nature communications

Article

https://doi.org/10.1038/s41467-024-54813-x

# Deep neural networks have an inbuilt Occam's razor

Received: 28 April 2023

Accepted: 18 November 2024

Published online: 14 January 2025

Check for updates

Chris Mingard[1,2,3], Henry Rees[1,3], Guillermo Valle-Pérez[1] & Ard A. Louis[1]

The remarkable performance of overparameterized deep neural networks (DNNs) must arise from an interplay between network architecture, training algorithms, and structure in the data. To disentangle these three components for supervised learning, we apply a Bayesian picture based on the functions expressed by a DNN. The prior over functions is determined by the network architecture, which we vary by exploiting a transition between ordered and chaotic regimes. For Boolean function classification, we approximate the likelihood using the error spectrum of functions on data. Combining this with the prior yields an accurate prediction for the posterior, measured for DNNs trained with stochastic gradient descent. This analysis shows that structured data, together with a specific Occam's razor-like inductive bias towards (Kolmogorov) simple functions that exactly counteracts the exponential growth of the number of functions with complexity, is a key to the success of DNNs.

Although deep neural networks (DNNs) have revolutionized modern machine learning[1,2], a fundamental theoretical understanding of why they perform so well remains elusive[3,4]. One of their most surprising features is that they work best in the overparameterized regime, with many more parameters than data points. As expressed in the famous quip: "*With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.*" (attributed by Enrico Fermi to John von Neumann[5]), it is widely believed that having too many parameters will lead to overfitting: a model will capture noise or other inconsequential aspects of the data, and therefore predict poorly.

In statistical learning theory[6] this intuition is formalized in terms of model capacity. It is not simply the number of parameters, but rather the complexity of the set of hypotheses a model can express that matters. The search for optimal performance is often expressed in terms of bias-variance trade-off. Models that are too simple introduce errors due to bias; they can't capture the underlying processes that generate the data. Models that are too complex are over-responsive to random fluctuations in the data, leading to variance in their predictions.

DNNs are famously highly expressive[7–9], i.e., they have extremely high capacity. Their ability to generalize therefore appears to break basic rules of statistical learning theory. Exactly how, without explicit regularization, DNNs achieve this feat is a fundamental question that has remained open for decades[3,4]. Although there has been much recent progress (see Supplementary Note 1 for a literature overview) there is no consensus for why DNNs work so well in the overparameterized regime.

Here we study this conundrum in the context of supervised learning for classification, where inputs $x_i$ are attached to labels $y_i$. Given a training set $S = \{(x_i, y_i)\}_{i=1}^{m}$ of $m$ input-output pairs, sampled i.i.d. from a data distribution $\mathcal{D}$, the task is to train a model on $S$ such that it performs well (has low generalization error) when predicting output labels $\hat{y}_i$ for a test set $T$ of unseen inputs, sampled from $\mathcal{D}$. For a DNN $\mathcal{N}(\Theta)$, with parameters $\Theta \subseteq \mathbb{R}^p$ (typically weights and biases), the accuracy on a training set can be captured by a loss function $L(\hat{y}_i, y_i)$ that measures how close, for input $x_i$, the prediction $\hat{y}_i$ of the DNN is to the true label $y_i$. Training is typically done via some variant of stochastic gradient descent (SGD) which uses derivatives of $L(\hat{y}_i, y_i)$ to adjust the parameters $\Theta$ in order to minimize the loss on $S$. Because DNNs are so highly expressive, and because SGD is typically a highly effective optimizer for DNNs, (near) zero training error (all correct labels after thresholding) on $S$ is routinely achieved[7].

## Functions and inductive bias

For classification, the question of why overparameterized DNNs don't overfit can conveniently be expressed in terms of functions. For a given

[1]Rudolf Peierls Centre for Theoretical Physics, University of Oxford, Oxford, UK. [2]Physical and Theoretical Chemistry Laboratory, University of Oxford, Oxford, UK. [3]These authors contributed equally: Chris Mingard, Henry Rees. e-mail: ard.louis@physics.ox.ac.uk

training set $S$ and test set $T$, a function $f$ can be defined on a restricted domain $S+T$. The inputs of $f$ are the $x_i \in S \cup T$, and the outputs include all possible sets of labels $\{\hat{y}_i\}$. Only one function gives the true labels $\{y_i\}$. For a given set of parameters $\Theta$, the DNN then represents a particular function $f$, which can be identified by the labels it outputs on the inputs $x_i \in S \cup T$, after thresholding. Under the assumption that zero training error can be achieved, functions need only be distinguished by how they behave on the test set $T$. For $C$ classes there are $N_T = C^{|T|}$ possible functions $f$ with zero error on the training set, this number is typically unimaginably large. The overwhelming majority of these functions will not generalize well. Since DNNs are highly expressive, they should be able to represent all (or nearly all) of these functions. The fundamental question of overparameterized DNN performance becomes a question of *inductive bias*: Why, from the unimaginably large set of functions that give zero error on $S$, do DNNs converge on a minuscule subset of functions that generalize well? Here, we will argue that a combination of structured data and specific Occam's razor-like inductive bias towards simple functions which cancels out the exponential growth of the number of functions with increasing complexity helps answer this question.

