Gaussian Process Surrogate Models for Neural Networks

Michael Y. Li[1] **Erin Grant**[2] **Thomas L. Griffiths**[3]

[1]Department of Computer Science, Stanford University, Stanford, California, USA
[2]Gatsby Computational Neuroscience Unit, University College London, London, UK
[3]Departments of Psychology and Computer Science, Princeton, NJ, USA

Abstract

Not being able to understand and predict the behavior of deep learning systems makes it hard to decide what architecture and algorithm to use for a given problem. In science and engineering, *modeling* is a methodology used to understand complex systems whose internal processes are opaque. Modeling replaces a complex system with a simpler, more interpretable surrogate. Drawing inspiration from this, we construct a class of surrogate models for neural networks using Gaussian processes. Rather than deriving kernels for infinite neural networks, we learn kernels empirically from the naturalistic behavior of finite neural networks. We demonstrate our approach captures existing phenomena related to the spectral bias of neural networks, and then show that our surrogate models can be used to solve practical problems such as identifying which points most influence the behavior of specific neural networks and predicting which architectures and algorithms will generalize well for specific datasets.

1 INTRODUCTION

Deep learning systems are ubiquitous in machine learning but sometimes exhibit unpredictable and undesirable behavior when deployed in real-world applications [Geirhos et al., 2020, D'Amour et al., 2020]. This gap between idealized and real-world performance has driven calls for explainability, transparency, and interpretability of deep learning systems [Lipton, 2016, Doshi-Velez and Kim, 2017, Samek et al., 2017], especially as these systems are more widely applied [Bommasani et al., 2021].

Machine learning is not unique in seeking to understand a complex system whose inputs and outputs are observable but whose internal processes are opaque—this challenge occurs across the empirical sciences and engineering. An explanatory tool that is foundational across these disciplines is that of *modeling*, that is, representing a complex and opaque system with a simpler one that is more amenable to interpretation.[1] Modeling makes precise assumptions about how a system may operate while abstracting away details that are irrelevant for a given level of understanding or a given downstream use case. These properties are valuable for a framework for understanding deep learning as they are in other scientific and engineering disciplines.

As the popularity of deep learning has grown, a number of proposals have been made for modeling these systems. Numerous mathematical models of deep learning have been developed [Roberts et al., 2022], and some surprising phenomena, such as adversarial examples [Szegedy et al., 2014], have been captured with a mathematical analysis [Ilyas et al., 2019]. However, existing mathematical models are unable to capture the properties of machine learning systems as applied in practice [Nakkiran, 2021]. Beyond mathematical models, specific models have aimed to explain the predictions of machine learning systems on a per-example basis [Ribeiro et al., 2016, Koh and Liang, 2017, Zhou et al., 2022, Ilyas et al., 2022], but these approaches are, by construction, only partial explanations of the end-to-end system.

In this paper, we pursue an alternative modeling approach that draws on two domains for inspiration. In engineering design, *surrogate models* [Wang and Shan, 2006] emulate the input-output behavior of a complex physical system, allowing practitioners to simulate effects that are consequential for design or analysis without relying on costly or otherwise prohibitive queries from the system itself. In cognitive science, *cognitive models* [Sun, 2008, McClelland, 2009] describe how unobservable mental processes such as memory or attention produce the range of people's observed be-

[1]Though some architectural components of a deep learning system are commonly referred to as a *model*—as in "neural network model"—we use *modeling* to refer to the methodology of idealizing a complex system as a simpler one.

Accepted for the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023).

haviors. Both domains abstract away internal details, such as real-world constraints on a physical system or neural circuitry in the brain, instead treating the target process or system as a *black box*. At the same time, both surrogate and cognitive models are constructed to replicate the end-to-end behavior of the target system and thus are complete where localized explanations are not.

We explore an analogous approach to investigate deep learning systems by constructing *Gaussian process surrogate models for neural networks*. Gaussian processes (GPs) are a natural choice, with appealing theoretical properties specific to the study of neural networks (NNs); namely, certain limiting cases of NN architectures are realizable as GPs [Neal, 1996, Li and Liang, 2018, Jacot et al., 2018, Allen-Zhu et al., 2019, Du et al., 2019]. However, in contrast to these theoretical approaches, we explore the scientific and practical utility of idealizing NNs with GPs using a *data-driven* approach to estimating the kernel functions of finite NNs. With this approach, we capture a number of known phenomena, including a bias towards low frequencies and pathological behavior at initialization, in a cohesive framework. Finally, we demonstrate the practical benefits of this framework by identifying points that strongly influence NN predictions and predicting the generalization behavior of models in a NN family.

2 BACKGROUND

In **surrogate modeling**, we approximate a complex function with a simpler surrogate model that emulates its input-output behavior. Surrogate models have many applications: In optimization, they are often used to approximate queries from expensive-to-evaluate functions [Snoek et al., 2012, Shahriari et al., 2016]; in other applications, surrogate models have been used to gain insight into large physical systems [Camps-Valls et al., 2015].

Cognitive models have been used by cognitive scientists since the 1950s to gain insight into the human mind [Newell et al., 1958]. Bayesian models of cognition offer a way to describe the inductive biases of learning systems as a prior distribution [Griffiths et al., 2010]. As deep NNs have become more prevalent, researchers have used methodologies from cognitive science to interrogate opaque models [Ritter et al., 2017, Geirhos et al., 2019, Hawkins et al., 2020]. The success of these efforts suggests that other methods from cognitive science—namely, cognitive modeling—may be applicable to machine learning systems.

Gaussian processes [GPs; Rasmussen and Williams, 2006] are probabilistic models that specify a distribution over functions. A GP models any *finite* set of N observations as a multivariate Gaussian distribution on $\mathbb{R}^N$, where the nth point is interpreted as the function value, $f(\mathbf{x}_n)$, at the input point $\mathbf{x}_n$. GPs are fully characterized by a mean function $m(\mathbf{x})$, usually taken to be degenerate as $m(\mathbf{x}) = \mathbf{0}, \forall \mathbf{x}$, and a positive-definite kernel function $k(\mathbf{x},\ \mathbf{x}')$ that gives the covariance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ as a function of $\mathbf{x}$ and $\mathbf{x}'$.

Formally, let $\mathbf{X}$ be a matrix of inputs and $\mathbf{y}$ be a vector of output responses. Due to the marginalization properties of the Gaussian distribution, the posterior predictive distribution of a GP for a new input $\mathbf{x}_*$, conditioned on dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$ and assuming centered Gaussian observation noise with variance σ^2, is Gaussian with closed-form expressions for the mean and variance:

$$\mathbb{E}[f(\mathbf{x}_*) \mid \mathcal{D}] = m(\mathbf{x}_*) + {\mathbf{k}_*}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - m(\mathbf{x})) \tag{1}$$

$$\mathbb{V}[f(\mathbf{x}_*) \mid \mathcal{D}] = k(\mathbf{x}_*, \mathbf{x}_*) - {\mathbf{k}_*}^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}_* \tag{2}$$

where $\mathbf{K}$ is the $N \times N$ Gram matrix of pairwise covariances, $k(\mathbf{x}_i, \mathbf{x}_j)$, and $\mathbf{k}_* = [k(\mathbf{x}_1, \mathbf{x}_*), \ldots, k(\mathbf{x}_N, \mathbf{x}_*)]^T$.

