $\ell = 1, \dots, L-1$ which evaluates with $K_\ell^{-1} k_\ell = K_{\ell-1}^{-1} k_{\ell-1}$ and $K_\ell = \sigma_\ell^2 K_{\ell-1} + \frac{\sigma_a^{2(L-\ell)}}{\prod_{\ell'=\ell+1}^L \sigma_{\ell'}^2} YU^{-(L-\ell+1)}Y^\top$ to $\kappa_\ell = \sigma_\ell^2 \kappa_{\ell-1} + \frac{\sigma_a^{2(L-\ell)}}{\prod_{\ell'=\ell+1}^L \sigma_{\ell'}^2} k_0^\top K_0^{-1} YU^{-(L-\ell+1)} Y^\top K_0^{-1} k_0$. Thus $K_\ell$ and $\kappa_\ell$ obey the same recursion except that $Y$ is replaced by $Y^\top K_{L-1}^{-1} k_{L-1}$, yielding

$$\kappa_\ell = \prod_{\ell'=1}^{\ell} [\sigma_{\ell'}^2] \kappa_0 + \frac{\sigma_a^{2(L-\ell)}}{\prod_{\ell'=\ell+1}^L \sigma_{\ell'}^2} k_0^\top K_0^{-1} YU^{-(L-\ell+1)} Y^\top K_0^{-1} k_0, \quad 2 \le \ell \le L-1. \tag{A41}$$

In the last layer, (A36) leads to

$$\kappa_L = \prod_{\ell=1}^{L} [\sigma_\ell^2] \kappa_0 + k_0^\top K_0^{-1} YU^{-1} Y^\top K_0^{-1} k_0. \tag{A42}$$

Evaluated for two test inputs $x_1, x_2$ and for $\sigma_\ell = 1$ leads to the kernel stated in the main text.

**Summary:** Test preactivation posterior on test inputs $x_*$

$$P(z^{\ell,*} \mid z^\ell) = \mathcal{N}(z^{\ell,*} \mid k_0^\top K_0^{-1} z^\ell, \prod_{\ell'=1}^{\ell} [\sigma_{\ell'}^2][\kappa_0 - k_0^\top K_0^{-1} k_0]), \quad \ell = 1, \dots, L, \tag{A43}$$

with

$$\kappa_\ell = \prod_{\ell'=1}^{\ell} [\sigma_{\ell'}^2] \kappa_0 + \frac{\sigma_a^{2(L-\ell)}}{\prod_{\ell'=\ell+1}^L \sigma_{\ell'}^2} k_0^\top K_0^{-1} YU^{-(L-\ell+1)} Y^\top K_0^{-1} k_0, \quad \ell = 1, \dots, L-1, \tag{A44}$$

$$\kappa_L = \prod_{\ell=1}^{L} [\sigma_\ell^2] \kappa_0 + k_0^\top K_0^{-1} YU^{-1} Y^\top K_0^{-1} k_0, \tag{A45}$$

and $K_0 = \frac{1}{N_0} X^\top X$, $k_0 = \frac{1}{N_0} X^\top x_*$, $\kappa_0 = \frac{1}{N_0} x_*^\top x_*$.
Predictor on test input $x$

$$f(x) = Y^\top K_0^{-1} k_0(x) \tag{A46}$$

with $K_0 = \frac{1}{N_0} X^\top X$ and $k_0(x) = \frac{1}{N_0} X^\top x$.

### 3. Consistency Check

The above derivation relies on a high dimensional saddle point approximation [2, 3]. This is an ad-hoc approximation without further assumptions about the structure of the integral [4, 5]. For the linear case the high dimensional saddle point approximation can be circumvented using the methods from [6]. Here, we show that this derivation leads to consistent results with the high dimensional saddle point approximation for non-lazy networks.

#### *a. Partition Function*

We start with the self consistency equation for the readout covariance $U$. To this end we consider the partition function

$$Z = \int dP_0(\Theta) \mathcal{N}\big(Y \mid f(X;\Theta), \frac{T}{N} I_m \otimes I_P\big) \tag{A47}$$

where we already rescaled $T \to T/N$, introduced the $m \times P$ matrix of training targets $Y$ and the $N_0 \times P$ matrix of training inputs $X$, and neglected irrelevant multiplicative factors (which will be done throughout) to rewrite the contribution from the loss as a matrix valued normal distribution (see, e.g., [1]) to ease the notation. Marginalizing the readout weights and changing variables from the $N \times N$ (or $N \times N_0$) weights $W_\ell$ to the $N \times P$ preactivations $Z_\ell = \frac{1}{\sqrt{N}} W_\ell Z_{\ell-1}$ and $Z_1 = \frac{1}{\sqrt{N_0}} W_1 X$ leads to

$$Z = \int \prod_{\ell=1}^{L} dP_0(Z_\ell) \mathcal{N}\big(Y \,|\, 0, I_m \otimes \big[\frac{T}{N} I_P + \frac{\sigma_a^2}{N} K_L\big]\big) \tag{A48}$$

with $P_0(Z_\ell) = \mathcal{N}(Z_\ell \,|\, 0, I_N \otimes \sigma_\ell^2 K_{\ell-1})$ and $K_\ell = \frac{1}{N} Z_\ell^\top Z_\ell$.
Next we marginalize the $Z_\ell$ layer by layer, starting from the last layer. Using (A2) makes the $P_0(Z_L)$ expectation tractable,

$$\big\langle e^{-\frac{\sigma_a^2}{2N^2} \operatorname{tr} t^\top Z_L^\top Z_L t} \big\rangle = e^{-\frac{N}{2} \log\det(I_m + \frac{\sigma_a^2 \sigma_L^2}{N^2} t^\top K_{L-1} t)}. \tag{A49}$$

Introducing the $m \times m$ matrix $H_L = \frac{\sigma_a^2 \sigma_L^2}{N^2} t^\top K_{L-1} t$ with conjugate variable $U_L$ leads to

$$Z = \int dU_L \, e^{\frac{N}{2} \log\det U_L - \frac{N}{2} \operatorname{tr} U_L} \int \prod_{\ell=1}^{L-1} dP_0(Z_\ell) \mathcal{N}\big(Y \,|\, 0, \frac{T}{N} I_m \otimes I_P + \frac{\sigma_a^2}{N} \sigma_L^2 U_L \otimes K_{L-1}\big) \tag{A50}$$

where $H_L = U_L^{-1} - I_m$ from a saddle-point approximation at large $N$ was used.
Repeating these steps layer by layer leads to

$$Z = \int \prod_{\ell=1}^{L} dU_\ell \, e^{-\frac{N}{2} \sum_{\ell=1}^{L} [\operatorname{tr} U_\ell - \log\det U_\ell]} \mathcal{N}\big(Y \,|\, 0, \frac{T}{N} I_m \otimes I_P + \frac{\sigma_a^2}{N} \prod_{\ell=1}^{L} \sigma_\ell^2 U_\ell \otimes K_0\big) \tag{A51}$$

with $K_0 = \frac{1}{N_0} X^\top X$. Note that without the rescaling of temperature the contribution $\frac{\sigma_a^2}{N} \prod_{\ell=1}^{L} \sigma_\ell^2 U_\ell \otimes K_0$ due to learning would be a $O(1/N)$ correction. The remaining $U_\ell$ integrals can be solved in a low dimensional saddle-point approximation; at zero temperature the corresponding saddle-point equations are

$$I_m = U_\ell + \alpha I_m - \frac{1}{\sigma_a^2 \prod_{\ell=1}^{L} \sigma_\ell^2} \big( \prod_{\ell'=1}^{\ell} U_{\ell'} \big)^{-1} Y K_0^{-1} Y^\top \big( \prod_{\ell'=\ell+1}^{L} U_{\ell'} \big)^{-1}. \tag{A52}$$

Since the equations are identical for every $\ell$, we can set $\sigma_a^2 U_\ell = U$, leading to

$$U = \sigma_a^2 I_m - \alpha \sigma_a^2 I_m + \frac{\sigma_a^{2L}}{\prod_{\ell=1}^{L} \sigma_\ell^2} U^{-L} Y K_0^{-1} Y^\top. \tag{A53}$$

In (A22) the second term proportional to $\alpha$ is missing, indicating that the high dimensional saddle point approximation is in general not valid. However, the scaling analysis of $U$ still applies, requiring $\sigma_a^2 = O(P^{-1/L})$ to ensure $U = O(1)$. In this case the first and second term can be neglected for large $P$ and we arrive at $U^{L+1} = \frac{\sigma_a^{2L}}{\prod_{\ell=1}^{L} \sigma_\ell^2} Y K_0^{-1} Y^\top$, consistent with the derivation based on the high dimensional saddle point approximation.

### *b. Readout Weight Posterior*

Next, we show that the readout weight posterior is consistent. We consider the neurons averaged readout posterior

$$P(a) = \frac{1}{N} \sum_{i=1}^{N} \langle \delta(a - a_i) \rangle_\Theta \tag{A54}$$

where $a_i$ denotes the $m$ readout weights of an arbitrary neuron $i$. Note that due to the permutation symmetry of the weight posterior, $P(a)$ is identical to the single neuron posterior $\langle \delta(a - a_i) \rangle_\Theta$. We compute $P(a)$ from the generating functional

$$Z[j] = \int dP_0(\Theta) \, \exp\big( - N\beta \mathcal{L}(\Theta) + \sum_{i=1}^{N} j(a_i) \big) \tag{A55}$$

via $P(a) = \frac{1}{N}\frac{\delta}{\delta j(a)} \log Z[j]\big|_{j=0}$. Changing variables from $W_\ell$ to $Z_\ell$ and using the matrix valued normal for convenience leads to

$$Z[j] = \int dP_0(A) \prod_{\ell=1}^{L} dP_0(Z_\ell)\, \mathcal{N}\big(Y \,|\, \frac{1}{N} A^\top Z_L, \frac{T}{N} I_m \otimes I_P\big) e^{\sum_{i=1}^N j(a_i)} \tag{A56}$$

where $A$ denotes the $N \times m$ matrix of all readout weights.
Again, the expectations $Z_\ell$ are calculated layer by layer, starting with $Z_L$. Using (A2) the $Z_L$ expectation is tractable,

$$\langle e^{\frac{i}{N} \operatorname{tr} t A^\top Z_L} \rangle = e^{-\frac{\sigma_L^2}{2N^2} \operatorname{tr} A t^\top K_{L-1} t A^\top}, \tag{A57}$$

which leads to

$$Z[j] = \int dP_0(A) \prod_{\ell=1}^{L-1} dP_0(Z_\ell)\, \mathcal{N}\big(Y \,|\, 0, \frac{T}{N} I_m \otimes I_P + \frac{\sigma_L^2}{N^2} A^\top A \otimes K_{L-1}\big) e^{\sum_{i=1}^N j(a_i)}. \tag{A58}$$

Introducing the $m \times m$ order parameter $\sigma_a^2 U_L = \frac{1}{N} A^\top A$ with conjugate variable $H_L$ and the single neuron partition function

$$\hat{Z}[j; H_L] = \int dP_0(a)\, e^{-\frac{1}{2\sigma_a^2} a^\top H_L a + j(a)} \tag{A59}$$

leads to

$$Z[j] = \int dU_L dH_L\, e^{\frac{N}{2} \operatorname{tr} U_L H_L + N \log \hat{Z}[j; H_L]} \int \prod_{\ell=1}^{L-1} dP_0(Z_\ell)\, \mathcal{N}\big(Y \,|\, 0, \frac{T}{N} I_m \otimes I_P + \frac{\sigma_a^2}{N} \sigma_L^2 U_L \otimes K_{L-1}\big). \tag{A60}$$

The structure for the remaining $Z_\ell$ expectations is identical to the partition function.
Integrating out the remaining $Z_\ell$ layer by layer yields

$$Z[j] = \int dH_L \prod_{\ell=1}^{L} dU_\ell\, e^{\frac{N}{2} \operatorname{tr} U_L H_L - \frac{N}{2} \sum_{\ell=1}^{L-1} [\operatorname{tr} U_\ell - \log\det U_\ell] + N \log \hat{Z}[j; H_L]}\, \mathcal{N}\big(Y \,|\, 0, \frac{T}{N} I_m \otimes I_P + \frac{\sigma_a^2}{N} \prod_{\ell=1}^{L} \sigma_\ell^2 U_\ell \otimes K_0\big). \tag{A61}$$

Taking the functional derivative and evaluating at $j = 0$ leads to

$$P(a) = \mathcal{N}(a \,|\, 0, \sigma_a^2 U_L) \tag{A62}$$

where $\log \hat{Z}[0; H_L] = -\frac{1}{2} \log\det(I_m + H_L)$ and the corresponding saddle point equation $H_L = (U_L^{-1} - I_m)$ were used. The saddle point equations for $U_\ell$ are unchanged and we arrive at $P(a) = \mathcal{N}(a \,|\, 0, U)$ with $U = \sigma_a^2 U_L$ determined by the self consistency equation for $U$.

### c. *Predictor*

Finally, we compute the predictor statistics. To this end we consider the generating function

$$Z(j) = \int dP_0(\Theta)\, \exp\big( - N\beta \mathcal{L}(\Theta) + i j^\top f(x; \Theta)\big) \tag{A63}$$

where $x$ denotes an arbitrary test input and $j$ is an $m$-dimensional vector. Changing variables from $W_\ell$ to the $N \times (P+1)$ matrix $\hat{Z}_\ell$ of preactivations on the training inputs and the test input $x$, using (A2), and marginalizing the readout weights leads to

$$Z(j) = \int dt \int \prod_{\ell=1}^{L} dP_0(\hat{Z}_\ell)\, \exp\Big( - i \operatorname{tr} t Y - \frac{T}{2N} \operatorname{tr} t^\top t - \frac{\sigma_a^2}{2N} \operatorname{tr} \hat{t}^\top \hat{Z}_L \hat{t}\Big) \tag{A64}$$

where $\hat{t}$ is the $(P+1)\times m$ matrix obtained by by concatenating $t$ and $j$, $\hat{K}_\ell = \frac{1}{N}\hat{Z}_\ell^\top \hat{Z}_\ell$ is the $(P+1)\times(P+1)$ dimensional kernel on the training and test input, and $P_0(\hat{Z}_\ell) = \mathcal{N}(\hat{Z}_\ell \,|\, 0, I_N \otimes \sigma_\ell^2 \hat{K}_{\ell-1})$. Marginalizing the $\hat{Z}_\ell$ can be done in analogy to the partition function, leading to

$$Z(j) = \int \prod_{\ell=1}^{L} dU_\ell \, e^{-\frac{N}{2}\sum_{\ell=1}^{L}[\mathrm{tr}\, U_\ell - \log\det U_\ell]} \int dt \, \exp\Big( - i\mathrm{tr}\, tY - \frac{T}{2N}\mathrm{tr}\, t^\top t - \frac{\sigma_a^2}{2N}\mathrm{tr} \prod_{\ell=1}^{L}(\sigma_\ell^2 U_\ell)\hat{t}^\top \hat{K}_0 \hat{t}\Big). \tag{A65}$$

We separate $j$ again,

$$\hat{t}^\top \hat{K}_0 \hat{t} = t^\top K_0 t + 2t^\top k_0(x) j^\top + \kappa_0(x) j j^\top \tag{A66}$$

leading to (at $T=0$ for simplicity)

$$Z(j) = e^{-\frac{NL}{2}[\mathrm{tr}\, U - \ln\det U] - \frac{\sigma_a^2 \prod_{\ell=1}^{L}(\sigma_\ell^2)}{2N}\kappa_0(x) j^\top U^L j} \mathcal{N}\big(Y \,|\, \frac{\sigma_a^2 \prod_{\ell=1}^{L}(\sigma_\ell^2)}{N} k_0(x) i j^\top U^L, \frac{\sigma_a^2 \prod_{\ell=1}^{L}(\sigma_\ell^2)}{N} U^L \otimes K_0\big) \tag{A67}$$

after solving the $t$ integral and performing the saddle-point approximation of the $U_\ell$ integrals.