## Distinguishing two questions about generalization

The main question we address here is what we call the *1st-order question of generalization* – Why do high-capacity learning models (such as DNNs) generalize at all? – This question of how inductive bias allows DNNs to break the conventional bias-variance trade-off expectations of classical learning theory has a long history. For example, it was famously highlighted in Leo Breiman's 1995 commentary on refereeing for the NeurIPS conference[3] (see also Appendix A) who phrased it by asking "Why don't heavily over-parameterised neural networks overfit the data?". While originally formulated for DNNs, the fact that infinite width limits of DNNs can reduce to neural network Gaussian processes (NNGPs)[10–12], or kernels such as the Neural Tangent Kernel[13], has stimulated a large volume of important theoretical work on GPs and kernels, see e.g.,[14,15]. In particular, these models recapitulate many properties of finite-width DNNs, including reaching small generalization errors on standard datasets such as CIFAR10[16,17]. While these methods are non-parametric, they have high capacity, and like DNNs[7], can memorize random data[14]. For small capacity (or fewer parameters than data points for DNNs) all these models exhibit classic bias-variance trade-offs, with optimal generalization performance at intermediate capacity. However, as capacity (or the number of parameters) increases further, the generalization error markedly diminishes. This phenomenon, known as double-descent[18], is observed in DNNs, kernels, and GPs, illustrating how these high-capacity models deviate from the conventional wisdom of classical statistical learning theory

The relative simplicity of GPs and kernels compared to DNNs has enabled the derivation of analytic estimates of the generalization error in terms of the kernel eigenfunctions and eigenvalues [15,19–23]. Good generalization occurs when the kernel eigenfunctions with large eigenvalues align well with the target function being learned. Thus, these analyses offer a quantitative measure of precisely *how* the inductive bias of a high-capacity kernel must align with that of the learning task. However, they do not provide a broader explanation of the nature and origin of the inductive bias in these models, nor why it often matches the data they are trained on. That is the big question we will attempt to address here.

We want to disambiguate the broader 1st-order question above from a more specific 2nd-order question of generalization – Given a high capacity DNN that generalizes reasonably well (e.g., it solves the 1st order overparameterization/large capacity problem), can we understand how to improve its performance further?—This second question is crucial for deep learning practitioners: variations in architecture, hyperparameter tuning, data augmentation, etc., can significantly enhance performance over basic vanilla DNNs. However, these adjustments and tricks start from a base of a high-capacity model that already confounds expectations from classical learning theory. Because the two questions are sometimes conflated, we want to emphasize up front that this paper will focus on the 1st-order question, which is relevant to all high-capacity models. A better understanding of this fundamental question should help frame important second-order questions about further improving DNN performance.

## Learning Boolean functions: a model system

Inspired by calls to study model systems[3,4], we first examine how a fully connected network (FCN) learns Boolean functions $f$: $\{0,1\}^n \to \{0,1\}$, which are key objects of study in computer science. Just as the Ising model does for magnetism, this simple but versatile model allows us to capture the essence of the overparameterization problem, while remaining highly tractable. For a system of size $n$, there are $2^n$ inputs, and $2^{2^n}$ Boolean functions. Given a Boolean target function $f_t$, the DNN is trained on a subset $S$ of $m < 2^n$ inputs, and then provides a prediction on a test set $T$ which consists of the rest of the inputs. A key advantage of this system is that data complexity can easily be varied by choice of target function $f_t$. Moreover, the model's tractability allows us to calculate the prior $P(f)$, likelihood, $P(S|f)$ and posterior $P(f|S)$ for different functions and targets, and so cast the tripartite schema of architecture, training algorithm, and structured data from[4] into a textbook Bayesian picture.

## Results

### Quantifying inductive bias with Bayesian priors

The prior over functions, $P(f)$, is the probability that a DNN $\mathcal{N}(\Theta)$ expresses $f$ upon random sampling of parameters over a parameter initialization distribution $P_{\text{par}}(\Theta)$:

$$P(f) = \int \mathbb{1}[\mathcal{N}(\Theta) = = f] P_{\text{par}}(\Theta) d\Theta, \tag{1}$$

where $\mathbb{1}$ is an indicator function (1 if its argument is true, and 0 otherwise). Explicitly, this term is 1 if the neural network $\mathcal{N}(\Theta)$ expresses $f$ with parameters $\Theta$, else 0. It was shown in ref. 24 that, for ReLU activation functions, $P(f)$ for the Boolean system was insensitive to different choices of $P_{\text{par}}(\Theta)$, and that it exhibits an exponential bias of the form $P(f) \lesssim 2^{-aK(f)+b}$ towards simple functions with low descriptional complexity $\tilde{K}(f)$, which is a proxy for the true (but uncomputable) Kolmogorov complexity. We will, as in ref. 24, calculate $\tilde{K}(f)$ using $C_{LZ}$, a Lempel-Ziv (LZ) based complexity measure from ref. 25 on the $2^n$ long bitstring that describes the function, taken on an ordered list of inputs. Other complexity measures give similar results[24,26], so there is nothing fundamental about this particular choice. To simplify notation, we will use $K(f)$ instead of $\tilde{K}(f)$. The exponential drop of $P(f)$ with $K(f)$ in the map from parameters to functions is consistent with an algorithmic information theory (AIT) coding theorem[27] inspired *simplicity bias* bound[25] which works for a much wider set of input-output maps. It was argued in ref. 24 that if this inductive bias in the priors matches the simplicity of structured data then it would help explain why DNNs generalize so well. However, the weakness of that work, and related works arguing for such a bias towards simplicity[24,26,28–35], is that it is typically not possible to significantly change this inductive bias towards simplicity, making it hard to conclusively show that it is not some other property of the network that instead generates the good performance. Here we exploit a particularity of tanh activation functions that enable us to significantly vary the inductive bias of DNNs. In particular, for a Gaussian $P_{\text{par}}(\Theta)$ with standard deviation $\sigma_w$, it was shown[36,37] that, as $\sigma_w$ increases, there is a transition to a chaotic regime. Moreover, it was recently demonstrated that the simplicity bias in $P(f)$ becomes weaker in the chaotic regime[38] (see also Supplementary Note 3). We will