The kernel function k specifies the prior on what kind of functions might be represented in observed data, for example, expressing expected smoothness or periodicity. Parametric kernels have hyperparameters θ that affect this prior and thus the posterior predictive. These kernel hyperparameters can be adapted to the properties of a dataset, thus defining a prior over functions that is appropriate for that context. GP kernel hyperparameters are typically learned via gradient-based optimization to maximize the GP marginal likelihood, $p(\mathbf{y} \mid \mathbf{X})$. Again due to properties of the GP, this marginal likelihood has the closed-form expression:

$$\begin{aligned}\log p(\mathbf{y} \mid \mathbf{X}) = &-\frac{1}{2}\mathbf{y}^T\left(\mathbf{K}_\theta + \sigma_n^2 I\right)^{-1}\mathbf{y} \\ &-\frac{1}{2}\log|\mathbf{K}_\theta + \sigma_n^2 I| - \frac{n}{2}\log 2\pi\ .\end{aligned} \tag{3}$$

We write the Gram matrix as $\mathbf{K}_\theta$ to indicate that it depends on kernel hyperparameters via a particular parameterization. In this work, we make use of two kernel parameterizations: the **Matérn kernel** [MK; Matérn, 1960] and the **spectral mixture kernel** [SMK; Wilson and Adams, 2013]. Specifically, following Snoek et al. [2012], we use the automatic relevance determination (ARD) $5/2$ MK, given by:

$$\begin{aligned}k(\mathbf{x}, \mathbf{x}') = \theta_0 &\left(1 + \sqrt{5r^2(\mathbf{x}, \mathbf{x}')} + \tfrac{5}{3}r^2(\mathbf{x}, \mathbf{x}')\right) \\ &\exp\left\{-5\sqrt{5r^2(\mathbf{x}, \mathbf{x}')}\right\}\end{aligned} \tag{4}$$

where $r^2(\mathbf{x}, \mathbf{x}') = \sum_{d=1}^{D}(x_d - x'_d)^2/\theta_d^2$ and each θ_d is a lengthscale parameter capturing smoothness along dimension d. The SMK is derived by modeling the spectral density associated with a kernel as a scale-location mixture of Gaussians [Wilson and Adams, 2013], giving:

$$k(\tau) = \sum_{q=1}^{Q} w_q \cos\left(2\pi^2\tau^T\boldsymbol{\mu_q}\right)\prod_{p=1}^{P}\exp\left\{-2\pi^2\tau_p^2 v_q^{(p)}\right\}. \tag{5}$$

Here, $k(\tau)$ gives the covariance between function values $f(\mathbf{x})$ and $f(\mathbf{x}')$ whose corresponding P- dimensional input values $\mathbf{x}$ and $\mathbf{x}'$ are a distance τ apart. For a Q-component spectral mixture, $w = \{w_i\}_{i=1}^{Q}$ are scalar mixture weights, the q-th component has mean vector $\boldsymbol{\mu_q} = (\mu_q^{(1)}, \ldots, \mu_q^{(P)})$ and covariance matrix $\text{diag}(v_q^{(1)}, \ldots, v_q^{(P)})$. See Section A.4 of the supplement for details on how the hyperparameters of the MK and the SMK control the priors on functions.

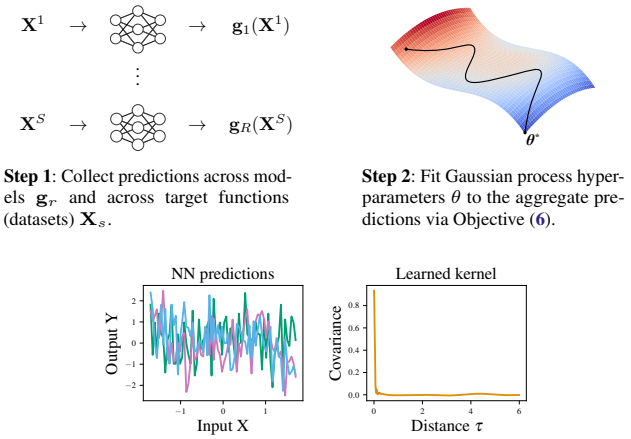
3 LEARNING A GP SURROGATE MODEL FROM NN PREDICTIONS

In this section, we detail the goals and approach of the surrogate modeling framework. In brief, our approach involves collecting neural network predictions across a set of initializations and datasets and estimating GP kernel hyperparameters from these predictions by maximizing the marginal likelihood across network-dataset pairs; see Fig. (1).

3.1 FORMAL FRAMEWORK

Our goal is to capture shared properties among a family of neural network models $\mathfrak{F}$ as applied to a family of datasets $\mathfrak{D}$; untrained neural networks are the special case where $\mathfrak{D}$ is empty. Here, a model family $\mathfrak{F}$ is a set of neural networks $\{\mathbf{g}_0, \ldots, \mathbf{g}_R\}$ that share in design choices (*e.g.,* architecture, training procedure, random initialization scheme) but differ in quantities that are randomized prior to or during training (*e.g.,* parameter initializations). Similarly, a dataset family $\mathfrak{D}$ is a set of datasets $\{\mathcal{D}_0, \ldots, \mathcal{D}_S\}$ that share some underlying structure as in multi-task and meta-learning settings [Caruana, 1997, Hospedales et al., 2020]. We consider supervised learning, in which each dataset consists of inputs and targets, $\mathcal{D} = (\mathbf{X}, \mathbf{y})$. Importantly, we fit surrogate model parameters θ to a *behavioral dataset* of the model family evaluated on the dataset family, and not the ground truth datasets themselves. By behavioral dataset, we mean a dataset of neural network predictions at some test inputs.

Data. We construct a component of the surrogate model training dataset as follows: We sample a model index r and a dataset index s. The corresponding dataset is split into a training set and an evaluation set, $\mathcal{D}_s = \mathcal{D}_s^{\text{train}} \cup \mathcal{D}_s^{\text{eval}}$. The corresponding model $\mathbf{g}_r$ is fit the training set $\mathcal{D}_s^{\text{train}} = (\mathbf{X}_s^{\text{train}}, \mathbf{y}_s^{\text{train}})$ according to the training procedure specified by the choice of model family $\mathfrak{F}$, producing $\mathbf{g}_r^{\text{fit}}$. We then collect the predictions of the trained model on the evaluation set, $\mathbf{g}_r^{\text{fit}}(\mathbf{X}_s^{\text{eval}})$, to produce the component $(\mathbf{X}_s^{\text{eval}}, \mathbf{g}_r^{\text{fit}}(\mathbf{X}_s^{\text{eval}}))$ consisting of the *ground truth inputs* paired with the *neural network behavioral targets* (*i.e.,* neural network predictions on ground truth inputs) from the evaluation set. We aggregate the ground truth inputs and the neural network behavioral targets across pairs to produce the *surrogate model training dataset*,



Step 1: Collect predictions across models $\mathbf{g}_r$ and across target functions (datasets) $\mathbf{X}_s$.

