Structured random receptive fields enable informative sensory encodings

Biraj Pandey[1], Marius Pachitariu[2], Bingni W. Brunton[3], Kameron Decker Harris[4,*]

[1] Department of Applied Mathematics, University of Washington, Seattle, WA 98195, USA

[2] Janelia Research Campus, Howard Hughes Medical Institute, Ashburn, VA 20147, USA

[3] Department of Biology, University of Washington, Seattle, WA 98195, USA

[4] Department of Computer Science, Western Washington University, Bellingham, WA 98225, USA

[*] Corresponding author, kameron.harris@wwu.edu.

May 26, 2022

Abstract

Brains must represent the outside world so that animals survive and thrive. In early sensory systems, neural populations have diverse receptive fields structured to detect important features in inputs, yet significant variability has been ignored in classical models of sensory neurons. We model neuronal receptive fields as random, variable samples from parameterized distributions and demonstrate this model in two sensory modalities using data from insect mechanosensors and mammalian primary visual cortex. Our approach leads to a significant theoretical connection between the foundational concepts of receptive fields and random features, a leading theory for understanding artificial neural networks. The modeled neurons perform a randomized wavelet transform on inputs, which removes high frequency noise and boosts the signal. Further, these random feature neurons enable learning from fewer training samples and with smaller networks in artificial tasks. This structured random model of receptive fields provides a unifying, mathematically tractable framework to understand sensory encodings across both spatial and temporal domains.

Keywords: random features, receptive fields, sensory encoding, Gaussian processes, linear-nonlinear model, artificial neural networks.

1 Introduction

It has long been argued that the brain uses a large population of neurons to represent the world [98, 31, 81, 89]. In this view, sensory stimuli are encoded by the responses of the population, which are then used by downstream areas for diverse tasks, including learning, decision-making, and movement control. These sensory areas have different neurons responding to differing stimuli while also providing a measure of redundancy. However, we still lack a clear understanding of what response properties are well-suited for different sensory modalities.

One way to approach sensory encoding is by understanding how a neuron would respond to arbitrary stimuli. Experimentally, we typically present many stimuli to the animal, measure the responses of sensory neurons, then attempt to estimate some kind of model for how the neurons respond to an arbitrary stimulus. A common assumption is that the neuron computes a linear filter of the stimulus, which then drives spiking through a nonlinear spike-generating mechanism. Mathematically, this assumption can be summarized as the number of measured spikes for a stimulus $\boldsymbol{x}$ being equal to $\sigma(\boldsymbol{w}^T\boldsymbol{x})$ for a weight vector $\boldsymbol{w}$ and nonlinearity σ. Here, the weights $\boldsymbol{w}$ define the filtering properties of the neuron, also known as its *receptive field* [84]. This model is

known as a *linear-nonlinear* (LN) model [18], and it is also the most common form of artificial neuron in artificial neural networks (ANNs). LN models have been used extensively to describe the firing of diverse neurons in various sensory modalities of vertebrates and invertebrates. In the mammalian visual system, LN models have been used to characterize retinal ganglion cells [80], lateral geniculate neurons [20], and simple cells in primary visual cortex (V1) [45]. They have also been used to characterize auditory sensory neurons in the avian midbrain [47] and somatosensory neurons in the cortex [79]. In insects, they have been used to understand the response properties of visual interneurons [77], mechanosensory neurons involved in proprioception [29, 70], and auditory neurons during courtship behavior [21].

Given the stimulus presented and neural response data, one can then estimate the receptive fields of a population of neurons. Simple visual receptive fields have classically been understood as similar to wavelets with particular spatial frequency and angular selectivity [45]. In mechanosensory areas, receptive fields are selective to temporal frequency over a short time window [29]. Commonly, parametric modeling (Gabor wavelets [89]) or smoothing (regularization, etc. [67]) are used to produce "clean" receptive fields. Yet, the data alone show noisy receptive fields that are perhaps best modeled using a random distribution [9]. As we will show, modeling receptive fields as random samples produces realistic receptive fields that reflect both the structure and noisiness seen in experimental data. More importantly, this perspective creates significant theoretical connections between foundational ideas from neuroscience and artificial intelligence. This connection helps us understand why receptive fields have the structures that they do and how this structure relates to the kinds of stimuli that are relevant to the animal.

Modeling the filtering properties of a population of LN neurons as samples from a random distribution leads to the study of networks with random weights [76, 15, 53]. In machine learning (ML), such networks are known as *random feature networks* (RFNs) [12, 41, 72, 54]. The study of RFNs has rapidly gained popularity in recent years, in large part because it offers a theoretically tractable way to study the learning properties of ANNs where the weights are tuned using data [3, 2, 17]. When the RFN contains many neurons, it can approximate functions that live in a well-understood function space. This function space is called a *reproducing kernel Hilbert space* (RKHS), and it depends on the network details, in particular the weight i.e. receptive field distribution [60, 93, 73]. Learning can then be framed as approximating functions in this space from limited data.

Several recent works highlight the RFN theory's usefulness for understanding learning in neural systems. Bordelon, Canatar, and Pehlevan, in a series of papers, have shown that neural codes allow learning from few examples when spectral properties of their second-order statistics aligns with the spectral properties of the task [10, 11, 14]. When applied to V1, they found that the neural code is aligned with tasks that depend on low spatial frequency components. Harris constructed an RFN model of sparse networks found in associative centers like the cerebellum and insect mushroom body and showed that these areas may behave like additive kernels [36], an architecture also considered by Hashemi et al. [37]. These classes of kernels are beneficial for learning in high dimensions because they can learn from fewer examples and remain resilient to input noise or adversarial perturbation. Xie et al. investigated the relationship between the fraction of active neurons in a model of the cerebellum—controlled by neuron thresholds—and generalization performance for learning movement trajectories [94]. In the vast majority of network studies with random weights, these weights $\boldsymbol{w}$ are drawn from a Gaussian distribution with independent entries. This sampling is equivalent to a fully *unstructured* receptive field, which looks like white noise.

Closely related to our work, a previous study of ANNs showed that directly learning structured receptive fields could improve image classification in deep networks [43]. Their receptive fields were parametrized as a sum of Gaussian derivatives up to fourth order. This led to better performance against rival architectures in low data regimes.

In this paper, we study the effect of having *structured yet random* receptive fields and how they lead to informative sensory encodings. Specifically, we consider receptive fields generated by a Gaussian process (GP), which can be thought of as drawing the weights $\boldsymbol{w}$ from a Gaussian distribution with a particular covariance matrix. We show that networks with such random weights project the input to a new basis and filter out particular components. This theory introduces realistic structure of receptive fields into random feature models which are crucial to our current understanding of artificial networks. Next, we show that receptive field datasets from two disparate sensory systems, mechanosensory neurons on insect wings and V1 cortical neurons from mice and monkeys, are well-modeled by GPs with covariance functions that have wavelet eigenbases. Given the success of modeling these data with the GP, we apply these weight distributions in RFNs that are used in synthetic learning tasks. We find that these structured weights improve learning by reducing the number of training examples and the size of the network needed to learn the task. Thus, structured random weights offer a realistic generative model of the receptive fields in multiple sensory areas, which we understand as performing a random change of basis. This change of basis enables the network to represent the most important properties of the stimulus, which we demonstrate to be useful for learning.

2 Results

We construct a generative model for the receptive fields of sensory neurons and use it for the weights of an ANN. We refer to such a network as a *structured* random feature network. In Section 2.1, we review the basics of random feature networks, the details and rationale behind our generative model, and the process by which we generate hidden weights. Our main theory result is that networks with such weights transform the inputs into a new basis and filter out particular components, thus bridging sensory neuroscience and the theory of neural networks. In Section 2.2, we show that neurons in two receptive field datasets—insect mechanosensory neurons and mammalian V1 cortical neurons—are well-described by our generative model. There is a close resemblance between the the second-order statistics, sampled receptive fields, and their principal components for both data and model. Finally, in Section 2.3 we show the performance of structured random feature networks on several synthetic learning tasks. The hidden weights from our generative model allows the network to learn from fewer training examples and smaller network sizes.

2.1 Theoretical analysis

We consider receptive fields generated by GPs in order to connect this foundational concept from sensory neuroscience to the theory of random features in artificial neural networks. GPs can be thought of as samples from a Gaussian distribution with a particular covariance matrix, and we initialize the hidden weights of RFNs using these GPs. We show that using a GP causes the network to project the input into a new basis and filter out particular components. The basis itself is determined by the covariance matrix of the Gaussian, and is useful for removing irrelevant and noisy components from the input. We use these results to study the space of functions that RFNs containing many neurons can learn by connecting our construction to the theory of kernel methods.

2.1.1 Random feature networks

We start by introducing the main learning algorithm and the neuronal model of our work, the RFN. Consider a two-layer, feedforward ANN. Traditionally, all the weights are initialized randomly and learned through backpropagation by minimizing some loss objective. In sharp contrast, RFNs have

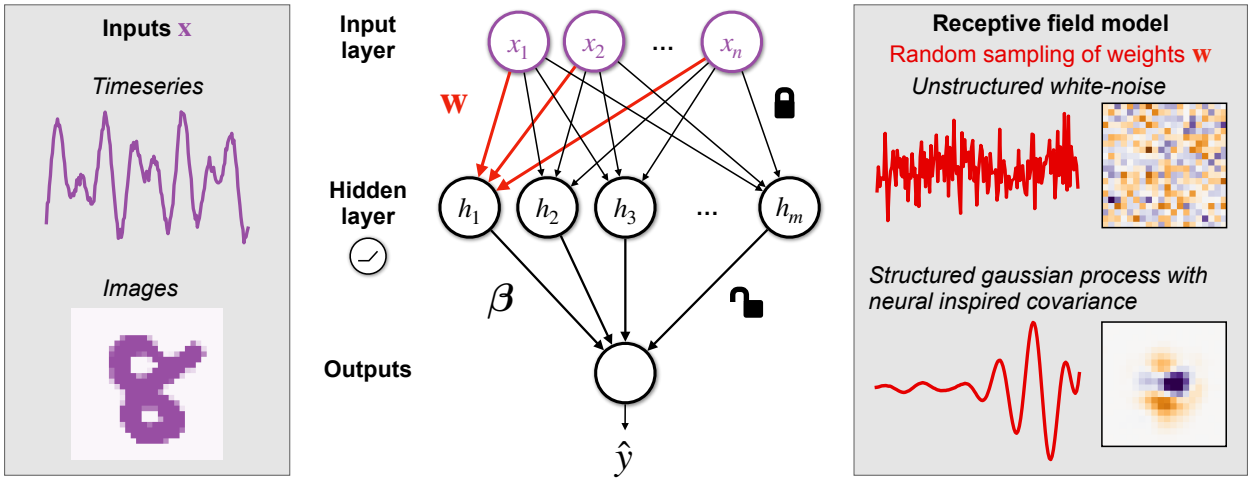

Figure 1: **Random feature networks with structured weights.** We study random feature networks as models for learning in sensory regions. In these networks, each neuron's weight $\boldsymbol{w}$ is fixed as a random sample from some specified distribution. Only the readout weights $\boldsymbol{\beta}$ are trained. In particular, we specify distributions to be Gaussian Processes (GPs) whose covariances are inspired by biological neurons; thus, each realization of the GP resembles a biological receptive field. We build GP models of two sensory areas that specialize in processing timeseries and image inputs. We initialize $\boldsymbol{w}$ from these *structured* GPs and compare them against initialization from *unstructured* white-noise distribution.

their hidden layer weights sampled randomly from some distribution and fixed. Each hidden unit computes a random feature of the input, and only the output layer weights are trained (Fig. 1). Note that the weights are randomly drawn but the neuron's response is a deterministic function of the input given the weights.

Mathematically, we have the hidden layer activations and output given by

$$\boldsymbol{h} = \sigma\left(\boldsymbol{W}\boldsymbol{x}\right), \qquad \hat{y} = \boldsymbol{\beta}^T \boldsymbol{h} + \beta_0, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is the stimulus, $\boldsymbol{h} = [h_1, h_2, \ldots, h_m]^T \in \mathbb{R}^m$ are the hidden neuron responses, and $\hat{y} \in \mathbb{R}$ is the predicted output. We use a rectified linear (ReLU) nonlinearity, $\sigma(x) = \max(0, x)$ applied entrywise in (1). The hidden layer weights $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_m]^T \in \mathbb{R}^{m \times d}$ are drawn randomly and fixed. Only the readout weights β_0 and $\boldsymbol{\beta}$ are trained in RFNs.

In our RFN experiments, we train the readout weights $\boldsymbol{\beta} \in \mathbb{R}^m$ and offset $\beta_0 \in \mathbb{R}$ using a support vector machine (SVM) classifier with squared hinge loss and ℓ^2 penalty with regularization strength of tuned in the range $[10^{-3}, 10^3]$ by 5-fold cross-validation. Our RFNs do not include a threshold for the hidden neurons, although this could help in certain contexts [94].

In the vast majority of studies with RFNs, each neuron's weights $\boldsymbol{w} \in \mathbb{R}^d$ are initialized i.i.d. from a spherical Gaussian distribution $\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{I}_d)$. We will call networks built this way *classical unstructured* RFNs (Fig. 1). We propose a variation where hidden weights are initialized $\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{C})$, where $\boldsymbol{C} \in \mathbb{R}^{d \times d}$ is a positive semidefinite covariance matrix. We call such networks *structured* RFNs (Fig. 1), to mean that the weights are random with a specified covariance. To compare unstructured and structured weights on equal footing, we normalize the covariance matrices so that $\mathrm{Tr}(\boldsymbol{C}) = \mathrm{Tr}(\boldsymbol{I}_d) = d$, which ensures that the mean square amplitude of the weights $\mathbb{E}[\|\boldsymbol{w}\|^2] = d$.

2.1.2 Receptive fields modeled by linear weights

Sensory neurons respond preferentially to specific features of their inputs. This stimulus selectivity is often summarized as a neuron's receptive field, which describes how features of how the sensory space elicits responses when stimulated [84]. Mathematically, receptive fields are modeled as a linear filter in the stimulus space. Linear filters are also an integral component of the widely used LN model of sensory processing [18]. According to this model, the firing rate of a neuron is a nonlinear function applied to the projection of the stimulus onto the low-dimensional subspace of the linear filter.

A linear filter model of receptive fields can explain responses of individual neurons to diverse stimuli. It has been used to describe disparate sensory systems like visual, auditory, and somatosensory systems of diverse species including birds, mammals, and insects [80, 77, 47, 21, 79]. If the stimuli are uncorrelated, the filters can be estimated by computing the spike triggered average (STA), the average stimulus that elicited a spike for the neuron. When the stimuli are correlated, the STA filter is whitened by the inverse of the stimulus covariance matrix [66]. Often these STAs are denoised by fitting a parametric function to the STA [18], such as Gabor wavelets for simple cells in V1 [45].

We model the receptive field of a neuron i as its weight vector $\boldsymbol{w}_i$ and its nonlinear function as σ. Instead of fitting a parametric function, we construct covariance functions so that each realization of the resulting Gaussian process resembles a biological receptive field (Fig. 1).

2.1.3 Structured weights project and filter input into the covariance eigenbasis

We generate network weights from Gaussian processes (GP) whose covariance functions are inspired by the receptive fields of sensory neurons in the brain. By definition, a GP is a stochastic process where finite observations follow a Gaussian distribution [74]. We find that networks with such weights project inputs into a new basis and filter out irrelevant components. In Section 2.3, we will see that this adds an inductive bias to classical RFNs for tasks with naturalistic inputs and improves learning.

We view our weight vector $\boldsymbol{w}$ as the finite-dimensional discretization of a continuous function $w(t)$ which is a sample from a GP. The continuous function has domain T, a compact subset of $\mathbb{R}^D$, and we assume that T is discretized using a grid of d equally spaced points $\{t_1, \ldots, t_d\} \subset T$, so that $w_i = w(t_i)$. Let the input be a real-valued function $x(t)$ over the same domain T, which could represent a finite timeseries ($D = 1$), an image of luminance on the retina ($D = 2$), or more complicated spatiotemporal sets like a movie ($D = 3$). In the continuous setting, the d-dimensional ℓ^2 inner product $\boldsymbol{w}^T\boldsymbol{x} = \sum_{i=1}^d w_i x_i$ gets replaced by the $L^2(T)$ inner product $\langle w, x\rangle = \int_{t\in T} w(t)x(t)\mathrm{d}t$.

Every GP is fully specified by its mean and covariance function $C(t, t')$. We will always assume that the mean is zero and study different covariance functions. By the Kosambi-Karhunen–Loève theorem [49], each realization of a zero-mean GP has a random series representation

$$w(t) = \sum_{i=1}^{\infty} z_i \lambda_i \phi_i(t), \tag{2}$$

in terms of standard Gaussian random variables $z_i \sim \mathcal{N}(0, 1)$, functions $\phi_i(t)$, and weights $\lambda_i \geq 0$. The pairs (λ_i^2, ϕ_i) are eigenvalue, eigenfunction pairs of the covariance operator $\mathcal{C} : L^2(T) \to L^2(T)$,

$$(\mathcal{C}f)(t) = \int_{t\in T} C(t, t')f(t')\mathrm{d}t',$$

which is the continuous analog of the covariance matrix $\boldsymbol{C}$. If $C(t,t')$ is positive definite, as opposed to just semidefinite, all $\lambda_i^2 > 0$ and these eigenfunctions ϕ_i form a complete basis for $L^2(T)$. Using (2), the inner product between a stimulus and a neuron's weights is

$$\langle w, x\rangle = \left\langle \sum_{i=1}^{\infty} z_i \lambda_i \phi_i, x \right\rangle = \sum_{i=1}^{\infty} z_i \lambda_i \langle \phi_i, x\rangle = \sum_{i=1}^{\infty} z_i \tilde{x}_i, \quad \text{where } \tilde{x}_i = \lambda_i \langle \phi_i, x\rangle. \tag{3}$$

Equation (3) shows that the structured weights compute a *projection* of the input x onto each eigenfunction $\langle \phi_i, x\rangle$ and reweight or *filter* by the eigenvalue λ_i before taking the ℓ^2 inner product with the random Gaussian weights z_i.

It is illuminating to see what these continuous equations look like in the d-dimensional discrete setting. Samples from the finite-dimensional GP are used as the hidden weights in RFNs, $\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{C})$. First, the GP series representation (2) becomes $\boldsymbol{w} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{z}$, where $\boldsymbol{\Lambda}$ and $\boldsymbol{\Phi}$ are matrices of eigenvalues and eigenvectors, and $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}_d)$ is a Gaussian random vector. By the definition of the covariance matrix, $\boldsymbol{C} = \mathbb{E}[\boldsymbol{w}\boldsymbol{w}^T]$, which is equal to $\boldsymbol{\Phi}\boldsymbol{\Lambda}^2\boldsymbol{\Phi}^T$ after a few steps of linear algebra. Finally, (3) is analogous to $\boldsymbol{w}^T\boldsymbol{x} = \boldsymbol{z}^T\boldsymbol{\Lambda}\boldsymbol{\Phi}^T\boldsymbol{x}$. Since $\boldsymbol{\Phi}$ is an orthogonal matrix, $\boldsymbol{\Phi}^T\boldsymbol{x}$ is equivalent to a change of basis, and the diagonal matrix $\boldsymbol{\Lambda}$ shrinks or expands certain directions to perform filtering. This can be summarized in the following theorem:

Theorem 1 (Basis change formula) *Assume $\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{C})$ with $\boldsymbol{C} = \boldsymbol{\Phi}\boldsymbol{\Lambda}^2\boldsymbol{\Phi}^T$ its eigenvalue decomposition. For $\boldsymbol{x} \in \mathbb{R}^d$, define*

$$\tilde{\boldsymbol{x}} := \boldsymbol{\Lambda}\boldsymbol{\Phi}^T\boldsymbol{x}. \tag{4}$$

Then $\boldsymbol{w}^T\boldsymbol{x} = \boldsymbol{z}^T\tilde{\boldsymbol{x}}$ for $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I}_d)$.

Theorem 1 says that projecting an input onto a structured weight vector is the same as first filtering that input in the GP eigenbasis and doing a random projection onto a spherical random Gaussian. The form of the GP eigenbasis is determined by the choice of the covariance function. If the covariance function is compatible with the input structure, the hidden weights filter out any irrelevant features or noise in the stimuli while amplifying the descriptive features. This inductive bias facilitates inference on the stimuli by any downstream predictor. Because the spherical Gaussian distribution is the canonical choice for unstructured RFNs, there is a simple way to evaluate the effective kernel of structured RFNs as $k_{\text{struct}}(\boldsymbol{x}, \boldsymbol{x}') = k_{\text{unstruct}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}')$ (see Appendix A.1).

Our expression for the structured kernel provides a concrete connection to the kernel theory of learning using nonlinear neural networks. For readers interested in such kernel theories, a full example of how these work is given in Appendix A.2. There we show that there can be an exponential reduction in the number of samples needed to learn frequency detection using a structured versus unstructured basis.

2.2 Examples of random yet structured receptive fields

Our goal is to model the weights of artificial neurons in a way that is inspired by biological neurons' receptive fields. Structured RFNs sample hidden weights from GPs with structured covariance, so we construct covariance functions that make the generated weights resemble neuronal receptive fields. We start with a toy example of a stationary GP with well-understood Fourier eigenbasis and show how the receptive fields generated from this GP are selective to frequencies in timeseries signals. Then, we construct locally stationary covariance models of the of insect mechanosensory and V1 neuron receptive fields. These models are shown to be a good match for experimental data.

2.2.1 Warm-up: frequency selectivity from stationary covariance

To illustrate some results from our theoretical analysis, we start with a toy example of temporal receptive fields that are selective to particular frequencies. This example may be familiar to readers comfortable with Fourier series and basic signal processing. Let the input be a finite continuous timeseries $x(t)$ over the interval $T = [0, L]$. We use the covariance function

$$C(t, t') = \overbrace{\sum_{k=0}^{\infty} \lambda_k^2 \cos\left(\omega_k (t - t')\right)}^{\text{stationary process}}, \tag{5}$$

where $\omega_k = 2\pi k/L$ is the kth natural frequency and λ_k^2 are the weight coefficients. The covariance function (5) is *stationary*, which means that it only depends on the difference between the timepoints $t - t'$. Applying the compound angle formula, we get

$$C(t, t') = \sum_{k=0}^{\infty} \lambda_k^2 \left(\cos(\omega_k t)\cos(\omega_k t') + \sin(\omega_k t)\sin(\omega_k t')\right). \tag{6}$$

Since the sinusoidal functions $\cos(\omega_k t)$ and $\sin(\omega_k t)$ form an orthonormal basis for $L^2(T)$, (6) is the eigendecomposition of the covariance, where the eigenfunctions are sines and cosines with eigenvalues λ_k^2. From (2), we know that structured weights with this covariance form a random series:

$$w(t) = \sum_{k=0}^{\infty} \lambda_k \left(z_k \cos(\omega_k t) + z_k' \sin(\omega_k t)\right), \tag{7}$$

where each $z_k, z_k' \sim \mathcal{N}(0, 1)$. Thus, the receptive fields are made up of sinusoids weighted by λ_k and the Gaussian variables z_k, z_k'.

Suppose we want receptive fields that only retain specific frequency information of the signal and filter out the rest. Take $\lambda_k = 0$ for any k where $\omega_k < f_{\text{lo}}$ or $\omega_k > f_{\text{hi}}$. We call this a *bandlimited* spectrum with passband $[f_{\text{lo}}, f_{\text{hi}}]$ and bandwidth $f_{\text{hi}} - f_{\text{lo}}$. As the bandwidth increases, the receptive fields become less smooth since they are made up of a wider range of frequencies. If the λ_k are all nonzero but decay at a certain rate, this rate controls the smoothness of the resulting GP [91].

When these receptive fields act on input signals $x(t)$, they implicitly transform the inputs into the Fourier basis and filter frequencies based on the magnitude of λ_k. In a bandlimited setting, any frequencies outside the passband are filtered out, which makes the receptive fields selective to a particular range of frequencies and ignore others. On the other hand, classical random features weight all frequencies equally, even though in natural settings high frequency signals are the most corrupted by noise.

2.2.2 Insect mechanosensors

We next consider a particular biological sensor that is sensitive to the time-history of forces. Campaniform sensilla (CS) are dome-shaped mechanoreceptors that detect local stress and strain on the insect exoskeleton [23]. They are embedded in the cuticle and deformation of the cuticle through bending or torsion induces depolarizing currents in the CS by opening mechanosensitive ion channels. The CS encode proprioceptive information useful for body state estimation and movement control during diverse tasks like walking, kicking, jumping, and flying [23].