The statistics of $f(x,\Theta)$ follow from the derivatives of $\log Z(j)$ evaluated at $j=0$. Thus, we only need the solution of the saddle-point equations at $j=0$. Computing the derivatives at $j=0$ leads to

$$\langle f(x)\rangle = k_0(x)^\top K_0^{-1} Y, \tag{A68}$$

$$\langle \delta f(x) \delta f(x)^\top \rangle = \frac{\sigma_a^2 \prod_{\ell=1}^{L}(\sigma_\ell^2)}{N} U^L \big(\kappa_0(x) - k_0(x)^\top K_0^{-1} k_0(x)\big). \tag{A69}$$

All higher cumulants vanish because $\log Z(j)$ is quadratic in $j$.

**Scaling Analysis:** If $\sigma_a = O(1)$ the variance is $O(P^{L/(L+1)}/N)$; if $\sigma_a$ is scaled to keep the readout norm $O(1)$ the variance is $O(P^{(L-1)/L}/N)$. In all cases, the variance is suppressed compared to the lazy case and the effect diminishes with increasing number of layers. The strongest reduction occurs for single hidden layer networks with scaled $\sigma_a$ where the predictor variance is $O(1/N)$. Thus, a benefit of feature learning is that it suppresses the variability of the predictor. Accordingly, the per sample generalization error $\epsilon_g(x) = [y_r - f_r(x)]^2 + \langle \delta f_r(x;\Theta)^2\rangle_\Theta$ is dominated by the $O(1)$ bias contribution $[y_r - f_r(x)]^2$, particularly in shallow networks with scaled $\sigma_a$.

### *d. Training Kernel*

We first consider the mean training kernel in the last layer $K_L = \frac{1}{N}\langle Z_L^\top Z_L\rangle_\Theta$. We marginalize $A$ following the steps used for the partition function, leading to

$$K_L = \frac{1}{N}\int \prod_{\ell=1}^{L}[dP_0(Z_\ell)]\, \mathcal{N}\big(Y \,|\, 0, I_m \otimes \big[\frac{T}{N} I_P + \frac{\sigma_a^2}{N^2} Z_L^\top Z_L\big]\big) Z_L^\top Z_L \tag{A70}$$

The presence of $\frac{1}{N} Z_L^\top Z_L$ modifies the relevant part for the marginalization of $Z_L$ to

$$\int dP_0(Z_L)\, \exp\Big( - \frac{\sigma_a^2}{2N^2}\mathrm{tr}\, t^\top Z_L^\top Z_L t\Big) \frac{1}{N} Z_L^\top Z_L = e^{-\frac{N}{2}\log\det(I_m + \frac{\sigma_a^2\sigma_L^2}{N^2} t^\top K_{L-1} t)} A(t), \tag{A71}$$

$$A(t) = \frac{\int dP_0(z_L)\, z_L z_L^\top \exp\Big( - \frac{\sigma_a^2}{2N^2}\|t^\top z_L\|^2\Big)}{\int dP_0(z_L)\, \exp\Big( - \frac{\sigma_a^2}{2N^2}\|t^\top z_L\|^2\Big)}. \tag{A72}$$

$A(t)$ is the second moment of a zero-mean Gaussian with covariance $\big((\sigma_L^2 K_{L-1})^{-1} + \frac{\sigma_a^2}{N^2} t t^\top\big)^{-1} = \sigma_L^2 K_{L-1} - \sigma_L^2 K_{L-1} \frac{\sigma_a^2}{N^2} t\big(I + H_L\big)^{-1} t^\top \sigma_L^2 K_{L-1}$, thus

$$A(t) = \sigma_L^2 \Big( K_{L-1} - \frac{\sigma_a^2 \sigma_L^2}{N^2} K_{L-1} t U_L t^\top K_{L-1}\Big) \tag{A73}$$

where we used $H_L = U_L^{-1} - I_m$. Since

$$\begin{aligned}\frac{\sigma_a^2\sigma_L^2}{N^2}\int dt\, tU_L t^\top e^{-i\mathrm{tr}\, t^\top Y - \frac{\sigma_a^2\sigma_L^2}{2N}\mathrm{tr}\, U_L t^\top K_{L-1} t} \\ &= -\frac{2}{N}\frac{\partial}{\partial K_{L-1}}\mathcal{N}\big(Y\,|\,0, \sigma_a^2\sigma_L^2 N^{-1} U_L \otimes K_{L-1}\big) \\ &= -\frac{1}{\sigma_a^2\sigma_L^2}(K_{L-1})^{-1} Y U_L^{-1} Y^\top (K_{L-1})^{-1}\mathcal{N}\big(Y\,|\,0, \sigma_a^2\sigma_L^2 N^{-1} U_L \otimes K_{L-1}\big)\end{aligned}$$

where we neglect a $O(m/N)$ term in the last step, we obtain

$$\langle K_L\rangle = \sigma_L^2\langle K_{L-1}\rangle + \frac{1}{\sigma_a^2} Y U_L^{-1} Y^\top. \tag{A74}$$

Using the same steps for the next layer leads to

$$\langle K_L\rangle = \sigma_L^2\sigma_{L-1}^2\langle K_{L-2}\rangle + \frac{1}{\sigma_a^2} Y (U_{L-1}U_L)^{-1} Y^\top + \frac{1}{\sigma_a^2} Y U_L^{-1} Y^\top. \tag{A75}$$

Iterating this to the first layer and using $\sigma_a^2 U_\ell = U$ at the saddle-point we obtain

$$\langle K_L\rangle = \prod_{\ell=1}^{L}(\sigma_\ell^2) K_0 + Y\Big(\sum_{\ell=1}^{L}\sigma_a^{2(\ell-1)} U^{-\ell}\Big) Y^\top. \tag{A76}$$

Due to the scaling $\sigma_a^2 = O(1/P^{1/L})$ only the first term in the sum is relevant, $\sum_{\ell=1}^{L}\sigma_a^{2(\ell-1)}U^{-\ell} = U^{-1} + O(1/P^{1/L})$. Alternatively, the sum can be performed using the geometric series.
Using the same steps we obtain for $\ell < L$

$$\langle K_\ell\rangle = \prod_{\ell'=1}^{\ell}(\sigma_{\ell'}^2) K_0 + \frac{1}{\prod_{\ell'=\ell+1}^{L}(\sigma_{\ell'}^2)} Y\Big(\sum_{\ell'=L-\ell+1}^{L}\sigma_a^{2(\ell'-1)} U^{-\ell'}\Big) Y^\top \tag{A77}$$

where $(M_\ell)_{ss'} = \delta_{ss'}\sum_{\ell'=L-\ell+1}^{L} u_s^{-\ell'} = \delta_{ss'}\frac{1}{u_s^L}\frac{1-u_s^\ell}{1-u_s}$ from the geometric series. Again only the first term in the sum is relevant, $\sum_{\ell'=L-\ell+1}^{L}\sigma_a^{2(\ell'-1)}U^{-\ell'} = \sigma_a^{2(L-\ell)}U^{-L+\ell-1}$, yielding once again consistent results with the high dimensional saddle point approximation.

## Appendix B: Sigmoidal Networks

### 1. Training Posterior

As in the linear case, we change variables from the weights $W_\ell$ to the preactivations $Z_\ell$. Repeating the derivation for the linear network leads to

$$P(A, Z_L, \ldots, Z_1) \propto \int dt \prod_{\ell=1}^{L-1}[dK_\ell d\tilde{K}_\ell]\, e^{-NE(t,\{K_\ell,\tilde{K}_\ell\})}\prod_{i=1}^{N}[P(a_i, z_i^L)]\prod_{\ell=1}^{L-1}[\prod_{j=1}^{N}P(z_j^\ell)], \tag{B1}$$

$$\begin{aligned}E(t,\{K_\ell,\tilde{K}_\ell\}) = &-\frac{T}{2}\mathrm{tr}\, t^\top t + \mathrm{tr}\, t^\top Y + \frac{1}{2}\sum_{\ell=1}^{L-1}\mathrm{tr}\,\tilde{K}_\ell^\top K_\ell - \log\int dP_0(a) d\mathcal{N}(z^L\,|\,0,\sigma_L^2 K_{L-1}) e^{a^\top t^\top \phi(z^L)} \\ &- \sum_{\ell=1}^{L-1}\log\int d\mathcal{N}(z^\ell\,|\,0,\sigma_\ell^2 K_{\ell-1}) e^{\frac{1}{2}\phi(z^\ell)^\top \tilde{K}_\ell \phi(z^\ell)},\end{aligned} \tag{B2}$$

where the single neuron posteriors are not Gaussian,

$$P(a, z^L) \propto P_0(a)\mathcal{N}(z^L \,|\, 0, \sigma_L^2 K_{L-1}) e^{a^\top t^\top \phi(z^L)}, \tag{B3}$$

$$P(z^\ell) \propto \mathcal{N}(z^\ell \,|\, 0, \sigma_\ell^2 K_{\ell-1}) e^{\frac{1}{2}\phi(z^\ell)^\top \tilde{K}_\ell \phi(z^\ell)}. \tag{B4}$$

The saddle point approximation of the $t$ and $K_\ell$, $\tilde{K}_\ell$ integrals leads to

$$P(A, Z_L, \dots, Z_1) = \prod_{i=1}^{N} [P(a_i) P(z_i^L \,|\, a_i)] \prod_{\ell=1}^{L-1} [\prod_{j=1}^{N} P(z_j^\ell)], \tag{B5}$$

the corresponding saddle point equations are

$$y_\mu^r = \langle a_r \phi(z_\mu^L) \rangle_{a, z^L} + T t_\mu^r, \tag{B6}$$

$$K_\ell = \langle \phi(z^\ell) \phi(z^\ell)^\top \rangle_{z^\ell}, \quad \ell = 1, \dots, L-1, \tag{B7}$$

$$\tilde{K}_{\ell-1} = \frac{1}{\sigma_\ell^2} K_{\ell-1}^{-1} \langle z^\ell z^{\ell\top} \rangle_{z^\ell} K_{\ell-1}^{-1} - K_{\ell-1}^{-1}, \quad \ell = 2, \dots, L. \tag{B8}$$

Note that $K_\ell$ depends on the activation kernel and $\tilde{K}_{\ell-1}$ on the preactivation kernel; in the linear case this distinction was not relevant.

We now set $\sigma_a^2 = 1/P$. The readout posterior is

$$\log P(a) = -P\Big(\frac{1}{2} a^\top a - \frac{1}{P} \log \int d\mathcal{N}(z^L \,|\, 0, \sigma_L^2 K_{L-1}) e^{a^\top t^\top \phi(z^L)}\Big) + \text{const.} \equiv -P E(a \,|\, K_{L-1}, t) + \text{const.} \tag{B9}$$

At large $P$ the posterior concentrates at the minima of $E(a \,|\, K_{L-1}, t)$,

$$P(a) = \sum_{\gamma=1}^{n} P_\gamma \delta(a - a_\gamma), \qquad P_\gamma = \frac{e^{-P E(a_\gamma \,|\, K_{L-1}, t)}}{\sum_{\gamma'=1}^{n} e^{-P E(a_{\gamma'} \,|\, K_{L-1}, t)}}, \tag{B10}$$

where the $a_\gamma$ are determined by

$$a_\gamma = \frac{1}{P} t^\top \langle \phi(z^L) \rangle_{z^L | a_\gamma}, \qquad P(z \,|\, a) = \frac{P_0(z) e^{(at)^\top \phi(z)}}{\int dP_0(z)\, e^{(at)^\top \phi(z)}}, \tag{B11}$$

with $P_0(z) = \mathcal{N}(z \,|\, 0, \sigma_L^2 K_{L-1})$.

An interesting aspect is that the weights are given by $P_\gamma \propto \exp[-P E(a_\gamma \,|\, K_{L-1}, t)]$ where the energy $E(a \,|\, K_{L-1}, t)$ is $O(1)$. Thus, the energy needs to be quasi-degenerate between the branches, $E(a_\gamma \,|\, K_{L-1}, t) - E(a_{\gamma'} \,|\, K_{L-1}, t) = O(1/P)$, to achieve finite weights $P_\gamma$. Since this quasi-degeneracy can occur even though the solutions are not connected by permutation symmetry, the order parameter $t$ as well as the readouts weights $a_\gamma$ must be precisely fine tuned.

In contrast to the readout posterior, the conditional distribution

$$P(z^L \,|\, a) \propto \mathcal{N}(z^L \,|\, 0, \sigma_L^2 K_{L-1}) e^{a^\top t^\top \phi(z^L)} \tag{B12}$$

does not concentrate; all quantities appearing in $P(z^L \,|\, a)$ are $O(1)$. Nonetheless, as evidenced by the empirical sampling, $P(z^L \,|\, a)$ can posses a pronounced mean and comparably modest fluctuations. The simplification of $P(a)$ due to the concentration makes it advantageous to factorize the joint posterior into the conditional distributions $P(a, z^L) = P(z^L \,|\, a) P(a)$, which simplifies the preactivation posterior to $P(z^L) = \sum_{\gamma=1}^{n} P_\gamma P(z^L \,|\, a_\gamma)$, the saddle point equation for $t$ to $y_\mu^r = \sum_{\gamma=1}^{n} P_\gamma a_\gamma^r \langle \phi(z_\mu^L) \rangle_{z^L | a_\gamma} + T t_\mu^r$, and the last layer kernel to $K_L = \sum_{\gamma=1}^{n} P_\gamma \langle \phi(z^L) \phi(z^L)^\top \rangle_{z^L | a_\gamma}$.

**Summary:** Posterior distributions

$$P(a) = \sum_{\gamma=1}^{n} P_\gamma \delta(a - a_\gamma) \tag{B13}$$

$$P(z^L \,|\, a) \propto \mathcal{N}(z^L \,|\, 0, \sigma_L^2 K_{L-1}) e^{a^\top t^\top \phi(z^L)} \tag{B14}$$

$$P(z^\ell) \propto \mathcal{N}(z^\ell \,|\, 0, \sigma_\ell^2 K_{\ell-1}) e^{\frac{1}{2}\phi(z^\ell)^\top \tilde{K}_\ell \phi(z^\ell)}, \quad \ell = 1, \dots, L-1, \tag{B15}$$

with

$$a_\gamma = \frac{1}{P} t^\top \langle \phi(z^L) \rangle_{z^L | a_\gamma}, \quad \gamma = 1, \dots, n, \tag{B16}$$

$$y_\mu^r = \sum_{\gamma=1}^{n} P_\gamma a_\gamma^r \langle \phi(z_\mu^L) \rangle_{z^L | a_\gamma} + T t_\mu^r, \quad r = 1, \dots, m, \quad \mu = 1, \dots, P \tag{B17}$$

$$K_\ell = \langle \phi(z^\ell) \phi(z^\ell)^\top \rangle_{z^\ell}, \quad \ell = 1, \dots, L-1, \tag{B18}$$

$$\tilde{K}_{\ell-1} = \frac{1}{\sigma_\ell^2} K_{\ell-1}^{-1} \langle z^\ell z^{\ell\top} \rangle_{z^\ell} K_{\ell-1}^{-1} - K_{\ell-1}^{-1}, \quad \ell = 2, \dots, L, \tag{B19}$$

$K_0 = \frac{1}{N_0} X^\top X$, and $\sigma_a^2 = 1/P$.