Step 2: Fit Gaussian process hyperparameters θ to the aggregate predictions via Objective (6).

Step 3: Analyze the kernel learned from the aggregate predictions. Here, the learned surrogate kernel reveals the quickly varying behavior of particular neural networks.

Figure 1: Outline of the surrogate modeling approach. We learn a Gaussian process surrogate model for a neural network family applied to a task family by learning kernel hyperparameters from neural network predictions across datasets. The learned kernel provides insight into the properties of the neural network family; e.g., biases towards particular frequencies (Section (4.1)), or expected generalization behavior (Section (4.4)).

$$\big((\mathbf{X}_{s_1}^{\text{eval}}, \mathbf{g}_{r_1}^{\text{fit}}(\mathbf{X}_{s_1}^{\text{eval}})), \ldots, (\mathbf{X}_{s_T}^{\text{eval}}, \mathbf{g}_{r_T}^{\text{fit}}(\mathbf{X}_{s_T}^{\text{eval}}))\big).$$

Surrogate model. We fit the GP using type-II maximum likelihood. Let $P_\theta(\mathbf{g}^{\text{fit}}(\mathbf{X}^{\text{eval}}) \mid \mathbf{X}^{\text{eval}})$ be the GP marginal likelihood of the dataset component $\big(\mathbf{X}^{\text{eval}}, \mathbf{g}^{\text{fit}}(\mathbf{X}^{\text{eval}})\big)$ under a GP with kernel hyperparameters θ, as given in Eq. (3). We fit the surrogate model jointly across model-and-task pairs in the surrogate model training dataset by maximizing the marginal likelihood with respect to θ:

$$\max_{\theta} \prod_{(r,s)} P_\theta(\mathbf{g}_r^{\text{fit}}(\mathbf{X}_s^{\text{eval}}) | \mathbf{X}_s^{\text{eval}}) \,. \tag{6}$$

By optimizing Objective (6), we encourage the kernel hyperparameters θ to capture the implicit posterior distribution over functions induced by the models in the family $\mathfrak{F}$ as applied to the datasets in the family $\mathfrak{D}$. Algorithm (1) gives the complete surrogate model training and evaluation process.

Motivating the framework. By estimating a posterior over functions for a neural network family, we aim to capture shared properties that determine the model family's behavior on data, *i.e.,* **the model family's *inductive biases***. The inductive biases of neural networks (*e.g.,* invariances and equivariances, Markovian assumptions, compositionality) play an important role in their performance by determining their extrapolation behavior [Mitchell, 1980].

GPs, in particular, offer several advantages as surrogate models of NNs. Firstly, GPs **are flexible models that are also often interpretable** in the sense that the learned hyperparameters can provide insights into properties of the datasets on which they are trained [Wilson and Adams, 2013]. Secondly, the use of GP surrogate models is also motivated

Algorithm 1: Training and evaluation of a GP surrogate.

hyperparameters : model family $\mathfrak{F}$,
dataset family $\mathfrak{D}$,
model-dataset count N,
GP parameterization θ

`// Step 1 in Fig. (1)`

for $n \in 1 \ldots N$ **do**

  Sample a model, $\mathbf{g}_{r_n} \sim \text{Unif}(\mathfrak{F})$

  Sample a dataset, $\mathcal{D}_{s_n} \sim \text{Unif}(\mathfrak{D})$

  Train the model, $\mathbf{g}^{\text{fit}}_{r_n} \leftarrow \text{train}(\mathbf{g}_{r_n}, \mathcal{D}^{\text{train}}_{s_n})$

  Evaluate $\mathbf{g}^{\text{fit}}_{r_n}(\mathcal{D}^{\text{eval}}_{s_n})$

end

`// Step 2 in Fig. (1)`

Optimize Objective (6) for θ^*

`// Step 3 in Fig. (1)`

Analyze θ^* via P_{θ^*}

by the **theoretical connections between GPs and NNs.** Neal [1996] showed that a prior over functions, implied by a prior over the weights of certain single-layer multilayer perceptrons (MLPs), converges to a GP as the MLP's width approaches infinity, and recent works [Lee et al., 2017, Matthews et al., 2018, Novak et al., 2019, Garriga-Alonso et al., 2019, Yang, 2019] have extended this correspondence to deep MLPs and more modern NN architectures. The connections between GPs and NNs can provide insight because they transform the priors implicit in NNs designs into explicit priors expressed through a GP. However, our strategy differs from this prior theoretical work that derives analytic kernels for limiting cases of NNs—we instead take an empirical approach by learning GP kernels from the predictions of arbitrary classes of finite NNs. Lastly, GPs have **analytic** marginal likelihoods and leave-one-out predictive distributions, a property that we will exploit in Sections (4.3) and (4.4) to reduce costly computations.

3.2 DEMONSTRATION: COMPARING LEARNED GP PRIORS WITH NN PRIORS

We briefly demonstrate the framework of Section (3.1) by verifying that GP surrogates learned from different NN families exhibit meaningful variation in behavior.

NN hyperparameters. We consider ensembles of 50 randomly initialized (about zero with weight variance $\sigma_w^2 = 1.5$ and bias variance $\sigma_b^2 = 0.05$) fully connected NNs with rectified linear unit (ReLU) or sine (sin) activations and 16 or 32 hidden layers of 128 hidden units each.

GP surrogate. For each ensemble, we learn the hyperparameters of a randomly initialized SMK with $Q = 10$ mixture components by optimizing Objective (6) for 350 iterations with batch gradient descent and the adaptive momentum (Adam) optimizer [Kingma and Ba, 2015] with learning rate $\eta = 0.1$. We choose the kernel hyperparameters with highest objective value across three random initializations.

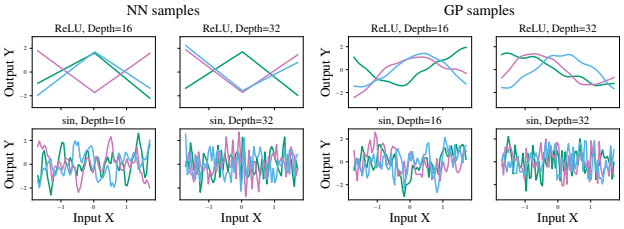


Figure 2: Demonstration: Comparing learned GP priors with NN priors. Samples from GP prior (**right**) with kernel hyperparameters inferred from the predictions of NN families (**left**). GPs are flexible enough to capture properties of each NN family; for example, the samples from the learned GP prior reflect the quickly varying behavior of the 32-layer sinusoidal NNs and the increasing-decreasing behavior of rectifier NNs.