We will model the receptive fields of CS that are believed to be critical for flight control, namely the ones found at the base of the halteres [97] and on the wings [70] (Fig. 2A). Halteres and wings

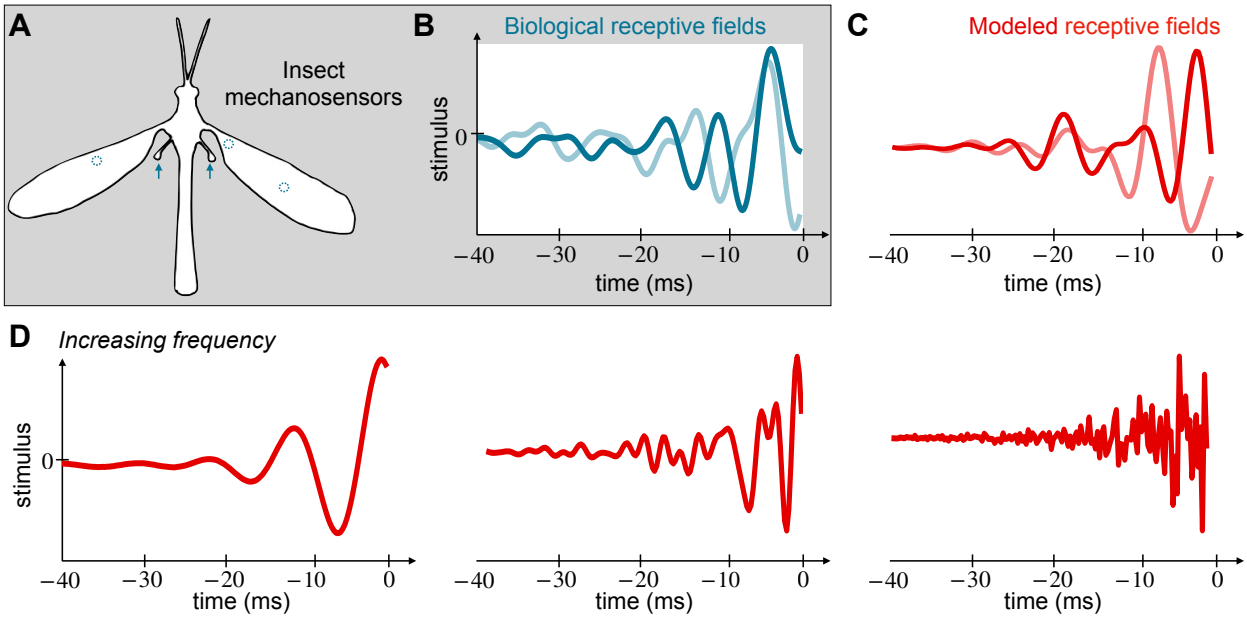

Figure 2: **Random receptive field model of insect mechanosensors.** (A) Diagram of the the cranefly, *Tipula hespera*. Locations of the mechanosensors, campaniform sensilla, are marked in blue on the wings and halteres. (B) Two receptive fields of campaniform sensilla are shown in blue. They are smooth, oscillatory, and decay over time. We model them as random samples from distributions parameterized by frequency and decay parameters. Data are from the hawkmoth [70]; cranefly sensilla have similar responses [29]. (C) Two random samples from the model distribution are shown in red. (D) The smoothness of the receptive fields is controlled by the frequency parameter. The decay parameter controls the rate of decay from the origin (not shown).

flap rhythmically during flight, and rotations of the insect's body induce torsional forces that can be felt on these active sensory structures. The CS detect these small strain forces, thereby encoding angular velocity of the insect body [97]. Experimental results show haltere and wing CS are selective to a broad range of oscillatory frequencies [28, 70], with STAs that are smooth, oscillatory, selective to frequency, and decay over time [29] (Fig. 2B).

We model these temporal receptive fields with a locally stationary GP [33] with bandlimited spectrum. Examples of receptive fields generated from this GP are shown in Fig. 2C. The inputs to the CS are modeled as a finite continuous timeseries $x(t)$ over the finite interval $T = [0, L]$. The neuron weights are generated from a covariance function

$$C(t,t') = \overbrace{\exp\left(-\frac{(t+t')}{\gamma}\right)}^{\text{localized}} \overbrace{\sum_{k=0}^{\infty} \lambda_k^2 \cos\left(\omega_k (t-t')\right)}^{\text{stationary process}}, \qquad \lambda_k = \overbrace{\begin{cases} 1 & f_{\text{lo}} \le \omega_k \le f_{\text{hi}} \\ 0 & \text{otherwise} \end{cases}}^{\text{bandlimited, flat-power spectrum}}, \tag{8}$$

where $\omega_k = 2\pi k/L$ is the kth natural frequency. As in Section 2.2.1, the frequency selectivity of their weights is accounted for by the parameters f_{lo} and f_{hi}. As the bandwidth $f_{\text{hi}} - f_{\text{lo}}$ increases, the receptive fields are built out of a wider selection of frequencies. This makes the receptive fields less smooth (Fig. 2D). Each field is localized to near $t = 0$, and its decay with t is determined by the parameter γ. As γ increases, the receptive field is selective to larger time windows.

The eigenbasis of the covariance function (8) is similar to a Fourier eigenbasis modulated by a decaying exponential. The eigenbasis is an orthonormal basis for the span of $\lambda_k e^{-t/\gamma} \cos(\omega_k t)$ and

$\lambda_k e^{-t/\gamma} \sin(\omega_k t)$, which are a non-orthogonal set of functions in $L^2(T)$. The hidden weights transform timeseries inputs into this eigenbasis and discard components outside the passband frequencies $[f_{\text{lo}}, f_{\text{hi}}]$.

We fit the covariance model to receptive field data from 95 CS neurons from wings of the hawkmoth *Manduca sexta* (data from [70]). Briefly, CS receptive fields were estimated as the spike-triggered average (STA) of experimental mechanical stimuli of the wings, where the stimuli were generated as bandpassed white noise (2–300 Hz).

To characterize the receptive fields of this population of CS neurons, we compute the data covariance matrix $\boldsymbol{C}_{\text{data}}$ by taking the inner product between the receptive fields. We normalize the trace to be the dimension of each receptive field (number of samples), which in this case is 40 kHz $\times$ 40 ms = 1600 samples. This normalization sets the overall scale of the covariance matrix. The data covariance matrix shows a tridiagonal structure (Fig. 3A). The main diagonal is positive while the off diagonals are negative. All diagonals decay away from the top left of the matrix.

To fit the covariance model to the data, we optimize the parameters (see Appendix A.3) $f_{\text{lo}}, f_{\text{hi}}$, and γ, finding $f_{\text{lo}} = 75$ Hz, $f_{\text{hi}} = 200$ Hz, and $\gamma = 12.17$ ms best fit the sensilla data. We do so by minimizing the Frobenius norm of the difference between $\boldsymbol{C}_{\text{data}}$ and the model (26). The resulting model covariance matrix (Fig. 3B) matches the data covariance matrix (Fig. 3A) remarkably well qualitatively. The normalized Frobenius norm of the difference between $\boldsymbol{C}_{\text{data}}$ and the model is 0.4386. Examples of biological receptive fields and random samples from this fitted covariance model are shown in the Appendix (Fig. 24). To simulate the effect of a finite number of neurons, we generate 95 weight vectors (equal to the number of neurons recorded) and recompute the model covariance matrix (Fig. 3C). We call this the finite neuron model covariance matrix $\boldsymbol{C}_{\text{finite}}$, and it shows the bump and blob-like structures evident in $\boldsymbol{C}_{\text{data}}$ but not in $\boldsymbol{C}_{\text{model}}$. This result suggests that these bumpy structures can be attributed to having a small number of recorded neurons. We hypothesize that these effects would disappear with a larger dataset and $\boldsymbol{C}_{\text{data}}$ would more closely resemble $\boldsymbol{C}_{\text{model}}$.

For comparison, we also calculate the Frobenius difference for null models, the unstructured covariance model and the Fourier model (5). For the unstructured model, the Frobenius norm difference is 0.9986 while that of the Fourier model is 0.9123. The sensilla covariance model has a much lower difference (0.4386) compared to the null models, fitting the data more accurately. We show the covariance matrices and sampled receptive fields from these null models in the Appendix A.4.1.

Comparing the eigenvectors and eigenvalues of the data and model covariance matrices, we find that the spectral properties of both $\boldsymbol{C}_{\text{model}}$ and $\boldsymbol{C}_{\text{finite}}$ are similar to that of $\boldsymbol{C}_{\text{data}}$. The eigenvalue curves of the models match that of the data quite well (Fig. 3E); these curves are directly comparable because each covariance is normalized by its trace, which makes the sum of the eigenvalues unity. Further, all of the data and the model covariance matrices are low-dimensional. The first 10 data eigenvectors explain 97% of the variance, and the top 5 explain 90%. The top 5 eigenvectors of the model and its finite sample match that of the data quite well (Fig. 3D).

2.2.3 Primary visual cortex

We now turn to visually driven neurons from the mammalian primary cortex. Primary visual cortex (V1) is the earliest cortical area for processing visual information (Fig. 4A). The neurons in V1 can detect small changes in visual features like orientations, spatial frequencies, contrast, and size.

Here, we model the receptive fields of simple cells in V1, which have clear excitatory and inhibitory regions such that light shone on the excitatory regions increase the cell's response and vice-versa (Fig. 4B). The shape of the regions determines the orientation selectivity, while their

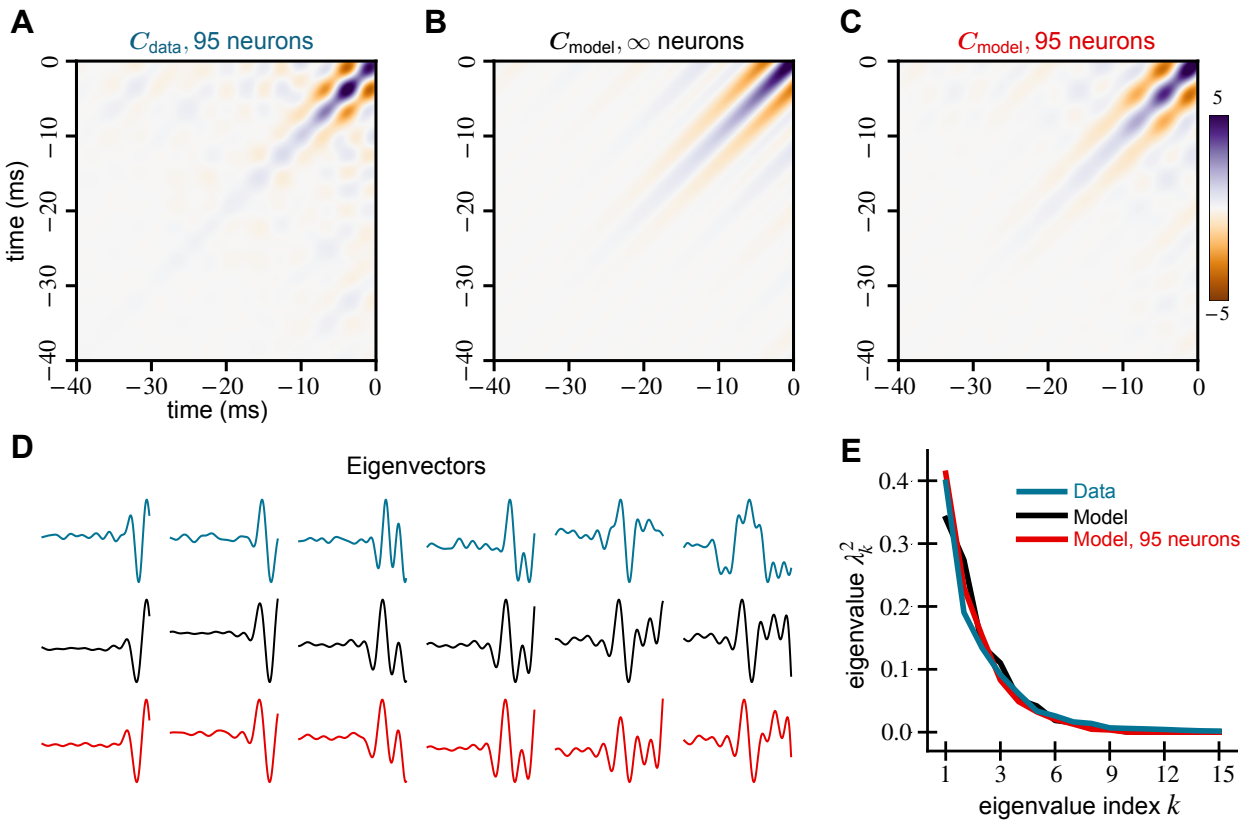

Figure 3: **Spectral properties of mechanosensory RFs and our model are similar.** We compare the covariance matrices generated from (A) receptive fields of 95 mechanosensors from [70], (B) the model (8), and (C) 95 random samples from the same model. All covariance matrices show a tri-diagonal structure that decays away from the origin. (D) The first five principal components of all three covariance matrices are similar and explain 90% of the variance in the RF data. (E) The leading eigenvalue spectra of the data and models show similar behavior.

widths determine the frequency selectivity. The receptive fields are centered to a location in the visual field and decay away from it. They integrate visual stimuli within a small region of this center [40]. Gabor functions are widely used as a mathematical model of the receptive fields of simple cells [45].

We model these receptive fields using another locally stationary GP [33] and show examples of generated receptive fields in Fig. 4C. Consider the inputs to the cortical cells to be a continuous two-dimensional image $x(\boldsymbol{t})$, where the domain $T = [0, L] \times [0, L']$ and $x : T \to \mathbb{R}$. Since the image is real-valued, $x(\boldsymbol{t})$ is the grayscale contrast or single color channel pixel values. The neuron weights are then generated from a covariance function of the following form:

$$C(\boldsymbol{t}, \boldsymbol{t}') = \overbrace{\exp\left(-\frac{\|\boldsymbol{t} - \boldsymbol{t}'\|^2}{2f^2}\right)}^{\text{smooth receptive fields}} \cdot \overbrace{\exp\left(-\frac{\|\boldsymbol{t} - \boldsymbol{c}\|^2 + \|\boldsymbol{t}' - \boldsymbol{c}\|^2}{2s^2}\right)}^{\text{localized to a center } c}. \tag{9}$$

The receptive field center is defined by $\boldsymbol{c}$, and the size of the receptive field is determined by the parameter s. As s increases, the receptive field extends farther from the center $\boldsymbol{c}$ (Fig. 4D). Spatial frequency selectivity is accounted for by the bandwidth parameter f. As f decreases, the spatial frequency of the receptive field goes up, making the weights less smooth (Fig. 4E).

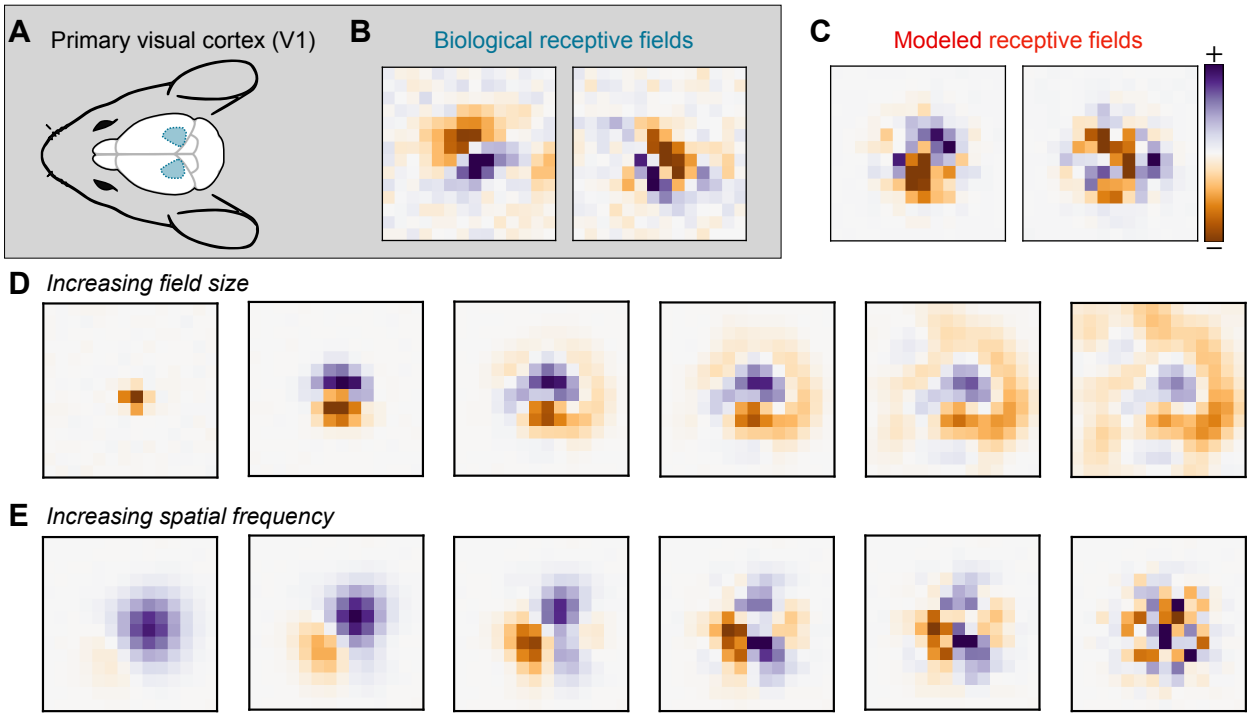

Figure 4: **Random receptive field model of Primary Visual Cortex (V1).** (A) Diagram of the mouse brain with V1 shown in blue. (B) Receptive fields of two mouse V1 neurons calculated from their response to white noise stimuli. The fields are localized to a region in a visual field and show "on" and "off" regions. (C) Random samples from the model (9) distribution. (D) Increasing the receptive field size parameter in our model leads to larger fields. (E) Increasing the model spatial frequency parameter leads to more variable fields.

The eigendecomposition of the covariance function (9) leads to an orthonormal basis of single scale *Hermite wavelets* [56, 57]. When $\boldsymbol{c} = 0$, the wavelet eigenfunctions are Hermite polynomials modulated by a decaying Gaussian:

$$\phi_{\boldsymbol{k}}(\boldsymbol{t}) \propto \prod_{i=1}^{D} e^{-c_1 {t_i}^2} H_{k_i}(c_2 t_i) \quad \text{and} \quad \lambda_{\boldsymbol{k}}^2 \propto \prod_{i=1}^{D} c_3^{k_i}, \tag{10}$$

where H_k is the kth (physicist's) Hermite polynomial; eigenfunctions for nonzero centers $\boldsymbol{c}$ are just shifted versions of (10). The detailed derivation and values of the constants c_1, c_2, c_3 and normalization are in Appendix A.5.

We use (9) to model receptive field data from 8,358 V1 neurons recorded with calcium imaging from transgenic mice expressing GCaMP6s; the mice were headfixed and running on an air-floating ball. We presented 24,357 unique white noise images of 14×36 pixels using the Psychtoolbox [46], where the pixels were white or black with equal probability. Images were upsampled to the resolution of the screens via bilinear interpolation. The stimulus was corrected for eye-movements online using custom code. The responses of 45,026 cells were collected using a two-photon mesoscope [86] and preprocessed using Suite2p [65]. Receptive fields were calculated from the white noise images and the deconvolved calcium responses of the cells using the STA. For the covariance analysis, we picked cells above the signal-to-noise (SNR) threshold of 0.4; this gave us 8,358 cells. The SNR was calculated from a smaller set of 2,435 images that were presented twice using the

method from [89]. As a preprocessing step, we moved the center of mass of every receptive field to the center of the visual field.

We compute the data covariance matrix $\boldsymbol{C}_{\text{data}}$ by taking the inner product between the receptive fields. We normalize the trace to be the dimension of each receptive field, which in this case is (14 $\times$ 36) pixels = 504 pixels. The data covariance matrix resembles a tridiagonal matrix. However, the diagonals are non-zero only at equally spaced segments. Additionally, their values decay away from the center of the matrix. We show $\boldsymbol{C}_{\text{data}}$ zoomed in at the non-zero region around the center of the matrix (Fig. 5A); this corresponds to the 180×180 pixel region around the center of the full 504×504 pixel matrix. The full covariance matrix is shown in the Appendix A.10 (Fig. 22).

In the covariance model, the number of off-diagonals, the center, the rate of their decay away from the center are determined by the parameters f, s and $\boldsymbol{c}$ respectively. The covariance between pixels decays as a function of their distance from $\boldsymbol{c}$. This leads to the the equally-spaced non-zero segments. On the other hand, the covariance also decays as a function of the distance between pixels. This brings the the diagonal structure to the model. When the frequency parameter f increases, the number of off-diagonals increases. Pixels in the generated weights become more correlated and the weights become spatially smoother. When the size parameter s increases, the diagonals decay slower from the center $\boldsymbol{c}$, increasing correlations with the center pixel and leading the significant weights to occupy more of the visual field.

We again optimize the parameters to fit the data (Appendix A.3.2), finding $s = 1.87$ and $f = 0.70$ pixels. We do so by minimizing the Frobenius norm of the difference between $\boldsymbol{C}_{\text{data}}$ and the model. We do not need to optimize over the center parameter $\boldsymbol{c}$, since we preprocess the data so that all receptive fields are centered at $\boldsymbol{c} = (7, 18)$, the center of the 14×36 grid. The resulting model covariance matrix (Fig. 5B) and the data covariance matrix (Fig. 5A) match remarkably well qualitatively. The normalized Frobenius norm of the difference between $\boldsymbol{C}_{\text{data}}$ and the model is 0.2993. Examples of biological receptive fields and random samples from the fitted covariance model are shown in Fig. 23 in the Appendix. To simulate the effect of a finite number of neurons, we generate 8,358 weights, equal to the number of neurons in our data, to compute $\boldsymbol{C}_{\text{finite}}$ shown in Fig. 5C. This finite matrix $\boldsymbol{C}_{\text{finite}}$ looks even more like $\boldsymbol{C}_{\text{data}}$, and it shows that some of the negative covariances far from center result from finite sample size but not all.

For comparison, we also calculate the normalized Frobenius difference for null models, the unstructured covariance model and the translation invariant V1 model (28). In the translation invariant model, we remove the spatially localizing exponential in Eq. 9 and only fit the spatial frequency parameter, f. For the unstructured model, the Frobenius norm difference is 0.9835 while that of the translation invariant model is 0.9727. The V1 covariance model has a much lower difference (0.2993) and is a better fit to the data. We show the covariance matrices and sampled receptive fields from these null models in the Appendix A.4.2.

Similar spectral properties are evident in the eigenvectors and eigenvalues of $\boldsymbol{C}_{\text{model}}$, $\boldsymbol{C}_{\text{finite}}$, $\boldsymbol{C}_{\text{data}}$, and the analytical forms derived in (10) (Fig. 5D,E). As in Section 2.2.2, the covariances are normalized to have unit trace. Note that the analytical eigenfunctions are shown on a finer grid than the model and data because the analysis was performed in continuous space. The differences between the eigenfunctions and eigenvalues of the analytical and model results are due to discretization. Examining the eigenvectors (Fig. 5D), we also see a good match, although there are some rotations and differences in ordering. These 10 eigenvectors explain 68% of the variance in the receptive field data. For reference, the top 80 eigenvectors explain 86% of the variance in the data and all of the variance in the model. The eigenvalue curves of both the models and the analytical forms match that of the data (Fig. 5E) reasonably well, although not as well as for the mechanosensors. In Appendix A.10, we repeat this analysis for receptive fields measured with different stimulus sets in the mouse and different experimental dataset from non-human primate

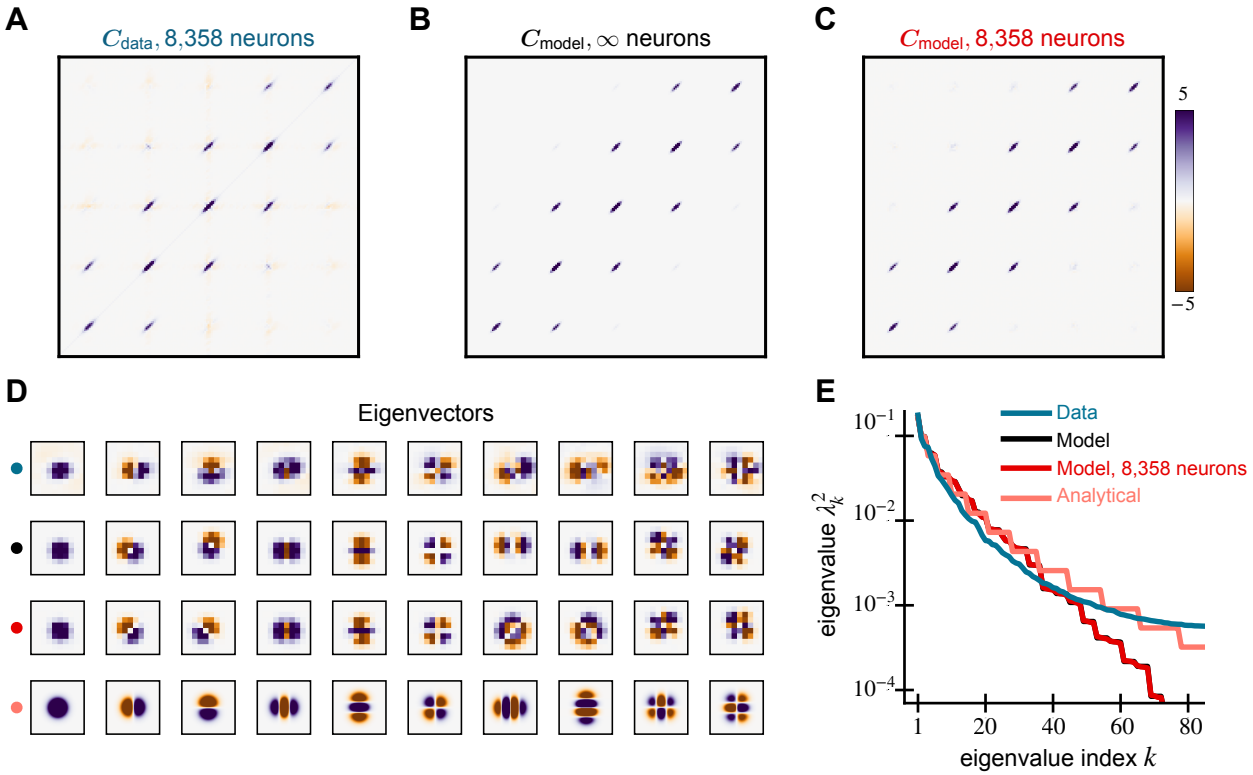

Figure 5: **Spectral properties of V1 RFs and our model are similar.** We compare the covariance matrices generated from the (A) receptive fields of 8,358 mouse V1 neurons, (B) the GP model (9), and (C) 8,358 random samples from the model. These resemble a tri-diagonal matrix whose diagonals are non-zero at equally-spaced segments. (D) The leading 10 eigenvectors of the data and model covariance matrices show similar structure and explain 68% of the variance in the data. Analytical Hermite wavelet eigenfunctions are in the last row and differ from the model due to discretization (both cases) and finite sampling (8,358 neurons only). (E) The eigenspectrum of the model matches well with the data. The staircase pattern in the model comes from repeated eigenvalues at each frequency. The model curve with infinite neurons (black) is obscured by the model curve with 8,358 neurons (red).