**Finite $P$ correction:** For finite $P$, we include quadratic fluctuations around the minima of $E(a \,|\, K_{L-1}, t) \equiv E(a)$, leading to

$$P(a) = \sum_{\gamma=1}^{n} P_\gamma \mathcal{N}(a \,|\, a_\gamma, \frac{1}{P} E''(a_\gamma)^{-1}) \tag{B20}$$

where $E''(a) = I_m - \frac{1}{P} t^\top \langle (\phi(z^L) - \langle \phi(z^L) \rangle_{z|a})(\phi(z^L) - \langle \phi(z^L) \rangle_{z|a})^\top \rangle_{z|a} t$. Thus, the sum of Dirac deltas is replaced by a Gaussian mixture.

**Orthogonal inputs, $L = 1$:** For single hidden layer $L = 1$ and orthogonal data $K_0 = I$, the energy simplifies to $E(a \,|\, t) = \frac{1}{2} a^\top a - \frac{1}{P} \sum_{\mu=1}^{P} \log \langle e^{\phi(\sigma_1 \xi) \sum_{r=1}^{m} a_r t_\mu^r} \rangle$ with $\xi \sim \mathcal{N}(0, 1)$. Similarly, the conditional distribution of the preactivations factorizes into $P(z \,|\, a) \propto \prod_{\mu=1}^{P} [\mathcal{N}(z_\mu \,|\, 0, \sigma_1^2) e^{\phi(z_\mu) \sum_{r=1}^{m} a_r t_\mu^r}]$.

**General inputs, $L = 1$:** For general $K_0$, we use an ad-hoc saddle point approximation $P(z \,|\, a_\gamma) \approx \delta(z - z_\gamma)$ justified by the strong mean apparent in the preactivations. This leads to $E(a \,|\, t) = \frac{1}{2} a^\top a + \frac{1}{P} E(z \,|\, a, t)$ with $E(z \,|\, a, t) = \frac{1}{2\sigma_1^2} z^\top K_0^+ z - (at)^\top \phi(z)$ and $z$ determined by the minimum of $E(z \,|\, a, t)$, $K_0^+ z_\gamma = \sigma_1^2 \phi'(z_\gamma) t a_\gamma$. It also simplifies the saddle point equations to $y_r = \sum_{\gamma=1}^{n} P_\gamma a_\gamma^r \phi(z_\gamma) + T t$ and $a_\gamma = \frac{1}{P} t^\top \phi(z_\gamma)$.

**Multimodality of lower layer preactivation posterior:** Since $\tilde{K}_\ell$ is symmetric we can use the eigenvalue decomposition $\tilde{K}_\ell = \frac{1}{P} V_\ell \Lambda_\ell V_\ell^\top$ (here we normalize the eigenvectors such that $\frac{1}{P} V_\ell^\top V_\ell = I$) to decouple $e^{\frac{1}{2}\phi(z^\ell)^\top \tilde{K}_\ell \phi(z^\ell)} = \int d\mathcal{N}(a^\ell \,|\, 0, \frac{1}{P} I)\, e^{a^{\ell\top} \sqrt{\Lambda_\ell} V_\ell^\top \phi(z^\ell)}$. Thus, we have $P(z^\ell) = \int dP(a^\ell)\, P(z^\ell \,|\, a^\ell)$ with $P(z^\ell \,|\, a^\ell) \propto \mathcal{N}(z^\ell \,|\, 0, \sigma_\ell^2 K_{\ell-1}) e^{a^{\ell\top} \sqrt{\Lambda_\ell} V_\ell^\top \phi(z^\ell)}$ and $P(a^\ell) \propto \mathcal{N}(a^\ell \,|\, 0, \frac{1}{P} I) \int d\mathcal{N}(z^\ell \,|\, 0, \sigma_\ell^2 K_{\ell-1}) e^{a^{\ell\top} \sqrt{\Lambda_\ell} V_\ell^\top \phi(z^\ell)}$ which recovers the structure of the last layer posterior. The difference is that $a^\ell$ is $\mathrm{rank}(\tilde{K}_\ell)$-dimensional whereas in the last layer $a$ is $m$-dimensional. Empirically $\tilde{K}_\ell$ is low rank and the dominating eigenvalues are positive and $O(1)$ such that the discussion of $P(z^L \,|\, a)$ and $P(a)$ applies identically here: $P(a^\ell)$ concentrates, $P(a^\ell) = \sum_{\gamma=1}^{n_\ell} P_\gamma \delta(a^\ell - a_\gamma^\ell)$, leading to a multimodal posterior $P(z^\ell) = \sum_{\gamma=1}^{n_\ell} P_\gamma P(z^\ell \,|\, a_\gamma^\ell)$. The difference leading to the frozen code in the last layer but not in the previous layers is that in the last layer the joint distribution $P(z^L \,|\, a) P(a)$ is sampled, which freezes due to the disconnected structure of $P(a)$, whereas in the previous layers the marginal $P(z^\ell)$ is sampled which is multimodal but not disconnected.

**Multiple self-consistent solutions:** The energy corresponding to the order parameters $E(t, \{K_\ell, \tilde{K}_\ell\})$ can also posses multiple minima, corresponding to different coding schemes. Taking for simplicity $L = 1$ such that $E(t, \{K_\ell, \tilde{K}_\ell\}) = E(t)$, the relative probability of two solutions $t_1, t_2$ is $e^{-N[E(t_1)-E(t_2)]}$. We did not observe degeneracy of $E(t)$ such that the sampled coding scheme is unique. We show multiple solutions in Fig. S3 on the toy task.

**Weight space dynamics:** Starting from an initial condition sampled from the prior, both Langevin dynamics and gradient descent on the negative log posterior first minimize the error and bring the network close to the solution space (see Fig. S7 and Fig. S8), as in lazy networks [7]. Already during this early stage a coding scheme emerges. Subsequently, the Langevin dynamics stay close to the solution space and are dominated by the regularization imposed by the prior, which further shapes the coding scheme and eventually leads to the coding scheme shown in Fig. 2(B) (see Fig. S7 and Fig. S8). Interestingly, also gradient descent without noise can arrive at the same solution (see Fig. S8).

### 2. Test Posterior

As in the linear case, we change variables from the weights $W_\ell$ to the matrix of training and test preactivations $\hat{Z}_\ell$. Repeating the steps for the linear network leads to

$$P(A, \hat{Z}_L, \dots, \hat{Z}_1) \propto \int dt \prod_{\ell=1}^{L-1} d\hat{K}_\ell d\tilde{\hat{K}}_\ell \, e^{-NE(t,\{\hat{K}_\ell, \tilde{\hat{K}}_\ell\})} \prod_{i=1}^{N} [P(a_i, \hat{z}_i^L)] \prod_{\ell=1}^{L-1} [\prod_{j=1}^{N} P(\hat{z}_j^\ell)], \tag{B21}$$

$$\begin{aligned} E(t, \{\hat{K}_\ell, \tilde{\hat{K}}_\ell\}) = &- \frac{T}{2} \operatorname{tr} t^\top t + \operatorname{tr} t^\top Y + \frac{1}{2} \sum_{\ell=1}^{L-1} \operatorname{tr} \tilde{\hat{K}}_\ell^\top \hat{K}_\ell - \log \int dP_0(a) \int d\mathcal{N}(\hat{z}^L \,|\, 0, \sigma_L^2 \hat{K}_{L-1}) \, e^{a^\top t^\top \phi(z^L)} \\ &- \sum_{\ell=1}^{L-1} \log \int d\mathcal{N}(\hat{z}^\ell \,|\, 0, \sigma_\ell^2 \hat{K}_{\ell-1}) \, e^{\frac{1}{2} \operatorname{tr} \phi(\hat{z}^\ell)^\top \tilde{\hat{K}}_\ell \phi(\hat{z}^\ell)}, \end{aligned} \tag{B22}$$

with the single neuron posteriors

$$P(a, \hat{z}^L) \propto P_0(a) \mathcal{N}(\hat{z}^L \,|\, 0, \sigma_L^2 \hat{K}_{L-1}) e^{a^\top t^\top \phi(z^L)} \tag{B23}$$

$$P(\hat{z}^\ell) \propto \mathcal{N}(\hat{z}^\ell \,|\, 0, \sigma_\ell^2 \hat{K}_{\ell-1}) e^{\frac{1}{2} \operatorname{tr} \phi(\hat{z}^\ell)^\top \tilde{\hat{K}}_\ell \phi(\hat{z}^\ell)} \tag{B24}$$

By the same arguments as in the linear case, the saddle-point equations for the train-test and test-test conjugate kernels are $\tilde{k}_\ell = \tilde{\kappa}_\ell = 0$ for $\ell = 1, \dots, L-1$. For $\tilde{K}_\ell$, $\ell = 1, \dots, L-1$, the saddle-point equations remain identical to the training case.

The saddle-point equations for the kernels are $\hat{K}_\ell = \langle \phi(\hat{z}^\ell) \phi(\hat{z}^\ell)^\top \rangle_{\hat{z}^\ell}$, $\ell = 1, \dots, L-1$. At $\tilde{k}_\ell = \tilde{\kappa}_\ell = 0$, the conditional distribution of the test preactivations given the training preactivations is identical to the prior distribution in all layers:

$$P(z^{\ell,*} \,|\, z^\ell) = P_0(z^{\ell,*} \,|\, z^\ell) = \mathcal{N}(z^{\ell,*} \,|\, k_{\ell-1}^\top K_{\ell-1}^+ z^\ell, \sigma_\ell^2 \kappa_{\ell-1} - \sigma_\ell^2 k_{\ell-1}^\top K_{\ell-1}^+ k_{\ell-1}) \tag{B25}$$

where $K^+$ denotes the pseudo inverse. The saddle-point equations for the training kernels $K_\ell$ remain unchanged. For the train-test and the test-test kernels, the saddle-point equations at $\tilde{k}_\ell = \tilde{\kappa}_\ell = 0$ are

$$k_\ell = \langle \phi(z^\ell) \langle \phi(z^{\ell,*})^\top \rangle_{z^{\ell,*}|z^\ell} \rangle_{z^\ell}, \tag{B26}$$

$$\kappa_\ell = \langle \langle \phi(z^{\ell,*}) \phi(z^{\ell,*})^\top \rangle_{z^{\ell,*}|z^\ell} \rangle_{z^\ell}, \tag{B27}$$

and $k_0 = \frac{1}{N_0} X^\top x_*$, $\kappa_0 = \frac{1}{N_0} x_*^\top x_*$. In the last layer, the conditional distribution of the test preactivations given the training preactivations is also identical to the prior and thus $P(a, \hat{z}^L) = P_0(z^{L,*} \,|\, z^L) P(a, z^L)$. In total, we arrive at

$$P(A, \hat{Z}_L, \dots, \hat{Z}_1) = \prod_{i=1}^{N} [P_0(z_i^{L,*} \,|\, z_i^L) P(z_i^L \,|\, a_i) P(a_i)] \prod_{\ell=1}^{L-1} [\prod_{j=1}^{N} [P_0(z_j^{\ell,*} \,|\, z_j^\ell) P(z_j^\ell)]] \tag{B28}$$

for the test posterior.

The mean predictor follows immediately from the definition of the joint posterior,

$$f_r(x) = \sum_{\gamma=1}^{n} P_\gamma a_\gamma^r \langle\langle \phi(z^{L,*})\rangle_{z^{L,*}|z^L}\rangle_{z^L|a_\gamma} \tag{B29}$$

where $z^{L,*}$ denotes the preactivation corresponding to the input $x$. For $\phi(z) = \frac{1}{2}[1 + \mathrm{erf}(\sqrt{\pi}z/2)]$ the Gaussian expectation $z^{L,*}|z^L$ is solvable [8], leading to

$$\langle \phi(z^{L,*})\rangle_{z^{L,*}|z^L} = \phi\big(c_L(x) k_{L-1}(x)^\top K_{L-1}^+ z^L\big), \qquad c_L(x) = \frac{1}{\sqrt{1 + \frac{\pi\sigma_L^2}{2}[\kappa_{L-1}(x) - k_{L-1}(x)^\top K_{L-1}^+ k_{L-1}(x)]}} \tag{B30}$$

where we made the dependence of $\kappa_{L-1}$ and $k_{L-1}$ on the input $x$ explicit. Thus, the mean predictor is

$$f_r(x) = \sum_{\gamma=1}^{n} P_\gamma a_\gamma^r \langle \phi[c(x) k_{L-1}(x)^\top K_{L-1}^+ z^L]\rangle_{z^L|a_\gamma}. \tag{B31}$$

The mean kernels for a pair of inputs $x_1, x_2$ also follow immediately from the posterior,

$$K_L(x_1, x_2) = \sum_{\gamma=1}^{n} P_\gamma \langle\langle \phi(z_1^L)\phi(z_2^L)\rangle_{z_1^L, z_2^L|z^L}\rangle_{z^L|a_\gamma}, \tag{B32}$$

$$K_\ell(x_1, x_2) = \langle\langle \phi(z_1^\ell)\phi(z_2^\ell)\rangle_{z_1^\ell, z_2^\ell|z^\ell}\rangle_{z^\ell}, \quad \ell = 1, \dots, L-1, \tag{B33}$$

where $z_1^\ell, z_2^\ell$ denote the preactivations corresponding to the inputs $x_1, x_2$. The preactivation kernels $K_\ell^z(x_1, x_2) = \langle \frac{1}{N}\sum_{i=1}^N z_i^\ell(x_1) z_i^\ell(x_2)\rangle_\Theta$ are

$$K_\ell^z(x_1, x_2) = \sigma_\ell^2 \kappa_{\ell-1}(x_1, x_2) + \sigma_\ell^2 k_{\ell-1}(x_1)^\top \tilde{K}_{\ell-1} k_{\ell-1}(x_2), \quad \ell = 1, \dots, L, \tag{B34}$$

with $\kappa_0(x_1, x_2) = \frac{1}{N_0} x_1^\top x_2$.