Results. We plot NN predictions and samples from the learned GP priors in Fig. (2). The learned GP captures the periodicity of the sinusoidal neural networks (sinusoidal NNs), and partially captures the increasing-decreasing behavior of rectifier neural networks (rectifier NNs) about a cusp; though, due to the SMK parameterization, it cannot capture the cusp. The GP also captures differences in depths for the sinusoidal NNs: The GP prior samples for the 32-layer networks are quickly varying, indicating shorter learned lengthscales. Taken together, the results of this demonstration show that GP surrogates can capture certain NN behavior.

4 EXPERIMENTS

We provide a series of demonstrations of the value of the approach of Section (3). Each experiment aims to investigate the properties of one or more neural network families, specified by **neural network (NN) hyperparameters**, as evaluated on one or more dataset families, parameterized as **target functions**, by analyzing the corresponding **Gaussian process (GP) surrogate model**. In Section (4.1) and Section (4.2), we capture previously established NN phenomena, in Section (4.3), we identify influential points for NNs, and in Section (4.4), we predict NN generalization behavior.

4.1 REPRODUCTION: SPECTRAL BIAS

Rahaman et al. [2019] demonstrated that deep rectifier NNs exhibit *spectral bias*, the preference to learn lower frequencies in the target function before higher frequencies. To demonstrate this, the authors studied the Fourier spectrum of rectifier NNs fit to a sum of sinusoidal functions of varying frequencies. We take an alternative approach, learning kernels from NN predictions at various stages of training. We show that these learned kernels capture the spectral bias.

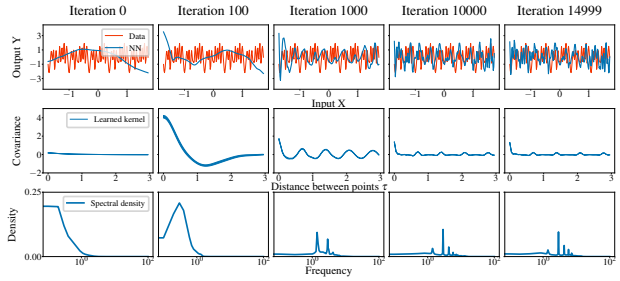


Figure 3: Capturing spectral bias in neural networks. (**Top**) Neural network predictions as training progresses on the sum-of-sines target function described in Section (4.1). (**Middle**) Spectral mixture kernel fit to neural network predictions as training progresses and (**Bottom**) corresponding spectral density. The kernel reveals a spectral bias for this neural network family: the range of spectral frequencies expressed in the kernel increases with the number of iterations of training.

NN hyperparameters. As in Rahaman et al. [2019], we train an ensemble of 20 NNs with 6 hidden layers of 256 units and ReLU activations using full-batch gradient descent with Adam and a learning rate of $\eta = 3 \times 10^{-4}$.

Target function. We consider 20 target functions which are sums of sine functions with frequencies in $(5, 10, \ldots, 45, 50)$ and phases drawn from $U(0, 2\pi)$, evaluated at 200 points evenly spaced between $[0, 1]$, as in Rahaman et al. [2019].

GP surrogate. We learn the parameters of a spectral mixture kernel (SMK) with $Q = 10$ mixture components by optimizing Objective (6) with Adam for 350 iterations with a learning rate of $\eta = 0.1$ [Kingma and Ba, 2015]. Since the marginal likelihood of the SMK is multi-modal in its frequency parameters, we repeat this optimization for three different random initializations of the kernel parameters and choose the hyperparameters with the largest marginal likelihood value (the value of Objective (6)). Following GPyTorch's initialization strategy, we draw the inverse lengthscale v_i from a truncated Gaussian distribution with variance set to the maximum distance between two points in the dataset and truncation value set to the maximum distance between points [Gardner et al., 2018]. We set the signal variances w to the variance of the target function values divided by the number of mixture components. The frequency hyperparameters of the SMK are sometimes initialized by sampling from a uniform distribution whose upper limit is the Nyquist frequency [Wilson and Adams, 2013]; since this target function's largest frequency is smaller than the Nyquist frequency, we instead set a smaller frequency.

Results. Fig. (3) displays the average NN prediction across the ensemble, the learned kernel at different iterations of NN training averaged across the ensemble and ten random initializations of the SMK, and the corresponding average spectral density.

The kernel function, which is given in Eq. (5), reflects how the similarity between function values varies with the distance between their input points.[2] The structure of the learned kernel reflects the properties of the NN family: Initially, the learned kernel only captures low frequencies in the NN's predictions—seen in the long period of the kernel—consistent with the spectral bias of Rahaman et al. [2019]. However, with more training, the periodicity of the surrogate kernel reflects both low and high frequencies. Consistent with this, the spectral density exhibits a wider range of frequencies.

4.2 REPRODUCTION: DEPTH PATHOLOGIES IN RANDOMLY INITIALIZED NN

Hyperparameter selection in NNs is not always theoretically grounded. Many recent studies thus characterize how hyperparameter choices affect the properties of NNs at random initialization [Schoenholz et al., 2017, Yang, 2019, Xiao et al., 2018]. Towards that end, recent work showed that increasing depth can actually induce pathologies in randomly initialized NNs [Labatie, 2019, Duvenaud et al., 2014]. For example, Duvenaud et al. [2014] proved that increasing depth in a certain class of infinitely wide NNs produces functions with ill-behaved derivatives.

We empirically study a similar pathology—quick variation in input space—that emerges in randomly initialized, finite-width, finite-depth NNs. To do this, we fit GP surrogates to randomly initialized NN ensembles with varying depths and activations. If NNs exhibit this pathology, the learned covariance will decay sharply with distance.

NN hyperparameters. We consider families of NNs with $\sin$ or ReLU activations and depths ranging from 16 to 512 layers. From each family, we randomly initialize an ensemble of 50 NNs, with 128 hidden units in each layer. We randomly initialize NN weights about zero with weight variance $\sigma_w^2 = 1.0$ and bias variance $\sigma_b^2 = 0.05$.

GP surrogate. We learn an SMK kernel by optimizing Objective (6) separately for each ensemble, running Adam for 750 iterations with $\eta = 0.1$. We choose the kernel hyperparameters with the highest mean marginal likelihood among three random initializations. To ensure our results are robust across an neural network ensemble, we consider the averaged learned kernel. That is, if we have n kernels, $k_1(\cdot), \ldots, k_n(\cdot)$, learned from n different ensembles, we consider the average learned kernel $\bar{k}(\tau) = \frac{1}{n}\sum_{i=1}^{N} k_i(\tau)$.