https://doi.org/10.1038/s41467-024-54813-x

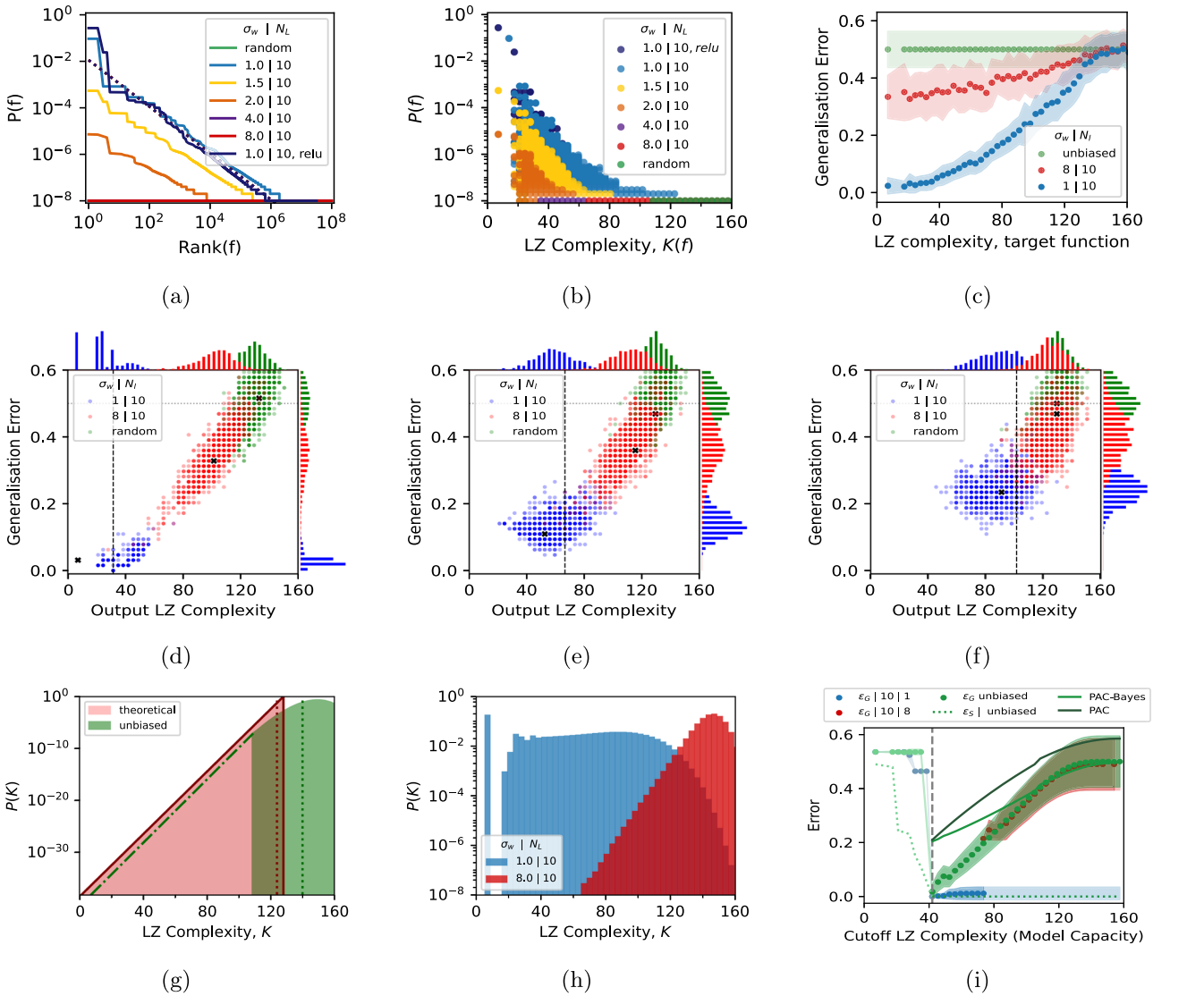


**Fig. 1 | Priors over functions and over complexity. a** Prior $P(f)$ that a $N_l$-layer FCN with tanh activations generates $n = 7$ Boolean functions $f$, ranked by probability of individual functions, generated from $10^8$ random samples of parameters $\Theta$ over a Gaussian $P_{\text{par}}(\Theta)$ with standard deviations $\sigma_w = 1 \ldots 8$. Also compared is a ReLU-activated DNN. The dotted blue line denotes a Zipf's law prior[24] $P(f) = 1/((128 \ln 2) Rank(f))$. **b** $P(f)$ versus LZ complexity $K$ for the networks from (**a**). **c** generalization error versus $K$ of the target function for an unbiased learner (green), and $\sigma_w = 1, 8$ tanh networks trained to zero error with advSGD[24] on cross-entropy loss with training set $S$ of size $m = 64$, for 1000 random initializations. The error is calculated on the remaining $|T| = 64$ functions. Error bars are one standard deviation (See Fig. S17 for PAC-Bayes bounds on this data). **d**, **e**, **f** Scatterplots of generalization error versus learned function LZ complexity, from 1000 random initializations for three target functions from subfigure (**c**). The dashed vertical line denotes the target function complexity. The black cross represents the mode function. The histograms at the top (side) of the plots show the posterior probability upon training as a function of complexity, $P_{\text{SGD}}(K|S)$ (error, $P_{\text{SGD}}(\epsilon_G|S)$). **g** The prior probability $P(K)$ to obtain a function of LZ complexity $K$ for uniform random sampling of $10^8$, compared to a theoretical perfect compressor. 90% of the probability mass lies to the right of the vertical dotted lines, and the dash-dot line denotes an extrapolation to low $K$. **h** $P(K)$ is relatively uniform on $K$ for the $\sigma_w = 1$ system, while it is highly biased towards complex functions for the $\sigma_w = 8$ networks. The large difference in these priors helps explain the significant variation in DNN performance. **i** generalization error for the K-learning restriction for the $\sigma_w = 1$, 8 DNNs and for an unbiased learner, all for $|S| = 100$. $\epsilon_S$ is the training error and $\epsilon_G$ is the generalization error on the test set. The vertical dashed line is the complexity $K_t$ of the target. Also compared are the standard realizable PAC and marginal-likelihood PAC-Bayes bounds for the unbiased learner. In $10^4$ samples, no solutions were found with $K \lesssim 70$ for the $\sigma_w = 8$ DNN, and with $K \gtrsim 70$ for the $\sigma_w = 1$ DNN.

exploit this behavior to systematically vary the inductive bias over functions in the prior.

In Fig. 1a, b we depict prior probabilities $P(f)$ for functions $f$ defined on all 128 inputs of a $n = 7$ Boolean system upon random sampling of parameters of an FCN with 10 layers and hidden width 40 (which is provably fully expressive for this system[31]), and tanh activation functions. The simplicity bias in $P(f)$ becomes weaker as the width $\sigma_w$ of the Gaussian $P_{\text{par}}(\sigma_w)$ increases. By contrast, for ReLU activations, the bias in $P(f)$ barely changes with $\sigma_w$ (see Fig. S3a). The effect of the decrease in simplicity bias on DNN generalization performance is demonstrated in Fig. 1c for a DNN trained to zero error on a training set $S$ of size $m = 64$ using advSGD (an SGD variant taken from ref. 24), and tested on the other 64 inputs $x_i \in T$. The generalization error (the fraction of incorrect predictions on $T$) varies as a function of the complexity of the target function. Although all these DNNs exhibit simplicity bias, weaker forms of the bias correspond to significantly worse generalization on the simpler targets (see also Supplementary Note 10). For very complex targets, both networks perform poorly. For reference, we also show an unbiased learner, where functions $f$ are chosen uniformly at random with the proviso that they exactly fit the