For $\phi(z) = \frac{1}{2}[1 + \mathrm{erf}(\sqrt{\pi}z/2)]$ the Gaussian expectation $\langle \phi(z_1^\ell)\phi(z_2^\ell)\rangle_{z_1^\ell, z_2^\ell|z^\ell}$ can be solved in terms of Owen's T function [8, 9]; for simplicity we assume $\kappa_{\ell-1}(x_1, x_2) - k_{\ell-1}(x_1)^\top K_{\ell-1}^+ k_{\ell-1}(x_2) \approx 0$ for $x_1 \neq x_2$. This decouples the fluctuations of $z_1^\ell, z_2^\ell|z^\ell$, leading to $\langle \phi(z_1^\ell)\phi(z_2^\ell)\rangle_{z_1^\ell, z_2^\ell|z^\ell} \approx \phi\big(c_\ell(x_1) k_{\ell-1}(x_1)^\top K_{\ell-1}^+ z^\ell\big)\phi\big(c_\ell(x_2) k_{\ell-1}(x_2)^\top K_{\ell-1}^+ z^\ell\big)$ for $x_1 \neq x_2$ with $c_\ell(x) = \frac{1}{\sqrt{1 + \frac{\pi\sigma_L^2}{2}[\kappa_{\ell-1}(x) - k_{\ell-1}(x)^\top K_{\ell-1}^+ k_{\ell-1}(x)]}}$.

**Summary:** Test preactivation posterior on test inputs $x_*$

$$P(z^{\ell,*} \,|\, z^\ell) = \mathcal{N}(z^{\ell,*} \,|\, k_{\ell-1}^\top K_{\ell-1}^+ z^\ell, \sigma_\ell^2 \kappa_{\ell-1} - \sigma_\ell^2 k_{\ell-1}^\top K_{\ell-1}^+ k_{\ell-1}), \quad \ell = 1, \dots, L, \tag{B35}$$

with

$$k_\ell = \langle \phi(z^\ell)\langle \phi(z^{\ell,*})^\top\rangle_{z^{\ell,*}|z^\ell}\rangle_{z^\ell}, \quad \ell = 1, \dots, L, \tag{B36}$$

$$\kappa_\ell = \langle\langle \phi(z^{\ell,*})\phi(z^{\ell,*})^\top\rangle_{z^{\ell,*}|z^\ell}\rangle_{z^\ell}, \quad \ell = 1, \dots, L, \tag{B37}$$

and $K_0 = \frac{1}{N_0} X^\top X$, $k_0 = \frac{1}{N_0} X^\top x_*$, $\kappa_0 = \frac{1}{N_0} x_*^\top x_*$.

Predictor on single test input $x$:

$$f_r(x) = \sum_{\gamma=1}^{n} P_\gamma a_\gamma^r \langle\langle \phi(z^{L,*})\rangle_{z^{L,*}|z^L}\rangle_{z^L|a_\gamma} \tag{B38}$$

where $z^{\ell,*}$ are the preactivations on $x$. For $\phi(z) = \frac{1}{2}[1 + \mathrm{erf}(\sqrt{\pi}z/2)]$: $\langle \phi(z^{\ell,*})\rangle_{z^{\ell,*}|z^\ell} = \phi\big(c_\ell k_{\ell-1}^\top K_{\ell-1}^+ z^\ell\big)$ with $c_\ell = 1/\sqrt{1 + \frac{\pi\sigma_\ell^2}{2}[\kappa_{\ell-1} - k_{\ell-1}^\top K_{\ell-1}^+ k_{\ell-1}]}$.

**Regime $P > N_0$:** For $P > N_0$, the first layer preactivations are restricted to a $N_0$ dimensional subspace and the kernel $K_0$ is singular. The pseudo-inverse appearing in the theory restricts the matrix inversion to the corresponding subspace. To demonstrate that the theory is correct beyond $P \leq N_0$ we randomly project the images to a $N_0 = 50$ dimensional subspace such that $P = 2N_0$ for $P = 100$ (see Fig. S18). The nature of the solution indeed stays the same, a redundant code with neurons coding for either one or two classes, and the theory is in accurate quantitative agreement.

**Predictor Variance:** Here, we show that the predictor variance is suppressed. To this end we consider the scaled generating function

$$Z(Nj) = \int dP_0(\Theta)\, \exp\big( - N\beta \mathcal{L}(\Theta) + Nj^\top f(x;\Theta)\big). \tag{B39}$$

Decoupling with $t$ and introducing the kernels leads to

$$Z(Nj) = \int dt \prod_{\ell=1}^{L-1} d\hat{K}_\ell d\tilde{\hat{K}}_\ell\, e^{-NE(t,j,\{\hat{K}_\ell,\tilde{\hat{K}}_\ell\})}, \tag{B40}$$

$$\begin{aligned} E(t,j,\{\hat{K}_\ell,\tilde{\hat{K}}_\ell\}) = &- \frac{T}{2}\mathrm{tr}\, t^\top t + \mathrm{tr}\, t^\top Y + \frac{1}{2}\sum_{\ell=1}^{L-1} \mathrm{tr}\, \tilde{\hat{K}}_\ell^\top \hat{K}_\ell - \log \int dP_0(a) \int d\mathcal{N}(\hat{z}^L \,|\, 0, \sigma_L^2 \hat{K}_{L-1})\, e^{a^\top \hat{t}^\top \phi(\hat{z}^L)} \\ &- \sum_{\ell=1}^{L-1} \log \int d\mathcal{N}(\hat{z}^\ell \,|\, 0, \sigma_\ell^2 \hat{K}_{\ell-1})\, e^{\frac{1}{2}\mathrm{tr}\, \phi(\hat{z}^\ell)^\top \tilde{\hat{K}}_\ell \phi(\hat{z}^\ell)}, \end{aligned} \tag{B41}$$

where $\hat{t}$ denotes the concatenation of $t$ and $j$. After the saddle-point approximation we obtain

$$\log Z(Nj) = -NE(t,j,\{\hat{K}_\ell,\tilde{\hat{K}}_\ell\}). \tag{B42}$$

The mean predictor is $\frac{1}{N}\partial_j \log Z(Nj)|_{j=0} = -\partial_j E(t,j,\{\hat{K}_\ell,\tilde{\hat{K}}_\ell\})|_{j=0} = O(1)$, its variance is $\frac{1}{N^2}\partial_j^2 \log Z(Nj)|_{j=0} = -\frac{1}{N}\partial_j^2 E(t,j,\{\hat{K}_\ell,\tilde{\hat{K}}_\ell\})|_{j=0} = O(1/N)$. More generally, the $k$-the cumulant of the predictor is $O(1/N^{k-1})$.

## Appendix C: ReLU Networks ($L = 1$)

### 1. Training & Test Posterior

We directly start with the joint posterior on training and test data but restrict the theory to $L = 1$ and set $\sigma_a^2 = 1/P$. Following the same initial steps as for the linear and sigmoidal networks leads for the joint posterior of the readout weight matrix $A$ and the $N \times (P + P_*)$ preactivation matrix $\hat{Z}$ on $P_*$ test inputs $x_*$ to

$$P(A,\hat{Z}) \propto \int dt\, e^{-NE(t)} \prod_{i=1}^{N} P(a_i,\hat{z}_i), \tag{C1}$$

$$E(t) = -\frac{T}{2}\mathrm{tr}\, t^\top t + \mathrm{tr}\, t^\top Y - \log \int dP_0(a) \int d\mathcal{N}(z \,|\, 0, \sigma_1^2 K_0)\, e^{a^\top t^\top \phi(z)} \tag{C2}$$

$$P(a,\hat{z}) \propto P_0(a)\mathcal{N}(\hat{z} \,|\, 0, \sigma_1^2 \hat{K}_0) e^{a^\top t^\top \phi(z)} \tag{C3}$$

where we dropped the layer index and denote the $(P + P_*) \times (P + P_*)$ input kernel comprising all training and test inputs as $\hat{K}_0$ and the $P \times P$ input kernel on the training inputs as $K_0$. We explicitly break permutation symmetry for $n$ outlier neurons, leading to

$$P(A,\hat{Z}) \propto \int dt\, e^{-NE(t)} \prod_{i=1}^{n} [P(a_i,\hat{z}_i)] \prod_{i=n+1}^{N} [P(a_i,\hat{z}_i)], \tag{C4}$$

$$E(t) = -\frac{T}{2}\operatorname{tr} t^\top t + \operatorname{tr} t^\top Y - \big(1-\frac{n}{N}\big)\log\int dP_0(a)\int d\mathcal{N}(z\,|\,0,\sigma_1^2 K_0)\,e^{a^\top t^\top \phi(z)} + \frac{1}{N}\sum_{i=1}^n \log\int dP_0(a_i)\int d\mathcal{N}(z_i\,|\,0,\sigma_1^2 K_0)\,e^{a_i^\top t^\top \phi(z_i)}. \tag{C5}$$

Rescaling $a_i \to \sqrt{N}\bar{a}_i$ and $z_i \to \sqrt{N}\bar{z}_i$ for $i = 1,\dots,n$ ensures that the contribution from the outliers to the energy $E(t)$ is not suppressed by $O(1/N)$. The corresponding integrals can be solved in saddle-point approximation,

$$\frac{1}{N}\log\int dP_0(\sqrt{N}\bar{a}_i)\int d\mathcal{N}(\sqrt{N}\bar{z}_i\,|\,0,\sigma_1^2 K_0)\,e^{N\bar{a}_i^\top t^\top \phi(\bar{z}_i)} = -\frac{1}{2\sigma_a^2}\bar{a}_i^\top\bar{a}_i - \frac{1}{2\sigma_1^2}\bar{z}_i^\top K_0^+\bar{z}_i + \bar{a}_i^\top t^\top\phi(\bar{z}_i), \tag{C6}$$

where we used the homogeneity of ReLU, $\phi(\sqrt{N}\bar{z}_i) = \sqrt{N}\phi(\bar{z}_i)$, with saddle-point equations

$$\bar{a}_i = \sigma_a^2 t^\top\phi(\bar{z}_i), \qquad K_0^+\bar{z}_i = \phi'(\bar{z}_i)t\bar{a}_i, \qquad i = 1,\dots,n, \tag{C7}$$

where $\phi'(z)$ denotes a diagonal matrix with elements $\phi'(z_\mu)$. The saddle point approximation of the $t$ integral leads to

$$P(A,\hat{Z}) = \prod_{i=1}^n [P(a_i,\hat{z}_i)] \prod_{i=n+1}^N [P(a_i,\hat{z}_i)]. \tag{C8}$$

The readout posterior of the $N-n$ bulk neurons concentrates, similar to the sigmoidal case, leading to

$$P(a_i,\hat{z}_i) = \delta(a_i - a_0)P(z\,|\,a_0)P_0(z^*\,|\,z), \quad i = n+1,\dots,N. \tag{C9}$$

Due to the rescaling with $\sqrt{N}$ all posteriors of the outlier neurons concentrate, leading to

$$P(a_i,\hat{z}_i) = \delta(a_i - \sqrt{N}\bar{a}_i)\delta(z_i - \sqrt{N}\bar{z}_i)\delta(z_i^* - \sqrt{N}\bar{z}_i^*), \quad i = 1,\dots,n, \tag{C10}$$

with $\bar{z}_i^* = k_0^\top K_0^+\bar{z}_i$. The saddle point equation for $t$ simplifies to

$$Y = \big(1-\frac{n}{N}\big)\langle\phi(z)\rangle_{z|a_0}a_0^\top + \sum_{i=1}^n \phi(\bar{z}_i)\bar{a}_i^\top + Tt \tag{C11}$$

where the contribution due to the outliers is $O(1)$ because both readout weight and preactivation are $O(\sqrt{N})$.
The predictor on a test input $x$ is

$$f_r(x) = \big(1-\frac{n}{N}\big)a_0^r\langle\langle\phi[z(x)]\rangle_{z(x)|z}\rangle_{z|a_0} + \sum_{i=1}^n \bar{a}_i^r\phi[k_0(x)^\top K_0^+\bar{z}_i] \tag{C12}$$

with $k_0(x) = \frac{1}{N_0}X^\top x$. The Gaussian expectation $\langle\phi[z(x)]\rangle_{z(x)|z}$ can be solved,

$$\langle\phi[z(x)]\rangle_{z(x)|z} = \frac{\sigma_{z(x)|z}}{\sqrt{2\pi}}e^{-\frac{\mu_{z(x)|z}^2}{2\sigma_{z(x)|z}^2}} + \mu_{z(x)|z}\,H[-\mu_{z(x)|z}/\sigma_{z(x)|z}] \tag{C13}$$

with $\mu_{z(x)|z} = k_0(x)^\top K_0^+ z$, $\sigma_{z(x)|z}^2 = \sigma_1^2[\kappa_0(x) - k_0(x)^\top K_0^+ k_0(x)]$, and $H(x) = \int_x^\infty d\mathcal{N}(z\,|\,0,1) = \frac{1}{2}\operatorname{erfc}(x/\sqrt{2})$.
The kernel on a pair of test inputs $x_1, x_2$ is

$$K(x_1,x_2) = \langle\langle\phi(z_1)\phi(z_2)\rangle_{z_1,z_2|z}\rangle_{z|a_0} + \sum_{i=1}^n \phi[k_0(x_1)^\top K_0^+\bar{z}_i]\phi[k_0(x_2)^\top K_0^+\bar{z}_i] \tag{C14}$$

where $z_1, z_2$ denotes the preactivation on the test inputs.

**Summary ($L = 1$):** Posterior distribution on a set of test inputs $x_*$ separates into $N - n$ bulk neurons where

$$P(a) = \delta(a - a_0), \tag{C15}$$

$$P(z \,|\, a) \propto \mathcal{N}(z \,|\, 0, \sigma_1^2 K_0) e^{a^\top t^\top \phi(z)}, \tag{C16}$$

$$P(z^* \,|\, z) = \mathcal{N}(z^* \,|\, k_0^\top K_0^+ z, \sigma_1^2 \kappa_0 - \sigma_1^2 k_0^\top K_0^+ k_0), \tag{C17}$$

with $a_0 = \frac{1}{P} t^\top \langle \phi(z) \rangle_{z|a_0}$ and $n$ outlier neurons where

$$P(a_i) = \delta(a_i - \sqrt{N}\bar{a}_i), \quad i = 1, \dots, n, \tag{C18}$$

$$P(z_i \,|\, a_i) = \delta(z_i - \sqrt{N}\bar{z}_i), \quad i = 1, \dots, n, \tag{C19}$$

$$P(z_i^* \,|\, z_i) = \delta(z_i^* - \sqrt{N} k_0^\top K_0^+ \bar{z}_i), \quad i = 1, \dots, n, \tag{C20}$$

with $\bar{a}_i = \frac{1}{P} t^\top \phi(\bar{z}_i)$ and $K_0^+ \bar{z}_i = \phi'(\bar{z}_i) t \bar{a}_i$; $t$ is determined by $Y = \big(1 - \frac{n}{N}\big) \langle \phi(z) \rangle_{z|a_0} a_0^\top + \sum_{i=1}^n \phi(\bar{z}_i) \bar{a}_i^\top + Tt$. Finally, $K_0 = \frac{1}{N_0} X^\top X$, $k_0 = \frac{1}{N_0} X^\top x_*$, $\kappa_0 = \frac{1}{N_0} x_*^\top x_*$, and $\sigma_a^2 = 1/P$.
Predictor on test input $x$:

$$f_r(x) = a_0^r \langle \langle \phi[z(x)] \rangle_{z(x)|z} \rangle_{z|a_0} + \sum_{i=1}^n \bar{a}_i^r \phi[k_0(x)^\top K_0^+ \bar{z}_i] \tag{C21}$$

**Necessary number of outliers:** The number of outliers must be at least $n - 1 \geq m$ for general tasks. For fewer outliers, the constraints imposed by the training $Y = \big(1 - \frac{n}{N}\big) \langle \phi(z) \rangle_{z|a_0} a_0^\top + \sum_{i=1}^n \phi(\bar{z}_i) \bar{a}_i^\top$ cannot be fulfilled if $Y$ is full rank $m$ because the predictor is rank $n + 1$. If the bulk does not contribute, $a_0 \approx 0$, at least $n = m$ outliers are necessary.