Results. Fig. (4) plots the average learned kernels for NN families with varying activation functions and depths, and the corresponding neural network predictions. Across both

[2] Since the SMK is a stationary covariance function, we graph against the distance between input points

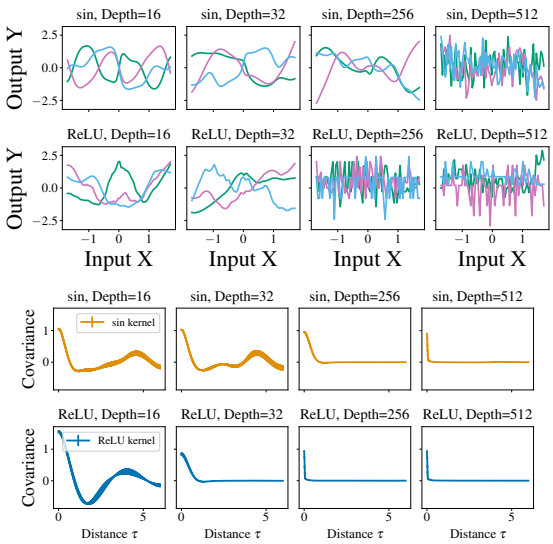


Figure 4: Depth pathologies in randomly initialized neural networks. Predictions of neural networks **(top)** with varying activations **(rows)** and depths **(columns)**; mean and standard error of the covariance of learned kernels **(bottom)**. Greater depth results in kernels with shorter lengthscales, and this pathology emerges earlier in rectifier NNs.

activation functions, the learned kernels reveal a pathology: For large depths, the covariance (Fig. (4), bottom) decays sharply towards zero with distance. The NN predictions (Fig. (4), top) explain this property: At large depths, the NNs vary quickly in the input domain, which leads the SMK to learn short lengthscales. Interestingly, this pathology emerges at different depths for different activation functions. We see rectifier NNs exhibit this pathology with 256 layers while sinusoidal NNs exhibit this pathology with 512 layers.

4.3 AMORTIZED INFLUENCE ESTIMATION

The primary computational bottleneck of fitting a GP is the $O(N^3)$ cost of Gram matrix inversion. However, once this matrix is inverted, certain computations that seem costly can actually be performed cheaply. For example, the leave-one-out predictive (LOO) distributions can be computed analytically given the inverted Gram matrix and the training data [Rasmussen and Williams, 2006]. The LOO distributions characterize how GP predictions change after removing a training point. In this section, we leverage this property of GPs to perform influence function analysis in the spirit of Koh and Liang [2017]. In particular, we use the LOO predictive distributions of a surrogate GP to detect influential points for NNs with Fourier feature layers [Tancik et al., 2020], which enable NNs to learn higher frequencies. Similar architectures have achieved state-of-art performance in a number of important tasks in computer vision and computer graphics [Mildenhall et al., 2020]. We show that the surrogate LOO predictions are consistent with NN LOO predictions obtained from costly retraining.

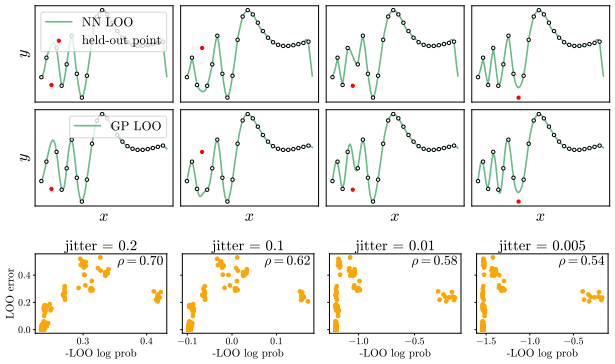


Figure 5: NN and GP surrogate leave-one-out predictions compared. (**Top**) Neural network predictions obtained from re-training on target function with influential (red) training points removed. (**Middle**) GP predictions on same datasets with kernel learned from trained NN predictions. Columns correspond to different held-out training points. (**Bottom**) Correlation between LOO error and (negative) GP LOO predictive probability of held-out points; a lower negative LOO predictive probability means a less influential point. The GP and NN LOO predictions are consistent, and the LOO error positively correlates with the negative LOO predictive probability.

NN hyperparameters. We train 10 NNs with 4 hidden layers of 1024 units, ReLU activations [Tancik et al., 2020]. The inputs to the NNs are passed through a single Fourier feature layer with parameter p that controls the spectral bias of the NN. For details see Tancik et al. [2020]. We set $p = 1$ and choose Fourier feature mappings with frequencies ranging from 1 to 4. We train the network using full-batch gradient descent with stochastic gradient descent (SGD) and a learning rate of $\eta = 10^{-3}$ for 2000 iterations.

Target functions. We consider a target function inspired by one used in the leave-one-out GP literature [Ginsbourger and Scharer, 2021]. The functional form is given by $f(x) = \sin(30(x-0.95)^4)\cos(2(x-0.95)) + (x-0.95)/2$. This target function is non-stationary, with higher frequency behavior in the left half of its input domain and lower frequency behavior in the right.

GP surrogate. We fit a SMK with $Q = 10$ mixture components to trained neural network predictions at 300 input points evenly spaced between (0, 3). We initialize frequency hyperparameters by sampling uniformly from 0 to 20.

We briefly overview leave-one-out cross-validation (LOO-CV) in the context of GPs. The GP LOO predictive probability when leaving out training example i is

$$\log p(y_i|X_{-i}, \mathbf{y}_{-i}) = -\frac{1}{2}\log\sigma_i^2 - \frac{(y_i-\mu_i)^2}{2\sigma_i^2} - \frac{1}{2}\log 2\pi \tag{7}$$

The notation $\mathbf{y}_{-i}$ indicates all training targets *besides* the ith target. The LOO-CV predictive mean μ_i and variance

σ_i^2 can be computed from the inverse of the Gram matrix:

$$\mu_i = y_i - \frac{[K^{-1}y]_i}{[K^{-1}]_{ii}}, \quad \text{and} \quad \sigma_i^2 = \frac{1}{[K^{-1}]_{ii}} \tag{8}$$

Importantly, to compute these predictive means and variances, we can reuse the inverse of the Gram matrix computed to perform GP inference (Eqs. (1) and (3)). In our analysis, we will quantify the influence of a point (x_i, y_i) using the predictive log probability $\log p(y_i | X_{-i}, \mathbf{y}_{-i})$. We can interpret points with lower log probability as more "influential" under the assumptions of a kernel.

Results In the first two rows of Figure 4, we compare the surrogate GP LOO and NN LOO predictions for some of the most influential points. Indeed, the GP LOO predictions at the influential points are consistent with the NN LOO prediction. In the last row of Figure 4, we plot the (negative) LOO log probability at each input point against the actual LOO error at the training input. Each plot in the last row contains the data across all 10 NNs. The LOO error is computed as the difference between the NN LOO predictions and the predictions of the NN trained on the full dataset. We validate this result across different levels of jitter added to the Gram matrix. The Spearman's rank correlation coefficient ρ shows a positive relationship between the negative LOO log probability at an input point and the LOO error at the training input. Interestingly, the outliers on the right region of each subplot always correspond to the leftmost points in the training data. This is consistent with previous work showing that the GP LOO can overestimate the influence of a boundary point [Ginsbourger and Scharer, 2021].