training set $S$. Not surprisingly, given the $2^{64} \approx 2 \times 10^{19}$ functions that can fit $S$, the performance of this unbiased learner is no better than random chance.

The scatter plots of Fig. 1d–f depict a more fine-grained picture of the behavior of the SGD-trained networks for three different target functions. For each target, 1000 independent initializations of the SGD optimizer, with initial parameters taken from $P_{\text{par}}(\sigma_w)$, are used. The generalization error and complexity of each function found when the DNN first reaches zero training error are plotted. Since there are $2^{64}$ possible functions that give zero error on the training set $S$, it is not surprising that the DNN converges to many different functions upon different random initializations. For the $\sigma_w = 1$ network (where $P(f)$ resembles that of ReLU networks) the most common function is typically simpler than the target. By contrast, the less biased network converges on functions that are typically more complex than the target. As the target itself becomes more complex, the relative difference between the two generalization errors decreases, because the strong inductive bias towards simple functions of the first network becomes less useful. No free lunch theorems for supervised learning tell us that when averaged over all target functions, the three learners above will perform equally badly[39,40] (see also Supplementary Note 43).

## Priors over complexity

To understand why relatively modest changes in the inductive bias towards simplicity lead to such significant differences in generalization performance, we need another important ingredient, namely how the *number* of functions vary with complexity. Basic counting arguments imply that the number of strings of a fixed length that have complexity $K$ scales exponentially as $2^K$ [27]. Therefore, the vast majority of functions picked at random will have high complexity. This exponential growth of the number of functions with complexity can be captured in a more coarse-grained prior, the probability $P(K)$ that the DNN expresses a function of complexity $K$ upon random sampling of parameters over a parameter initialization function $P_{\text{par}}(\Theta)$, which can also be written in terms of functions as $P(K') = \sum_{f \in \mathcal{H}_{K'}} P(f)$, the weighted sum over the set $\mathcal{H}_{K'}$ of all functions with complexity $\tilde{K}(f) = K'$. In Fig. 1g $P(K)$ is shown for uniform random sampling of functions for $10^8$ samples using the LZ measure, and also for the theoretical ideal compressor with $P(K) = 2^{K - K_{max} - 1}$ over all $2^{128} \approx 3 \times 10^{38}$ functions (see also Supplementary Note 9). In (h) we display $P(K)$ for functions not sampled at random, but rather from the two networks. There is a dramatic difference between random sampling functions (as in (g)) and between the network with $\sigma_w = 1$, where $P(K)$ is nearly flat. This behavior follows from the interesting fact that the AIT coding theorem-like scaling[24,25] of the prior over functions $P(f) \sim 2^{-\tilde{K}(f)}$ counters the $2^K$ growth in the number of functions.