V1. Our findings are consistent with the results shown above.

2.3 Advantages of structured random weights for artificial learning tasks

Our hypothesis is that neuronal inductive bias from structured receptive fields allows networks to learn with fewer neurons, training examples, and steps of gradient descent for classification tasks with naturalistic inputs. To examine this hypothesis, we compare the performance of structured receptive fields against classical ones on several classification tasks. We find that, for most artificial learning tasks, structured random networks learn more accurately from smaller network sizes, fewer training examples, and gradient steps.

2.3.1 Frequency detection

CS naturally encode the time-history of strain forces acting on the insect body and sensors inspired by their temporal filtering properties have been shown to accurately classify spatiotemporal data

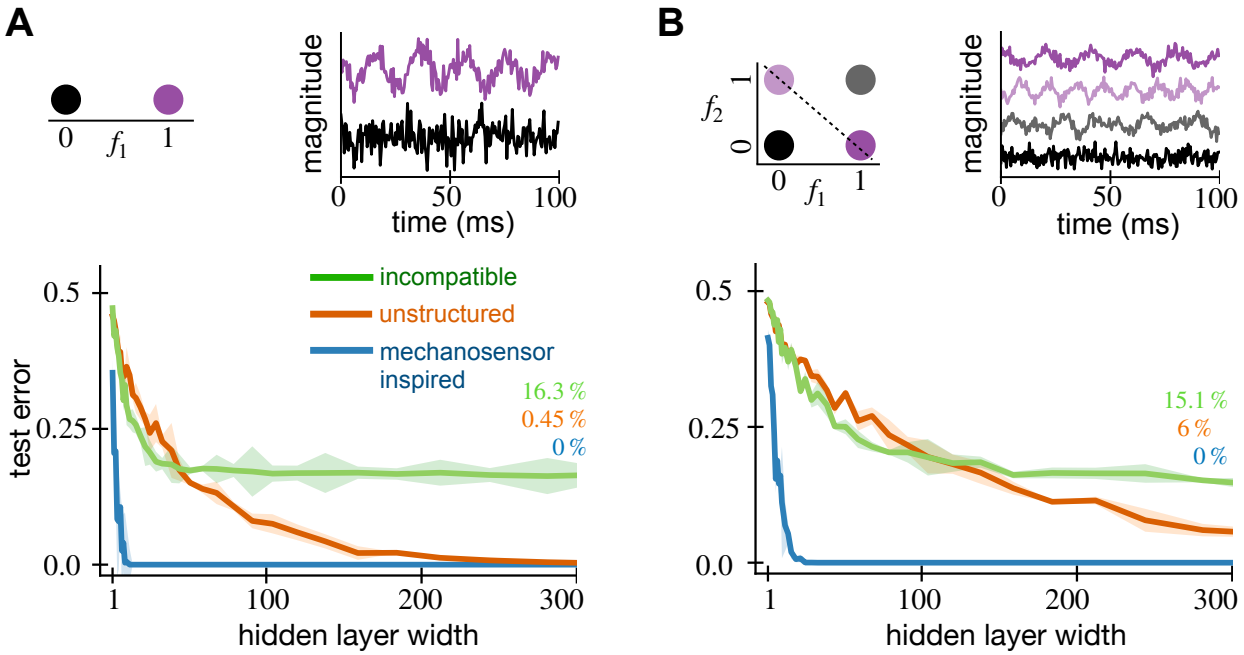

Figure 6: **Random mechanosensory weights enable learning with fewer neurons in timeseries classification tasks.** We show the test error of random feature networks with both mechanosensory and classical white-noise weights against the number of neurons in their hidden layer. For every hidden layer width, we generate five random networks and average their test error. In the error curves, the solid lines show the average test error while the shaded regions represent the standard error across five generations of the random network. The top row shows the timeseries tasks that the networks are tested on. (A, top) In the frequency detection task, a $f_1 = 50$ Hz frequency signal (purple) is separated from white noise (black). (B, top) In the frequency XOR task, $f_1 = 50$ Hz (purple) and $f_2 = 80$ Hz (light purple) signals are separated from white noise (black) and mixtures of 50 Hz and 80 Hz (gray). When their covariance parameters are tuned properly, mechanosensor-inspired networks achieve lower error using fewer hidden neurons on both frequency detection (A, bottom) and frequency XOR (B, bottom) tasks. However, the performance of bio-inspired networks suffer if their weights are incompatible with the task.

[59]. Inspired by this result, we test sensilla-inspired mechanosensory receptive fields from Section 2.2.2 on a timeseries classification task (Fig. 6A, top). Each example presented to the network is a 100 ms timeseries sampled at 2 kHz so that $d = 200$, and the goal is to detect whether or not each example contains a sinusoidal signal. The positive examples are sinusoidal signals with $f_1 = 50$ Hz and corrupted by noise so that their SNR $= 1.76$ (2.46 dB). The negative examples are Gaussian white noise with matched amplitude to the positive examples. Note that this frequency detection task is not linearly separable because of the random phases in positive and negative examples. See Section A.7 for additional details including the definition of SNR and how cross-validation was used to find the optimal parameters $f_{\text{lo}} = 10$ Hz, $f_{\text{hi}} = 60$ Hz, and $\gamma = 50$ ms.

For the same number of hidden neurons, the structured RFN significantly outperforms a classical RFN. We show test performance using these tuned parameters in Fig. 6A. Even in this noisy task, it achieves 0.5% test error using only 25 hidden neurons. Meanwhile, the classical network takes 300 neurons to achieve similar error.

Predictably, the performance suffers when the weights are *incompatible* with the task. We show

results when $f_{\text{lo}} = 10$ Hz and $f_{\text{hi}} = 40$ Hz and the same γ (Fig. 6A). The incompatible RFN performs better than chance (50% error) but much worse than the classical RFN. It takes 300 neurons just to achieve 16.3% test error. The test error does not decrease below this level even with additional hidden neurons.

2.3.2 Frequency XOR task

To challenge the mechanosensor-inspired networks on a more difficult task, we build a frequency Exclusive-OR (XOR) problem (Fig. 6B, top). XOR is a binary function which returns true if and only if the both inputs are different, otherwise it returns false. XOR is a classical example of a function that is not linearly separable and thus harder to learn. Our inputs are again 100 ms timeseries sampled at 2 kHz. The inputs either contain a pure frequency of $f_1 = 50$ Hz or $f_2 = 80$ Hz, mixed frequency signals with both f_1 and f_2, or white noise. In both the pure and mixed frequency cases, we add noise so that the SNR = 1.76. See A.7 for details. The goal of the task is to output true if the input contains either pure tone and false if the input contains mixed frequencies or is white noise.

We tune the GP covariance parameters f_{lo}, f_{hi}, and γ from (8) using cross-validation. The cross validation procedure and algorithmic details are identical to that of the frequency detection task in Section 2.3.1. Using cross validation, we find the optimal parameters to be $f_{\text{lo}} = 50$ Hz, $f_{\text{hi}} = 90$ Hz, and $\gamma = 40$ ms. For incompatible weights, we take $f_{\text{lo}} = 10$ Hz, $f_{\text{hi}} = 60$ Hz, and the same γ.

The structured RFN significantly outperform classical RFN for the same number of hidden neurons. We show network performance using these parameters in Fig. 6B. Classification error of 1% can be achieved with 25 hidden neurons. In sharp contrast, the classical RFN requires 300 hidden neurons just to achieve 6% error. With incompatible weights, the network needs 300 neurons to achieve just 15.1% test error and does not improve with larger network sizes. Out of the four input subclasses, it consistently fails to classify pure 80 Hz sinusoidal signals which are outside its passband.

2.3.3 Image classification

We next test the V1-inspired receptive fields from Section 2.2.3 on two standard digit classification tasks, MNIST [51] and KMNIST [19]. The MNIST and KMNIST datasets each contain 70,000 images of handwritten digits. In MNIST, these are the Arabic numerals 0–9, whereas KMNIST has 10 Japanese *hiragana* phonetic characters. Both datasets come split into 60,000 training and 10,000 test examples. With 10 classes, there are 6,000 training examples per class. Every example is a 28×28 grayscale image with centered characters.

Each hidden weight has its center $\boldsymbol{c}$ chosen uniformly at random from all pixels. This ensures that the network's weights uniformly cover the image space and in fact means that the network can represent any sum of locally-smooth functions (see Section A.6). We use a network with 1,000 hidden neurons and tune the GP covariance parameters s and f from (9) using 3-fold cross validation on the MNIST training set. Each parameter ranges from 1 to 20 pixels, and the optimal parameters are found with a grid search. We find the optimal parameters to be $s = 5$ pixels and $f = 2$ pixels. We then refit the optimal model using the entire training set. The parameters from MNIST were used on the KMNIST task without additional tuning.

The V1-inspired achieves much lower average classification error as compared to the classical RFN for the same number of hidden neurons. We show learning performance using these parameters on the MNIST task in Fig. 7A. To achieve 6% error on the MNIST task requires 100 neurons versus 1,000 neurons for the classical RFN, and the structured RFN achieves 2.5% error with 1,000

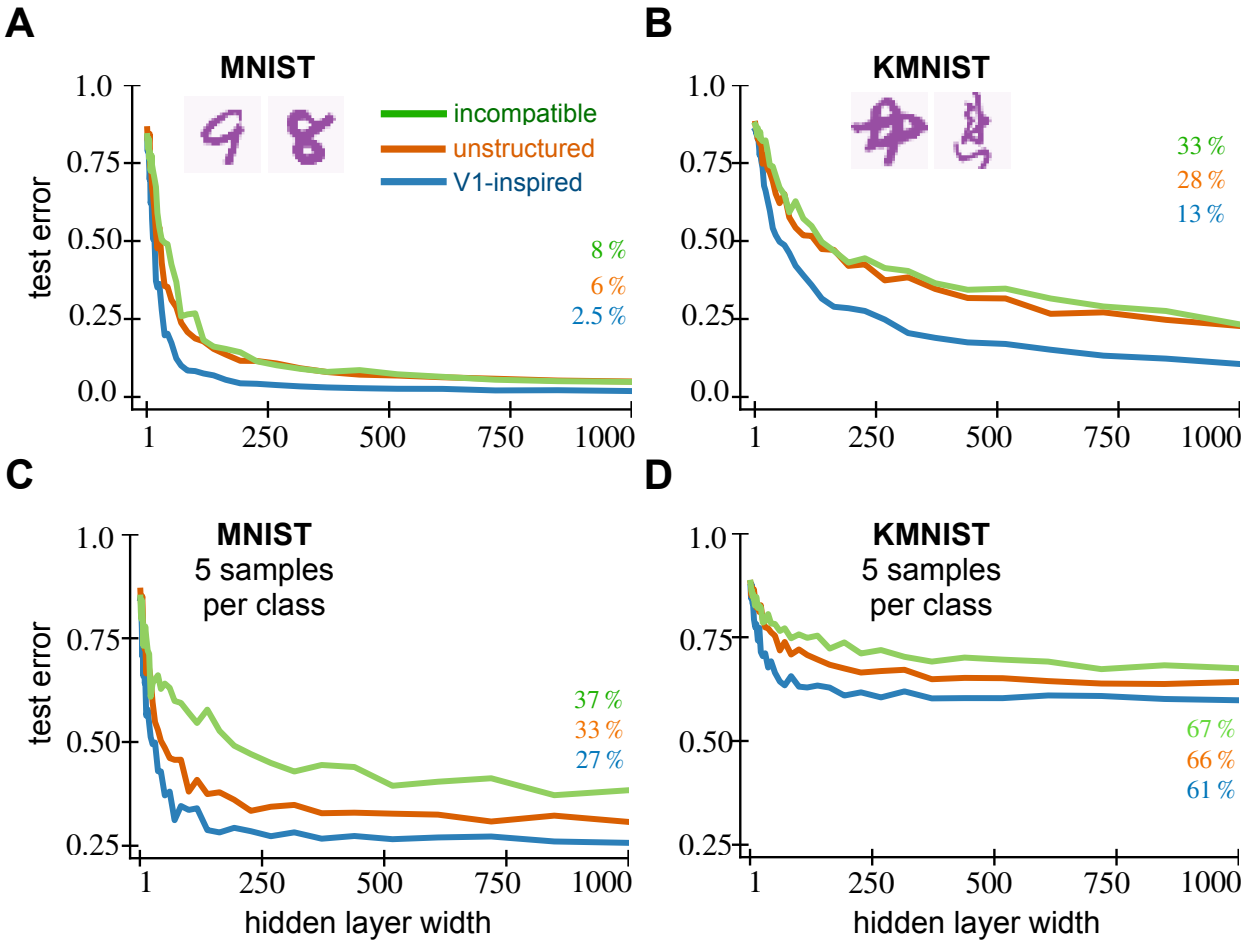

Figure 7: **Random V1 weights enable learning with fewer neurons and fewer examples on digit classification tasks.** We show the average test error of random feature networks with both V1 and classical white-noise weights against the number of neurons in their hidden layer. For every hidden layer width, we generate five random networks and average their test error. The solid lines show the average test error while the shaded regions represent the standard error across five generations of the random network. The top row shows the network's test error on (A) MNIST and (B) KMNIST tasks. When their covariance parameters are tuned properly, V1-inspired networks achieve lower error using fewer hidden neurons on both tasks. The network performance deteriorates when the weights are incompatible to the task. (C) MNIST and (D) KMNIST with 5 samples per class. The V1 network still achieves lower error on these fewshot tasks when the parameters are tuned properly.

neurons. Qualitatively similar results hold for the KMNIST task (Fig. 7B), although the overall errors are larger, reflecting the harder task. To achieve 28% error on KMNIST requires 100 neurons versus 1,000 neurons for the classical RFN, and the structured RFN achieves 13% error with 1,000 neurons.

Again, network performance suffers when GP covariance parameters do not match the task. This happens if the size parameter s is smaller than the stroke width or spatial scale f doesn't match the stroke variations in the character. Taking the incompatible parameters $s = 0.5$ and $f = 0.5$ (Fig. 7A, B), the structured weights performs worse than the classical RFN in both tasks.

With 1,000 hidden neurons, it achieves the relatively poor test errors of 8% on MNIST (Fig. 7A) and 33% on KMNIST (Fig. 7B).

2.3.4 Structured weights improve generalization with limited data

Alongside learning with fewer hidden neurons, V1 structured RFNs also learn more accurately from fewer examples. We test few-shot learning using the image classification datasets from Section 2.3.3. The training examples are reduced from 60,000 to 50, or only 5 training examples per class. The test set and GP parameters remain the same.

Structured encodings allow learning with fewer samples than unstructured encodings. We show these few-shot learning results in Fig. 7C and D. The networks' performance saturate past a few hundred hidden neurons. For MNIST, the lowest error achieved by V1 structured RFN is 27% versus 33% for the classical RFN and 37% using incompatible weights (Fig. 7C). The structured network acheives 61% error using structured features on the KMNIST task, as opposed to 66% for the classical RFN and 67% using incompatible weights (Fig. 7D).

2.3.5 Networks train faster when initialized with structured weights

Now we study the effect of structured weights as an initialization strategy for fully-trained neural networks where all weights in the network vary. We hypothesized that structured initialization allows networks to learn faster, i.e. that the training loss and test error would decrease faster than with unstructured weights. We have shown that the performance of RFNs improves with biologically inspired weight sampling. However, in RFNs (1) only the readout weights $\boldsymbol{\beta}$ are modified with training, and the hidden weights $\boldsymbol{W}$ are frozen at their initial value.

We compare the biologically-motivated initialization with a classical initialization where the variance is inversely proportional to the number of hidden neurons, $\boldsymbol{w}_{\text{unstruct}} \sim \mathcal{N}(0, \frac{2}{d}\boldsymbol{I})$. This initialization is widely known as the "Kaiming He normal" scheme and is thought to stabilize training dynamics by controlling the magnitude of the gradients [39]. The classical approach ensures that $\text{Tr}(\frac{2}{d}\boldsymbol{I}) = 2$, so for fair comparison we scale our structured weight covariance matrix to have $\text{Tr}(\boldsymbol{C}) = 2$. In our studies with RFNs the trace is equal to d, but this weight scale can be absorbed into the readout weights $\boldsymbol{\beta}$ due to the homogeneity of the ReLU.

We again compare structured and unstructured weights on MNIST and KMNIST tasks, common benchmarks for fully-trained networks. The architecture is a single hidden layer feedforward neural network (Fig. 1) with 1,000 hidden neurons. The cross-entropy loss over the training sets are minimized using simple gradient descent (GD) for 3,000 epochs. For a fair comparison, the learning rate is optimized for each network separately. We define the area under the training loss curve as a metric for the speed of learning. Then, we perform a grid search in the range of $(1e^{-4}, 1e^{0})$ for the learning rate that minimizes this metric, resulting in the parameters $0.23, 0.14, 0.14$ for structured, unstructured and incompatible networks respectively. All other parameters are the same as in Section 2.3.3.

In both the MNIST and KMNIST tasks, the V1-initialized network minimizes the loss function faster than the classically initialized network. For the MNIST task, the V1 network achieves a loss value of 0.05 after 3,000 epochs compared to 0.09 for the other network (Fig. 8A). We see qualitatively similar results for the KMNIST task. At the end of training, the V1-inspired network's loss is 0.08, while the classically initialized network only reaches 0.12 (Fig. 8B). We find that the the V1-initialized network performs no better than classical initialization when the covariance parameters do not match the task. With incompatible parameters, the V1-initialized network achieves a loss value of 0.11 on MNIST and 0.15 on KMNIST.

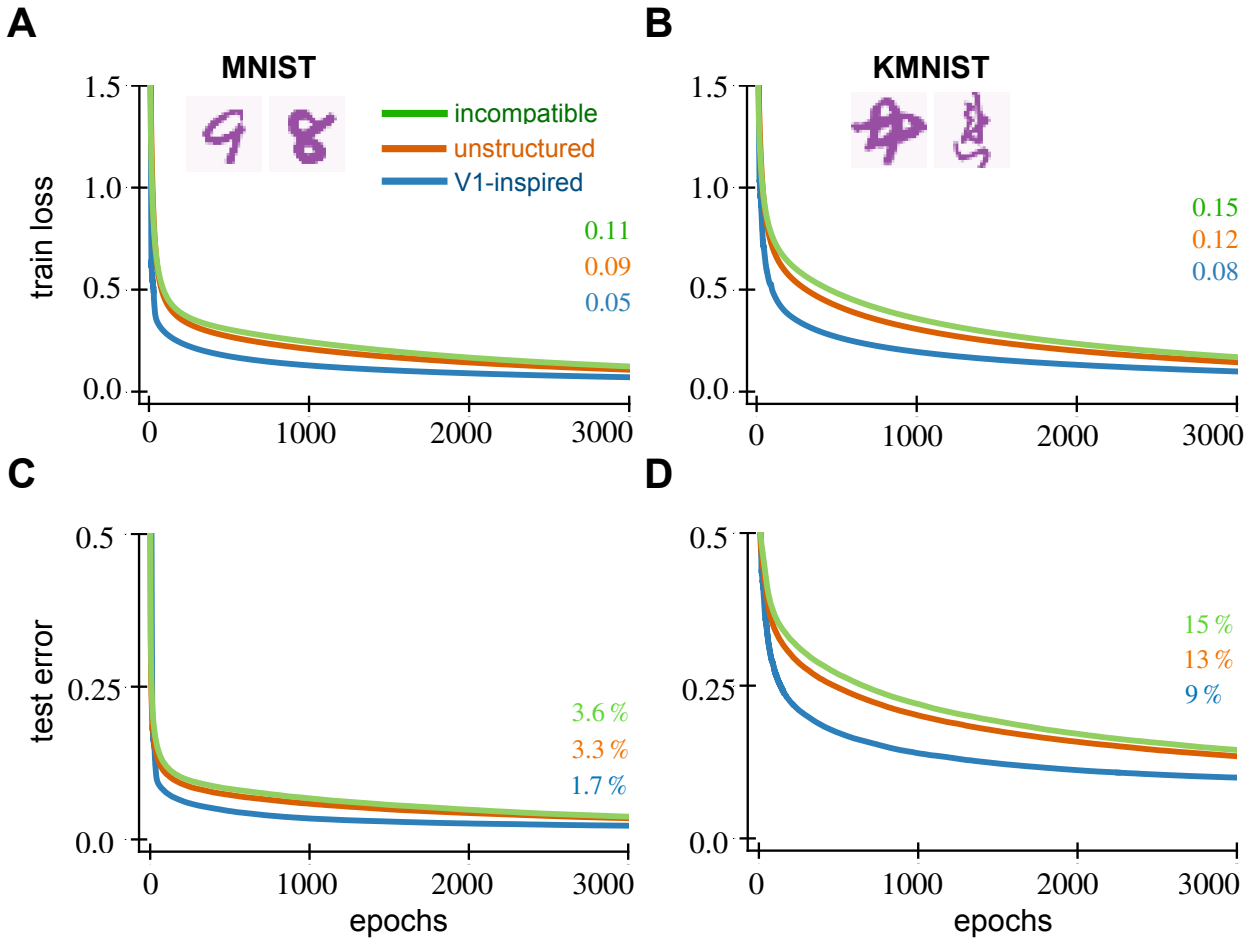

Figure 8: **V1 weight initialization for fully-trained networks enables faster training on digit classification tasks.** We show the average test error and the train loss of fully-trained neural networks against the number of training epochs. The hidden layer of each network contains 1,000 neurons. We generate five random networks and average their performance. The solid lines show the average performance metric across five random networks while the shaded regions represent the standard error. The top row shows the network's training loss on (A) MNIST and (B) KMNIST tasks. The bottom row shows the corresponding test error on (C) MNIST and (D) KMNIST tasks. When their covariance parameters are tuned properly, V1-initialized networks achieve lower training loss and test error under fewer epochs on both MNIST and KMNIST tasks. The network performance is no better than unstructured initialization when the weights are incompatible with the task.

Not only does it minimize the training loss faster, the V1-initialized network also generalizes well and achieves a lower test error at the end of training. For MNIST, it achieves 1.7% test error compared to 3.3% error for the classically initialized network, and 3.6% using incompatible weights (Fig. 8C). For KMNIST, we see 9% error compared to 13% error with classical initalization and 15% using incompatible weights (Fig. 8D).

We see similar results across diverse hidden layer widths and learning rates (Fig. 25–28), with the benefits most evident for wider networks and smaller learning rates. Furthermore, the structured weights show similar results when trained for 10,000 epochs (rate 0.1; 1,000 neurons; not shown)

and with other optimizers like minibatch Stochastic Gradient Descent (SGD) and ADAM (batch size 256, rate 0.1; 1,000 neurons; not shown). Structured initialization facilitates learning across a wide range of networks.

However, the improvement is not universal: no significant benefit was found by initializing the early convolutional layers of the deep network AlexNet [50] and applying it to the ImageNet dataset [78], as shown in Appendix A.12. The large amounts of training data and the fact that only a small fraction of the network was initialized with structured weights could explain this null result. Also, in many of these scenarios the incompatible structured weights get to performance on par with the compatible ones by the end of training, when the poor inductive bias is overcome.

2.3.6 Improving representation with structured random weights

We have shown how structured receptive field weights can improve the performance of RFNs and fully-trained networks on a number of supervised learning tasks. As long as the receptive fields are compatible with the task itself, then performance gains over unstructured features are possible. If they are incompatible, then the networks performs no better or even worse than using classical unstructured weights.

These results can be understood with the theoretical framework of Section 2.1. Structured weights effectively cause the input $\boldsymbol{x}$ to undergo a linear transformation into a new representation $\tilde{\boldsymbol{x}}$ following Theorem 1. In all of our examples, this new representation is bandlimited due to how we design the covariance function.[1] By moving to a bandlimited representation, we both filter out noise—high-frequency components—and reduce dimensionality—coordinates in $\tilde{\boldsymbol{x}}$ outside the passband are zero. In general, noise and dimensionality both make learning harder.

It is easiest to understand these effects in the frequency detection task. For simplicity, assume we are using the stationary features of our warm-up Section 2.2.1 to do frequency detection. In this task, all of the signal power is contained in the $f_1 = 50$ Hz frequency, and everything else is due to noise. If the weights are compatible with the task, this means that $\boldsymbol{w}$ is a sum of sines and cosines of frequencies ω_k in some passband which includes f_1. The narrower we make this bandwidth while still retaining the signal, the higher the SNR of $\tilde{\boldsymbol{x}}$ becomes since more noise is filtered out (Appendix A.8).

3 Discussion

In this paper, we describe a random generative model for the receptive fields of sensory neurons. Specifically, we model each receptive field as a random filter sampled from a Gaussian process (GP) with covariance structure matched to the statistics of experimental neural data. We show that two kinds of sensory neurons—insect mechanosensory and simple cells in mammalian V1—have receptive fields that are well-described by GPs. In particular, the generated receptive fields, their second-order statistics, and their principal components match with receptive field data. Theoretically, we show that individual neurons perform a randomized transformation and filtering on the inputs. This connection provides a framework for sensory neurons to compute input transformations like Fourier and wavelet transforms in a biologically plausible way.

Our numerical results using these structured random receptive fields show that they offer better learning performance than unstructured receptive fields on several benchmarks. The structured networks achieve higher test performance with fewer neurons and fewer training examples, unless the frequency content of their receptive fields is incompatible with the task. In networks that are

[1] The V1 weights have all eigenvalues nonzero, but the spectrum decays exponentially, so it acts as a lowpass filter.

fully trained, initializing with structured weights leads to better network performance (as measured by training loss and generalization) in fewer iterations of gradient descent. Structured random features may be understood theoretically as transforming inputs into an informative basis that retains the important information in the stimulus while filtering away irrelevant signals.