**Dynamics:** The Langevin dynamics are similar to the dynamics of the sigmoidal networks (see Fig. S14): first, the loss decreases rapidly which brings the network close to the solution space. Already at this early stage a coding scheme is visible which is, however, not yet sparse. Subsequently, the dynamics is dominated by the regularizer. In this phase sparsity increases over time until the solution with three outliers is reached. In contrast to the sigmoidal case, gradient descent without noise does not reach this maximally sparse solution (see Fig. S15).

**Predictor Variance:** Here, we show that the predictor variance is suppressed. As in the sigmoidal case we consider the scaled generating function

$$Z(Nj) = \int dP_0(\Theta) \, \exp\big( - N\beta \mathcal{L}(\Theta) + N j^\top f(x; \Theta) \big). \tag{C22}$$

Decoupling with $t$, explicitly separating the $n$ outliers, rescaling $a_i \to \sqrt{N}\bar{a}_i$ and $\hat{z}_i \to \sqrt{N}\hat{\bar{z}}_i$ for $i = 1, \dots, n$, and solving the $\bar{a}_i$ and $\hat{\bar{z}}_i$ in saddle-point approximation leads to

$$Z(Nj) = \int dt \, e^{-NE(t,j)}, \tag{C23}$$

$$\begin{aligned} E(t, j) = &- \frac{T}{2} \mathrm{tr}\, t^\top t + \mathrm{tr}\, t^\top Y - \big(1 - \frac{n}{N}\big) \log \int dP_0(a) \int d\mathcal{N}(\hat{z} \,|\, 0, \sigma_1^2 \hat{K}_0) \, e^{a^\top \hat{t}^\top \phi(\hat{z})} \\ &- \sum_{i=1}^n \Big( \frac{1}{2\sigma_a^2} \bar{a}_i^\top \bar{a}_i + \frac{1}{2\sigma_1^2} \hat{\bar{z}}_i^\top K_0^+ \hat{\bar{z}}_i - \bar{a}_i^\top \hat{t}^\top \phi(\hat{\bar{z}}_i) \Big) \end{aligned} \tag{C24}$$

where $\hat{t}$ denotes the concatenation of $t$ and $j$. After the saddle-point approximation we obtain

$$\log Z(Nj) = -NE(t, j, \{\hat{K}_\ell, \tilde{\hat{K}}_\ell\}). \tag{C25}$$

The mean predictor is $\frac{1}{N} \partial_j \log Z(Nj)|_{j=0} = -\partial_j E(t, j, \{\hat{K}_\ell, \tilde{\hat{K}}_\ell\})|_{j=0} = O(1)$, its variance is $\frac{1}{N^2} \partial_j^2 \log Z(Nj)|_{j=0} = -\frac{1}{N} \partial_j^2 E(t, j, \{\hat{K}_\ell, \tilde{\hat{K}}_\ell\})|_{j=0} = O(1/N)$. More generally, the $k$-the cumulant of the predictor is $O(1/N^{k-1})$.

## Appendix D: Numerics

The code is available on figshare (doi:10.6084/m9.figshare.26539129).

### 1. Empirical Sampling

For all experiments we relied heavily on NumPy [10] and SciPy [11]. To obtain samples from the weight posterior we use Hamiltonian Monte Carlo with the no-u-turn sampler (see, e.g., [12, 13]) implemented in numpyro [14, 15] with jax [16] as a back-end; the model itself is implemented using the front-end provided by pymc [17]. Unless specified otherwise, in all sampling experiments we use a small but finite (rescaled) temperature $T = 10^{-4}$ and keep $\sigma_\ell = 1$ for $1 \leq \ell \leq L$. For all experiments, we preprocess each input by subtracting the mean and normalizing such that $\|x_\mu\|^2 = N$ for all $\mu$.

### 2. Theory

For linear networks, the solution is straightforward to implement. For nonlinear networks, we solve the saddle point equations by constructing an auxiliary squared loss function with its global minima at the self consistent solutions, e.g., $\sum_{r=1}^{m}\sum_{\mu=1}^{P}[y_\mu^r - \langle a_r \phi(z_\mu)\rangle_{a,z} - T t_\mu^r]^2$ for $t$. This auxiliary loss is minimized using the gradient based minimizer available in jaxopt.ScipyMinimize [18]; after convergence it is checked that the solution achieves zero auxiliary loss, i.e., that a self-consistent solution is obtained.

On the toy example, the minimization is performed jointly for all $a_\gamma$ and $t$; the one dimensional $z$ integrals are solved numerically using Gauss-Hermite quadrature with fixed order for sigmoidal and softplus networks or analytical solutions for ReLU networks.

Beyond the toy example, the minimization is performed jointly for all $z_\gamma$, $a_\gamma$, and $t$. For ReLU networks, we approximate $\max(0, x) \approx \frac{1}{c}\log(1 + e^{cx})$ with $c = 10$ to make the auxiliary loss differentiable.

### 3. GP Theory

The GP theory corresponds to lazy networks (with $1/\sqrt{N}$ scaling of the outputs) in the infinite-width limit ($N \to \infty$ with finite $P$). In the GP theory [19–22] the predictor mean and variance are

$$f_r(x) = k_{GP}(x)^\top K_{GP}^{-1} y_r \tag{D1}$$

$$\langle \delta f_r(x;\Theta)^2\rangle_\Theta = \kappa_{GP}(x) + k_{GP}(x)^\top K_{GP}^{-1} k_{GP}(x) \tag{D2}$$

such that the generalization error is $\epsilon_g(x) = [y_r - f_r(x)]^2 + \langle \delta f_r(x;\Theta)^2\rangle_\Theta$ where, unlike in non-lazy networks, the second contribution due to the variance cannot be neglected. The associated GP kernel is determined recursively for arbitrary inputs $x_1, x_2$ by $K_\ell(x_1, x_2) = \langle \phi(z_1^\ell)\phi(z_2^\ell)\rangle$ where $z_1^\ell, z_2^\ell$ are zero mean Gaussian with covariance $\langle z_1^\ell z_2^\ell\rangle = \sigma_\ell^2 K_{\ell-1}(x_1, x_2)$ and $K_0(x_1, x_2) = \frac{1}{N_0} x_1^\top x_2$. In the predictor the last layer kernel appears, $K_{GP}(x_1, x_2) = K_L(x_1, x_2)$, and $K_{GP}$ denotes the associated $P \times P$ matrix on the training inputs $K_{GP}(x_\mu, x_\nu)$, $k_{GP}(x)$ the $P$-dimensional vector containing $K_{GP}(x, x_\mu)$, and $\kappa_{GP}(x) = K_{GP}(x, x)$.

For the nonlinearities considered here the Gaussian expectation $\langle \phi(z_1^\ell)\phi(z_2^\ell)\rangle$ is tractable for ReLU [23] and erf [8, 20]. We employed the analytical formulas for the implementation and performed the inversion using Cholesky decomposition.

## Appendix E: Supplemental Tables and Figures

Brief summary and discussion of the supplemental figures (since we include both pre- and postactivations, we here refer to the latter as postactivations instead of activations):

- Table I: Scaling of the prior variance of the readout weights and approximations used in the theory for each nonlinearity.

- Fig. S1: Extended comparison between sampling and theory for a two-hidden-layer linear network including the theoretical kernel (C,F) and a comparison of the readout weight posterior across samples for a single neuron (G) and across neurons for a given sample (H).

- Fig. S2: Extended view of the results for sigmoidal nonlinearity presented in Fig. 2 including a comparison of the readout weight posterior across samples for a single neuron (A) and across neurons for a given sample (B), the training preactivations (E), and the theoretical kernel (I).

- Fig. S3: Multiple self-consistent theoretical solutions for the sigmoidal setup from Fig. 2. Each solution corresponds to a saddle point of the $t$ integral. Comparing $E(t)$ shows that the sampled solution in Fig. 2 has the lowest energy (A). The lowest energy solution does not correspond to the minimal number of codes needed to solve the task: the lowest energy solution comprises 4 codes (B); the other solutions comprise only 3 codes (C,D).

- Fig. S4: Shows the emerging code using the same setup as in the sigmoidal part of Fig. 2 except that the one-hot encoding of the targets $y_-$ increases from $y_- = 0.1$ to $y_- = 0.5$ (which was used in Fig. 2), keeping $y_+ = 1$ as before. At $y_- = 0.1$, the coding scheme contains all six permutations of the 1-0-0 and 1-1-0 codes (Fig. S4(A)). At $y_- = 0.3$ the codes 0-1-0 and 0-0-1 disappeared (Fig. S4(B)) and at $y_- = 0.4$ also the code 1-0-0 is lost (Fig. S4(C)). Finally, at $y_- = 0.5$ the new code 1-1-1 emerged (Fig. S4(D)).

- Fig. S5: Temperature dependent phase transition of the redundant coding scheme from Fig. 2. At a high (rescaled) temperature $T = 0.1$ the activations do not exhibit a code and the readout weight posterior is unimodal (A,E,I). At a lower temperature, a coding scheme emerges: $T = 0.08$ is still above the critical temperature (B,F,J) but at $T = 0.07$ a coding scheme emerges involving a subset of the neurons and the readout weight posterior becomes multimodal (C,G,K). Further lowering the temperature to $T = 0.01$ leads to the solution identical to the one shown in Fig. 2 at $T = 10^{-4}$.

- Fig. S6: Extended version of Fig. 5 including the feature layer and readout weight dynamics, showing their frozen code.

- Fig. S7: Approach to equilibrium using Langevin dynamics for the setup from the sigmoidal part of Fig. 2. Starting from initial conditions sampled from the prior, the Langevin dynamics first rapidly decrease the loss (A; note the logarithmic x-axis) until the solution space is approximately reached. Afterwards, the dynamics decrease the regularizer imposed by the prior while staying at the solution space (B) until the final redundant coding scheme is reached (C,F). After the first phase of approaching the solution space, a coding scheme starts to emerge (D). During the minimization of the regularizer the coding scheme is sharpened (E) until it reaches the final solution (F).

- Fig. S8: Gradient ascent on the log posterior for the setup from the sigmoidal part of Fig. 2, i.e., identical to Fig. S7 except that the dynamics are noiseless ($\frac{d}{dt}\Theta = -\nabla\mathcal{L}(\Theta) + T\nabla \log P_0(\Theta)$). The dynamics go through the same phases of loss minimization (A) and minimization of the regularizer at the solution space (B). The coding scheme is identical to the Langevin case except for missing variability due to the missing noise (C,F). This, however, is dependent on the random initialization; for other random initializations there can be a few neurons with a deviating code (not shown). Similar to Langevin dynamics a code appears after the first phase of approaching the solution space and is further sharpened during the minimization of the regularizer (D,E,F).

- Fig. S9: Identical to Fig. S7 except that $\sigma_a^2 = 1$ instead of $\sigma_a^2 = 1/P$. Again, the dynamics start with an abrupt decrease of the loss (A). In contrast to Fig. S7 the slow relaxation of the regularizer is absent (B). This leads to a representation which displays only a weak, rather noisy structure. This structure still corresponds to a redundant coding scheme but, unlike in Fig. S7, it would not be apparent without sorting the neurons appropriately. Put differently, allowing the readout weights to grow leads to a representation which exhibits but a shadow of the coding scheme.

- Fig. S10: Identical to Fig. S8, i.e., noiseless gradient ascent ($\frac{d}{dt}\Theta = -\nabla\mathcal{L}(\Theta) + T\nabla \log P_0(\Theta)$), except that $\sigma_a^2 = 1$ instead of $\sigma_a^2 = 1/P$. Qualitatively, the dynamics are similar to the ones in Fig. S8: After a sharp decrease of the MSE the regularizer slowly decays and the final solution displays a clear redundant coding scheme. The differences to Fig. S8 are that the sigmoidal nonlinearity is less saturated (F) and the readout weights are more distributed (C). The striking difference to the noisy dynamics with $\sigma_a^2 = 1$ is that without noise a clear coding

scheme emerges. This indicates that the regularization leads to a minimum norm solution which is characterized by a redundant coding scheme; furthermore it suggests that the scaling of $\sigma_a^2$ is necessary to balance the impact of the noise. While in this case the solution with a clear coding scheme could be reached without noise, we expect in general that there are barriers in the loss landscape which can only be overcome by noise.

- Fig. S11: Gradient descent on the loss in the absence of regularization ($\frac{d}{dt}\Theta = -\nabla\mathcal{L}(\Theta)$). Unlike in the regularized but noiseless cases, Figs. S8 and S10, the final solution does not exhibit a clear redundant coding scheme (F). Instead, the solution is qualitatively similar to the noisy case with $\sigma_a^2 = 1$: Only a shadow of a redundant coding scheme is apparent in the representation. This suggests that gradient descent without explicit regularization does not reach the minimum norm solution.

- Fig. S12: Extended view of the ReLU results presented in Fig. 2 including a comparison of the readout weight posterior across samples for a single neuron (A) and across neurons for a given sample (B), the training preactivations of the outlier neurons (E), and the theoretical kernel (I).

- Fig. S13: Readout weight and representation dynamics in a two-hidden-layer ReLU network on the toy task. Similar to the sigmoidal case in Fig. S6 the code in the feature layer is frozen. In contrast to the sigmoidal case in Fig. S6 the lower layer is also frozen.

- Fig. S14: Approach to equilibrium using Langevin dynamics for the ReLU part of Fig. 2. Similar to the sigmoidal case, the dynamics go through the phases of loss minimization (A) and minimization of the regularizer at the solution space (B) until the final sparse code is reached (C,F). After the first phase of approaching the solution space, a coding scheme starts to emerge which is not yet sparse (D). During the minimization of the regularizer the coding scheme becomes sparser (E) until it reaches the final solution (F).

- Fig. S15: Gradient ascent on the log posterior for the setup from the ReLU part of Fig. 2, i.e., identical to Fig. S14 except that the dynamics are noiseless. The dynamics go through the same phases of loss minimization (A) and minimization of the regularizer at the solution space (B). However, unlike in the sigmoidal case, the final solution is different from the Langevin case (C,F). This holds for all tested random initializations. Similar to Langevin dynamics a coding scheme involving all neurons appears after the first phase of approaching the solution space and is further sharpened during the minimization of the regularizer (D,E,F). However, the final solution is less sparse than the solution based on Langevin dynamics. This is likely due to local minima (A) which cannot be overcome without noise.

- Fig. S16: Extended comparison between sampling and theory for a two-hidden-layer linear network on MNIST including training and test kernels in all layers from theory and sampling (D-I).

- Fig. S17: Extended view of the sigmoidal results presented in Fig. 6 including the training preactivations (C), training preactivations of neurons with 0-0-1 code (D), the training preactivation kernel from sampling (E) and theory (F), and the test postactivation kernel from sampling (G) and theory (H). Note that the variability of the preactivations across inputs (C,D) is not apparent in the postactivations since the preactivations saturate the sigmoidal nonlinearity; correspondingly the preactivation kernel exhibits more structure (E,F).