By contrast, even though, relative to the 38 or so orders of magnitude scale on which $P(f)$ varies, the more artefactual $\sigma_w = 8$ system has strong simplicity bias (we estimate that for the simplest functions, $P(f)$ is about $10^{25}$ times higher than the mean probability $\langle P(f) \rangle = 2^{-128} \approx 3 \times 10^{-39}$), this is not enough to counter the $2^K$ growth in the number of functions with complexity. Therefore, this DNN is exponentially more likely to throw up complex functions, an effect that SGD is unable to overcome.

More generally, the fact that the number of complex functions grows exponentially with complexity $K$ lies at the heart of the classical explanation of why an insufficiently biased agent suffers from variance: It can too easily find many different functions that all fit the data. The marked differences in the generalization performance between the two networks observed in Fig. 1c–f can be therefore traced to differences in the inductive bias of the networks, as measured by the differences in their priors.

## Artificially restricting model capacity

To further illustrate the effect of inductive bias we create a K-learner that only allows functions with complexity $\leq K_M$ to be learned and discards all others. As can be seen in Fig. 1i, the learners typically cannot reach zero training error on the training set if $K_M$ is less than the target function complexity $K_t$. For $K_M \geq K_t$, zero training error can be reached and not surprisingly, the lowest generalization error occurs when $K_M = K_t$. As the upper limit $K_M$ is increased, all three learning agents are more likely to make errors in predictions due to variance. The random learner has an error that grows linearly with $K_M$. This behavior can be understood with a classic Probably Approximately Correct (PAC) bound[6] where the generalization error (with confidence $0 \leq (1 - \delta) \leq 1$) scales as $\epsilon_G \leq (\ln |\mathcal{H}_{\leq K_M}| - \ln \delta)/m$, where $|\mathcal{H}_{\leq K_M}| K \leq K_M$ is the size of the hypothesis class of all functions with $K \leq K_M$; the bound scales linearly in $K_M$, as the error does (see Supplementary Note 7 for further discussion including the more sophisticated PAC-Bayes bound[41,42]). The generalization error for the $\sigma_w = 1$ DNN does not change much with $K_M$ for $K_M > K_t$ because the strong inductive bias towards simple solutions means access to higher complexity solutions doesn't significantly change what the DNN converges on.

Finally, we show data for DNNs in the ordered regime with $\sigma_w \ll 1$, and for other optimizers, loss functions, and activation functions in Figs. S6–S11. These results broadly exhibit the same behavior we describe here.

## Calculating the Bayesian posterior and likelihood

To better understand the generalization behavior observed in Fig. 1 we apply Bayes' rule, $P(f|S) = P(S|f)P(f)/P(S)$ to calculate the Bayesian posterior $P(f|S)$ from the prior $P(f)$, the likelihood $P(S|f)$, and the marginal likelihood $P(S)$. Since we condition on zero training error, the likelihood takes on a simple form. $P(S|f) = 1$if $\forall\ x_i \in S, f(x_i) = y_i$, while $P(S|f) = 0$ otherwise. For a fixed training set, all the variation in $P(f|S)$ for $f \in U(S)$, the set of all functions compatible with $S$, comes from the prior $P(f)$ since $P(S)$ is constant. Therefore, in this Bayesian picture, the bias in the prior is translated over to the posterior.

The marginal likelihood also takes a relatively simple form for discrete functions, since $P(S) = \sum_f P(S|f)P(f) = \sum_{f \in U(S)} P(f)$. It is equivalent to the probability that the DNN obtains zero error on the training set $S$ upon random sampling of parameters, and so can be interpreted as a measure of the inductive bias towards the data. The Marginal-likelihood PAC-Bayes bound[42] makes a direct link $P(S) \lesssim e^{-m\epsilon_G}$ to the generalization error $\epsilon_G$ which captures the intuition that, for a given $m$, a better inductive bias towards the data (larger $P(S)$) implies better performance (lower $\epsilon_G$).

One can also define the posterior probability $P_{\text{SGD}}(f|S)$, that a network trained with SGD (or another optimizer) on training set $S$, when initialized with $P_{\text{par}}(\Theta)$, converges on function $f$. For simplicity, we take this probability at the epoch where the system first reaches zero training error. Note that in Fig. 1d–f it is this SGD-based posterior that we plot in the histograms at the top and sides of the plots, with functions grouped either by complexity, which we will call $P_{\text{SGD}}(K|S)$, or by generalization error $\epsilon_G$, which we will call $P_{\text{SGD}}(\epsilon_G|S)$.

DNNs are typically trained by some form of SGD, and not by randomly sampling over parameters which is much less efficient. However, a recent study[43] which carefully compared the two posteriors has shown that to first order, $P_{\text{B}}(f|S) \approx P_{\text{SGD}}(f|S)$, for many different data sets and DNN architectures. We demonstrate this close similarity in Fig. S15 explicitly for our $n = 7$ Boolean system. This evidence suggests that Bayesian posteriors calculated by random sampling of parameters, which are much simpler to analyze, can be used to understand the dominant behavior of an SGD-trained DNN, even if, for example, hyperparameter tuning can lead to 2nd-order deviations between the two methods (see also Supplementary Note 1).