3.1 Modeling other sensory neurons and modalities

The random feature formulation is a natural extension of the traditional linear-nonlinear (LN) neuron model. This approach may be applied to other brain regions where LN models are successful, for instance sensory areas with primarily feedforward connectivity like somatosensory and auditory regions. The neurons in auditory and somatosensory systems are selective to both spatial and temporal structures in their stimuli [47, 79, 71], and spatial structure emerges in networks trained on artificial tactile tasks [100]. Their receptive fields could be modeled by GPs with spatiotemporal covariance functions [95]; these could be useful for artificial tasks with spatiotemporal stimuli such as movies and multivariate timeseries. Neurons with localized but random temporal responses were found to be compatible with manifold coding in a decision-making task [48]. Our GPs are a complementary approach to traditional sparse coding [63] and efficient coding [5, 16] hypotheses; the connections to these other theories are interesting for future research.

3.2 Receptive fields in development

Our generative model offers new directions to explore the biological basis and computational principles behind receptive fields. Development lays a basic architecture that is conserved from animal to animal [90, 88], yet the details of every neural connection cannot be specified [99], leading to some amount of inevitable randomness at least initially [15]. If receptive fields are random with constrained covariance, it is natural to ask how biology implements this. Unsupervised Hebbian dynamics with local inhibition can allow networks to learn principal components of their input [61, 68]. An interesting future direction is how similar learning rules may give rise to overcomplete, nonorthogonal structure similar to what has been studied here. This may prove more biologically plausible than weights that result from task-driven optimization.

The above assumes that receptive field properties actually lie within synaptic weights. For spatial receptive fields, this assumption is plausible [75], but the temporal properties of receptive fields are more likely a result of neurons' intrinsic dynamics, for which the LN framework is just a model [64, 92, 25]. Heterogeneous physiological (e.g. resonator dynamics) and mechanical (position and shape of mechanosensor relative to body structure) properties combine to give the diverse temporal receptive field structures [7]. Development thus leverages different mechanisms to build structure into receptive field properties of sensory neurons.

3.3 Connections to compressive sensing

Random projections have seen extensive use in the field of compressive sensing, where a high-dimensional signal can be found from only a few measurements so long as it has a sparse representation [24, 27, 32]. Random compression matrices are known to have optimal properties, however in many cases structured randomness is more realistic. Recent work has shown that structured random projections with local wiring constraints (in one dimension) were compatible with dictionary learning [26], supporting previous empirical results [6]. Our work shows that structured random receptive fields are equivalent to employing a wavelet dictionary and dense Gaussian projection.

3.4 Machine learning and inductive bias

An important open question for both neuroscience and machine learning is why certain networks, characterized by features such as their architecture, weights, and nonlinearities, are better than others for certain problems. One perspective is that a network is good for a problem if it is biased towards approximating functions that are close to the target, known as an *inductive bias*, which depends on an alignment between the features encoded by neurons and the task at hand [11]. Our approach shows that structured receptive fields are equivalent to a linear transformation of the input that can build in such biases. Furthermore, we can describe the nonlinear properties of the network using the kernel, which varies depending on the receptive field structure. If the target function has a small norm in this RKHS, then there is an inductive bias and it is easier to learn [83, 82]. A small norm in the RKHS means that the target function varies smoothly over the inputs. Smooth functions are easier to learn compared to fast varying ones. In this way, the receptive field structure influences the ease of learning of the target function. We conjecture that receptive fields from neural-inspired distributions affect the RKHS geometry such that the target function's norm is small in that RKHS, compared to the RKHS of random white-noise receptive fields. We leave to future work to verify this conjecture in detail.

Networks endowed with principles of neural computation like batch normalization, pooling of inputs, and residual connections have been found to contain inductive biases for certain learning problems [96, 38]. Learning data-dependent kernels is another way to add in inductive bias [85]. We also saw that initializing fully-trained networks from our generative models improved their speed of convergence and generalization compared to unstructured initialization. This result is consistent with known results that initialization has an effect on generalization [4]. The initialization literature has mostly been focused on avoiding exploding/vanishing gradients [39, 34]. Here, we conjecture that the inductive bias in our structured connectivity places the network closer to a good solution in the loss landscape [99].

The random V1-inspired receptive fields that we model can be seen as similar to what happens in a convolutional neural network (CNN) [62], which have similarities and differences compared to brains [52]. A recent study showed that CNNs with a fixed V1-like convolutional layer are more robust to adversarial perturbations to their inputs [22]. In a similar vein to our work, using randomly sampled Gabor receptive fields in the first layer of a deep network was also shown to improve its performance [42]. The wavelet scattering transform is a multi-layer network where wavelet coefficients are passed through nonlinearities, a model which is similar to deep CNNs [55, 13, 1]. Our framework differs as a randomized model and yields wavelets of a single scale, and similar studies of robustness and learning in deep networks with our weights are possible. Adding layers to our model or sampling weights with a variety of spatial frequencies and field sizes would yield random networks that behave similar to the scattering transform, offering an another connection between the brain and CNNs. Directly learning filters in a Hermite wavelet basis led to good perfomance in ANNs with little data [43], and this idea was extended to multiple scales by [69]. Our structured random features can be seen as an RFN version of those ideas with supporting evidence that these principles are used in biology.

3.5 Limitations and future directions

There are several limitations to the random feature approach. We model neuron responses with a scalar firing rates instead of discrete spikes, and we ignore complex neuronal dynamics, neuromodulatory context, and many other details. Like most LN models, the random feature model assumes zero plasticity in the hidden layer neurons. However, associative learning can drive changes in

receptive fields of individual neurons in sensory areas like V1 and auditory cortex [35, 30]. Further, our RFN is purely feedforward and cannot account for feedback connections. Recent work suggests that feedforward architecture lacks sufficient computational power to serve as a detailed input-output model for a network of cortical neurons; it might need additional layers with convolutional filters [8]. It can be difficult to interpret the parameters found from fitting receptive field data and connect them to experimental conditions. Also, the GP model of weights only captures covariance (second moments) and neglects higher-order statistics. It remains to be shown how the theory can yield concrete predictions that can be tested *in vivo* experimental conditions.

The random feature receptive field model is a randomized extension of the LN neuron model. The LN model fits a parameterized function to each receptive field [18]. In contrast, the random feature framework fits a distribution to an entire population of receptive fields and generates realistic receptive fields from that distribution. A natural question is how they compare. If the goal is to capture individual differences between neuronal receptive fields, one should resort to an LN model where each neuron's receptive field is fit to data. The random feature model is not as flexible, but it provides a direct connection to random feature theory, and it is mathematically tractable and generative. This connection to kernel learning opens the door to using techniques which are a mainstay in machine learning theory literature, for instance to estimate generalization error and sample complexity [83], in the context of learning in more biologically realistic networks.

We see several future directions of structured random features in connecting computational neuroscience and machine learning. As already stated, the auditory, somatosensory, and tactile regions are good candidates for further study as well as developmental principles that could give rise to random yet structured receptive field properties. To account for plasticity in the hidden layer, one could also analyze the neural tangent kernel (NTK) associated with structured features [44]. These kernels are often used to analyze ANNs trained with gradient descent when the number of hidden neurons is large and the step size is small [2]. To incorporate lateral and feedback connections, the weights could be sampled from GPs with recurrent covariance functions [58]. Our theory may also help explain why CNNs with fixed V1-like convolutional layer are more robust to adversarial input perturbations [22] as filtering out high frequency corruptions. It seems likely that structured random features will also be more robust. It would be interesting to study intermediate layer weights of fully-trained networks as approximate samples from a GP by studying their covariance structure. Finally, one could try and develop other covariance functions and further optimize these RFNs for most sophisticated learning tasks to see if near high performance—lower error, faster training, etc.—on more difficult tasks is possible.

Acknowledgements

We thank Dario Ringach for providing the macaque V1 data and Brandon Pratt for the hawkmoth mechanosensor data. We are grateful to Ali Weber, Steven Peterson, and Owen Levin for useful discussions. We thank Sarah Lindo, Michalis Michaelos, and Carsen Stringer for help with mouse surgeries, calcium imaging, and data processing, respectively.

Author contributions

Conceptualization: K.D.H.; Mathematical Analysis: B.P., K.D.H.; Data Acquisition: M.P.; Writing–original draft: B.P., K.D.H.; Writing–review & editing: B.P, M.P., B.W.B., K.D.H.; Figures: B.P., B.W.B., K.D.H.; Supervision: B.W.B., K.D.H.

Funding

B.P. was supported by a UW Applied Math Frederic Wan Endowed Fellowship, Terry Keegan Memorial ARCS Endowment Fellowship, and Natural Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1762114. M.P. was supported by the Janelia Research Campus, Howard Hughes Medical Institute. B.W.B. was supported by grants FA9550-19-1-0386 & FA9550-18-1-0114 from the Air Force Office of Scientific Research. K.D.H. was supported by the Washington Research Foundation postdoctoral fellowship and Western Washington University.

4 References

[1] Joakim Andén and Stéphane Mallat. Deep Scattering Spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, August 2014. Conference Name: IEEE Transactions on Signal Processing.

[2] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *arXiv:1904.11955 [cs, stat]*, Nov 2019. arXiv: 1904.11955.

[3] Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv:1901.08584 [cs, stat]*, May 2019. arXiv: 1901.08584.

[4] Devansh Arpit, Víctor Campos, and Yoshua Bengio. How to initialize your network? robust initialization for weightnorm & resnets. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[5] Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1(01), 1961.

[6] Victor J. Barranca, Gregor Kovačič, Douglas Zhou, and David Cai. Improved compressive sensing of natural scenes using localized random sampling. *Scientific Reports*, 6(1):31976, Aug 2016.

[7] Friedrich G. Barth. Mechanics to pre-process information for the fine tuning of mechanoreceptors. *Journal of Comparative Physiology A*, 205(5):661–686, October 2019.

[8] David Beniaguev, Idan Segev, and Michael London. Single cortical neurons as deep artificial neural networks. *Neuron*, 109(17):2727–2739.e3, Sep 2021.

[9] Vincent Bonin, Mark H. Histed, Sergey Yurgenson, and R. Clay Reid. Local diversity and fine-scale organization of receptive fields in mouse visual cortex. *Journal of Neuroscience*, 31(50):18506–18521, Dec 2011.

[10] Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. *arXiv:2002.02561 [cs, stat]*, Feb 2020. arXiv: 2002.02561.

[11] Blake Bordelon and Cengiz Pehlevan. Population codes enable learning from few examples by shaping inductive bias. *bioRxiv*, page 2021.03.30.437743, Apr 2021.

[12] David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

[13] Joan Bruna and Stephane Mallat. Invariant Scattering Convolution Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, August 2013. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[14] Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *arXiv:2006.13198 [cond-mat, stat]*, Feb 2021. arXiv: 2006.13198.

[15] Sophie J. C. Caron, Vanessa Ruta, L. F. Abbott, and Richard Axel. Random convergence of olfactory inputs in the drosophila mushroom body. *Nature*, 497(74477447):113–117, May 2013.

[16] Matthew Chalk, Olivier Marre, and Gašper Tkačik. Toward a unified theory of efficient, predictive, and sparse coding. *Proceedings of the National Academy of Sciences*, 115(1):186–191, January 2018.

[17] Lin Chen and Sheng Xu. Deep neural tangent kernel and laplace kernel have the same rkhs. *arXiv:2009.10683 [cs, math, stat]*, Mar 2021. arXiv: 2009.10683.

[18] E. J. Chichilnisky. A simple white noise analysis of neuronal light responses. *Network: Computation in Neural Systems*, 12(2):199–213, Feb 2001.

[19] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. *arXiv:1812.01718 [cs, stat]*, 9999. arXiv: 1812.01718.

[20] R. Clay Reid and Jose-Manuel Alonso. Specificity of monosynaptic connections from thalamus to visual cortex. *Nature*, 378(6554):281–284, Nov 1995.

[21] Jan Clemens and Bernhard Ronacher. Feature extraction and integration underlying perceptual decision making during courtship behavior. *Journal of Neuroscience*, 33(29):12136–12145, Jul 2013.

[22] Joel Dapello, Tiago Marques, Martin Schrimpf, Franziska Geiger, David Cox, and James J. DiCarlo. Simulating a primary visual cortex at the front of cnns improves robustness to image perturbations. *Advances in Neural Information Processing Systems*, 33, 2020.

[23] Bradley H Dickerson, Jessica L Fox, and Simon Sponberg. Functional diversity from generic encoding in insect campaniform sensilla. *Current Opinion in Physiology*, 19:194–203, Feb 2021.

[24] Yonina C Eldar and Gitta Kutyniok, editors. *Compressed Sensing: Theory and Applications*. Cambridge University Press, 2012.

[25] Adrienne Fairhall. The receptive field is dead. long live the receptive field? *Current Opinion in Neurobiology*, 25:ix–xii, Apr 2014.

[26] Kion Fallah, Adam A. Willats, Ninghao Liu, and Christopher J. Rozell. Learning sparse codes from compressed representations with biologically plausible local wiring constraints. *bioRxiv*, 2020.

[27] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser Basel, 2013.

[28] J. L. Fox and T. L. Daniel. A neural basis for gyroscopic force measurement in the halteres of holorusia. *Journal of Comparative Physiology A*, 194(10):887–897, Oct 2008.

[29] Jessica L. Fox, Adrienne L. Fairhall, and Thomas L. Daniel. Encoding properties of haltere neurons enable motion feature detection in a biological gyroscope. *Proceedings of the National Academy of Sciences*, 107(8):3840–3845, Feb 2010.

[30] Jonathan Fritz, Shihab Shamma, Mounya Elhilali, and David Klein. Rapid task-related plasticity of spectrotemporal receptive fields in primary auditory cortex. *Nature Neuroscience*, 6:1216–1223, Nov 2003.

[31] Stefano Fusi, Earl K Miller, and Mattia Rigotti. Why neurons mix: high dimensionality for higher cognition. *Current Opinion in Neurobiology*, 37:66–74, Apr 2016.

[32] Surya Ganguli and Haim Sompolinsky. Compressed sensing, sparsity, and dimensionality in neuronal information processing and data analysis. *Annual Review of Neuroscience*, 35(1):485–508, 2012. PMID: 22483042.

[33] Marc G. Genton. Classes of kernels for machine learning: A statistics perspective. *Journal of Machine Learning Research*, 2(Dec):299–312, 2001.

[34] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, page 249–256. JMLR Workshop and Conference Proceedings, Mar 2010.

[35] Pieter M Goltstein, Guido T Meijer, and Cyriel MA Pennartz. Conditioning sharpens the spatial representation of rewarded stimuli in mouse primary visual cortex. *eLife*, 7:e37683, Sep 2018.

[36] Kameron Decker Harris. Additive function approximation in the brain. *arXiv:1909.02603 [cs, q-bio, stat]*, Sep 2019. arXiv: 1909.02603.

[37] Abolfazl Hashemi, Hayden Schaeffer, Robert Shi, Ufuk Topcu, Giang Tran, and Rachel Ward. Generalization bounds for sparse random feature expansions. *arXiv:2103.03191 [cs, math, stat]*, Aug 2021. arXiv: 2103.03191.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv:1512.03385 [cs]*, Dec 2015. arXiv: 1512.03385.

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, page 1026–1034, Dec 2015.

[40] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591, Oct 1959.

[41] B. Igelnik and Y. H. Pao. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. *IEEE transactions on neural networks*, 6(6):1320–1329, 1995.

[42] Bernd Illing, Wulfram Gerstner, and Johanni Brea. Biologically plausible deep learning - but how far can we go with shallow networks? *Neural Networks*, 118:90–101, Oct 2019.

[43] Jörn-Henrik Jacobsen, Jan van Gemert, Zhongyu Lou, and Arnold W. M. Smeulders. Structured receptive fields in cnns. *arXiv:1605.02971 [cs]*, May 2016. arXiv: 1605.02971.

[44] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[45] J. P. Jones and L. A. Palmer. An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of Neurophysiology*, 58(6):1233–1258, Dec 1987.

[46] M Kleiner, D Brainard, and D Pelli. What's new in psychtoolbox-3? In *Perception - ECVP Abstract Supplement. European Conference on Visual Perception (ECVP-2007), August 27-31, Arezzo, Italy*, 2007.

[47] E. I. Knudsen and M. Konishi. Center-surround organization of auditory receptive fields in the owl. *Science*, 202(4369):778–780, Nov 1978.

[48] Sue Ann Koay, Adam S. Charles, Stephan Y. Thiberge, Carlos D. Brody, and David W. Tank. Sequential and efficient neural-population coding of complex task information. *bioRxiv*, 2021.

[49] D. D. Kosambi. Statistics in function space. *The Journal of the Indian Mathematical Society. New Series*, 7:76–88, 1943.

[50] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv:1404.5997 [cs]*, April 2014. arXiv: 1404.5997.

[51] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[52] Grace W. Lindsay. Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future. *Journal of Cognitive Neuroscience*, pages 1–15, February 2020.

[53] Ashok Litwin-Kumar, Kameron Decker Harris, Richard Axel, Haim Sompolinsky, and L.F. Abbott. Optimal degrees of synaptic connectivity. *Neuron*, 93(5):1153–1164.e7, Mar 2017.

[54] Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan A. K. Suykens. Random features for kernel approximation: A survey in algorithms, theory, and beyond. *arXiv:2004.11154 [cs, stat]*, Apr 2020. arXiv: 2004.11154.

[55] Stéphane Mallat. Group Invariant Scattering. *Communications on Pure and Applied Mathematics*, 65(10):1331–1398, 2012.

[56] D. Marr, E. Hildreth, and Sydney Brenner. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, February 1980. Publisher: Royal Society.

[57] J.-B. Martens. The Hermite transform-theory. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(9):1595–1606, September 1990. Conference Name: IEEE Transactions on Acoustics, Speech, and Signal Processing.

[58] César Lincoln C. Mattos, Zhenwen Dai, Andreas Damianou, Jeremy Forth, Guilherme A. Barreto, and Neil D. Lawrence. Recurrent gaussian processes. *arXiv:1511.06644 [cs, stat]*, Feb 2016. arXiv: 1511.06644.

[59] Thomas L. Mohren, Thomas L. Daniel, Steven L. Brunton, and Bingni W. Brunton. Neural-inspired sensors enable sparse, efficient classification of spatiotemporal data. *Proceedings of the National Academy of Sciences*, 115(42):10564–10569, Oct 2018.

[60] Radford M. Neal. *Priors for Infinite Networks*, pages 29–53. Springer New York, New York, NY, 1996.

[61] Erkki Oja. Principal components, minor components, and linear neural networks. *Neural Networks*, 5(6):927–935, November 1992.

[62] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[63] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, December 1997.

[64] Srdjan Ostojic and Nicolas Brunel. From Spiking Neuron Models to Linear-Nonlinear Models. *PLOS Computational Biology*, 7(1):e1001056, January 2011. Publisher: Public Library of Science.

[65] Marius Pachitariu, Carsen Stringer, Mario Dipoppa, Sylvia Schröder, L Federico Rossi, Henry Dalgleish, Matteo Carandini, and Kenneth D Harris. Suite2p: beyond 10,000 neurons with standard two-photon microscopy. *BioRxiv*, 2017.

[66] Liam Paninski. Convergence properties of some spike-triggered analysis techniques. In *Network: Computation in Neural Systems*, page 2003, 2003.

[67] Mijung Park and Jonathan W Pillow. Receptive field inference with localized priors. *PLoS computational biology*, 7(10):e1002219, 2011.

[68] Cengiz Pehlevan, Anirvan M. Sengupta, and Dmitri B. Chklovskii. Why Do Similarity Matching Objectives Lead to Hebbian/Anti-Hebbian Networks? *Neural Computation*, 30(1):84–124, January 2018.

[69] Silvia L. Pintea, Nergis Tomen, Stanley F. Goes, Marco Loog, and Jan C. van Gemert. Resolution learning in deep convolutional networks using scale-space theory. *arXiv:2106.03412 [cs]*, Jun 2021. arXiv: 2106.03412.

[70] Brandon Pratt, Tanvi Deora, Thomas Mohren, and Thomas Daniel. Neural evidence supports a dual sensory-motor role for insect wings. *Proceedings of the Royal Society B: Biological Sciences*, 284(1862):20170969, Sep 2017.

[71] J. Andrew Pruszynski and Roland S. Johansson. Edge-orientation processing in first-order tactile neurons. *Nature Neuroscience*, 17(10):1404–1409, Oct 2014.

[72] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates, Inc., 2008.

[73] Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, page 555–561. IEEE, Sep 2008.

[74] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[75] Dario L. Ringach. Haphazard Wiring of Simple Receptive Fields and Orientation Columns in Visual Cortex. *Journal of Neurophysiology*, 92(1):468–476, July 2004.

[76] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 19590501.

[77] Juha Rusanen, Roman Frolov, Matti Weckström, Michiyo Kinoshita, and Kentaro Arikawa. Non-linear amplification of graded voltage signals in the first-order visual interneurons of the butterfly papilio xuthus. *Journal of Experimental Biology*, 221(12), Jun 2018.

[78] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015.

[79] Hsiao S. Central mechanisms of tactile shape perception. *Current opinion in neurobiology*, 18(4), Aug 2008.

[80] H. M. Sakai and K. Naka. Signal transmission in the catfish retina. v. sensitivity and circuit. *Journal of Neurophysiology*, 58(6):1329–1350, Dec 1987.

[81] Shreya Saxena and John P Cunningham. Towards the neural population doctrine. *Current Opinion in Neurobiology*, 55:103–111, Apr 2019.

[82] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[83] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[84] Charles Sherrington. *The Integrative Action of the Nervous System*. Cambridge University Press, 1907.

[85] Aman Sinha and John C Duchi. Learning kernels with random features. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[86] Nicholas James Sofroniew, Daniel Flickinger, Jonathan King, and Karel Svoboda. A large field of view two-photon mesoscope with subcellular resolution for in vivo imaging. *Elife*, 5:e14472, 2016.

[87] Gilbert Strang. The Discrete Cosine Transform. *SIAM Review*, 41(1):135–147, January 1999.

[88] Nicholas James Strausfeld. *Arthropod Brains: Evolution, Functional Elegance, and Historical Significance*. Harvard University Press, 2012.

[89] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Matteo Carandini, and Kenneth D Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765):361–365, 2019.

[90] Larry W. Swanson. *Brain architecture: Understanding the basic plan*, volume xv. Oxford University Press, New York, NY, US, 2003.

[91] Grace Wahba. *Spline Models for Observational Data*. SIAM, September 1990.

[92] Alison I. Weber and Jonathan W. Pillow. Capturing the Dynamical Repertoire of Single Neurons with Generalized Linear Models. *Neural Computation*, 29(12):3260–3289, December 2017.

[93] Christopher K. I. Williams. Computation with infinite neural networks. *Neural Computation*, 10(5):1203–1216, Jul 1998.

[94] Marjorie Xie, Samuel Muscinelli, Kameron Decker Harris, and Ashok Litwin-Kumar. Understanding the role of sparseness in cerebellar granule cell representations. In *Computational and Systems Neuroscience (Cosyne)*. 2021.

[95] Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. A temporal kernel approach for deep learning with continuous-time information. *arXiv:2103.15213 [cs]*, Mar 2021. arXiv: 2103.15213.

[96] Daniel L. K. Yamins, Ha Hong, Charles F. Cadieu, Ethan A. Solomon, Darren Seibert, and James J. DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624, 2014.

[97] Alexandra M. Yarger and Jessica L. Fox. Dipteran halteres: Perspectives on function and integration for a unique sensory organ. *Integrative and Comparative Biology*, 56(5):865–876, Nov 2016.

[98] Rafael Yuste. From the neuron doctrine to neural networks. *Nature Reviews Neuroscience*, 16(8):487–497, Aug 2015.

[99] Anthony M. Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications*, 10(1):3770, Dec 2019.

[100] Charlie W. Zhao, Mark J. Daley, and J. Andrew Pruszynski. Neural network models of the tactile system develop first-order units with spatially complex receptive fields. *PLOS ONE*, 13(6):e0199196, Jun 2018.

A Appendix

Abbreviation	Meaning
ANN	artificial neural network
DCT	discrete cosine transform
DFT	discrete Fourier transform
DHT	discrete Hartley transform
GP	Gaussian process
LN	linear-nonlinear model of a neuron
ReLU	rectified linear unit, $\max(0, x)$ nonlinearity
RFN	random feature network
RKHS	reproducing kernel Hilbert space
SNR	signal-to-noise ratio
STA	spike triggered average
V1	primary visual cortex
XOR	exclusive-or, boolean function

Table 1: List of abbreviations

Symbol	Meaning
$\boldsymbol{x}$	an input or stimulus to the network
$\boldsymbol{w}$	input-hidden weights for a neuron
$\boldsymbol{\beta}, \beta_0$	readout weights and offset
$y, \hat{y}$	true and predicted output of the network
d	dimension of the stimulus as a vector
m	number of neurons in the hidden layer
T	structured input space
D	dimensions of input space T
$L^2(T)$	space of square-integrable functions over a domain T
ℓ^2	vector space with norm $\|\boldsymbol{u}\| = \sqrt{\boldsymbol{u}^T\boldsymbol{u}}$
$\|\cdot\|$	the L_2 or ℓ_2 norm for function or vector argument
$\|\cdot\|_F$	the Frobenius norm of a matrix
$\langle a, b\rangle$	the $L^2(T)$ inner product between functions, $\langle a, b\rangle = \int_{t\in T} a(t)b(t)\mathrm{d}t$
$\boldsymbol{u}^T\boldsymbol{v}$	finite-dimensional ℓ^2 inner product, $\boldsymbol{u}^T\boldsymbol{v} = \sum_{i=1}^d u_i v_i$
$\boldsymbol{I}_d$	$d \times d$ identity matrix
$\boldsymbol{C}$	$d \times d$ covariance matrix
$\mathcal{H}$	RKHS, comes with inner product $\langle a, b\rangle_{\mathcal{H}}$ and norm $\|a\|_{\mathcal{H}} = \sqrt{\langle a, a\rangle_{\mathcal{H}}}$

Table 2: List of important symbols

A.1 Function spaces for wide networks with structured receptive fields

RFNs are intimately connected to a popular class of supervised learning algorithms called kernel methods. As the network width grows, the inner product between the feature representations of two inputs $\boldsymbol{x}, \boldsymbol{x}'$ converges to a reproducing kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') := \mathbb{E}_{\boldsymbol{w}}\left[h(\boldsymbol{x})h(\boldsymbol{x}')\right]. \tag{11}$$

The kernel defines a reproducing kernel Hilbert space (RKHS) of functions. The explicit form of the kernels corresponding to classical RFNs are known for several non-linear activation functions. For example, with the ReLU nonlinearity, no threshold, and unstructured Gaussian weights $\boldsymbol{w} \sim \mathcal{N}(0, \boldsymbol{I}_d)$, $k_{\text{ReLU}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{\pi}\|\boldsymbol{x}\|\|\boldsymbol{x}'\|(\sin\theta + (\pi - \theta)\cos\theta)$ where $\theta = \arccos\left(\frac{\boldsymbol{x}^T\boldsymbol{x}'}{\|\boldsymbol{x}\|\|\boldsymbol{x}'\|}\right)$ [7].