4.4 PREDICTING THE GENERALIZATION GAP

In these sections, we predict generalization using kernels learned from trained NNs. To do this, we characterize trained NN properties on the *validation set* and compare these properties to the training data. We focus on the validation set because it is more informative of extrapolation. If the NN extrapolates well, its predictions on the validation set should be "similar" in some sense to the dataset. On the other hand, significant discrepancies could indicate poor extrapolation. This intuition motivates us to compare a kernel from the training data with a surrogate kernel fit to NN predictions on a validation set. We find that lower similarity between these kernels correlates with a larger *generalization gap* (*i.e.,* poorer extrapolation), defined as the difference between test error and training error [*e.g.,* Jiang et al., 2020].

4.4.1 Initial demonstration

NN hyperparameters. We train ensembles of randomly initialized NNs with sigmoid-weighted linear unit (SiLU) [Elfwing et al., 2018], Gaussian error linear unit (GELU) [Hendrycks and Gimpel, 2016], ReLU [Fukushima, 1975, Nair and Hinton, 2010], or hyperbolic tangent ($\tanh$) activations, and two layers of 128 hidden units. We use the LeCun normal initialization with a scale of 1.5 [LeCun et al., 2012]. We train 25 NNs with full-batch gradient descent using Adam with a learning rate of $\eta = 0.003$. We want to assess if our approach can distinguish between NNs with similar training behavior but varying generalization performance, so we train NNs either for a maximum number of iterations, a hyperparameter, or until training error reaches zero.

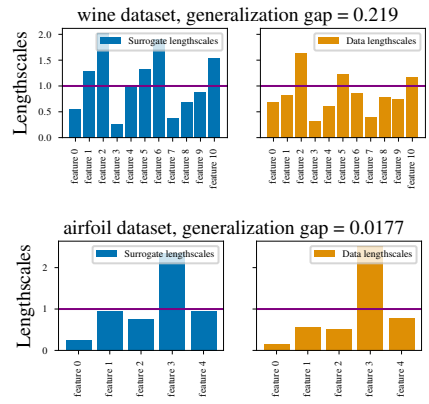


Figure 6: Qualitative connection between lengthscale profile discrepancy and generalization gap. Each subfigure compares normalized lengthscales learned from neural network predictions on validation set (*i.e.,* surrogate lengthscales) after training and normalized lengthscales learned from training data (*i.e.,* data lengthscales). A lengthscale greater than 1 indicates an "unimportant" feature. The title indicates the UCI dataset and generalization gap defined in Fig. (7). Data and surrogate lengthscales for some features are different (*e.g.,* features 1, 4, 6), reflected in a high generalization gap (**top**). Data and surrogate lengthscales for the same features are generally similar, reflected in a low generalization gap (**bottom**). This suggests a connection between the generalization gap and discrepancy between surrogate and data lengthscales.

Target functions. We consider a set of naturalistic regression tasks from the UC Irvine Machine Learning Repository (UCI) dataset [Dua and Graff, 2017], spanning a range of dataset sizes and input dimensions. We split each of the datasets into a 72/8/20 train/validation/test split. Both the data input and output are standardized by mean-centering and dividing by the standard deviation dimension-wise so that the target values and each dimension of the data input have near zero mean and unit variance. We subsample 2,000 datapoints for datasets with more than 2,000 datapoints, as in Simpson et al. [2021] and Liu et al. [2020].

GP surrogate. We learn a *data kernel* directly from the training dataset. We also learn a *surrogate kernel* from NN predictions on the validation set. In both cases, we use the Matérn kernel (MK) since the SMK can struggle for higher-dimensional inputs. We learn a separate lengthscale for each input dimension (*i.e.,* feature) of the data. We denote the lengthscales for a kernel as its *lengthscale profile*. We call

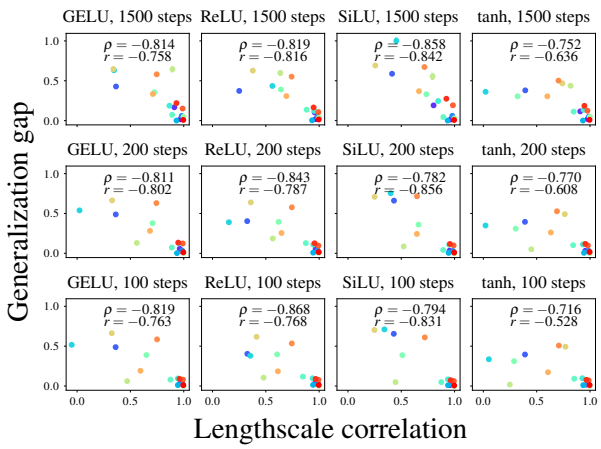


Figure 7: Inverse relationship between generalization error and lengthscale correlation on UCI datasets. Each point represents the lengthscale correlation (between surrogate and data lengthscales) and the average generalization gap for a neural network ensemble to which the surrogate model is fit, on a single UCI dataset; we also report the standard error of the generalization gap. Each panel corresponds to a particular neural family; see Section (4.4) for details about hyperparameters of these families, including architectures. Colors correspond to UCI datasets. Across datasets and architectures, a larger lengthscale correlation (*i.e.,* higher similarity between the data and surrogate representations) corresponds to a lower generalization gap (*i.e.,* better extrapolation).

the data kernel's lengthscales the *data lengthscales* and the surrogate kernel's lengthscales the *surrogate lengthscales*. To quantify the mismatch between NN validation predictions and the training data, we consider the *correlation in lengthscale profiles across features*. This is the correlation between the data and surrogate lengthscales.

Results. Fig. (6) gives intuition for our more general result in Fig. (7). For two UCI datasets, we compare the data lengthscales and the surrogate lengthscales for a two-layer GELU NN. The vertical axis corresponds to (normalized) learned lengthscales for each input dimension.[3] When the generalization gap is small, the data kernel and surrogate kernel are similar; the same features have similar lengthscales (Fig. (6), bottom). When the generalization gap is large, the data kernel and surrogate kernel have discrepancies. For example, the surrogate lengthscales for features 1 and 6 are larger than 1, but the data lengthscales for feature 1 and 6 are smaller than 1 (Fig. (6), top).

In Fig. (7), we summarize our results across different architectures, datasets, and maximum training iterations. We display the generalization gap against the correlation in lengthscale profiles across features. We report the mean generalization gap along with its standard error. The similarity in lengthscale profiles negatively correlates with generalization gap across a range of architectures and max iterations. The Pearson correlation coefficients (r) range from –0.856 to –0.528. The Spearman correlation coefficients (ρ) range from –0.868 to –0.716. In Section A.3, we demonstrate these results are insensitive to outlier datasets.