To test the predictive power of our Bayesian picture, we first define the function error $\epsilon(f)$ as the fraction of incorrect labels $f$ produces on the full set of inputs. Next, we average Bayes' rule over all

training sets $S$ of size $m$:

$$\langle P(f|S)\rangle_m = P(f)\left\langle \frac{P(S|f)}{P(S)} \right\rangle_m \approx \frac{P(f)(1-\epsilon(f))^m}{\langle P(S)\rangle_m} \quad (2)$$

where the mean likelihood $\langle P(S|f)\rangle_m = (1-\epsilon(f))^m$ is the probability of a function $f$ obtaining zero error on a training set of size $m$. In the second step, we approximate the average of the ratio with the ratio of the averages which should be accurate if $P(S)$ is highly concentrated, as is expected if the training set is not too small.

Equation (2) is hard to calculate, so we coarse-grain it by grouping together functions by their complexity:

$$\langle P(K|S)\rangle_m = \sum_{C_{LZ}(f)=K} \langle P(f|S)\rangle_m \propto P(K)\langle (1-\epsilon_G(K))^m\rangle_l, \quad (3)$$

and in the second step make a *decoupling approximation* where we average the likelihood term over a small number $l$ of functions with complexity $K$ with lowest generalization error $\epsilon_G(K)$ since the smallest errors in the sum dominate exponentially since $(1-\epsilon_G) \approx e^{-\epsilon_G}$ for $|\epsilon_G| \ll 1$. We then multiply by $P(K)$, which takes into account the value of the prior and the multiplicity of functions at that $K$, and normalize $\sum_K P(K|S) = 1$. For a given target, we make the ansatz that this decoupling approximation provides an estimate that scales as the true (averaged) posterior.

To test our approximations, we first plot, in Fig. 2a–c, the likelihood term in Equation (3) for three different target functions. To obtain these curves, we considered a large number of functions (including all functions with up to 5 errors w.r.t. the target, with further functions sampled). For each complexity, we average this term over the $l = 5$ functions with smallest $\epsilon_G$. Not surprisingly, functions close to