- Fig. S18: Application of the theory to the regime $P > N_0$. Identical to Fig. S17 except that the inputs are randomly projected to a $N_0 = 50$ dimensional subspace such that $P = 2N_0$.

- Fig. S19: Application of the theory to three categories of gray scaled CIFAR10 instead of MNIST. Identical to Fig. S17 except for the replacement of MNIST by gray scaled CIFAR10 inputs. While the coding scheme on training data is identical for CIFAR10 (B,C) and MNIST (Fig. S17(B,C)), the test kernels are drastically different and exhibit no structure corresponding to the task (G,H) and the predictor is almost random (I), leading to a high generalization error (A). Put differently, the solution fits the training data but does not generalize.

- Fig. S20: Drifting representation in the first layer of a two-hidden-layer sigmoidal network on MNIST. Similar to Fig. S6 except that MNIST is used instead of the toy task.

- Fig. S21: Extended view of the ReLU results presented in Fig. 6 including the training preactivations (C), the training preactivation kernel from sampling (E) and theory (F), and the test postactivation kernel from sampling (G) and theory (H). Note that the preactivations are negative on inputs belonging to classes that the neuron does not code (C), leading to zero postactivations on those inputs.

- Fig. S22: Sparse coding scheme in single layer softplus networks on random classification task. Identical to the ReLU results in Fig. 2 except that the nonlinearity is "softplus" $\phi(z) = \log(1 + e^z)$ instead of ReLU $\phi(z) = \max(0, z)$.

- Fig. S23: Sparse coding scheme in single layer softplus networks on random classification task. Identical to the ReLU results in Fig. 2 except that the nonlinearity is "rectified quadratic" $\phi(z) = \max(0, z^2)$ instead of ReLU $\phi(z) = \max(0, z)$.

- Fig. S24: Coding scheme and generalization of single layer softplus networks on MNIST. Identical to the ReLU results in Fig. 6 except that the nonlinearity is "softplus" $\phi(z) = \log(1 + e^z)$ instead of ReLU $\phi(z) = \max(0, z)$.

- Fig. S25: Coding schemes in a two-hidden-layer network with ReLU activation function $\phi(z) = \max(0, z)$ in the first and sigmoidal activation function $\phi(z) = \frac{1}{2}[1 + \mathrm{erf}(\sqrt{\pi}z/2)]$ in the second layer. In the first (ReLU) layer, the coding is sparse; in the second (sigmoidal) layer the coding is redundant. The code is frozen in both layers, as in the two-hidden-layer case with ReLU activation in both layers. Here, we used $\sigma_a^2 = 1/P$.

- Fig. S26: Coding schemes in a two-hidden-layer network with sigmoidal activation function $\phi(z) = \frac{1}{2}[1 + \mathrm{erf}(\sqrt{\pi}z/2)]$ in the first and ReLU activation function $\phi(z) = \max(0, z)$ in the second layer. In the first (sigmoidal) layer, the coding is redundant; in the second (ReLU) layer the coding is sparse. Interestingly, the code seems to be frozen in both layers, in contrast to the two-hidden-layer case with sigmoidal activation in both layers where the code is not frozen in the first layer. Here, we used $\sigma_a^2 = 1/P$.

| Nonlinearity | Scaling of $\sigma_a^2$ | Theory |
|---|---|---|
| $\phi(z) = z$ | $\sigma_a^2 = 1/P^{1/L}$ | High-dimensional saddle-point approximation (SPA) of $P \times m$- or $P \times P$-dim. integrals $t, K_\ell, \tilde{K}_\ell$, $\ell = 1, \dots, L-1$, using $N \to \infty$; identical results obtained from a low-dimensional SPA of $m \times m$-dim. integrals $U_\ell$, $\ell = 1, \dots, L-1$, using $N \to \infty$. |
| $\phi(z) = \frac{1}{2}[1 + \mathrm{erf}(\sqrt{\pi} z/2)]$ | $\sigma_a^2 = 1/P$ | High-dimensional SPA of $P \times m$- or $P \times P$-dim. integrals $t, K_\ell, \tilde{K}_\ell$, $\ell = 1, \dots, L-1$, using $N \to \infty$ followed by low-dimensional SPA of $m$-dim. integrals $a$ using $P \to \infty$. For the numerical solution of the theory on MNIST the fluctuations of the preactivations were neglected. |
| $\phi(z) = \max(0, z)$ | $\sigma_a^2 = 1/P^{1/L}$ | Only $L = 1$: Explicit breaking of permutation symmetry by selecting outliers, high-dimensional SPA of $P \times m$-dim. integral $t$ using $N \to \infty$ followed by high- and low-dimensional SPA for outliers of $m$- and $P$-dim. integrals $a_i, z_i$, $i = 1, \dots, n$, using $N \to \infty$ and low-dimensional SPA for bulk of $m$-dim. integrals $a_0$ using $P \to \infty$. For the numerical solution of the theory on MNIST the contribution due to the bulk was neglected. |

Table I. Summary of prior readout variance scaling and main steps in the derivation of the theory for each nonlinearity.

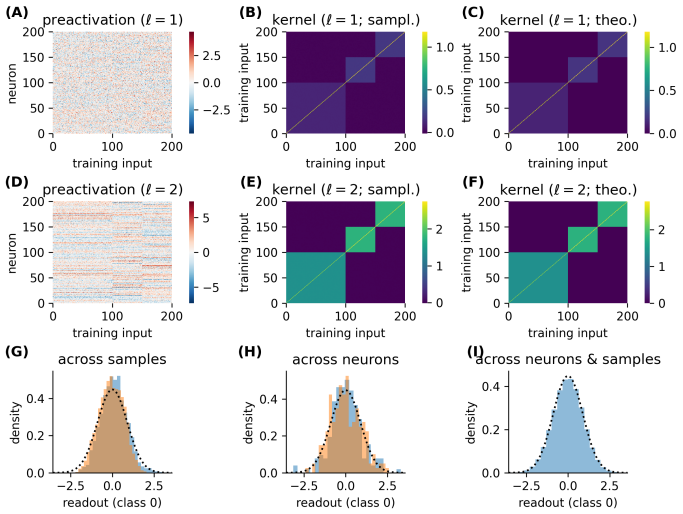


Figure S1. Coding scheme in two-hidden-layer linear networks on random classification task. (**A**,**D**) Preactivations of all neurons on all training inputs for a given weight sample in the first (A) and last (D) hidden layer. (**B**,**C**,**E**,**F**) Kernels on training data from sampling (B,E) and theory (C,F) in the first (B,C) and last (E,F) hidden layer. (**G**-**I**) Distribution of readout weight on first class across samples for two given neurons (G), across neurons for two given samples (H), and across neurons and samples (I). Theoretical distribution (A26) as black dashed line. Parameters: $N = P = 200$, $N_0 = 220$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = 0$.

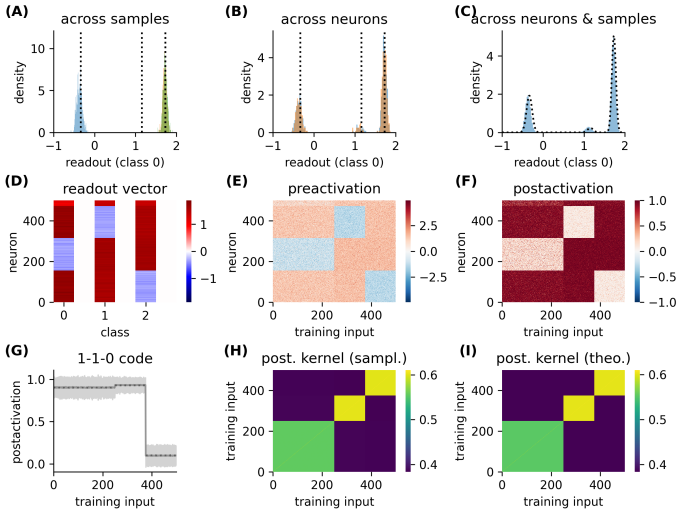


Figure S2. Coding scheme in single-hidden-layer sigmoidal networks on random classification task. (**A**-**C**) Distribution of readout weight on first class across samples for three given neurons (A), across neurons for two given samples (B), and across neurons and samples (C). Theoretical distribution (B13) as black dashed line, in (C) including finite $P$ correction. (**D**) Sample of the readout weights. (**E**,**F**) Pre- (E) and postactivations (F) of all neurons on all training inputs for a given weight sample. (**G**) Postactivation of neurons with 1-0-1 code from sampling (gray) and theory (black dashed). Shaded area shows standard deviation across neurons for a given weight sample. (**H**,**I**) Postactivation kernel on training data from sampling (H) and theory (I). Parameters: $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = 1/2$.

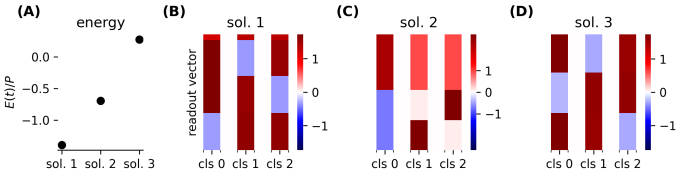


Figure S3. Multiple self-consistent solutions. (**A**) Energy $E(t)$ for multiple self-consistent solutions for $t$. Solutions with higher energy are suppressed with probability $\exp[-N\Delta E]$. (**B**-**D**) Readout vector corresponding to the self consistent solutions for $t$. Parameters: $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = 1/2$.

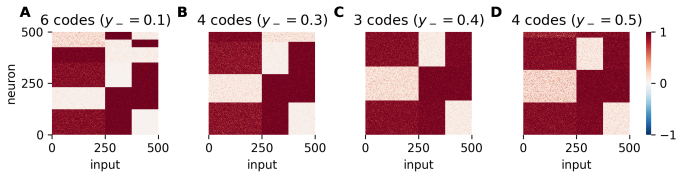


Figure S4. Coding scheme transitions in single-hidden-layer sigmoidal networks on random classification task. (**A**-**D**) Activations of all neurons on all training inputs for a given weight sample with changing training target $y_-$. Parameters as in Fig. 2 except that $y_-$ changes as indicated in the figure.

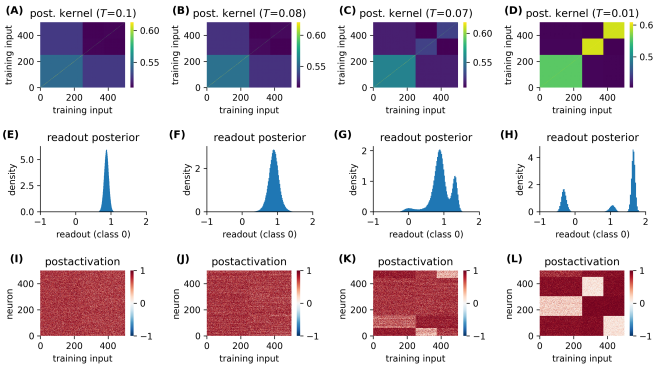


Figure S5. Emergence of code with decreasing temperature in single-hidden-layer sigmoidal networks on random classification task. (**A**-**D**) Postactivation kernel on training data from sampling for varying temperature. (**E**-**H**) Readout posterior of first class for varying temperature. (**I**-**L**) Postactivations of all neurons on all training inputs for a given weight sample for varying temperature. Parameters: $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = 1/2$.

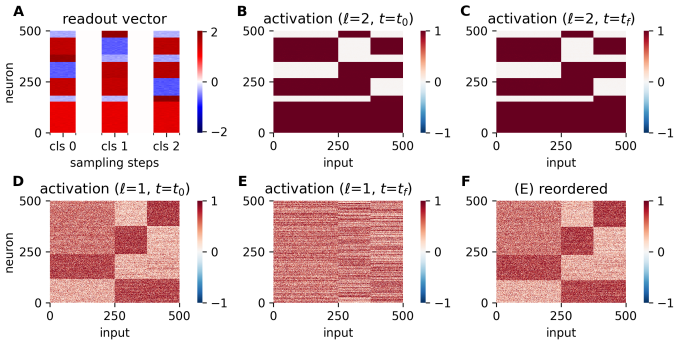


Figure S6. Frozen and drifting representations in two-hidden-layer sigmoidal networks. (**A**) Samples of the readout weights of all three classes. (**B**,**C**) Activations of all second layer neurons on all training inputs using first (B) and last (C) weight sample. (**D**,**E**) Activations of all first layer neurons on all training inputs using first (D) and last (E) weight sample. (**F**) Activations from (E) but neurons are reordered. Parameters: $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = 1/2$.

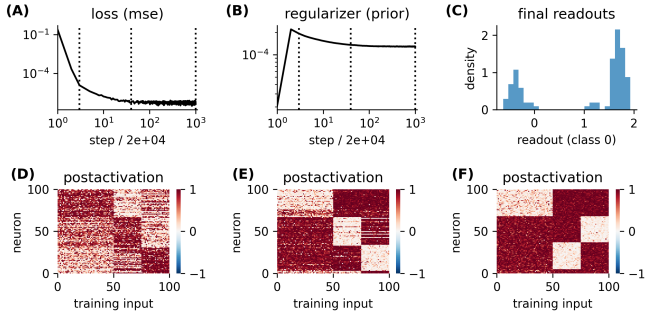


Figure S7. Langevin dynamics of single-hidden-layer sigmoidal network on random classification task. (**A**,**B**) Mean squared error (A) and regularizer (B) over time. (**C**) Readout posterior of first class at final step. (**D**-**F**) Postactivations of all neurons on all training inputs for a given weight sample at the steps indicated in (A,B). Parameters: $N = P = 100$, $N_0 = 120$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = 1/2$.

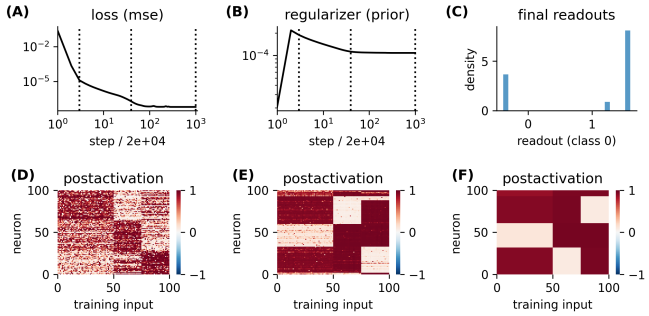


Figure S8. Gradient descent dynamics of single-hidden-layer sigmoidal network on random classification task. (**A**,**B**) Mean squared error (A) and regularizer (B) over time. (**C**) Readout posterior of first class at final step. (**D**-**F**) Postactivations of all neurons on all training inputs for a given weight sample at the steps indicated in (A,B). Parameters: $N = P = 100$, $N_0 = 120$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = 1/2$.