We derive the kernel induced by our RFNs with hidden weights initialized from GPs. In this section we work in the discrete setting, but the continuous version is analogous. By definition (11), network equation (1), and basis change Theorem 1, the kernel for structured features

$$\begin{aligned} k_{\text{struct}}(\boldsymbol{x}, \boldsymbol{x}') &= \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0,\boldsymbol{C})}\left[h(\boldsymbol{x})h(\boldsymbol{x}')\right] \\ &= \mathbb{E}_{\boldsymbol{w}\sim\mathcal{N}(0,\boldsymbol{C})}\left[\sigma(\boldsymbol{w}^T\boldsymbol{x})\sigma(\boldsymbol{w}^T\boldsymbol{x}')\right] \\ &= \mathbb{E}_{\boldsymbol{z}\sim\mathcal{N}(0,\boldsymbol{I}_d)}\left[\sigma\left(\boldsymbol{z}^T\tilde{\boldsymbol{x}}\right)\sigma\left(\boldsymbol{z}^T\tilde{\boldsymbol{x}}'\right)\right] \\ &:= k_{\text{unstruct}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}'). \end{aligned} \tag{12}$$

Thus, the induced kernels from structured weights can be found in terms of unstructured weight kernels acting on the transformed inputs $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}'$. Taking ReLU as the nonlinearity for example, we get that $k_{\text{struct}}(\boldsymbol{x}, \boldsymbol{x}') = k_{\text{ReLU}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}')$.

Every RKHS $\mathcal{H}$ comes with an inner product $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ and norm $\|\cdot\|_{\mathcal{H}} = \sqrt{\langle\cdot,\cdot\rangle_{\mathcal{H}}}$. The norm and inner product can be expressed in terms of eigenvalues and eigenfunctions of the kernel itself, analogous to the eigendecomposition of the covariance function of the GP weights. Although it is beyond the scope of our paper to explain the theory in detail, there are well-established results showing that functions with small $\mathcal{H}$-norm are easier to learn than those with larger norm for a wide variety of kernel-based algorithms [83, 82]. In ridge regression, this effect is again equivalent to projection and filtering in the kernel eigenbasis, i.e. linear filtering in function space. Finally, end-to-end trained networks where the weights $\boldsymbol{W}$ are optimized may be studied with the related neural tangent kernel (NTK) when the step size is small [44]. The basis change formula, Theorem 1, and (12) give us a way to understand the RKHS of the stuctured network in terms of an unstructured network's RKHS acting on the transformed inputs $\tilde{\boldsymbol{x}}$.

A.1.1 Kernel eigenfunctions differ with structured weights

The structured RKHS has eigenfunctions which are different from the eigenfunctions of the unstructured RKHS. To see this, it's necessary to introduce a probability measure $\mu(\boldsymbol{x})$ for the data $\boldsymbol{x} \in \mathbb{R}^d$. Kernel learning is often understood [14] in the orthonormal basis for $L^2(\mu)$ given by the eigenfunctions ψ_i of the integral operator $\mathcal{T}_k$ defined by

$$(\mathcal{T}_k f)(\boldsymbol{x}) = \int k(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}')\mathrm{d}\mu(\boldsymbol{x}').$$

A natural question to ask is, how does the eigensystems of $\mathcal{T}_{\text{struct}}$ and $\mathcal{T}_{\text{unstruct}}$ compare? The mapping from $\boldsymbol{x}$ to $\tilde{\boldsymbol{x}}$ induces a pushforward measure on $\tilde{\boldsymbol{x}}$ which we will call $\nu(\tilde{\boldsymbol{x}})$, and since the mapping is linear $\nu(\tilde{\boldsymbol{x}}) = \mu(\boldsymbol{\Phi}^T\boldsymbol{\Lambda}^{-1}\boldsymbol{x})$. (Note that $\mathrm{d}\nu(\boldsymbol{x}) = |\boldsymbol{\Lambda}|^{-1}\mathrm{d}\mu(\boldsymbol{\Phi}^T\boldsymbol{\Lambda}^{-1}\boldsymbol{x})$, and if μ is multivariate Gaussian, then ν is Gaussian with a different covariance.) Thus the integral operator $\mathcal{T}_{\text{struct}}$ under the data measure is equivalent to integrating kernel k_{unstruct} under the pushforward measure. Because

$$\int k_{\text{unstruct}}(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}')\mathrm{d}\mu(\boldsymbol{x}') \qquad \text{and} \qquad \int k_{\text{unstruct}}(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}')\mathrm{d}\nu(\boldsymbol{x}')$$

are different, there is no general relationship that holds between the eigenfunctions. Different measures leading to different kernel eigenfunctions and eigenvalues can explain why structured

weights have strong effects on learning, as shown by recent work demonstrating that areas of low input density are learned more slowly [4].

A.2 Kernel theory for frequency detection: a fully worked, highly nutritious, simplified example

To demonstrate how the kernel theory can also explain the benefits of structured random features, we developed a simplified frequency detection task. We fully explain the orthonormal basis, target function in that basis, kernels, and how the target function is harder to learn without knowledge of the structure. To our knowledge, this is the first technical result that explains how transforming the input data into an informative, lower-dimensional representation (essentially, preprocessing) leads to improved learning with kernel methods.

A.2.1 Data distribution and orthonormal basis

Frequency detection as presented in Appendix A.7.1 is difficult to work with because of the different distributions of x_+ and x_- samples. Kernel theory requires us to work with basis functions which are orthonormal with respect to the data measure, which in that case would be a mixture. To get around this issue, we simplify the data distribution so that all of our data are Gaussian white noise $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{I}_d)$. The Gaussian measure in 1-d has the natural orthogonal basis of (probabilist's) Hermite polynomials $He_k(x)$ [1]. The first few of these are

$$He_0(x) = 1, \quad He_1(x) = x, \quad He_2(x) = x^2 - 1.$$

We can normalize these as $\psi_k(x) = (k!)^{-1/2} He_k(x)$ so that orthonormality under the Gaussian distribution

$$\int_{-\infty}^{\infty} \psi_k(x)\psi_{k'}(x) \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \mathrm{d}x = \delta_{k,k'} \tag{13}$$

is satisfied. To construct an orthonormal basis in $d > 1$ dimensions, we take the tensor product of Hermite polynomials,

$$\psi_{\boldsymbol{k}}(\boldsymbol{x}) = \prod_{i=1}^{d} \psi_{k_i}(x_i), \tag{14}$$

which is orthonormal for the multivariate Gaussian

$$\int_{\mathbb{R}^d} \psi_{\boldsymbol{k}}(\boldsymbol{x})\psi_{\boldsymbol{k}'}(\boldsymbol{x})(2\pi)^{-d/2} e^{-\|\boldsymbol{x}\|^2/2} \mathrm{d}\boldsymbol{x} = \delta_{\boldsymbol{k},\boldsymbol{k}'} \tag{15}$$

due to the separability of the integrals over each dimension.

A.2.2 Target function

The labels $y = f^*(\boldsymbol{x})$ are chosen to only depend on the amplitude of frequency f_1 in the white noise signal. Writing $\tilde{\boldsymbol{x}} = \boldsymbol{\Phi}^T \boldsymbol{x}$ for the input signal in frequency space, with $\boldsymbol{\Phi}$ the discrete cosine transform (DCT) matrix [87], the target function f^* only depends on $|\tilde{x}_{f_1}|$. A simplified classfication task would then use

$$f^*(\boldsymbol{x}) = \begin{cases} +1 & \text{if } |\tilde{x}_{f_1}| \geq \theta \\ -1 & \text{if } |\tilde{x}_{f_1}| < \theta \end{cases}, \tag{16}$$

where the threshold θ is chosen so that 50% of the points are labeled ± 1. However, we need to write the target function f^* in the polynomial basis (14), and (16) is a step function which has an infinite polynomial expansion. A much simpler target function

$$f^*(\boldsymbol{x}) = |\tilde{x}_{f_1}|^2 - \theta^2 \tag{17}$$

still captures some of the behavior of (16): It's negative when the power in frequency f_1 is below θ and positive above. We will analyze the quadratic surrogate (17).

A.2.3 Kernel decomposition

We consider kernels $k_{\text{struct}}, k_{\text{unstruct}}$ of the form

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\boldsymbol{k}} \rho_{\boldsymbol{k}} \psi_{\boldsymbol{k}}(\boldsymbol{x}) \psi_{\boldsymbol{k}}(\boldsymbol{x}'), \tag{18}$$

where $\boldsymbol{k} = (k_1, k_2, \ldots, k_d)$ is a multi-index. We build κ out of 1-d kernels

$$\kappa_i(x_i, x_i') = \sum_{k_i=0}^{\infty} \rho_{k_i} \psi_{k_i}(x_i) \psi_{k_i}(x_i'), \tag{19}$$

where the eigenvalue sequence $\rho_0, \rho_1, \ldots$ is shared across different κ_i. Each κ_i defines an RKHS over functions of a single variable $\mathcal{H}_i$, and $\kappa(\boldsymbol{x}, \boldsymbol{x}') = \prod_{i=1}^d \kappa_i(x_i, x_i')$ defines an RKHS which is the tensor product $\mathcal{H} = \otimes_{i=1}^d \mathcal{H}_i$ with eigenvalues given as $\rho_{\boldsymbol{k}} = \prod_{i=1}^d \rho_{k_i}$. These tensor product polynomial kernels were considered in [3] for multiple kernel learning. We normalize the 1-d kernels (19) so that $\sum_{k=0}^{\infty} \rho_k = 1$. This implies that

$$\sum_{\boldsymbol{k}} \rho_{\boldsymbol{k}} = \sum_{k_1,\ldots,k_d=0}^{\infty} \prod_{i=1}^{d} \rho_{k_i} = \left(\sum_{k_1=0}^{\infty} \rho_{k_1} \right) \cdots \left(\sum_{k_d=0}^{\infty} \rho_{k_d} \right) = 1, \tag{20}$$

by the separability of the eigenvalues, so that the d-dimensional kernel (18) is also normalized. Furthermore, we can assume that all of the $\rho_i < 1$.

In general, random feature kernels will not be tensor products unless we make very specific choices of weights and nonlinearities. We only use the product structure for convenience with normalization. The form (18) is, on the other hand, quite general and will hold for a variety of random feature kernels.

A.2.4 Target function in unstructured and structured networks

Now we analyze the target function in both unstructured and unstructured spaces. Using structured random feature map is equivalent to some deterministic remapping $\tilde{\boldsymbol{x}} = \boldsymbol{\Lambda}\boldsymbol{\Phi}^T \boldsymbol{x}$, where $\boldsymbol{\Phi}$ is an orthogonal matrix encompassing the basis change and $\boldsymbol{\Lambda}$ is a filtering matrix. Like in Section A.8, let's use stationary bandpass features so that $\boldsymbol{\Lambda}$ just contains d' entries which are equal to 1 with the rest 0, and $\boldsymbol{\Phi}$ is the DCT. Since the DCT is unitary, $\tilde{\boldsymbol{x}} \sim \mathcal{N}(0, I_{d'})$. (We use the DCT rather than the DFT to avoid complications dealing with complex variables.) Thus the transformed variables $\tilde{\boldsymbol{x}}$ and the original variables $\boldsymbol{x}$ both follow similar spherical Gaussian distributions just in different dimensions. This means that the Hermite polynomial basis (14) is an orthonormal basis for both spaces.

Let $\tilde{\mathcal{H}}$ be the RKHS of functions after applying the transformation $\boldsymbol{x} \mapsto \tilde{\boldsymbol{x}}$. Then $\tilde{\mathcal{H}}$ has the kernel $\kappa(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{x}}')$ and eigenbasis $\psi_{\boldsymbol{k}}(\tilde{\boldsymbol{x}})$ under measure $\tilde{\boldsymbol{x}} \sim \mathcal{N}(0, I_{d'})$. We take $\mathcal{H}$ to be the RKHS of

functions without ever transforming coordinates, i.e. the RKHS with kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}')$ with eigenbasis $\psi_{\boldsymbol{k}}(\boldsymbol{x})$ under data measure $\boldsymbol{x} \sim \mathcal{N}(0, I_d)$.

The target function (17) is simple to express in the eigenbasis of $\tilde{\mathcal{H}}$:

$$\begin{aligned} f^*(\tilde{\boldsymbol{x}}) &= |\tilde{x}_{f_1}|^2 - \theta^2 \\ &= He_2(\tilde{x}_{f_1}) - (\theta^2 - 1) He_0(\tilde{x}_{f_1}) \\ &= \sqrt{2}\psi_2(\tilde{x}_{f_1}) - (\theta^2 - 1)\psi_0(\tilde{x}_{f_1}). \end{aligned}$$

Any RKHS $\mathcal{F}$ with eigenvalues ρ_i and eigenfunctions ψ_i defines an inner-product

$$\langle f, g \rangle_{\mathcal{F}} = \sum_i \frac{\langle f, \psi_i \rangle_{L^2(\mu)} \langle g, \psi_i \rangle_{L^2}(\mu)}{\rho_i}, \tag{21}$$

where the $L^2(\mu)$ inner product is with respect to the data measure μ and $\|f\|_{\mathcal{F}}^2 = \langle f, f \rangle_{\mathcal{F}}$ [2]. The norm of our target function thus becomes

$$\|f^*\|_{\tilde{\mathcal{H}}}^2 = \frac{2}{\rho_2 \rho_0^{d'-1}} + \frac{(\theta^2 - 1)^2}{\rho_0^{d'}}. \tag{22}$$

We will see that working with the less informative RKHS $\mathcal{H}$ leads to a significantly larger norm.

To express the target function (17) in the eigenbasis of $\mathcal{H}$ we must use the basis change matrix. Let $\boldsymbol{u}$ be the vector corresponding to row f_1 of $\boldsymbol{\Phi}^T$ so that $\tilde{x}_{f_1} = \boldsymbol{u}^T \boldsymbol{x} = \sum_{i=1}^d u_i x_i$. Thus,

$$\begin{aligned} f^*(\boldsymbol{x}) &= |\tilde{x}_{f_1}|^2 - 1 \\ &= \left(\sum_{i=1}^d u_i x_i \right)^2 - \theta^2 \\ &= \sum_{i=1}^d u_i^2 x_i^2 + 2 \sum_{i<j} u_i u_j x_i x_j - \theta^2 \\ &= \sum_{i=1}^d u_i^2 (\sqrt{2}\psi_2(x_i) + 1) + 2 \sum_{i<j} u_i u_j \psi_1(x_i) \psi_1(x_j) - \theta^2. \end{aligned}$$

Reading off the coefficients of the basis terms (recall that $1 = \psi_{\boldsymbol{0}}(\boldsymbol{x})$), we get that

$$\begin{aligned} \|f^*\|_{\mathcal{H}}^2 &= \frac{2}{\rho_2 \rho_0^{d-1}} \left(\sum_{i=1}^d u_i^4 \right) + \frac{1}{\rho_0^d} \left(\sum_{i=1}^d u_i^2 - \theta^2 \right)^2 + \frac{4}{\rho_1^2 \rho_0^{d-2}} \sum_{i<j} (u_i u_j)^2 \\ &= \frac{2}{\rho_2 \rho_0^{d-1}} \left(\sum_{i=1}^d u_i^4 \right) + \frac{(\theta^2 - 1)^2}{\rho_0^d} + \frac{4}{\rho_1^2 \rho_0^{d-2}} \sum_{i<j} (u_i u_j)^2, \end{aligned} \tag{23}$$

where the $\sum_{i=1}^d u_i^2$ terms are equal to 1 because $\boldsymbol{\Phi}$ is unitary. Note that

$$1 = \|\boldsymbol{u}\boldsymbol{u}^T\|_F^2 = \sum_{i,j} (u_i u_j)^2 = 2 \sum_{i<j} (u_i u_j)^2 + \sum_{i=1}^d u_i^4, \tag{24}$$

so the terms involving sums combined are $O(\rho_0^d)$. Specifically for the DCT-II [87], components $u_i = \sqrt{\frac{2}{d}} \cos\left(\frac{\pi f_1}{d}\left(i - \frac{1}{2}\right)\right)$, so for any $f_1 \in \{1, \dots (d-1)\}$

$$\sum_{i=1}^{d} u_i^4 = \frac{4}{d^2} \sum_{k=0}^{d-1} \cos^4\left(\frac{\pi f_1}{d}\left(k + \frac{1}{2}\right)\right) = \frac{3}{2d},$$

which implies that $\sum_{i<j}(u_i u_j)^2 = \frac{1}{2} - \frac{3}{4d}$. (For general unit vectors $\boldsymbol{u}$ we have that $1 \geq \sum_{i=1}^{d} u_i^4 \geq d^{-1}$ by Hölder's inequality.) Thus for DCT-II the norm is

$$\|f^*\|_{\mathcal{H}}^2 = \frac{3}{d\rho_2\rho_0^{d-1}} + \frac{(\theta^2-1)^2}{\rho_0^d} + \frac{2}{\rho_1^2\rho_0^{d-2}}\left(1 - \frac{3}{2d}\right). \tag{25}$$

A.2.5 Comparison of the learning performance in structured and unstructured RKHS

A very standard but rough bound on the generalization performance of kernel ridge regression or classification can be found by analyzing the Rademacher complexity of the class of linear functions in the RKHS [82, 83]. These upper bound the expected loss over new data in terms of the training loss plus an error term, c.f. Theorem 7.39 in [83] for kernel ridge regression. The error term controls the generalization gap between test and training losses and is typically proportional to the $\|f^*\|_{\mathcal{F}}$ (for 0/1 loss) or $\|f^*\|_{\mathcal{F}}^2$ (for square loss).[2] Thus, a function with small $\mathcal{F}$-norm is *easier to learn* in the *precise sense* that it takes a smaller training set size to achieve a given generalization gap when the norm is smaller.

We have computed $\|f^*\|_{\mathcal{H}}^2$ and $\|f^*\|_{\tilde{\mathcal{H}}}^2$ in (23) and (22). Examining those two expressions, we see that $\|f^*\|_{\mathcal{H}}^2 \gg \|f^*\|_{\tilde{\mathcal{H}}}^2$ due to two factors: First, there is the splitting of the quadratic target into x_i^2 and $x_i x_j$ terms which creates the extra term $\frac{4}{\rho_1^2\rho_0^{d-2}}\sum_{i<j}(u_i u_j)^2$ in the norm. This arises from *nonlinear effects* of the quadratic target function on the different components of the input signal. This nonlinear term is $O(\rho_0^{-d})$, the same order as the other terms, but (23) will be larger than (22) for $d' = d$ when $\rho_1^2 < \rho_0\rho_2$. Secondly, the *dimension reduction* from d to d' dimensions means that analogous terms in (23) and (22) scale like c^d and $c^{d'}$, respectively. This means that $\|f^*\|_{\mathcal{H}}^2 \geq c^{(d-d')}\|f^*\|_{\tilde{\mathcal{H}}}^2$. Note that the projection $\boldsymbol{x} \mapsto \tilde{\boldsymbol{x}}$ is a linear operation but has nonlinear consequences for the norm. The norm in the unstructured kernel space is exponentially larger than the norm in the structured kernel space.

Interpreting this exponential norm separation in light of the Rademacher generalization bounds, you would need to train on exponentially more samples with the unstructured versus the structured kernel to acheive the same bound. The majority of this effect is due to the dimension reduction factor, since the nonlinear factor only grows the norm by a constant. However, for target functions with contributions from many higher-order polynomials, the nonlinear factor would have a non-linearly stronger effect. This is an interesting avenue for future research. More precise estimates of generalization performance are possible using theories such as [10]. However, the exponential gap between the performance of these two kernels is fundamental and not an artificial result from shortcomings of the Rademacher analysis.

A.2.6 Simulation results support the theory

We ran simulations of the simplified frequency detection task with both kernel methods and random feature networks to check that this simplified task was similar to the task presented in Section 2.3.1

[2] The trace of the kernel matrix also appears in these bounds, but this is unity due to normalization (20).

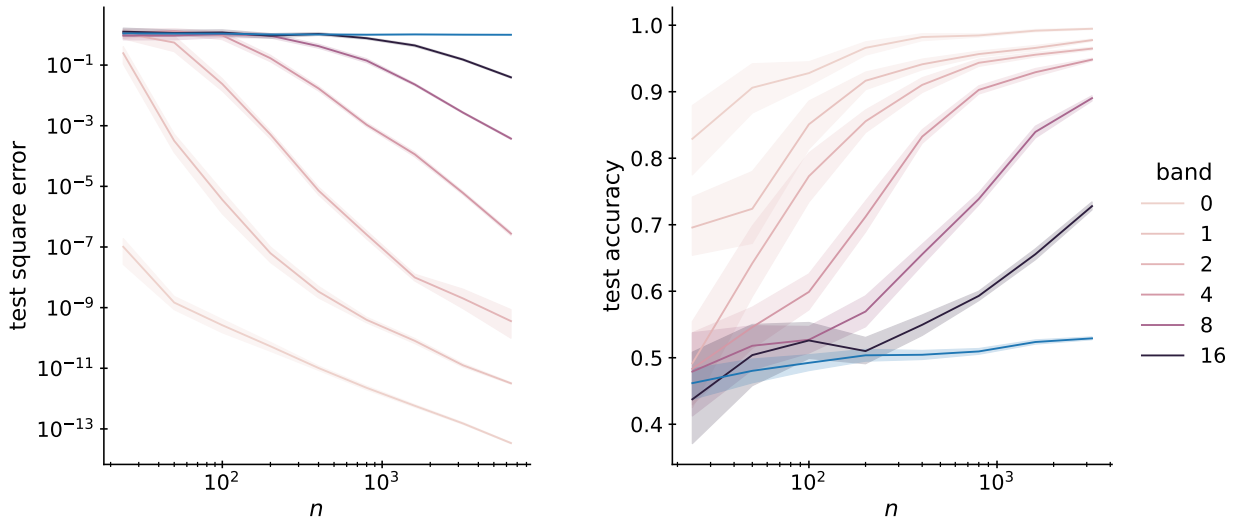

Figure 9: Simulation results for the simplified frequency detection task. On the left, test error versus dataset size for kernel ridge regression using structured kernels (purple lines, by bandwidth) and the unstructured kernel (blue). On the right, test accuracy versus dataset size for SVM classifier readout trained on structured random features (purple lines, by bandwidth) and unstructured random features (blue). In both cases, task structural information improves performance, leading to less error and higher accuracy.

and Appendix A.7.1. These simulations were performed with regression tasks with targets like (17) and classification tasks equivalent to (16). The results are consistent for a broad range of estimators and for both kernel and random feature networks.

For regression tasks, we used a tensor product kernel of the form (18) with kernel ridge regression and the target function $f^*(\boldsymbol{x}) = a(|\tilde{x}_{f_1}|^2 - \theta^2)$. The constant a was chosen so that the labels had standard deviation 1, meaning that a mean square error of 1 is equivalent to chance. For varying n logarithmically spaced between 24 and 6,400 we generated training and testing sets of size $n/2$ in d =1,000 dimensions with $f_1 = 16$. The training set was used to select the ridge parameter from powers of 10 in the range $[10^{-3}, \ldots, 10^3]$ by 5-fold cross-validation and error was computed on the test set. Each of these experiments was repeated 20 times. We compare unstructured and structured kernels by either passing in the raw vectors $\boldsymbol{x}$ (unstructured) or first performing the DFT and passing in $(\tilde{\boldsymbol{x}})_{f_1-\texttt{band}:f_1+\texttt{band}}$. Here, `band` is a bandwidth parameter determining the number of components that are used in the structured kernel. When `band` $= 0$ we keep only the target frequency component.

The classification tasks were similar. We used the exact target function (16), the range of n from 24 to 3,200, same split into training and test sets, $d = 100$, and $f_1 = 16$. We classified using a random feature network with 2,000 neurons, ReLU nonlinearity, and zero bias trained with a linear SVM classifier readout. The SVM regularization strength was also swept over the range $[10^{-3}, \ldots, 10^3]$ by powers of 10 and selected via 5-fold cross-validation on the training set. The accuracy of the selected estimator was then computed on the test set. We compare both classical and mechanosensory receptive fields with bandwidth parameter `band` equivalent to $f_{\text{lo}} = f_1 - \texttt{band}$ and $f_{\text{hi}} = f_1 + \texttt{band}$ and no decay ($\gamma = \infty$).

The results are shown in Fig. 9. For either regression or classification, the best performance is acheived by the structured method with smallest bandwidth: error increases and accuracy degrades

with bandwidth. The unstructured kernel and unstructured random feature methods both only perform at chance levels, square error ≈ 1 or accuracy ≈ 0.5. Similar results occur with kernel SVM on the classification task, kernel support vector regression, and for kernels which are not tensor product kernels (not shown). Furthermore, the random feature network's limiting kernel is an arc-cosine kernel which is not a tensor product kernel.

These simulations show that simplified frequency detection still exhibits the main features of the other tasks we study while remaining amenable to theoretical analysis. The simulation results are not sensitive to details of the experiment and are qualitatively similar when assumptions of the theory are broken. Getting quantitative predictions of training and testing errors from the theory is left to future work.