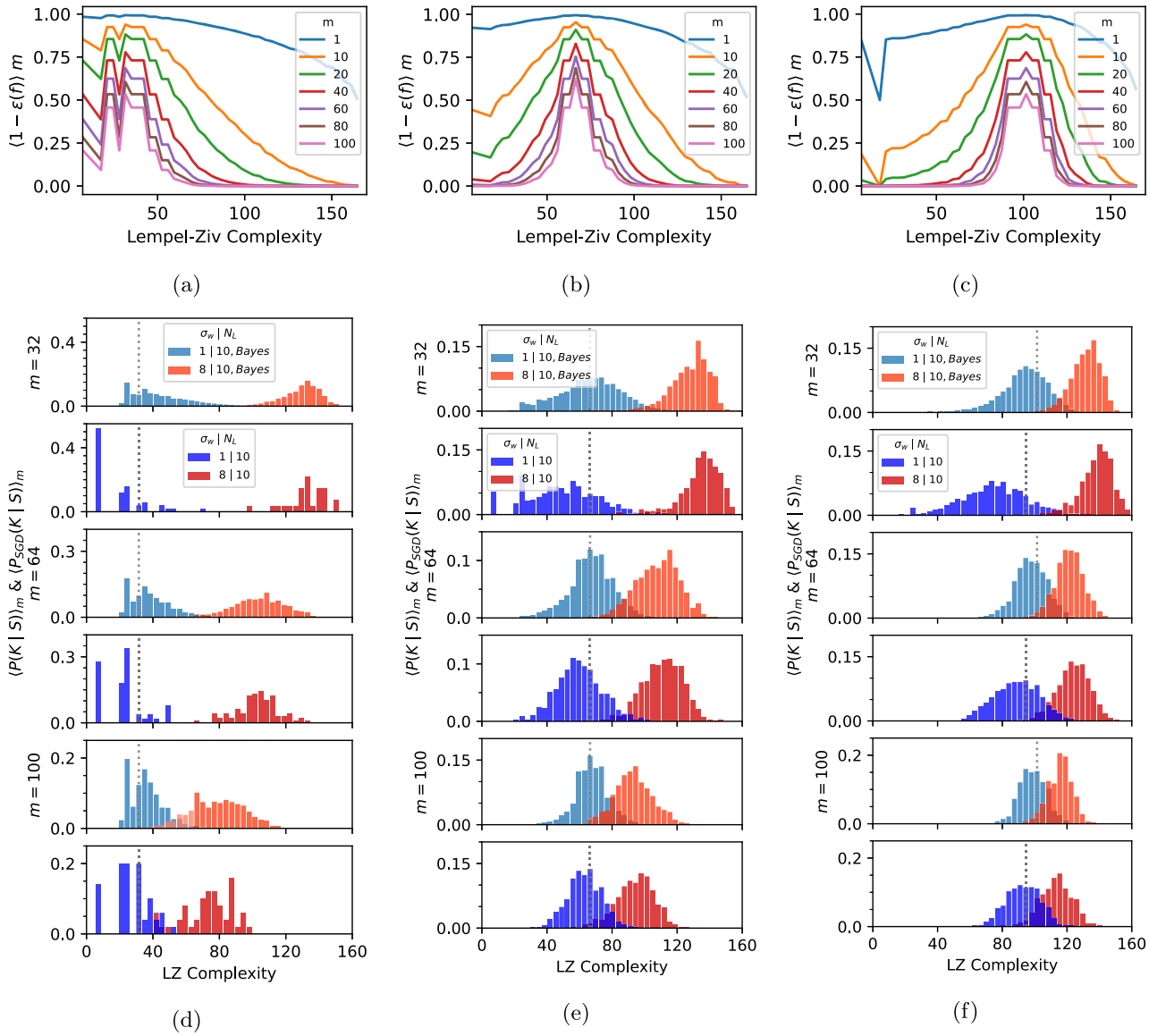


**Fig. 2 | How training data affects the posteriors. a–c** depict the mean likelihood $\langle (1-\epsilon_G(K))^m\rangle_5$ from Equation (3), averaged over training sets, and over the 5 lowest error functions at each $K$. This term depends on data and is independent of the DNN architecture. With increasing $m$ it peaks more sharply around the complexity of the target. In (**d–f**) we compare the posteriors over complexity, $\langle P_{SGD}(K|S)\rangle_m$, for SGD (darker blue and red) averaged over training sets of size $m$, to the prediction of $\langle P(K|S)\rangle_m$ from Equation (3) (lighter blue and orange), calculated by multiplying the Bayesian likelihood curves in (**a–c**) by the prior $P(K)$ shown in Fig. 1h. The light (Bayes) and dark (DNN) blue histograms are from the $\sigma_w = 1$ system, and the orange (Bayes) and red (DNN) histograms are from the $\sigma_w = 8$ system which has less bias towards simple functions. The Bayesian decoupling approximation (Equation (3)) captures the dominant trends in the behavior of the SGD-trained networks as a function of data complexity and training set size. Quantitative measures of the similarity between the posteriors can be found in Fig. S22.