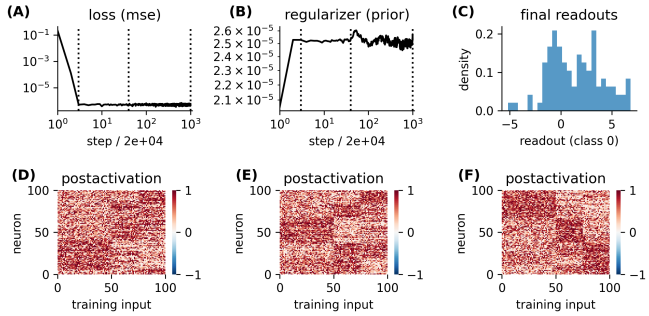


Figure S9. Langevin dynamics of single-hidden-layer sigmoidal network on random classification task with $\sigma_a^2 = 1$ instead of $\sigma_a^2 = 1/P$. (**A**,**B**) Mean squared error (A) and regularizer (B) over time. (**C**) Readout posterior of first class at final step. (**D**-**F**) Postactivations of all neurons on all training inputs for a given weight sample at the steps indicated in (A,B). Parameters: $N = P = 100$, $N_0 = 120$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = 1/2$.

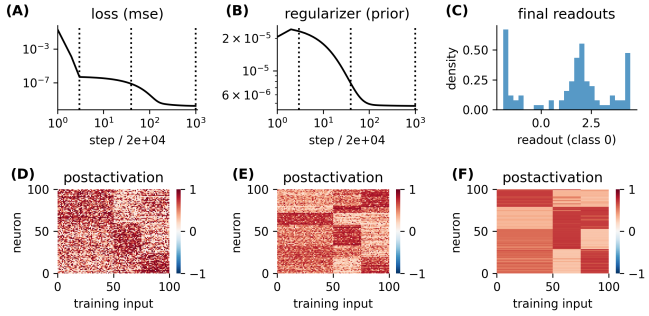


Figure S10. Gradient descent dynamics of single-hidden-layer sigmoidal network on random classification task with $\sigma_a^2 = 1$ instead of $\sigma_a^2 = 1/P$. (**A**,**B**) Mean squared error (A) and regularizer (B) over time. (**C**) Readout posterior of first class at final step. (**D**-**F**) Postactivations of all neurons on all training inputs for a given weight sample at the steps indicated in (A,B). Parameters: $N = P = 100$, $N_0 = 120$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = 1/2$.

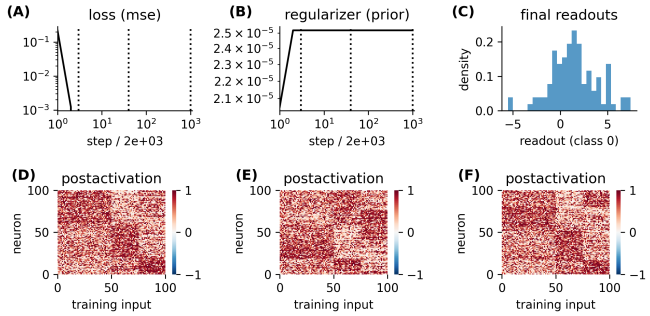


Figure S11. Gradient descent dynamics of single-hidden-layer sigmoidal network on random classification task without regularization. (**A**,**B**) Mean squared error (A) and regularizer (B) over time. (**C**) Readout posterior of first class at final step. (**D**-**F**) Postactivations of all neurons on all training inputs for a given weight sample at the steps indicated in (A,B). Parameters: $N = P = 100$, $N_0 = 120$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = 1/2$.

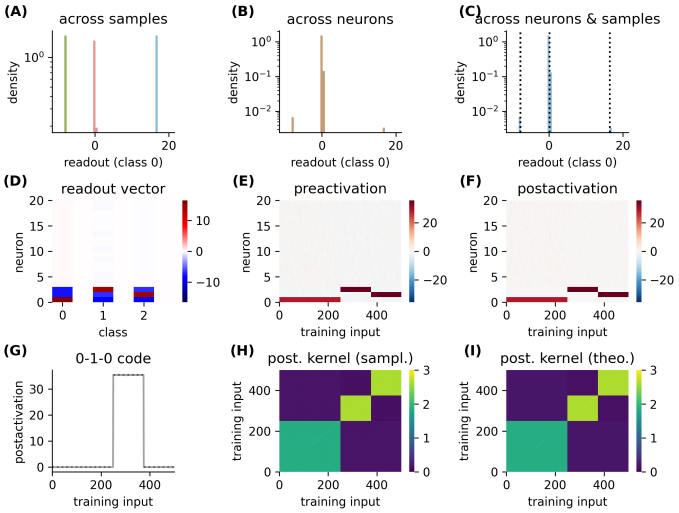


Figure S12. Coding scheme in single-hidden-layer ReLU networks on random classification task. (**A**-**C**) Distribution of readout weight on first class across samples for four given neurons (A), across neurons for two given samples (B), and across neurons and samples (C). Theoretical distribution (C15) and (C18) as black dashed line. (**D**) Sample of the readout weights for a sorted subset of 20 neurons. (**E**,**F**) Pre- (E) and postactivations (F) for a sorted subset of 20 neurons on all training inputs for a given weight sample. (**G**) Postactivation of the neuron with 0-1-0 code from sampling (gray) and theory (black dashed). (**H**,**I**) Postactivation kernel on training data from sampling (H) and theory (I). Parameters: $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = -1/2$.

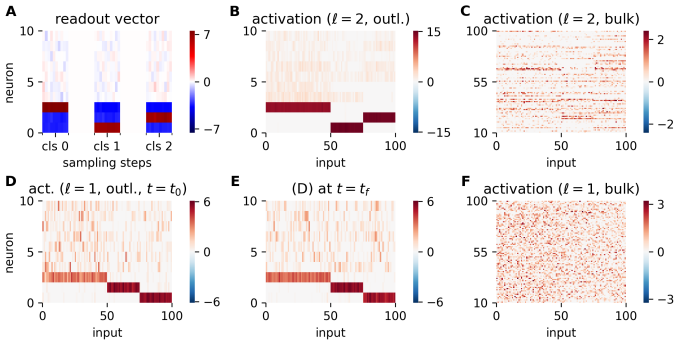


Figure S13. Frozen representations in two-hidden-layer ReLU networks. (**A**) Samples of the readout weights of all classes for the most active neurons. (**B**,**C**) Activations of the most active (B) and the remaining (C) neurons in the second layer on all training inputs for a given weight sample. (**D**,**E**,**F**) Activations of the most active (D,E) and the remaining (F) neurons in the first layer using the first (D,F) or the last (E) weight sample. Parameters: $N = P = 100$, $N_0 = 120$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = -1/2$.

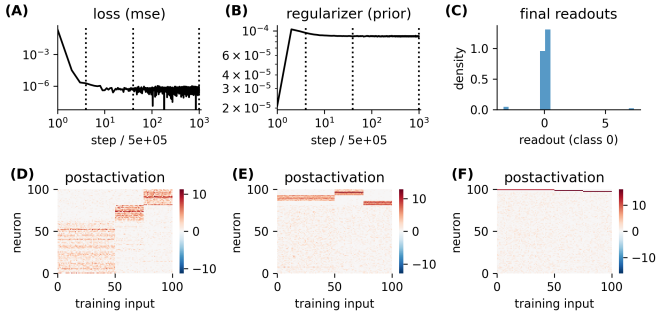


Figure S14. Langevin dynamics of single-hidden-layer ReLU network on random classification task. (**A**,**B**) Mean squared error (A) and regularizer (B) over time. (**C**) Readout posterior of first class at final step. (**D**-**F**) Postactivations of all neurons on all training inputs for a given weight sample at the steps indicated in (A,B). Parameters: $N = P = 100$, $N_0 = 120$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = -1/2$.

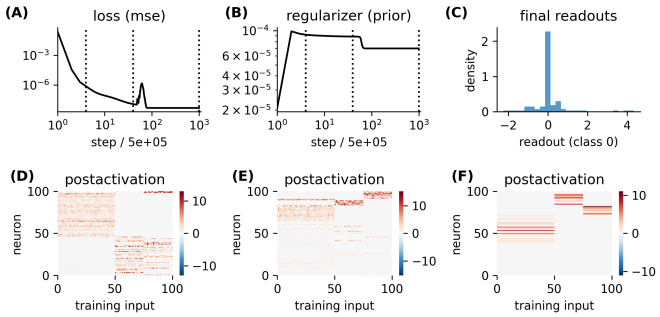


Figure S15. Gradient descent dynamics of single-hidden-layer ReLU network on random classification task. (**A**,**B**) Mean squared error (A) and regularizer (B) over time. (**C**) Readout posterior of first class at final step. (**D**-**F**) Postactivations of all neurons on all training inputs for a given weight sample at the steps indicated in (A,B). Parameters: $N = P = 100$, $N_0 = 120$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = -1/2$.

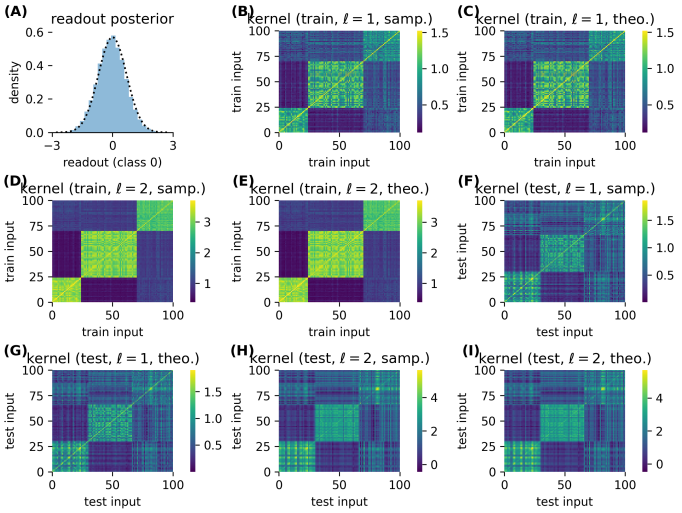


Figure S16. Generalization of two-hidden-layer linear networks on MNIST. (**A**) Readout posterior of first class. (**B-I**) Kernel on training (A-E) and test (F-I) data in first (B,C,F,G) and second layer (D,E,H,I) from sampling (B,D,F,H) and theory (C,E,G,I). Parameters: $N = P = 100$, $N_0 = 784$, classes 0, 1, 2 assigned randomly with probability 1/3, targets $y_+ = 1$ and $y_- = 0$.

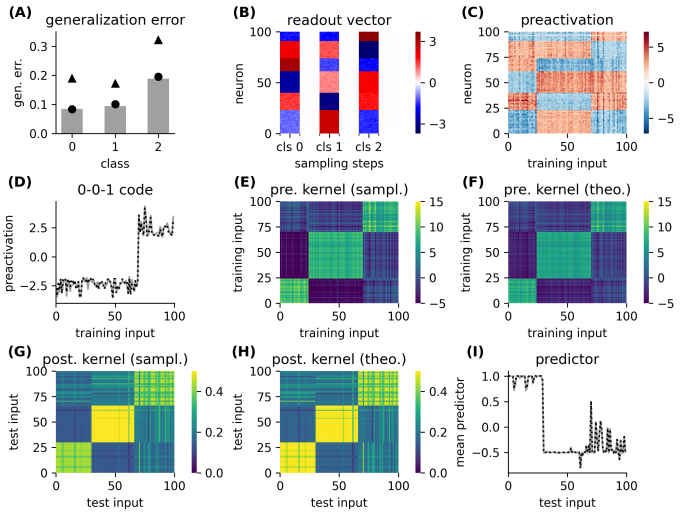


Figure S17. Generalization of single-hidden-layer sigmoidal networks on MNIST. (**A**) Generalization error for each class averaged over 1,000 test inputs from sampling (gray bars), theory (Eq. (B38), black circles), and GP theory (back triangles). (**B**) Sample of the readout weights for all neurons. (**C**) Preactivations of all neurons on all training inputs for a given weight sample. (**D**) Averaged preactivation of neurons with 0-0-1 code from sampling (gray) and theory (black dashed). (**E**,**F**) Preactivation kernel on training data from sampling (E) and theory (F). (**G**,**H**) Postactivation kernel on test data from sampling (G) and theory (H). (**I**) Mean predictor for class 0 from sampling (gray) and theory (black dashed). Parameters: $N = P = 100$, $N_0 = 784$, classes 0, 1, 2 assigned randomly with probability 1/3, targets $y_+ = 1$ and $y_- = -1/2$.

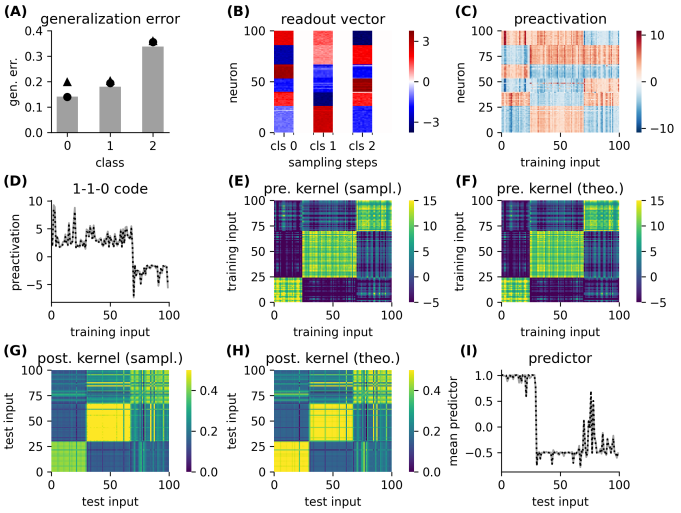


Figure S18. Generalization of single-hidden-layer sigmoidal networks on projected MNIST. Identical to Fig. S17 except that the data was randomly projected to a $N_0 = 50$ dimensional subspace. (**A**) Generalization error for each class averaged over $1,000$ test inputs from sampling (gray bars), theory (Eq. (B38), black circles), and GP theory (back triangles). (**B**) Sample of the readout weights for all neurons. (**C**) Preactivations of all neurons on all training inputs for a given weight sample. (**D**) Averaged preactivation of neurons with 0-0-1 code from sampling (gray) and theory (black dashed). (**E**,**F**) Preactivation kernel on training data from sampling (E) and theory (F). (**G**,**H**) Postactivation kernel on test data from sampling (G) and theory (H). (**I**) Mean predictor for class 0 from sampling (gray) and theory (black dashed). Parameters: $N = P = 100$, $N_0 = 50$, classes 0, 1, 2 assigned randomly with probability 1/3, targets $y_+ = 1$ and $y_- = -1/2$.

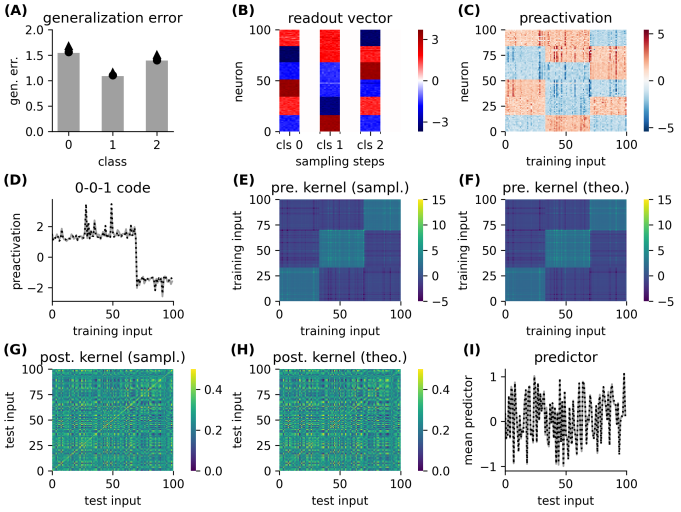


Figure S19. Generalization of single-hidden-layer sigmoidal networks on grayscale CIFAR10. Identical to Fig. S17 except that CIFAR10 instead of MNIST was used. The images were converted to grayscale such that $N_0 = 1024$. (**A**) Generalization error for each class averaged over $1,000$ test inputs from sampling (gray bars), theory (Eq. (B38), black circles), and GP theory (back triangles). (**B**) Sample of the readout weights for all neurons. (**C**) Preactivations of all neurons on all training inputs for a given weight sample. (**D**) Averaged preactivation of neurons with 0-0-1 code from sampling (gray) and theory (black dashed). (**E**,**F**) Preactivation kernel on training data from sampling (E) and theory (F). (**G**,**H**) Postactivation kernel on test data from sampling (G) and theory (H). (**I**) Mean predictor for class 0 from sampling (gray) and theory (black dashed). Parameters: $N = P = 100$, $N_0 = 1024$, classes 0, 1, 2 assigned randomly with probability 1/3, targets $y_+ = 1$ and $y_- = -1/2$.