[3]For this visualization, we divide the learned lengthscale for each dimension by the difference between the maximum feature value and minimum feature value for each dimension. Therefore, a lengthscale that is much greater than 1 suggests that the NN predictions do not vary much along that dimension.

4.4.2 Larger-scale demonstration

We now extend the analysis in the previous section to a larger hyperparameter sweep, more closely emulating the analysis a practitioner would use in selecting a model for a dataset.

NN hyperparameters. We train ensembles of randomly initialized NNs with four activations (SiLU, GELU, ReLU, or tanh), two depths (2, 4), two widths (32, 64), and two learning rates with Adam (0.03, 0.003), generating 32 hyperparameter combinations. We use the LeCun normal initialization with a scale of 1.5. We train 50 NNs for 500 iterations with full-batch gradient descent using Adam.

Target functions. We consider a subset of UCI datasets that satisfy two criteria: more than 2 unique feature values for each dimension, and a range in lengthscale correlations between neural network and datasets greater than 0.025; the first condition ensures that we can learn sensible lengthscales for the dataset, while the second ensures that there is meaningful variation in NN behavior within a dataset. As in Section (4.4.1), we split each of the datasets into a 72/8/20 train/validation/test split, subsample 2,000 datapoints if necessary, and standardize data inputs and outputs.

GP surrogate. In addition to the correlation in lengthscale profiles across features from Section (4.4.1), we consider the marginal likelihood of the training dataset under the kernel learned for each neural network family.

Results. Figure (8) plots the average generalization gap and standard error against the correlation in lengthscale profiles across features. In contrast to Fig. (7), each panel corresponds to a particular dataset and each data point corresponds to a particular neural network family with a set of hyperparameters described above. The majority of datasets exhibit a negative relationship between correlation in lengthscale profiles and generalization gap. There are two exceptions: pendulum and protein. For the pendulum dataset, the positive relationship is driven by neural networks with tanh activations (the dots in different shades of red). The Matern kernel may struggle to model the tanh networks, consistent with Fig. (7) where the correlations were consistently lower for the tanh networks. For the protein dataset, the positive relationship may be due to challenges in fitting GPs to this dataset; recent work showed that exact GP regression on a training dataset (without any subsampling) from the protein dataset attains high test RMSE [Wang et al., 2019].

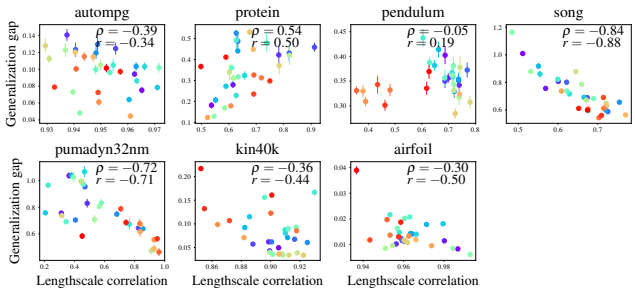


Figure 8: Inverse relationship between generalization error and lengthscale correlation on UCI datasets across hyperparameter sweep of neural networks. Each point represents the lengthscale correlation (between surrogate and data lengthscales) and the generalization gap for a neural network ensemble to which the surrogate model is fit. Each panel corresponds to a particular UCI dataset; see Section (4.4.2) for details about hyperparameters and architectures. In 5/7 datasets, a larger lengthscale correlation (*i.e.,* higher similarity between the data and surrogate representations) corresponds to a lower generalization gap (*i.e.,* better extrapolation).

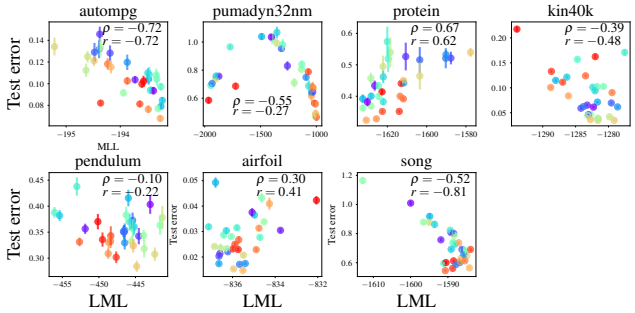


Figure 9: Inverse relationship between test error and marginal likelihood on UCI datasets Each point represents the marginal likelihood of the training data (using the surrogate kernel for a neural network family) and the test error for a neural network ensemble to which the surrogate model is fit. Each panel corresponds to a particular UCI dataset; see Section (4.4.2) for details about hyperparameters of these families, including architectures. Each color corresponds to a particular neural network ensemble. In 5/7 datasets, a larger marginal likelihood correlates with lower test error.

In Fig. (9), we plot the test error against the marginal likelihood of the training data under the various surrogate kernels for each neural network ensemble. That is, each point corresponds to the marginal likelihood of the training data using the surrogate kernel learned from each neural network family. We find the marginal likelihood correlates with test error on several of the datasets (each dataset is indicated by the subplot title). Consistent with previous work [Lotfi et al., 2022], we find that this result is sensitive to the jitter (which we set to 0.5). In Section A.2 of the supplement, we sanity check this analysis on synthetic datasets.

5 DISCUSSION

In this work, we illustrated the potential of empirically characterizing neural networks with Gaussian process surrogates. We captured the spectral bias of deep rectifier networks by examining how the surrogate kernel evolves during training (Section (4.1)). We captured pathologies that emerge in randomly initialized neural networks by examining the learned kernels of neural networks with varying depths (Section (4.2)). We further demonstrated that Gaussian process surrogates can be used to identify influential datapoints (Section (4.3)) and predict neural network generalization (Section (4.4)). Taken together, these results suggest that Gaussian process surrogates may be a valuable empirical tool for investigating deep learning, and future work could aim to use this framework to complement existing approaches to interpretability [*e.g.,* Ribeiro et al., 2016] and examining extrapolation behavior [*e.g.,* Xu et al., 2021].

We note some limitations. First, though the framework is, in principle, applicable to broader settings, we restricted this first exploration to regression tasks and feed-forward neural network architectures. A broader study of more architectures and tasks would be challenging due to the need to scale Gaussian processes but potentially rewarding, as characterizing properties of neural networks is a open problem with far-reaching implications [Sejnowski, 2020]. To tackle this, we could leverage techniques such as inducing points [*e.g.,* Titsias, 2009]. Another limitation is that the data generation process involves training many networks. We could leverage recent work on fast neural network training [Leclerc et al., 2022].

Second, we learn point estimates of kernel hyperparameters [type II maximum likelihood; Gelman et al., 2013]. Although this is standard, we could infer the posterior over hyperparameters using Markov chain Monte Carlo (MCMC) or variational inference [Lalchand and Rasmussen, 2020, Murray and Adams, 2010] to perform a fully Bayesian analysis. We could also explore a richer set of kernels [Duvenaud et al., 2013]. These are exciting avenues for future work.

Acknowledgements

We thank Jake Snell for helpful discussions. This work was supported by ONR grant number N00014-18-1-2873.

References

Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. In *Proc. ICML*, 2019.