A.3 Covariance parameter optimization

Here we describe the details of how the GP covariances were fit to our various datasets.

A.3.1 Mechanosensor covariance

We aim to minimize the difference between the matrix generated by the covariance model $\boldsymbol{C}_{\text{model}}$ and the data $\boldsymbol{C}_{\text{data}}$, while keeping f_{lo} smaller than f_{hi}. For simplicity, we measure the covariance mismatch with the Frobenius norm, solving

$$\begin{aligned} &\min_{f_{\text{lo}}, f_{\text{hi}}, \gamma} \|\boldsymbol{C}_{\text{model}}(f_{\text{lo}}, f_{\text{hi}}, \gamma) - \boldsymbol{C}_{\text{data}}\|_F \\ &\text{subject to: } f_{\text{hi}} \geq f_{\text{lo}}. \end{aligned} \tag{26}$$

We use the trust region algorithm provided by the `scipy.optimize.minimize` to solve (26).

A.3.2 V1 covariance

To fit the covariance model to the data, we formulate an optimization problem over the model parameters s and f, where we minimize the Frobenius norm of the difference between the covariance matrix $\boldsymbol{C}_{\text{model}}$ and $\boldsymbol{C}_{\text{data}}$:

$$\min_{s,f} \|\boldsymbol{C}_{\text{model}}(s, f) - \boldsymbol{C}_{\text{data}}\|_F. \tag{27}$$

We solve (27) using the Broyden–Fletcher–Goldfarb–Shannon (BFGS) algorithm provided by the `scipy.optimize.minimize` package.

A.4 Null receptive field models

We construct null GP models for both mechanosensors and V1 for comparison.

A.4.1 Mechanosensor covariance

We compare the data covariance matrix with the unstructured model and the Fourier model (Eq. 5). For the Fourier model, we fit the f_{hi} and f_{lo} parameters to data by minimizing the Frobenius error in the covariance matrix, finding $f_{lo} = 75$ Hz, $f_{hi} = 200$ Hz. The resulting covariance matrices are shown in Fig. 10 and Fig. 11. Receptive field samples from the null models are shown in Fig. 12. The samples from the Fourier model are smooth but do not decay in time like biological receptive fields.

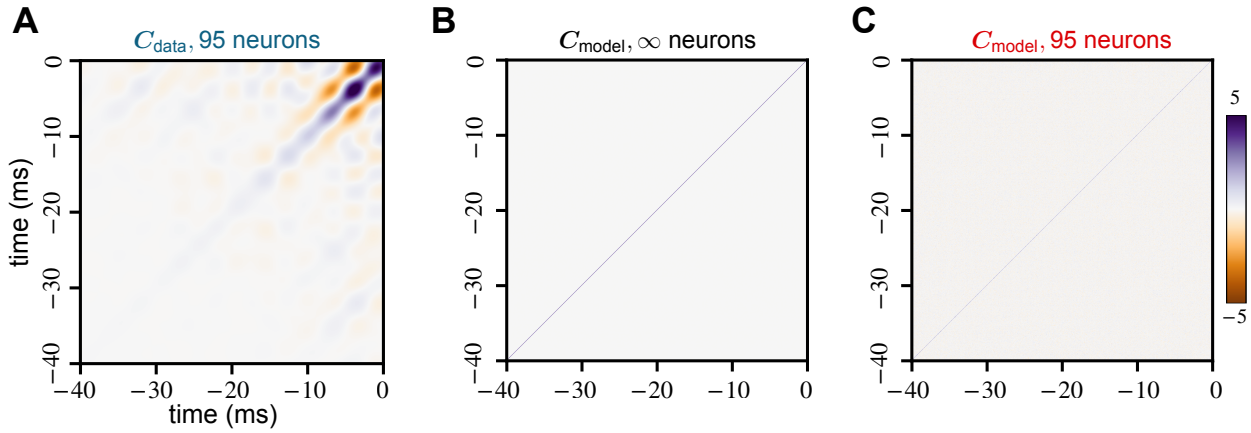

Figure 10: **Covariance matrix of mechanosensory receptive fields and unstructured model.** We compare the covariance matrices generated from the (A) receptive fields of 95 mechanosensory neurons, (B) unstructured GP model and (C) 95 random samples from the model.

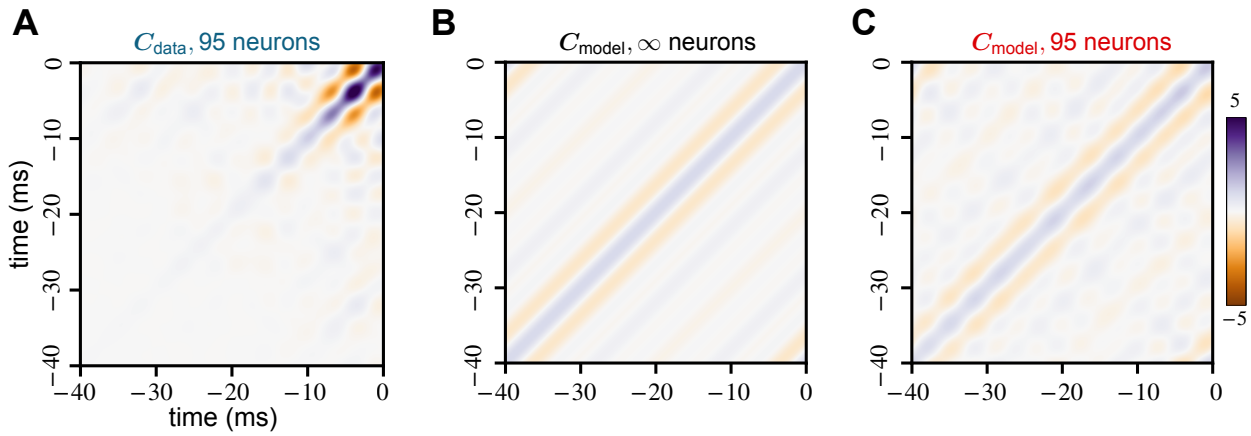

Figure 11: **Covariance matrix of mechanosensory receptive fields and the Fourier model (5).** We compare the covariance matrices generated from the (A) receptive fields of 95 mechanosensory neurons, (B) Fourier GP model and (C) 95 random samples from the model.

A.4.2 V1 covariance

We compare the data covariance matrix with the unstructured model and the translation invariant version of the V1 model (9). For the translation invariant model, we remove the localizing exponential and only fit the spatial frequency parameter, f (finding $f = 0.73$ pixels). The neuron weights are generated from a covariance function of the following form:

$$C(\boldsymbol{t}, \boldsymbol{t}') = \overbrace{\exp\left(-\frac{\|\boldsymbol{t} - \boldsymbol{t}'\|^2}{2f^2}\right)}^{\text{smooth receptive fields}} \tag{28}$$

The resulting covariance matrices are shown in Fig. 13 and Fig. 14. Receptive field samples from the null models are show in Fig. 15.

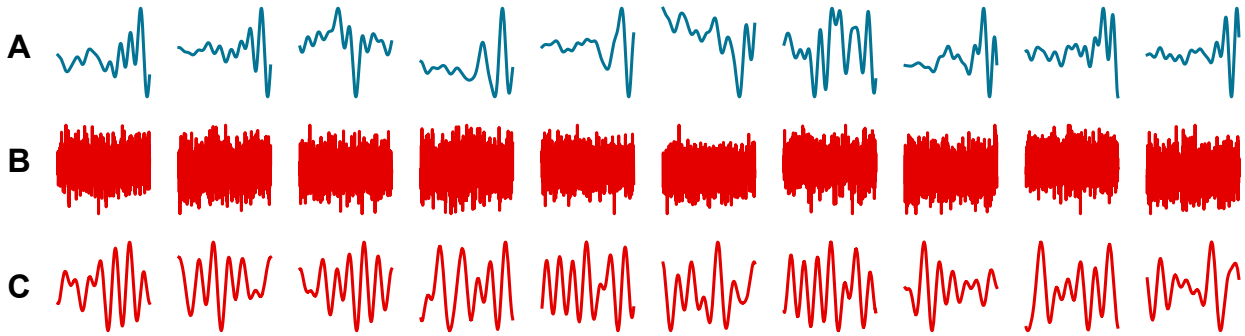

Figure 12: **Receptive fields from mechanosensory neurons, the unstructured model and the Fourier model (5).** We show the receptive fields from the (A) mechanosensory neurons, (B) unstructured GP model and (C) the Fourier GP model.

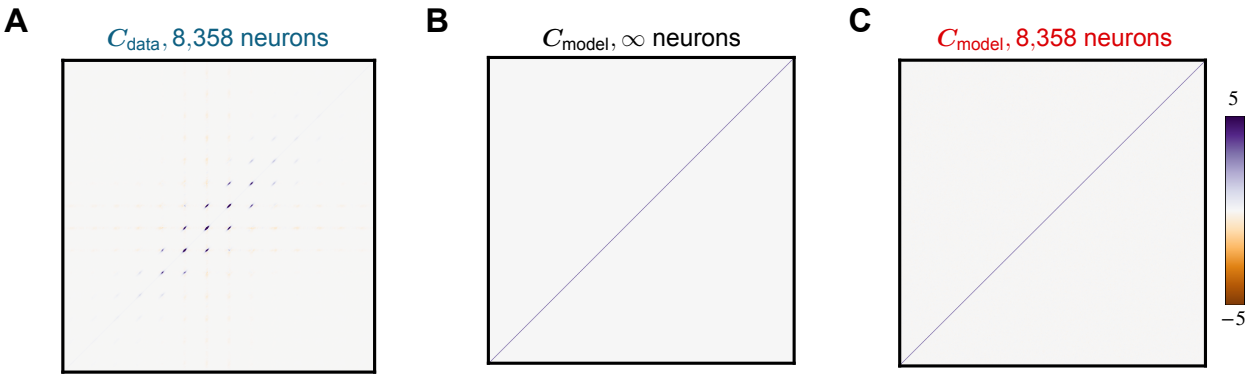

Figure 13: **Covariance matrix of V1 receptive fields and unstructured model for white noise stimuli.** We compare the covariance matrices generated from the (A) receptive fields of 8,358 mice V1 neurons, (B) unstructured GP model and (C) 8,358 random samples from the model.

Figure 14: **Covariance matrix of V1 receptive fields and translation invariant V1 model (Eq. 28) for white noise stimuli.** We compare the covariance matrices generated from the (A) receptive fields of 8,358 mice V1 neurons, (B) translation invariant version of the V1 GP model and (C) 8,358 random samples from the model.

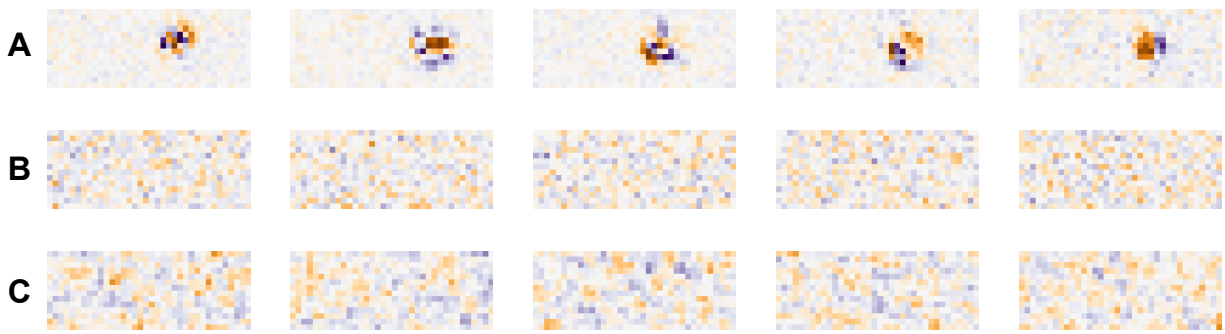

Figure 15: **Receptive fields from V1 neurons, the unstructured model and the translation invariant V1 model (28).** We show the receptive fields from the (A) V1 neurons, (B) unstructured GP model and (C) the translation invariant V1 GP model.

A.5 Derivation of eigenfunctions of V1 covariance function

The covariance between two pixel locations $\boldsymbol{t} = (t_1, t_2), \boldsymbol{t}' = (t_1', t_2') \in \boldsymbol{R}^2$ is given by

$$\begin{aligned} C(\boldsymbol{t}, \boldsymbol{t}') &= e^{-\frac{\|\boldsymbol{t}-\boldsymbol{t}'\|^2}{2f^2}} e^{-\frac{\|\boldsymbol{t}\|^2+\|\boldsymbol{t}'\|^2}{2s^2}} \\ &= e^{-\frac{(t_1-t_1')^2}{2f^2}} e^{-\frac{(t_1^2+t_1'^2)}{2s^2}} \cdot e^{-\frac{(t_2-t_2')^2}{2f^2}} e^{-\frac{(t_2^2+t_2'^2)}{2s^2}} \\ &= \prod_{i=1}^{2} e^{-\alpha(t_i-t_i')^2} e^{-\delta(t_i^2+t_i'^2)} \qquad \text{for } \alpha := \frac{1}{2f^2},\ \delta := \frac{1}{2s^2}. \end{aligned}$$

Since covariance function factors into a product of functions of variables t_1, t_1' and t_2, t_2', the multidimensional eigenfunctions $\phi_{\boldsymbol{k}}(\boldsymbol{t})$ and eigenvalues $\lambda_{\boldsymbol{k}}^2$ also factor into a product of 1-dimensional eigenfunction and eigenvalues, i.e. $\phi_{\boldsymbol{k}}(\boldsymbol{t}) = \prod_{i=1}^2 \phi_{k_i}(t_i)$ and $\lambda_{\boldsymbol{k}}^2 = \prod_{i=1} \lambda_{k_i}^2$. This holds for $d > 2$ dimensions as well. So we work in 1-d and search for eigenfunctions and eigenvalues such that,

$$\int_{-\infty}^{\infty} C(t, t')\phi_k(t)dt = \lambda_k^2 \phi_k(t'),$$

with $C(t, t') = e^{-\alpha(t-t')^2} e^{-\delta(t^2+t'^2)}$.

We make the ansatz that $\phi_k(t) = e^{-c_1 t^2} H_k(c_2 t)$, where H_k is the kth Hermite polynomial (physicists' convention) [8] and c_1, c_2 are constants. With this guess for the eigenfunctions,

$$\begin{aligned} \int_{-\infty}^{\infty} C(t, t')\phi_k(t)dt &= \int_{-\infty}^{\infty} e^{-\alpha(t-t')^2} e^{-\delta(t^2+t'^2)} e^{-c_1 t^2} H_k(c_2 t)dt \\ &= \int_{-\infty}^{\infty} e^{-Xt^2+Yt-t'^2(\alpha+\delta)} H_k(c_2 t)dt && (X := \alpha+\delta+c_1, Y := 2\alpha t') \\ &= e^{-t'^2(\alpha+\delta)+\frac{Y^2}{4X}} \int_{-\infty}^{\infty} e^{-(\sqrt{X}t-\frac{Y}{2\sqrt{X}})^2} H_k(c_2 t)dt && \text{(completing the square)} \\ &= \frac{1}{\sqrt{X}} e^{-t'^2(\alpha+\delta)+\frac{Y^2}{4X}} \int_{-\infty}^{\infty} e^{-\left(u-\frac{Y}{2\sqrt{X}}\right)^2} H_k(u\frac{c_2}{\sqrt{X}}) && (u = \sqrt{X}t) \\ &= \underbrace{\sqrt{\frac{\pi}{X}}\left(1-\frac{c_2^2}{X}\right)^{k/2}}_{\lambda_k^2} \underbrace{e^{-t'^2(\alpha+\delta-\frac{\alpha^2}{X})} H_k\left(\frac{c_2\alpha}{X(1-\frac{c_2^2}{X})^{1/2}}t'\right)}_{\hat{\phi}_k(t')}. && \text{([8], 7.374.8)} \end{aligned}$$

Solving for the unknown constants leads to the equations

$$c_1 = \alpha + \delta - \frac{\alpha^2}{X} \qquad \implies c_1 = \sqrt{\delta(2\alpha+\delta)},$$

$$c_2 = \frac{c_2\alpha}{X\left(1-\frac{c_2^2}{X}\right)^{1/2}} \qquad \implies c_2 = \sqrt{(\alpha+\delta+c_1)\left(1-\frac{\alpha^2}{(\alpha+\delta+c_1)}\right)} = \sqrt{2c_1}.$$

The last step is to find the normalization constant for the eigenfunctions:

$$\begin{aligned}\int_{-\infty}^{\infty} |\hat{\phi}_k(t)|^2 dt &= \int_{\infty}^{\infty} e^{-2c_1t^2} H_k^2(c_2t)dt \\ &= \frac{1}{c_2}\int_{\infty}^{\infty} e^{-u^2} H_k^2(u)du && (u = c_2t,\ c_2^2 = 2c_1) \\ &= \frac{2^k k!\sqrt{\pi}}{c_2}. && ([8],\ 7.374.1)\end{aligned}$$

Therefore, our orthonormal eigenfunctions and eigenvalues for the 1-dimensional covariance are

$$\phi_k(t) = \frac{c_2}{2^k k!\sqrt{\pi}} e^{-c_1t^2} H_k(c_2t), \qquad \lambda_k^2 = \sqrt{\frac{\pi}{\alpha+\delta+c_1}}\left(1-\frac{c_2^2}{\alpha+\delta+c_1}\right)^{k/2}, \tag{29}$$

where $c_1 = \sqrt{\delta(2\alpha+\delta)}, c_2 = \sqrt{2c_1}$. Note that $\lambda_k^2 \propto c_3^k$ with $c_3 = \sqrt{1-\frac{c^2}{\alpha+\delta+c_1}}$, so that the spectrum decays exponentially.

A.6 Distributed receptive field centers imply a sum kernel space

To generate our V1-inspired weights, we first sample a center $\boldsymbol{c}$ uniformly at random from the pixels in the image; call this set of pixels S. We will now derive the kernel for this weight sampling.

Suppose that all of the weights are sampled with a single center $\boldsymbol{c}$. Then (12) tells us that the structured kernel associated with the RFN

$$\begin{aligned}k_{\text{struct}}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{c}) &= k_{\text{unstruct}}(\boldsymbol{\Lambda_c}\boldsymbol{\Phi_c}^T\boldsymbol{x}, \boldsymbol{\Lambda_c}\boldsymbol{\Phi_c}^T\boldsymbol{x}') \\ &= k_{\text{unstruct}}(\tilde{\boldsymbol{x}}_{\boldsymbol{c}}, \tilde{\boldsymbol{x}}'_{\boldsymbol{c}}),\end{aligned} \tag{30}$$

where we have defined the *local basis change*

$$\tilde{\boldsymbol{x}}_{\boldsymbol{c}} = \boldsymbol{\Lambda_c}\boldsymbol{\Phi_c}^T\boldsymbol{x}. \tag{31}$$

This local basis change projects into a basis of Hermite wavelets $\boldsymbol{\Phi_c}$ centered at $\boldsymbol{c}$ and filters according to the eigenvalues $\boldsymbol{\Lambda_c}$. The reproducing kernel (30) defines an RKHS of functions $\mathcal{H}_{\boldsymbol{c}}$ which take images as their input and produce a real-valued output. The RKHS is a Hilbert space and thus has a norm $\|\cdot\|_{\mathcal{H}_{\boldsymbol{c}}}$. Functions with small $\mathcal{H}_{\boldsymbol{c}}$-norm are, informally, smooth functions of the local wavelet coefficients $\tilde{\boldsymbol{x}}_{\boldsymbol{c}}$.

In our experiments, we actually sample weights from all centers $\boldsymbol{c} \in S$ with equal probability. Taking the expectation over the centers, this means that the kernel will be an average over all of the local kernels (30),

$$k_{\text{struct}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{|S|}\sum_{\boldsymbol{c}\in S} k_{\text{struct}}(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{c}). \tag{32}$$

Let $\mathcal{H}$ be the RKHS associated with $k_{\text{struct}}(\cdot,\cdot)$, another space of functions that take in images and output a real number. The sum (32) implies that $\mathcal{H} = \bigoplus_{\boldsymbol{c}\in S} \mathcal{H}_{\boldsymbol{c}}$, i.e. the RKHS is a direct sum of local RKHS's [83]. This means that any function $f \in \mathcal{H}$ can be written as $f = \sum_{\boldsymbol{c}\in S} f_{\boldsymbol{c}}$, with every $f_{\boldsymbol{c}} \in \mathcal{H}_{\boldsymbol{c}}$. The norm of this function comes from taking a minimum over all such decompositions

$$\|f\|_{\mathcal{H}} = \min_{f_{\boldsymbol{c}}: f=\sum_{\boldsymbol{c}\in S} f_{\boldsymbol{c}}} \sqrt{|S| \sum_{\boldsymbol{c}\in S} \|f_{\boldsymbol{c}}\|^2_{\mathcal{H}_{\boldsymbol{c}}}}.$$

We can think of functions with small $\mathcal{H}$-norm, which will be easiest to learn, as sums of smooth functions of local wavelet coefficients.

A.7 Timeseries data generation

We detail how the two frequency classification tasks from Sections 2.3.1 and 2.3.2 are generated. In both tasks, each example is an L ms timeseries sampled at f Hz, making each $\boldsymbol{x}$ a vector of length $d = L \times f$. Thus in the discrete setting, we only have d total frequencies. While the math below show continuous signals, in our code we generate analagous discrete signals using the discrete Fourier transform basis.

A.7.1 Frequency detection

The frequency detection task from Section 2.3.1 is a binary classification task. The positive examples contain a pure sinusoidal signal with frequency f_1 and additive Gaussian noise. The negative examples are just white noise. They are generated in the following way:

$$x_+(t) = a \overbrace{\sqrt{\frac{2}{L}} \left(\eta_f \cos(f_1 t) - \xi_f \sin(f_1 t)\right)}^{\text{pure frequency}} + \overbrace{\sqrt{1-a^2} \sqrt{\frac{2}{(d-1)L}} \sum_{\substack{j=0 \\ j:\omega_j \neq f_1}}^{d-1} \left(\eta_j \cos(\omega_j t) - \xi_j \sin(\omega_j t)\right)}^{\text{additive Gaussian noise}}$$

$$x_-(t) = \sqrt{\frac{2}{dL}} \sum_{j=0}^{d-1} \left(\eta_j \cos(\omega_j t) - \xi_j \sin(\omega_j t)\right)$$

where $\omega_j = 2\pi j/L$ is the j-th natural frequency, a is a parameter that sets the SNR, and the coefficients $\eta_j, \xi_j, \eta_f, \xi_f$ are random variables uniformly sampled from the unit circle (which gives each frequency component a random phase).

We define

$$\text{SNR} := \frac{a^2}{1-a^2}, \tag{33}$$

with $a \in [0,1]$. Larger a means a larger contribution of the pure tone and smaller amplitude noise. Note that $\|x_-(t)\|^2_{L^2([0,L])} = \|x_+(t)\|^2_{L^2([0,L])} = 1$. The generation process ensures that the L^2 energy of both the negative and positive examples are matched and that the SNR is equal to the ratio of energy captured in frequency f_{lo} to the total energy in all other components.

We generate a balanced dataset with 7,000 timeseries signals which we split into a training set with 5,600 examples and a test set with 1,400 examples. We tuned the GP covariance parameters f_{lo}, f_{hi}, and γ from (8) using 3-fold cross validation on the training set. We found the optimal parameters for a network with 20 hidden neurons and used them for all hidden layer widths. We tested f_{lo} and f_{hi} parameters from 10 Hz to 200 Hz at increments of 10 Hz. For γ, we set the parameter range to be from 10 ms to 100 ms and used all parameters at increments of 10 ms. Using

grid search, we tested all combinations of these parameters. The optimal model was refit using all training 5,600 samples, and the errors we report were measured on the test set.

A.7.2 Frequency XOR

We use a similar set up to generate the timeseries for the frequency exclusive-or (XOR) task in Section 2.3.2. The positive examples are either frequency f_1 or f_2 Hz pure sinusoids with additive Gaussian noise. The negative examples are either mixed frequency timeseries (with both f_1 and f_2 Hz signals) or pure Gaussian noise. They are generated in the following way:

$$x_{+,k}(t) = \overbrace{a\sqrt{\frac{2}{L}}\left(\eta_f \cos(f_k t) - \xi_f \sin(f_k t)\right)}^{\text{pure frequency}} + \overbrace{\sqrt{\frac{2(1-a^2)}{(d-1)L}} \sum_{\substack{j=0 \\ j:\omega_j \neq f_j}}^{d-1} \left(\eta_j \cos(\omega_j t) - \xi_j \sin(\omega_j t)\right)}^{\text{additive Gaussian noise}}$$

$$x_{-,\text{noise}}(t) = \sqrt{\frac{2}{dL}} \sum_{j=0}^{d-1} \left(\eta_j \cos(\omega_j t) - \xi_j \sin(\omega_j t)\right)$$

$$x_{-,\text{mixed}}(t) = \frac{a}{\sqrt{L}} \sum_{j \in \{1,2\}} \left(\eta_j \cos(f_j t) - \xi_j \sin(f_j t)\right) + \sqrt{\frac{2(1-a^2)}{(d-2)L}} \sum_{\substack{j=0 \\ j:\omega_j \neq f_1, f_2}}^{d-1} \left(\eta_j \cos(\omega_j t) - \xi_j \sin(\omega_j t)\right),$$

for $k \in \{1, 2\}$ in the $x_{+,k}(t)$ function. The constants, random variables, and details of SNR are identical to the frequency detection section. The datasets we generate have balanced proportions of $x_{+,1}$, $x_{+,2}$, $x_{-,\text{noise}}$, and $x_{-,\text{mixed}}$ signals.