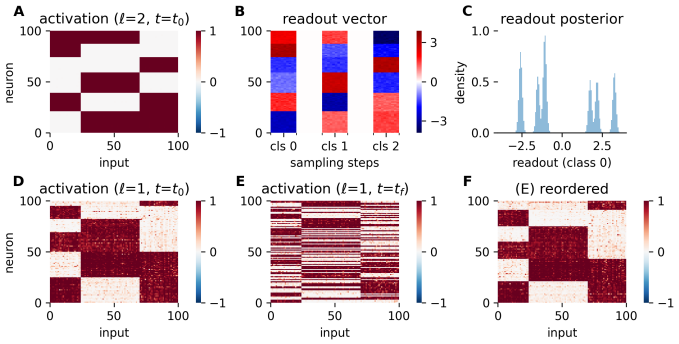


Figure S20. Coding scheme in two-hidden-layer sigmoidal networks on MNIST. (**A**) Postactivations of all second layer neurons on all training inputs for a given weight sample. (**B**) Samples of the readout weights of all three classes. (**C**) Readout posterior of first class. (**D**,**E**) Postactivations of all first layer neurons on all training inputs using first (D) and last (E) weight sample. (**F**) Postactivations from (E) but neurons are reordered. Parameters: $N = P = 100$, $N_0 = 784$, classes 0, 1, 2 assigned randomly with probability 1/3, targets $y_+ = 1$ and $y_- = -1/2$.

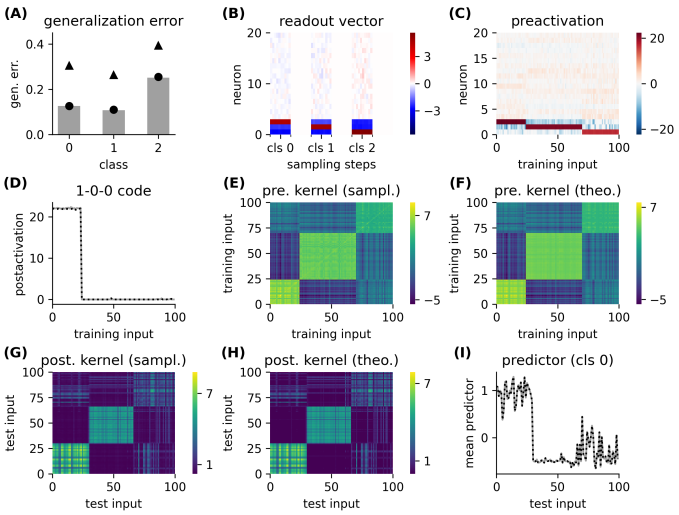


Figure S21. Generalization of single-hidden-layer ReLU networks on MNIST. (**A**) Generalization error for each class averaged over 1,000 test inputs from sampling (gray bars), theory (Eq. (C21), black circles), and GP theory (back triangles). (**B**) Sample of the readout weights for outlier neurons. (**C**) Preactivations of outlier neurons on all training inputs for a given weight sample. (**D**) Averaged postactivation of neurons with 1-0-0 code from sampling (gray) and theory (black dashed). (**E**,**F**) Preactivation kernel on training data from sampling (E) and theory (F). (**G**,**H**) Postactivation kernel on test data from sampling (H) and theory (I). (**I**) Mean predictor for class 0 from sampling (gray) and theory (black dashed). Parameters: $N = P = 100$, $N_0 = 784$, classes 0, 1, 2 assigned randomly with probability 1/3, targets $y_+ = 1$ and $y_- = -1/2$.

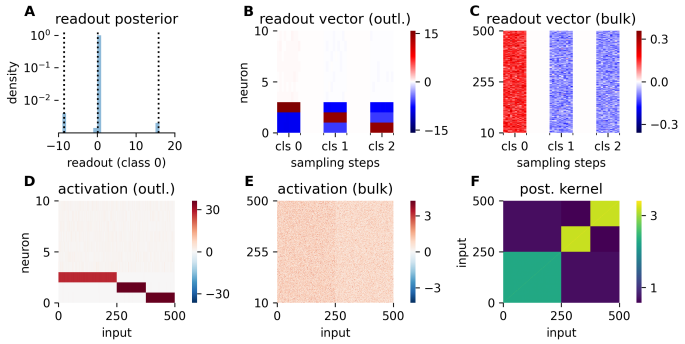


Figure S22. Sparse coding in single-hidden-layer softplus networks on random classification task. (**A**) Readout posterior of first class (blue) and theory (black dashed). (**B**,**C**) Samples of the readout weights of all classes of the most active (B) and the remaining (C) neurons. (**D**,**E**) Activations of the most active (D) and the remaining (E) neurons on all training inputs for a given weight sample. (**F**) Kernel on training data from sampling. Parameters: $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = -1/2$.

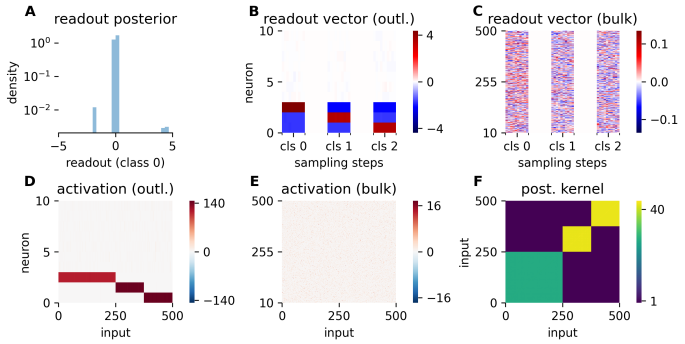


Figure S23. Sparse coding in single-hidden-layer rectified quadratic networks on random classification task. (**A**) Readout posterior of first class (blue) and theory (black dashed). (**B**,**C**) Samples of the readout weights of all classes of the most active (B) and the remaining (C) neurons. (**D**,**E**) Activations of the most active (D) and the remaining (E) neurons on all training inputs for a given weight sample. (**F**) Kernel on training data from sampling. Parameters: $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = -1/2$.

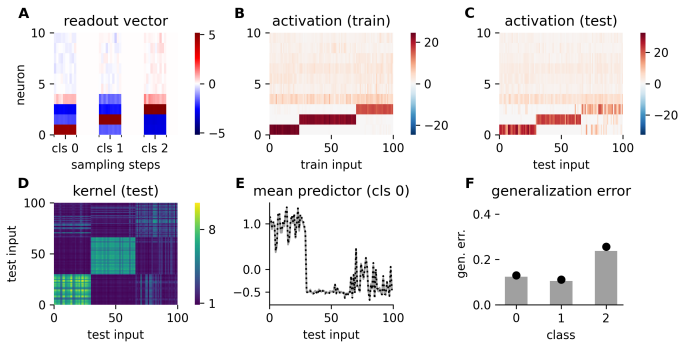


Figure S24. Coding scheme and generalization of single-hidden-layer softplus networks on MNIST. (**A**) Samples of all three readout weights for most active neurons. (**B**,**C**) Activations of most active neurons on all training (B) and 100 test (C) inputs for a given weight sample. (**D**) Kernel on 100 test inputs. (**E**) Mean predictor for class 0 from sampling (gray) and theory (Eq. (C21), black dashed). (**F**) Generalization error for each class averaged over 1,000 test inputs from sampling (gray bars) and theory (Eq. (B38), black circles). Parameters: $N = P = 100$, $N_0 = 784$, classes 0, 1, 2 assigned randomly with probability 1/3, targets $y_+ = 1$ and $y_- = -1/2$.

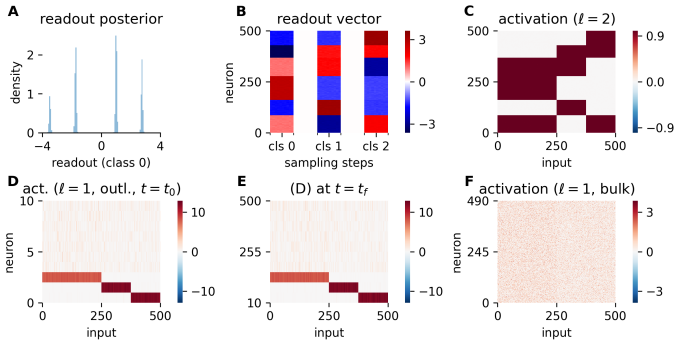


Figure S25. Redundant and sparse coding in two-hidden-layer networks with ReLU activation in the first layer and sigmoid activation in the second layer on random classification task. (**A**) Readout posterior of first class (blue). (**B**) Samples of the readout weights of all classes. (**C**) Activations of all second layer neurons on all training samples using the first weight sample. (**D**,**E**,**F**) Activations of the most active (D,E) and the remaining (F) first layer neurons on all training inputs for the first (D,F) and last (E) weight sample.Parameters: $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = -1/2$.

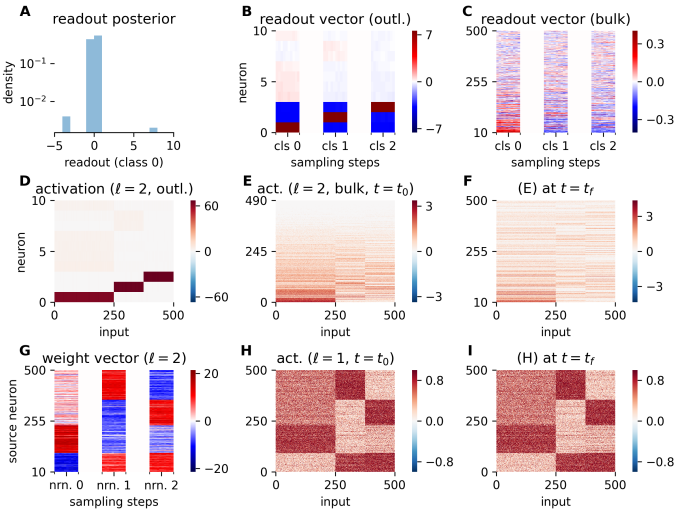


Figure S26. Sparse and redundant coding in two-hidden-layer networks with sigmoidal activation in the first layer and ReLU activation in the second layer on random classification task. (**A**) Readout posterior of first class (blue). (**B**,**C**) Samples of the readout weights of all classes of the most active (B) and the remaining (C) neurons. (**D**,**E**,**F**) Activations of the most active (D) and the remaining (E,F) second layer neurons on all training inputs for the first (D,E) and last (F) weight sample. (**G**) Samples of the weight vectors $W^2_{ij}$ to the most active neurons $i \in \{1, 2, 3\}$ in the second layer. (**H**,**I**) Activations of all first layer neurons on all training inputs using first (D) and last (E) weight sample. Parameters: $N = P = 500$, $N_0 = 520$, classes assigned with fixed ratios $[1/2, 1/4, 1/4]$, targets $y_+ = 1$ and $y_- = -1/2$.

## REFERENCES


[1] A. K. Gupta and D. K. Nagar, *Matrix Variate Distributions*, Monographs and Surveys in Pure and Applied Mathematics (Chapman & Hall/CRC, Philadelphia, PA, 1999).

[2] M. Fedoryuk, The saddle-point method (1977).

[3] M. V. Fedoryuk, Asymptotic methods in analysis, in *Encyclopaedia of Mathematical Sciences* (Springer Berlin Heidelberg, 1989) p. 83–191.

[4] Z. Shun and P. McCullagh, Laplace approximation of high dimensional integrals, Journal of the Royal Statistical Society. Series B (Methodological) **57**, 749 (1995).

[5] V. Spokoiny, Dimension free nonasymptotic bounds on the accuracy of high-dimensional laplace approximation, SIAM/ASA Journal on Uncertainty Quantification **11**, 1044 (2023).

[6] Q. Li and H. Sompolinsky, Statistical Mechanics of Deep Linear Neural Networks: The Backpropagating Kernel Renormalization, Physical Review X **11**, 031059 (2021).

[7] Y. Avidan, Q. Li, and H. Sompolinsky, Connecting ntk and nngp: A unified theoretical framework for neural network learning dynamics in the kernel regime (2023), arXiv:2309.04522.

[8] D. B. Owen, A table of normal integrals, Communications in Statistics - Simulation and Computation **9**, 389 (1980).

[9] A. van Meegen and S. J. van Albada, Microscopic theory of intrinsic timescales in spiking neural networks, Phys. Rev. Res. **3**, 043077 (2021).

[10] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, Array programming with NumPy, Nature **585**, 357–362 (2020).

[11] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python, Nature Methods **17**, 261 (2020).

[12] M. Betancourt, A conceptual introduction to hamiltonian monte carlo (2017).

[13] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson, What are bayesian neural network posteriors really like?, in *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 139, edited by M. Meila and T. Zhang (PMLR, 2021) pp. 4629–4640.

[14] D. Phan, N. Pradhan, and M. Jankowiak, Composable effects for flexible and accelerated probabilistic programming in numpyro, arXiv preprint arXiv:1912.11554 (2019).

[15] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. A. Szerlip, P. Horsfall, and N. D. Goodman, Pyro: Deep universal probabilistic programming, J. Mach. Learn. Res. **20**, 28:1 (2019).

[16] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018).

[17] J. Salvatier, T. V. Wiecki, and C. Fonnesbeck, Probabilistic programming in python using PyMC3, PeerJ Computer Science **2**, e55 (2016).

[18] M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert, Efficient and modular implicit differentiation, arXiv preprint arXiv:2105.15183 (2021).

[19] R. M. Neal, *Bayesian Learning for Neural Networks* (Springer New York, 1996).

[20] C. Williams, Computing with infinite networks, in *Advances in Neural Information Processing Systems*, Vol. 9, edited by M. Mozer, M. Jordan, and T. Petsche (MIT Press, 1996).

[21] J. Lee, J. Sohl-Dickstein, J. Pennington, R. Novak, S. Schoenholz, and Y. Bahri, Deep neural networks as gaussian processes, in *International Conference on Learning Representations* (2018).

[22] A. G. d. G. Matthews, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani, Gaussian process behaviour in wide deep neural networks, in *International Conference on Learning Representations* (2018).

[23] Y. Cho and L. Saul, Kernel methods for deep learning, in *Advances in Neural Information Processing Systems*, Vol. 22, edited by Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (Curran Associates, Inc., 2009).