R. Bommasani et al. On the opportunities and risks of foundation models. 2021.

G. Camps-Valls et al. Ranking drivers of global carbon and energy fluxes over land. In *Proc. Int. Geosci. Remote. Se.*, 2015.

R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. 2020.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. 2017.

S. S. Du, X. Zhai, B. Poczos, and A. Singh. Gradient descent provably optimizes over-parameterized neural networks. In *Proc. ICLR*, 2019.

D. Dua and C. Graff. UCI Machine Learning Repository, 2017.

D. Duvenaud, J. Lloyd, R. Grosse, J. Tenenbaum, and G. Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *Proc. ICML*, 2013.

D. Duvenaud, O. Rippel, R. P. Adams, and Z. Ghahramani. Avoiding pathologies in very deep networks. In *Proc. AISTATS*, 2014.

S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

K. Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20:121–136, 1975.

J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson. GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU acceleration. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Adv. NeurIPS*, 2018.

A. Garriga-Alonso, C. E. Rasmussen, and L. Aitchison. Deep convolutional networks as shallow Gaussian processes. In *Proc. ICLR*, 2019.

R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. ImageNet-trained CNNs are biased towards texture; Increasing shape bias improves accuracy and robustness. In *Proc. ICLR*, 2019.

R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann. Shortcut learning in deep neural networks. *Nat. Mach. Intell.*, 2:665–673, 2020.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. 2013.

D. Ginsbourger and C. Scharer. Fast calculation of Gaussian Process multiple-fold cross-validation residuals and their covariances. 2021.

T. L. Griffiths, N. Chater, C. Kemp, A. Perfors, and J. B. Tenenbaum. Probabilistic models of cognition: Exploring representations and inductive biases. *Trends in cognitive sciences*, 14(8):357–364, 2010.

R. Hawkins, T. Yamakoshi, T. L. Griffiths, and A. Goldberg. Investigating representations of verb bias in neural language models. In *Proc. EMNLP*, 2020.

D. Hendrycks and K. Gimpel. Gaussian error linear units (GELUs). 2016.

T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey. Meta-learning in neural networks: A survey. *Trans. PAMI*, 2020.

A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. *Adv. NeurIPS*, 2019.

A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry. Datamodels: Understanding predictions with data and data with predictions. In *Proc. ICML*, 2022.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Adv. NeurIPS*, volume 31, 2018.

Y. Jiang, P. Foret, S. Yak, D. M. Roy, H. Mobahi, G. K. Dziugaite, S. Bengio, S. Gunasekar, I. Guyon, and B. Neyshabur. NeurIPS 2020 Competition: Predicting generalization in deep learning, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proc. ICML*, 2017.

A. Labatie. Characterizing well-behaved vs. pathological deep neural networks. In *Proc. ICML*, 2019.

V. Lalchand and C. E. Rasmussen. Approximate inference for fully Bayesian Gaussian process regression. In *Proc. AABI*, 2020.

G. Leclerc, A. Ilyas, L. Engstrom, S. M. Park, H. Salman, and A. Madry. FFCV: Accelerating training by removing data bottlenecks. `https://github.com/libffcv/ffcv/`, 2022.

Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, 2012.

J. Lee, Y. Bahri, R. Novak, S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as Gaussian processes. In *Proc. ICLR*, 2017.

Y. Li and Y. Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Adv. NeurIPS*, 2018.

Z. C. Lipton. The mythos of model interpretability. 2016.

S. Liu, X. Sun, P. J. Ramadge, and R. P. Adams. Task-agnostic amortized inference of Gaussian process hyperparameters. In *Adv. NeurIPS*, 2020.

S. Lotfi, P. Izmailov, G. Benton, M. Goldblum, and A. G. Wilson. Bayesian model selection, the marginal likelihood, and generalization. 2022.

B. Matérn. *Spatial variation*, volume 36. Springer Science & Business Media, 1960.

A. G. d. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. In *Proc. ICLR*, 2018.

J. L. McClelland. The place of modeling in cognitive science. *Topics in Cognitive Science*, 1(1):11–38, 2009.

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

T. M. Mitchell. The need for biases in learning generalizations. Technical report, 1980.

I. Murray and R. P. Adams. Slice sampling covariance hyperparameters of latent Gaussian models. In *Adv. NeurIPS*, 2010.

V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proc. ICML*, 2010.

P. Nakkiran. *Towards an empirical theory of deep learning*. PhD thesis, Harvard University, 2021.

R. M. Neal. Priors for infinite networks. In *Bayesian Learning for Neural Networks*, pages 29–53. Springer, 1996.

A. Newell, J. C. Shaw, and H. A. Simon. Elements of a theory of human problem solving. *Psych. Rev.*, 65(3):151, 1958.

R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, D. A. Abolafia, J. Pennington, and J. Sohl-dickstein. Bayesian deep convolutional networks with many channels are Gaussian processes. In *Proc. ICLR*, 2019.

N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In *Proc. ICML*, 2019.

C. E. Rasmussen and C. K. Williams. *Gaussian processes for machine learning*. Jan. 2006.

M. T. Ribeiro, S. Singh, and C. Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *Proc. KDD*, 2016.

S. Ritter, D. G. T. Barrett, A. Santoro, and M. M. Botvinick. Cognitive psychology for deep neural networks: A shape bias case study. In *Proc. ICML*, 2017.

D. A. Roberts, S. Yaida, and B. Hanin. *The principles of deep learning theory*. Cambridge University Press, 2022.

W. Samek, T. Wiegand, and K.-R. Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. 2017.

S. S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein. Deep information propagation. In *Proc. ICLR*, 2017.

T. J. Sejnowski. The unreasonable effectiveness of deep learning in artificial intelligence. *PNAS*, 117, 2020.

B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE*, 104:148–175, 2016.

F. Simpson, I. Davies, V. Lalchand, A. Vullo, N. Durrande, and C. E. Rasmussen. Kernel identification through transformers. In *Adv. NeurIPS*, 2021.

J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Adv. NeurIPS*, 2012.

R. Sun. *The Cambridge handbook of computational psychology*. Cambridge University Press, 2008.

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. ICLR*, 2014.

M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Adv. NeurIPS*, pages 7537–7547, 2020.

M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proc. AISTATS*, 2009.

G. G. Wang and S. Shan. Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380, 05 2006.

K. A. Wang, G. Pleiss, J. R. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson. Exact Gaussian processes on a million data points. In *Adv. NeurIPS*, 2019.

A. G. Wilson and R. P. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *Proc. ICML*, 2013.

L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. S. Schoenholz, and J. Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In *Proc. ICML*, 2018.

K. Xu, M. Zhang, J. Li, S. S. Du, K.-I. Kawarabayashi, and S. Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *Proc. ICLR*, 2021.

G. Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. 2019.

Y. Zhou, M. T. Ribeiro, and J. Shah. ExSum: From local explanations to model understanding. In *Proc. NAACL*, 2022.