A.8 SNR amplification via filtering

Let's consider the simple frequency detection task with stationary bandpass features. Since these features are stationary, their eigenvectors are Fourier modes, i.e. $\mathbf{\Phi}$ is the discrete Fourier transform (DFT) matrix. Assume the features encode a bandpass filter, which means that $\mathbf{\Lambda} = \text{diag}(\lambda_i)$, $i \in \{0, \ldots, d-1\}$, with $\lambda_i = 1$ for $i_{\text{lo}} \leq i \leq i_{\text{hi}}$ and 0 otherwise.

In frequency detection, a single frequency component (discrete Fourier mode) contains the signal with energy a^2. The other $d-1$ components each have energy $\frac{1-a^2}{d-1}$, for a total energy of $1-a^2$ contained in this white noise. After the basis change (4) is applied to any input $\boldsymbol{x}$, the transformed vector $\tilde{\boldsymbol{x}} = \mathbf{\Lambda}\mathbf{\Phi}^*\boldsymbol{x}$ will have zeros in all entries outside the passband. (We use the conjugate transpose $\mathbf{\Phi}^*$ here rather than the transpose since the DFT matrix is complex; the interpretation is the same.) This makes the new representation $\tilde{\boldsymbol{x}}$ effectively d'-dimensional, where $d' = i_{\text{hi}} - i_{\text{lo}}$.

Now, first assume that the signal is within the passband. The total noise energy in the transformed representation becomes $(1-a^2)\frac{d'-1}{d-1}$, since one of the d' components is still signal, and no energy is lost in the retained components because $\mathbf{\Phi}$ is unitary. The overall noise is shrunk by a factor of $\frac{d'-1}{d-1}$, so the SNR gets boosted by $\frac{d-1}{d'-1}$. In the limiting case where $d' = 1$, the noise energy is 0 and SNR is infinite. On the other hand, if the signal lies outside the passband the SNR is reduced to 0.

A.9 Implementation details and code availability

In all experiments with RFNs, the training algorithm is an SVM classifier with squared hinge loss provided by the `sklearn.svm.LinearSVC` package and all other parameters set to their defaults.

We used `scipy` [17] and `numpy` [9] to construct both classical unstructured and neural-inspired structured weights. For the experiments with fully-trained networks, we used `pytorch` [12]. The cross entropy loss function was optimized using full batch stochastic gradient gescent (SGD) optimizer, i.e. gradient descent (GD). Our code is available at `https://github.com/BruntonUWBio/structured-random-features`.

A.10 Covariance of V1 neurons with other stimuli

We repeat the covariance analysis from Section 2.2.3 on three additional datasets of V1 neurons. Different stimuli were shown in each dataset to calculate the receptive field.

The first dataset was provided by Ringach et al. from their work on characterizing the spatial structure of simple receptive fields in macaque (*Macaca fascicularis*) V1 [13]. The spikes of 250 neurons were recorded in response to drifting sinusoidal gratings. The receptive fields were calculated from the stimuli and responses using subspace reverse correlation. Because of the bandlimited properties of sinusoidal stimuli, this experiment biases the reconstruction towards smooth receptive fields. The receptive fields were of various sizes: 32 pixels $\times$ 32 pixels, 64 pixels $\times$ 64 pixels, and 128 pixels $\times$ 128 pixels. We resized them to a common dimension of 32 pixels $\times$ 32 pixels using local mean averaging. We find the optimal covariance parameters that fit the data to be $s = 2.41$ and $f = 0.95$ pixels. The covariance matrices and eigenfunctions are shown in Fig. 16. Examples of biological receptive fields and random samples from the fitted model are shown in Fig. 17 in the Appendix.

The second dataset contains the responses of 69,957 neurons recorded from the primary visual cortex of mice bred to express GCaMP6s. We presented 5,000 static natural images of 24 $\times$ 27 pixels in random order for 3 trials each. We calculated the receptive fields from the natural images and calcium responses of cells using ridge regression with an ℓ^2 penalty set to 0.1 after each image pixel was z-scored across images. We used the average receptive field over all three trials. For the covariance analysis, we picked cells with SNR > 0.4. This gave us 10,782 cells. The optimal covariance parameters that fit the data are $s = 5.40$ and $f = 1.17$ pixels. Examples of biological receptive fields and random samples from the model are shown in Fig. 19. The covariance matrices and eigenfunctions are shown in Fig. 18. Examples of biological receptive fields and random samples from the fitted model are shown in Fig. 19. Repeating this analysis using receptive fields from individual trials yields identical results.

The third dataset contains the responses of 4,337 neurons also recorded from the primary visual cortex of mice bred to express GCaMP6s. The mice were shown static discrete Hartley transform (DHT, similar to a real-valued discrete Fourier transform) basis functions of size 30 $\times$ 80 pixels, and the calcium responses of neurons were recorded. The receptive fields were calculated using ridge regression without any ℓ^2 penalty. Here, we picked cells with SNR > 1 for analysis. We were left with 2,698 cells. The optimal covariance parameters that fit the data are $s = 10.46$ and $f = 1.20$ pixels. The covariance matrices and eigenfunctions are shown in Fig. 20. Examples of biological receptive fields and random samples from the fitted model are shown in Fig. 21.

A.11 Initialization of networks with structured weights

We show results of initializing fully trained neural networks across a range of network widths (50, 100, 400, and 1,000) and learning rates (10^{-3}, 10^{-2}, and 10^{-1}) in Figures 25, 26, 27, and 28.

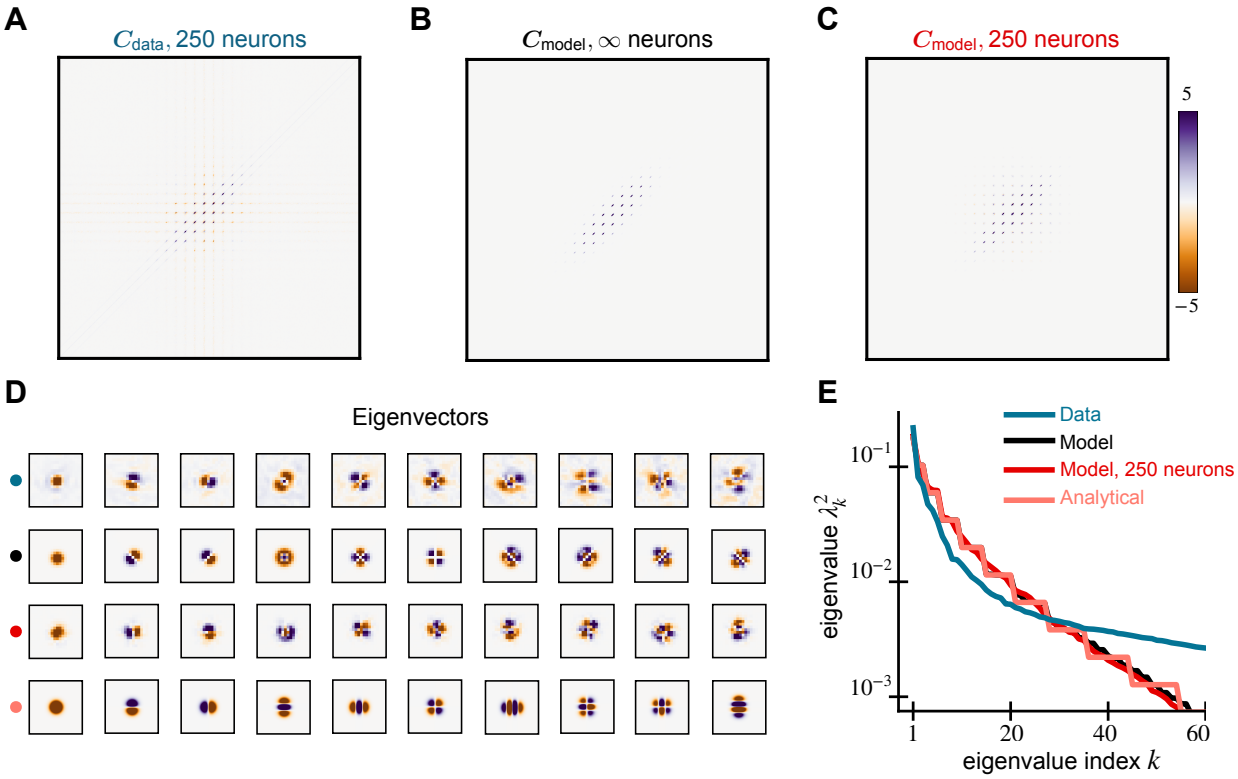

Figure 16: **Spectral properties of V1 receptive fields and our model for Ringach dataset.** We compare the covariance matrices generated from the (A) receptive fields of 250 macaque V1 neurons, (B) the GP model (9), and (C) 250 random samples from the model. The data is from [13]. (D) The leading 10 eigenvectors of the data and model covariance matrices show similar structure and explain 57% of the variance in the data. Analytical Hermite wavlet eigenfunctions are in the last row. (E) The eigenspectrum of the model matches well with the data.

A.12 Deep network experiments

We experimented with using the V1-inspired weight initialization in the first two convolutional layers of AlexNet [50] and training on the ImageNet Large Scale Visual Recognition Challenge from 2012 [78]. Our implementation was based on the example provided by `pytorch` and `torchvision` [12] and used the same optimization routine, parameters, and schedule as in `https://github.com/pytorch/examples/tree/master/imagenet`.

All convolutional layers were initialized with weights drawn from a Gaussian distribution with variance $(c_{\text{in}} d_{\text{x}} d_{\text{y}})^{-1}$, where c_{in} was the number of input channels, and d_{x} and d_{y} are the dimensions of the filter. This is equal to the reciprocal of the fan-in. In the case of classical initialization, this Gaussian distribution has covariance proportional to the identity, whereas in the structured case we use the V1-inspired covariance centered in the center of the filter with independent draws for each input channel. All biases and weights in the other layers are set with their `pytorch` defaults. The structured weights were only used in the first two convolutional layers of dimensions $d_{\text{x}} \times d_{\text{y}} = 11{\times}11$ and $5{\times}5$. The size parameter was set to $s = \max(d_{\text{x}}, d_{\text{y}}) \cdot 3$ and frequency bandwidth was $f = \max(d_{\text{x}}, d_{\text{y}})/5$.

We show training and testing loss over the first 10 epochs for both the classical and structured initializations in Fig. 29. The structured initialization at first shows an advantage over classical,

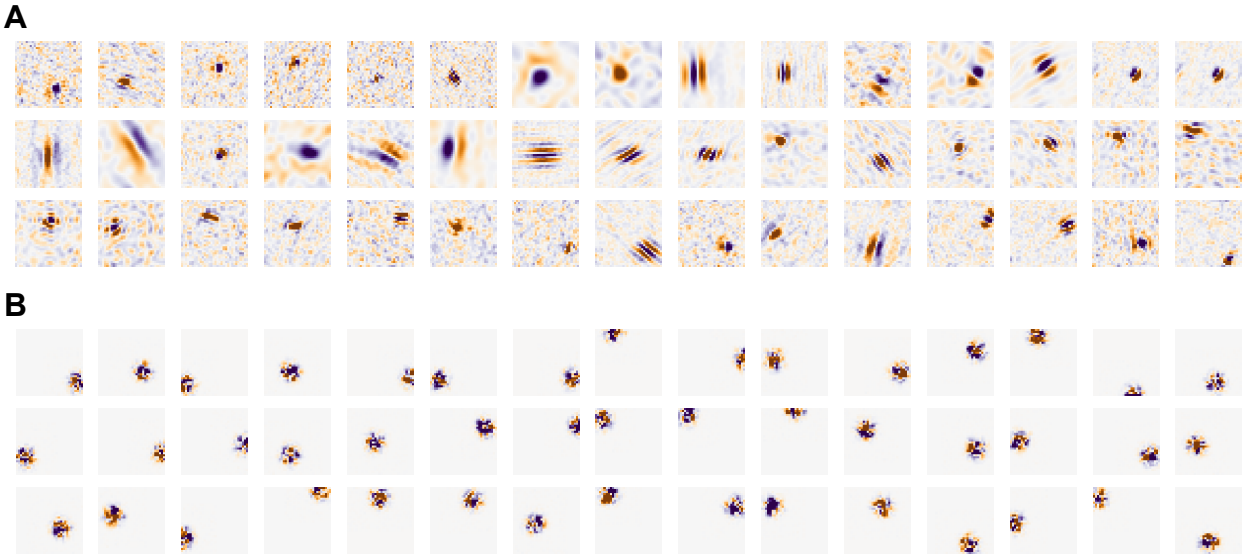

Figure 17: **Receptive fields of V1 neurons from the Ringach dataset.** We show (A) biological receptive fields and (B) random samples from the fitted covariance model.

Initialization	Loss	Top-1 accuracy (%)	Top-5 accuracy (%)
Classical	2.059 (1.907)	55.2 (56.5)	76.3 (79.2)
Structured	2.074 (1.920)	53.0 (56.4)	76.0 (79.0)

Table 3: **Results after training AlexNet for 90 epochs on ImageNet.** The classical initialization leads to slightly smaller loss and higher accuracy over the structured initialization. Test values are shown in parentheses; these are better than the training values due to dropout.

with consistently lower losses for the first 4 epochs, but eventually the classical network catches up. From this point onwards (until the 90 training epochs are complete), the classical network has the same or lower loss. Both networks end up performing well, reaching accuracies close to those reported in [50] and the `torchvision` documentation (`https://pytorch.org/vision/stable/models.html`), as shown in Table 3. The classical initialization performs slightly better overall.

These null results are perhaps not surprising: The initial layers of AlexNet contain only 64 and 192 output channels (i.e. filters) respectively, making up only a small fraction of the total weights in the network. The deeper convolutional layers contain many more channels and are built with small 3×3 filters where our initialization is unlikely to help. It is also possible that the effects of initialization are less important for overparametrized models or with large amounts of training data.

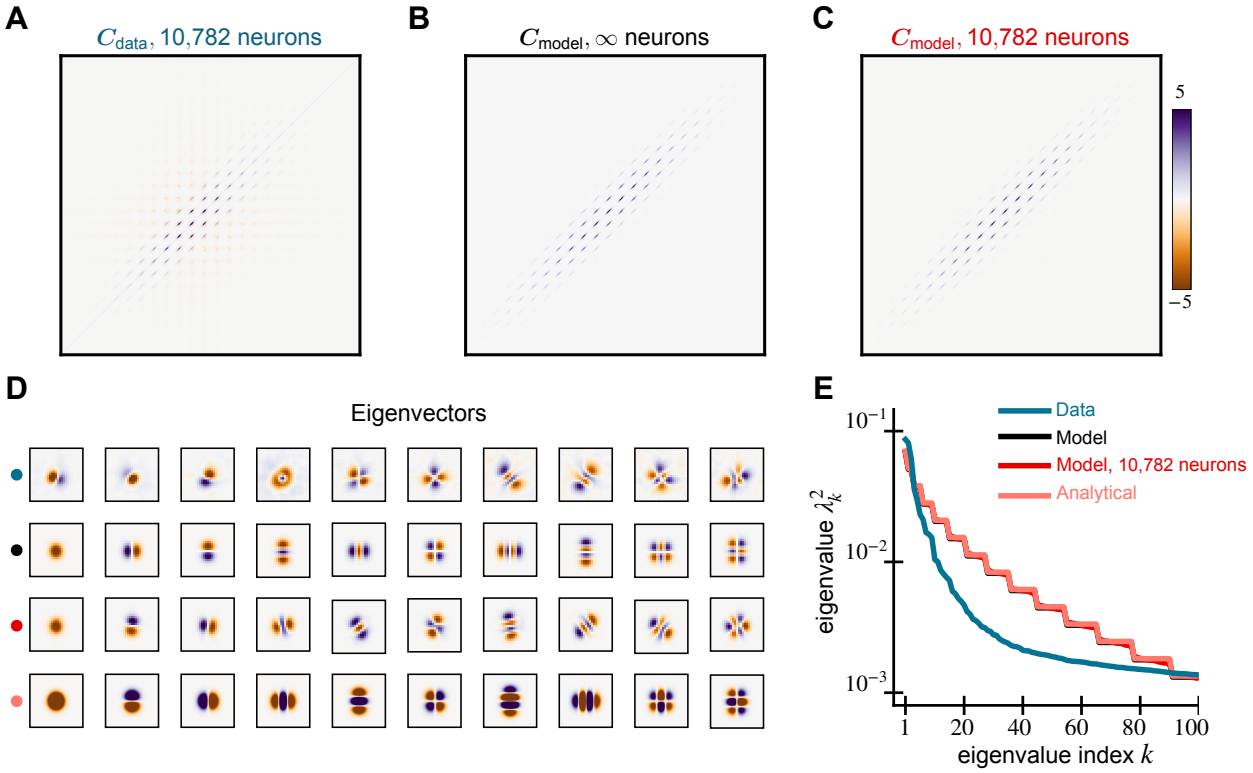

Figure 18: **Spectral properties of V1 receptive fields and our model for natural image stimuli.** We compare the covariance matrices generated from the (A) receptive fields of 10,782 mice V1 neurons, (B) the GP model (9), and (C) 10,782 random samples from the model. (D) The leading 10 eigenvectors of the data and model covariance matrices show similar structure and explain 39% of the variance in the data. Analytical Hermite wavelet eigenfunctions are in the last row. (E) The eigenspectrum of the model compared to the data.

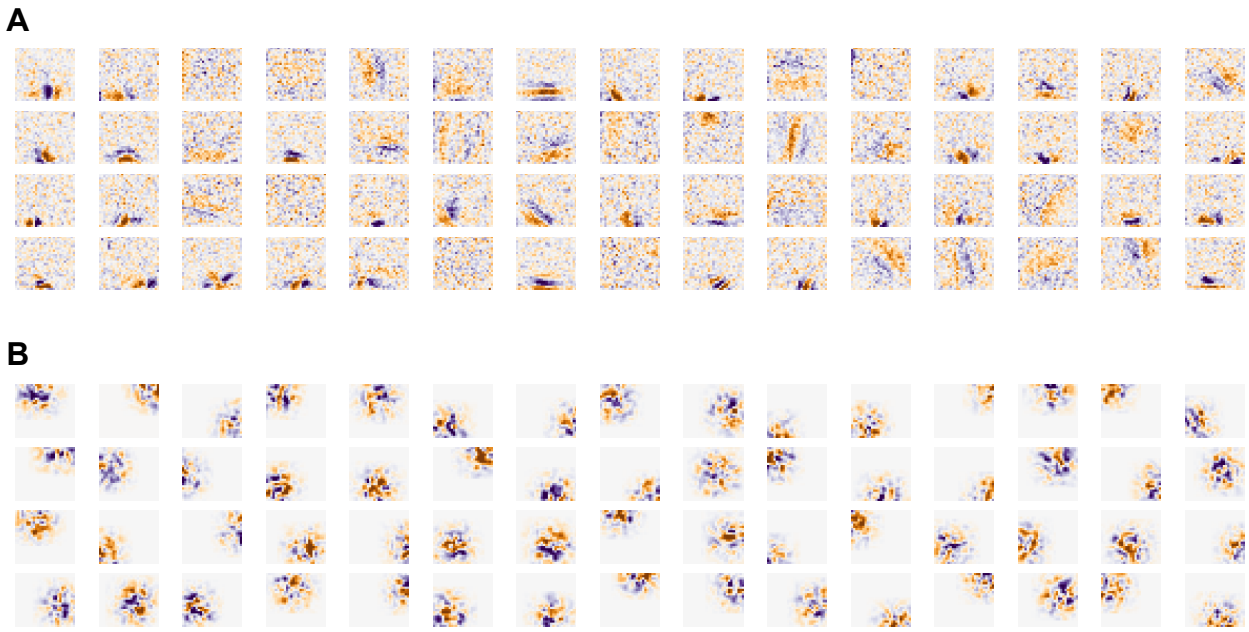

Figure 19: **Receptive fields of V1 neurons from natural images stimuli.** We show (A) biological receptive fields and (B) random samples from the fitted covariance model.

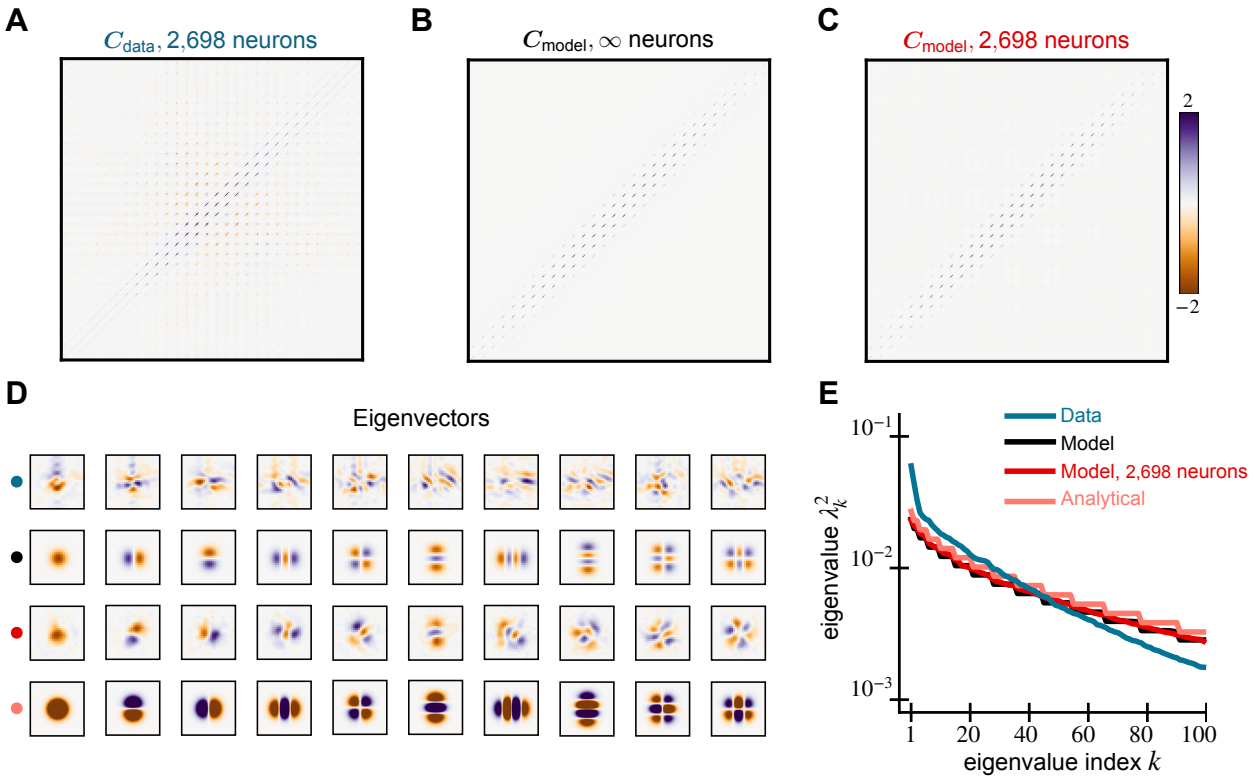

Figure 20: **Spectral properties of V1 receptive fields and our model for DHT stimuli.** We compare the covariance matrices generated from the (A) receptive fields of 2,698 mice V1 neurons, (B) the GP model (9), and (C) 2,698 random samples from the model. (D) The leading 10 eigenvectors of the data and model covariance matrices. They explain 29% of the variance in the data. Analytical Hermite wavelet eigenfunctions are in the last row. (E) The eigenspectrum of the model matches well with the data.

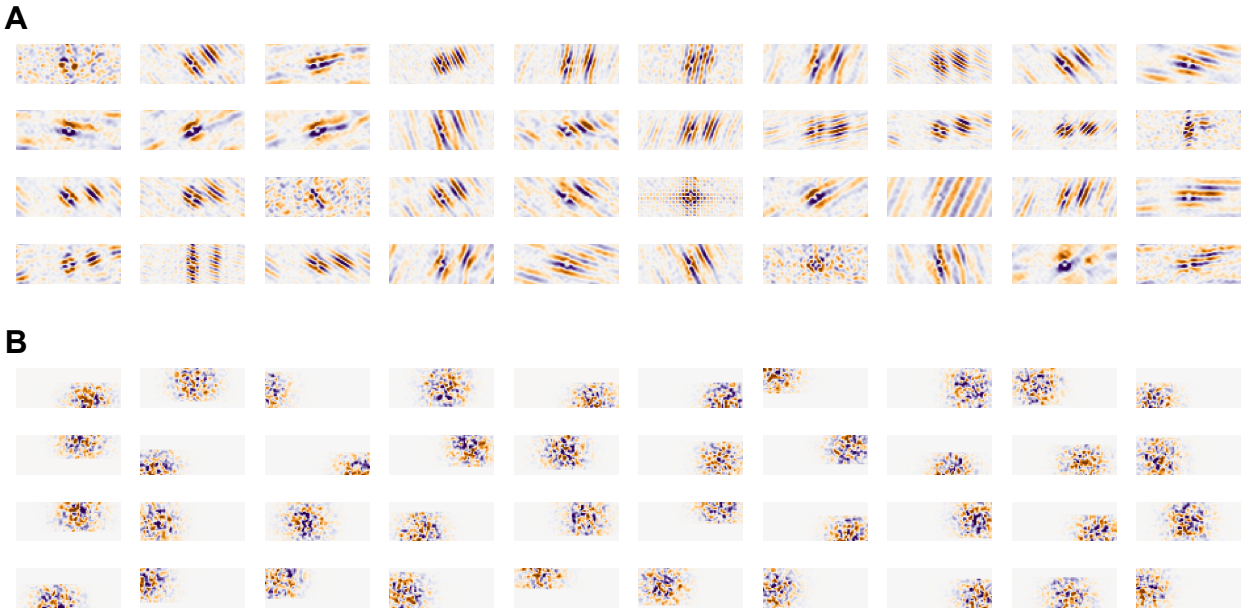

Figure 21: **Receptive fields of V1 neurons from DHT stimuli.** We show (A) biological receptive fields and (B) random samples from the fitted covariance model.

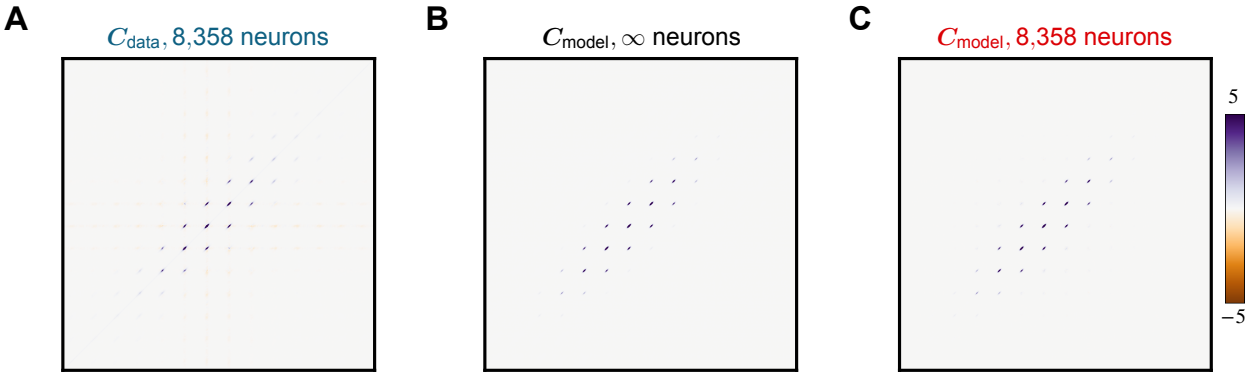

Figure 22: **Covariance matrix of V1 receptive fields and our model for white noise stimuli.** We show the full structure of the covariance matrices shown in Fig. 5, which are the 180×180 pixel region around the centers of these 504×504 pixel matrices. .These matrices are generated from the (A) receptive fields of 8,358 mouse V1 neurons, (B) the GP model (9), and (C) 8,358 random samples from the model.

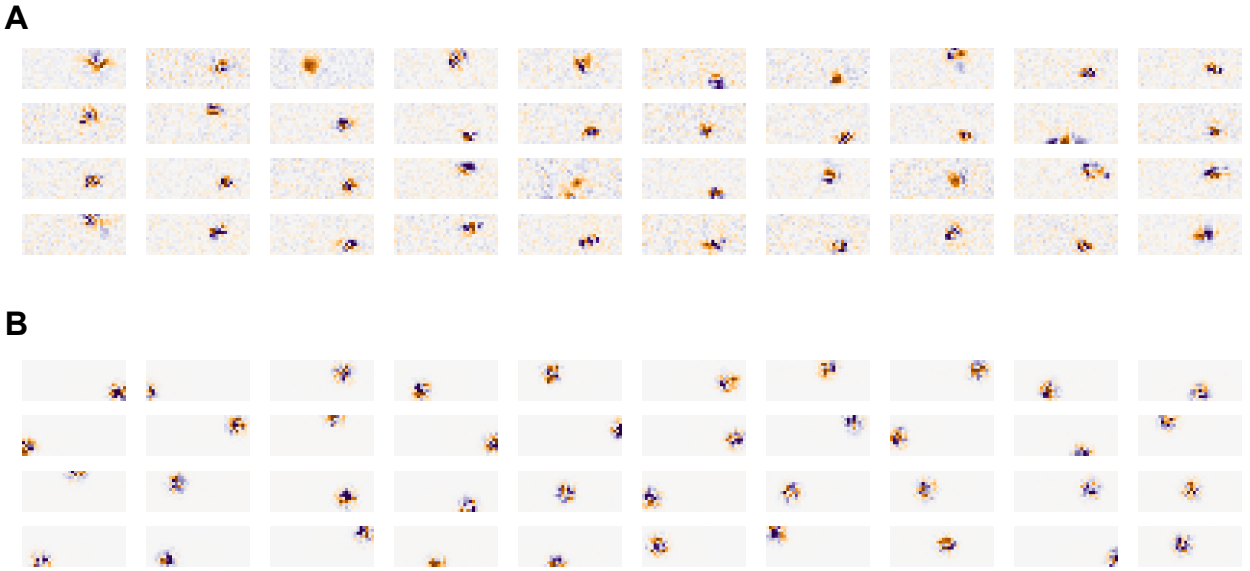

Figure 23: **Receptive fields of V1 neurons from white noise stimuli.** We show (A) biological receptive fields and (B) random samples from the fitted covariance model.

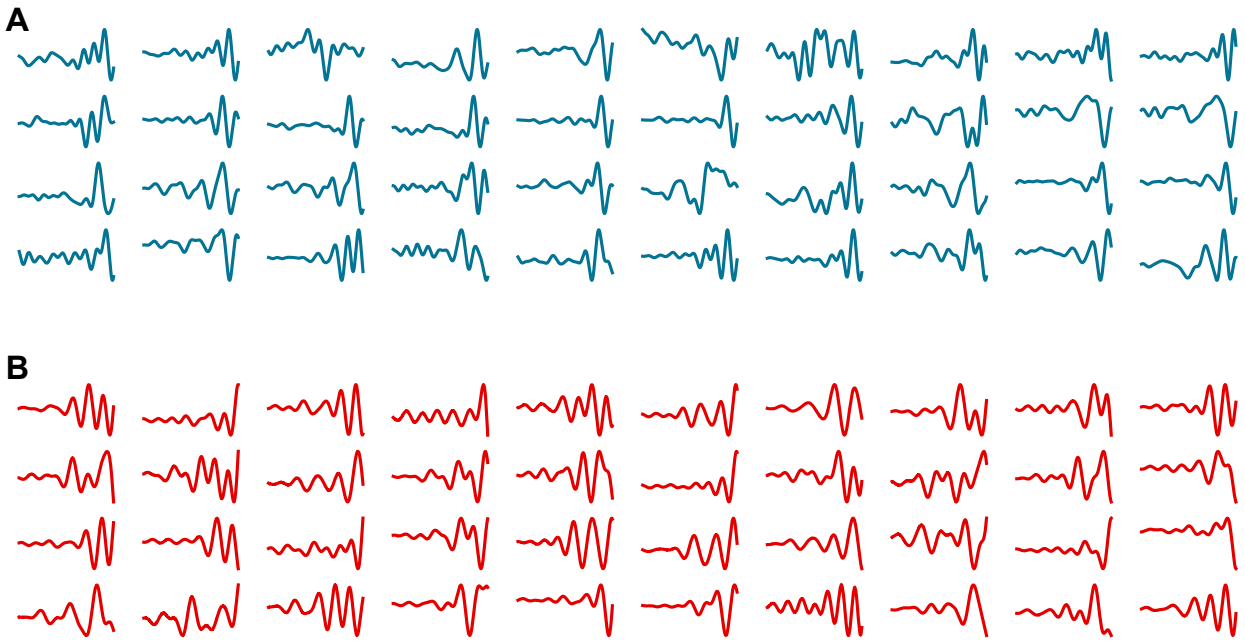

Figure 24: **Receptive fields of mechanosensory neurons.** We show (A) biological receptive fields and (B) random samples from the fitted covariance model.

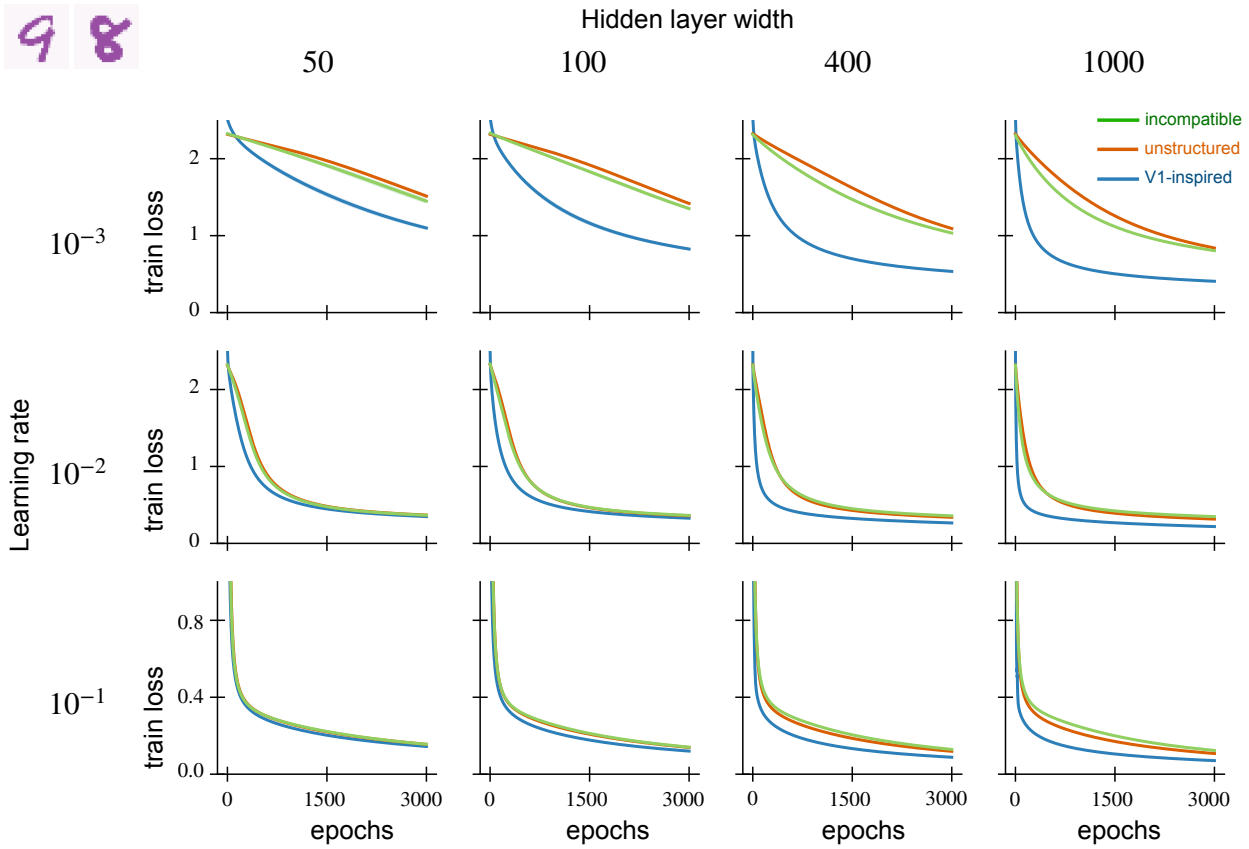

Figure 25: **Training loss on MNIST for fully-trained neural networks initialized with V1 weights.** We show the average training loss of fully-trained networks against the number of training epochs across diverse hidden layer widths (50, 100, 400, and 1000) and learning rates ($10^{-1}, 10^{-2}$, and 10^{-3}). For every hidden layer width, we generate five random networks and average their performance. The solid lines show the average training loss while the shaded region represents the standard error. When the covariance parameters are tuned properly, V1-initialized networks achieve lower training loss over fewer epochs. The benefits are more significant at larger network widths and lower learning rates. With incompatible weights, V1 initialization leads to similar performance as unstructured initialization.

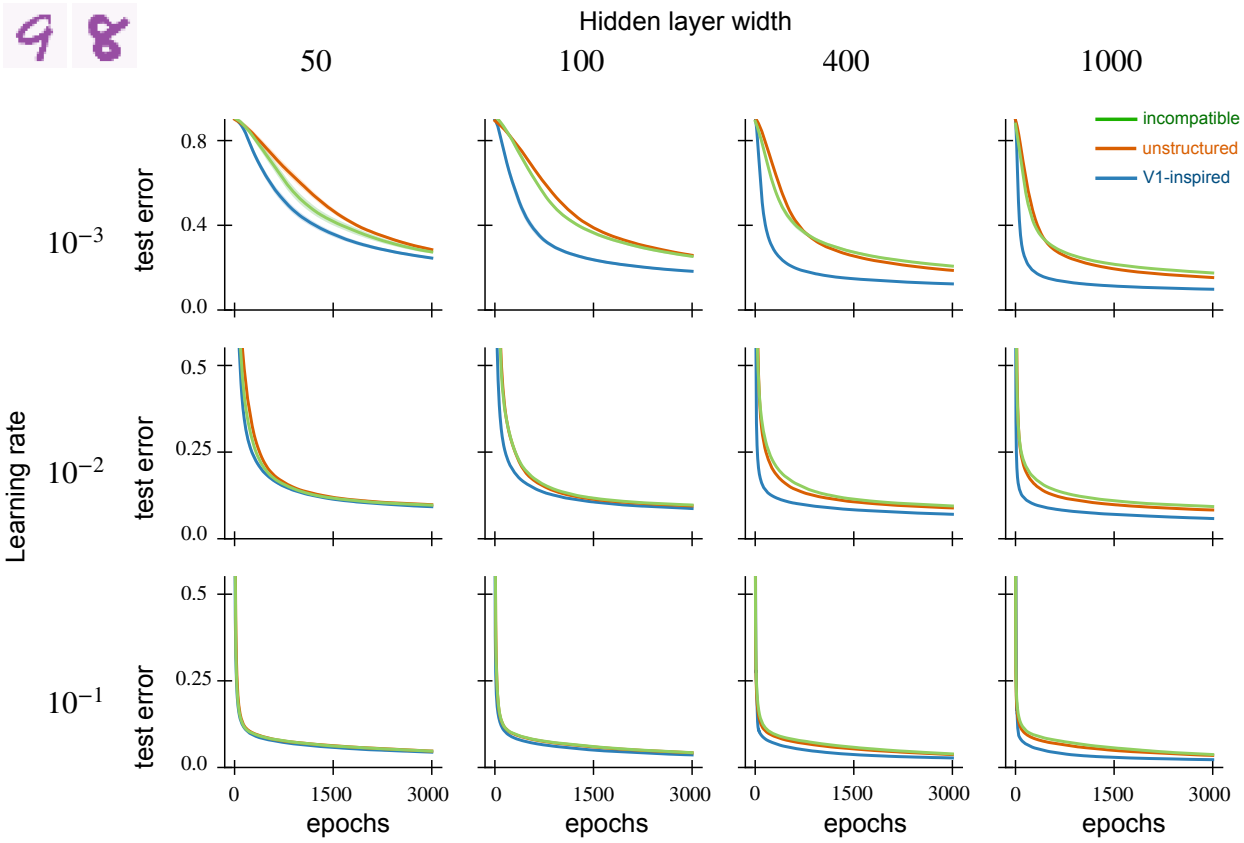

Figure 26: **Test error on MNIST for fully-trained neural networks initialized with V1 weights.** We show the average test error of fully-trained networks against the number of training epochs across diverse hidden layer widths (50, 100, 400, and 1000) and learning rates ($10^{-1}, 10^{-2}$, and 10^{-3}). For every hidden layer width, we generate five random networks and average their performance. The solid lines show the average test error while the shaded regions represent the standard error. When the covariance parameters are tuned properly, V1-initialized networks achieve lower test error over fewer epochs. The benefits are more significant at larger network widths and lower learning rates. With incompatible weights, V1 initialization leads to similar performance as unstructured initialization.

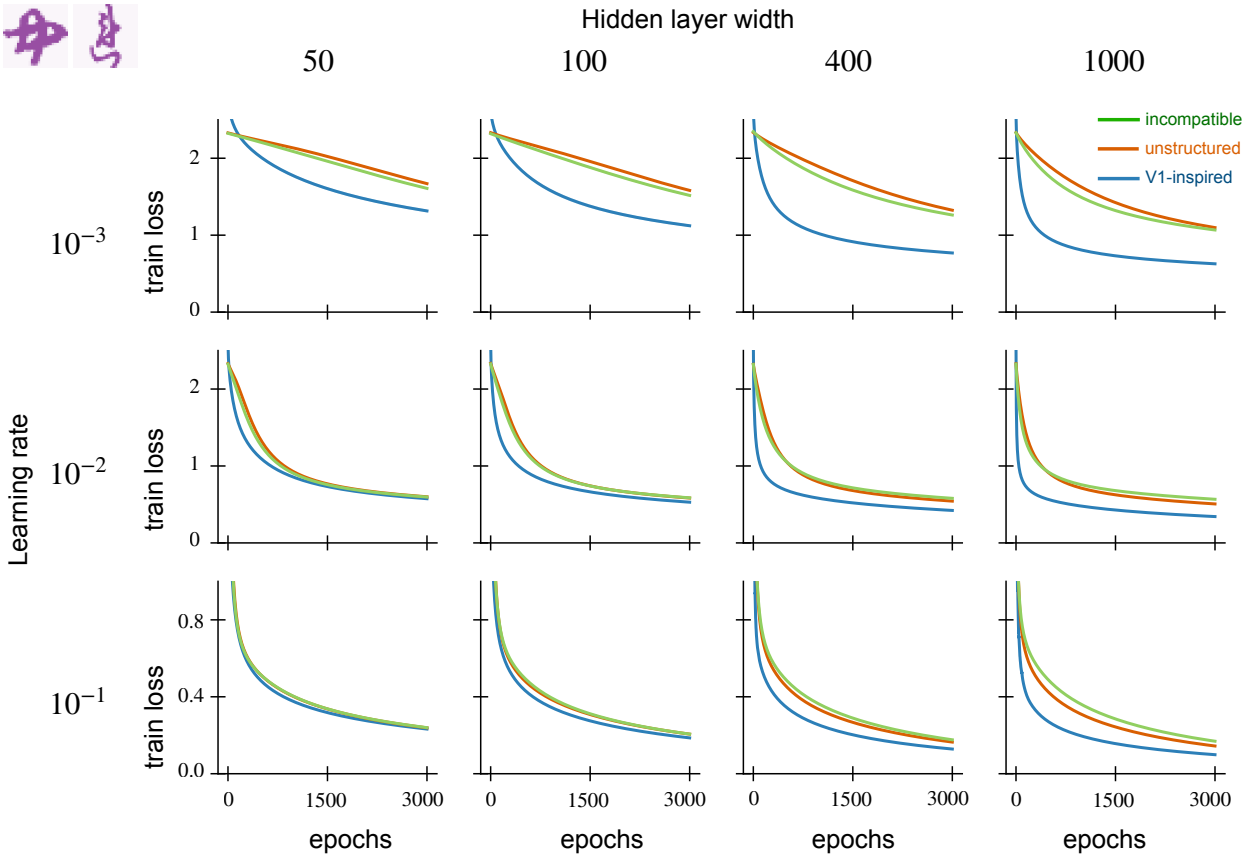

Figure 27: **Training loss on KMNIST for fully-trained neural networks initialized with V1 weights.** We show the average training loss of fully-trained networks against the number of training epochs across diverse hidden layer widths (50, 100, 400, and 1000) and learning rates ($10^{-1}, 10^{-2}$, and 10^{-3}). For every hidden layer width, we generate five random networks and average their performance. The solid lines show the average training loss while the shaded regions represent the standard error. When the covariance parameters are tuned properly, V1-initialized networks achieve lower training loss over fewer epochs. The benefits are more significant at larger network widths and lower learning rates. With incompatible weights, V1 initialization leads to similar performance as unstructured initialization.

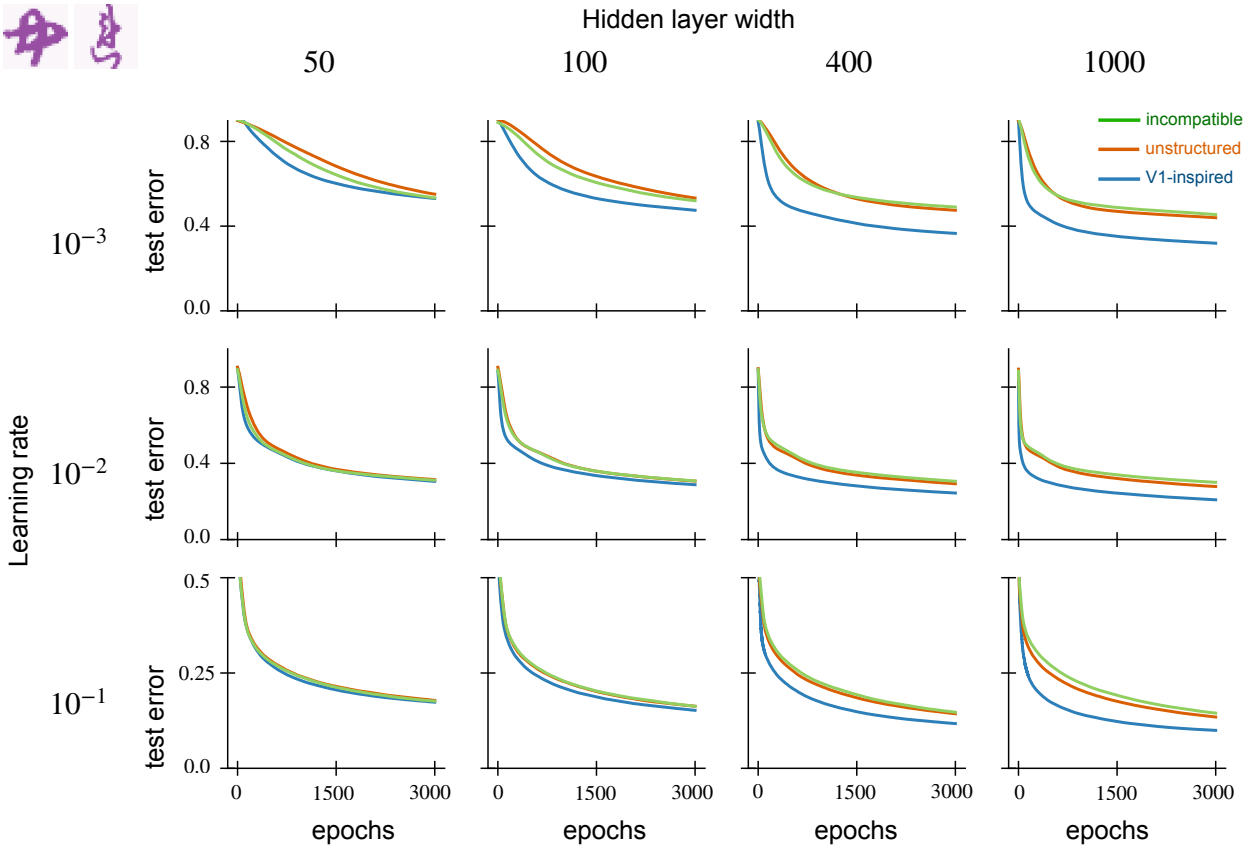

Figure 28: **Test error on KMNIST for fully-trained neural networks initialized with V1 weights.** We show the average test error of fully-trained networks against the number of training epochs across diverse hidden layer widths (50, 100, 400, and 1000) and learning rates ($10^{-1}, 10^{-2}$, and 10^{-3}). For every hidden layer width, we generate five random networks and average their performance. The solid lines show the average test error while the shaded regions represent the standard error. When the covariance parameters are tuned properly, V1-initialized networks achieve lower test error over fewer epochs. The benefits are more significant at larger network widths and lower learning rates. With incompatible weights, V1 initialization leads to similar performance as unstructured initialization.

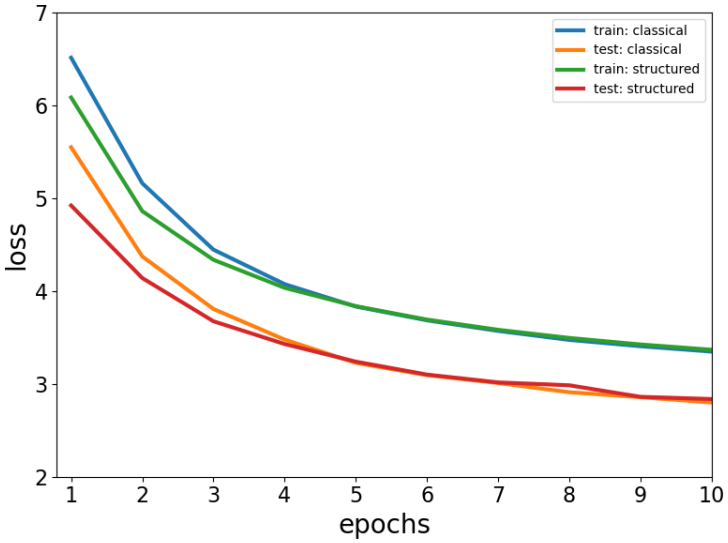

Figure 29: **Initializing AlexNet using structured random features shows little benefit for ImageNet.** Training and testing loss are shown for classical and structured random initializations of convolutional layers in AlexNet. These losses are initially lower for structured features, but by 6 epochs the classical initialization catches up and it eventually reaches a slightly lower loss than the structured initialization. Note that the training losses are higher than testing due to dropout applied in the training phase.

B Appendix References

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. U.S. Government Printing Office, 1964.

[2] Francis Bach. On the Equivalence Between Kernel Quadrature Rules and Random Feature Expansions. *J. Mach. Learn. Res.*, 18(1):714–751, January 2017.

[3] Francis R. Bach. Exploring Large Feature Spaces with Hierarchical Multiple Kernel Learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 105–112. Curran Associates, Inc., 2009.

[4] Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency Bias in Neural Networks for Input of Non-Uniform Density. In *Proceedings of the 37th International Conference on Machine Learning*, pages 685–694. PMLR, November 2020. ISSN: 2640-3498.

[5] Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. *arXiv:2002.02561 [cs, stat]*, Feb 2020. arXiv: 2002.02561.

[6] Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *arXiv:2006.13198 [cond-mat, stat]*, Feb 2021. arXiv: 2006.13198.

[7] Youngmin Cho and Lawrence Saul. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

[8] I. S. Gradshteyn and I. M. Ryzhik. *Table of integrals, series, and products*. Elsevier/Academic Press, Amsterdam, seventh edition, 2007. Translated from the Russian, Translation edited and with a preface by Alan Jeffrey and Daniel Zwillinger, With one CD-ROM (Windows, Macintosh and UNIX).

[9] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[10] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[11] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv:1404.5997 [cs]*, April 2014. arXiv: 1404.5997.

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style,

high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[13] Dario L. Ringach. Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. *Journal of Neurophysiology*, 88(1):455–463, Jul 2002.

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015.

[15] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[16] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[17] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.