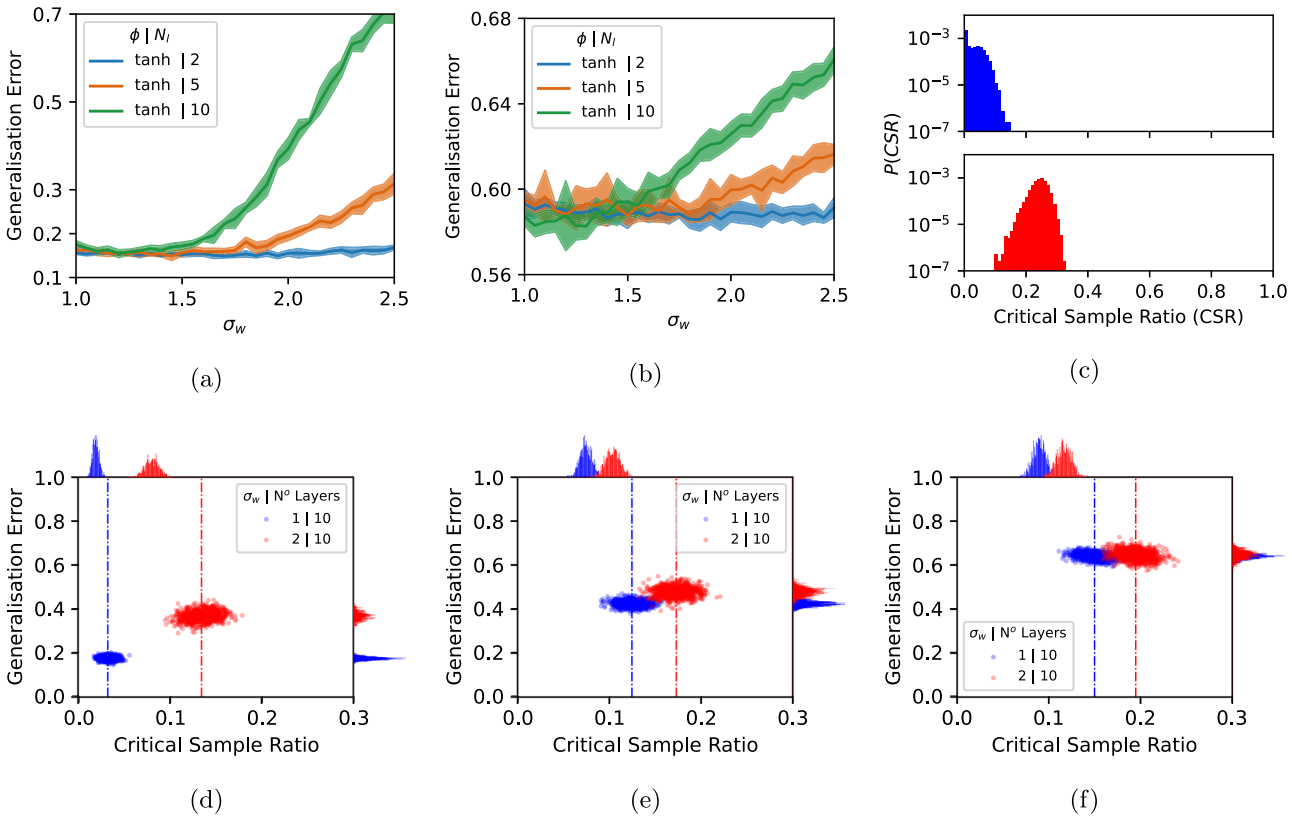


**Fig. 3 | MNIST and CIFAR-10 data. a** MNIST generalization error for FCNs on a 1000 image training set versus $\sigma_w$ for three depths. **b** CIFAR10 generalization error for FCNs trained on a 5000 image training set versus $\sigma_w$ for three depths. The FCNs, made of multiple hidden layers of width 200, were trained with SGD with batch size 32 and lr=$10^{-3}$ until 100% accuracy was first achieved on the training set. Error bars are one standard deviation. **c** Complexity prior $P(K)$, for CSR complexity, for 1000 MNIST images for randomly initialized networks of 10 layers and $\sigma_w$ = 1, 2. Probabilities are estimated from a sample of $2 \times 10^4$ parameters. **d**, **e**, **f** are scatterplots of generalization error versus the CSR for 1000 networks trained to 100% accuracy on a training set of 1000 MNIST images and tested on 1000 different images. In (**d**) the training labels are uncorrupted, in (**e**, **f**) 25% and 50% of the training labels are corrupted respectively. Note the qualitative similarity to the scatter plots in Fig. 1d–f.

the complexity of the target have the smallest error. These graphs help illustrate how the DNN interacts with data. As the training set size $m$ increases, functions that are most likely to be found upon training to zero training error are increasingly concentrated close to the complexity of the target function.

To test the decoupling approximation from Eq. (3), we compare in Fig. 2d–f the posterior $\langle P(K|S)\rangle_m$, calculated by multiplying the Bayesian likelihood curve from Fig. 2a–c with the two Bayesian priors $P(K)$ from Fig. 1h, i, to the posteriors $\langle P_{rmSGD}(K|S)\rangle_m$ calculated by advSGD[24] over a 1000 different parameter initializations and training sets. It is remarkable to see how well the simple decoupling approximation performs across target functions and training set sizes. In Figs. S13 and S14 we demonstrate the robustness of our approach by showing that using $l$ = 1 or $l$ = 50 functions does not change the predictions much. This success suggests that our simple approach captures the essence of the interaction between the data (measured by the likelihood, which is independent of the learning algorithm), and the DNN architecture (which is measured by the prior and is independent of the data).

We have therefore separated out two of the three parts of the tripartite scheme, which leaves the training algorithm. In the figures above our Bayesian approximation captures the dominant behavior of an SGD-trained network. This correspondence is consistent with the results and arguments of ref. 43. We checked this further in Fig. S15 for a similar set-up using MSE loss, where Bayesian posteriors can be exactly calculated using Gaussian processes (GPs). The direct Bayesian GP calculation closely matches SGD-based results for our much smaller network. Note that, in the spirit of model calculations, as called for in ref. 3,4, we mainly used a much smaller DNN. But their agreement with the GP-based posteriors, calculated for the infinite width limit, shows that at the scale of our Bayesian approach to the 1st-order generalization question we are addressing here, the size of the DNN is not an important factor. The width of a DNN can, of course, be a factor for 2nd order generalization questions.

## Beyond the Boolean model: MNIST & CIFAR-10

Can the principles worked out for the Boolean system be observed in larger systems that are closer to the standard practice of DNNs? To this end, we show, in Fig. 3a, b how the generalization error for the popular image datasets MNIST and CIFAR-10 changes as a function of the initial parameter width $\sigma_w$ and the number of layers $N_l$ for a standard FCN, trained with SGD on cross-entropy loss with tanh activation functions. Larger $\sigma_w$ and larger $N_l$ push the system deeper into the chaotic regime[36,37] and result in decreasing generalization performance, similar to what we observe for the Boolean system for relatively simple targets. In Fig. 3, we plot the prior over complexity $P(K)$ for a complexity measure called the critical sample ratio (CSR)[28], an estimate of the density of decision boundaries that should be appropriate for this problem. Again, increasing $\sigma_w$ greatly increases the prior probability that the DNN produces more complex functions upon random sampling of parameters. Thus the decrease in generalization performance is consistent with the inductive bias of the network becoming less simplicity biased, and therefore less well aligned with structure in the

data. Indeed, datasets such as MNIST and CIFAR-10 are thought to be relatively simple[44,45].

These patterns are further illustrated in Fig. 3d–f where we show scatterplots of generalization error v.s. CSR complexity for three target functions that vary in complexity (here obtained by corrupting labels). The qualitative behavior is similar to that observed for the Boolean system in Fig. 1. The more simplicity-biased networks perform significantly better on the simpler targets, but the difference with the less simplicity-biased network decreases for more complex targets. While we are unable to directly calculate the likelihoods because these systems are too big, we argue that the strong similarities to our simpler model system suggest that the same basic principles of inductive bias are at work here.

## Discussion

In order to generalize, high capacity models need a clear inductive bias towards functions that perform well on the data being studied[46]. Here we show that DNNs and also NNGPs and DNN-inspired kernels have a very specific kind of inbuilt Occam's razor (see Supplementary Note 4 for background on Occam's razor(s)). This inductive bias can be quantified by an AIT coding theorem-like scaling of the prior as $P(f) \propto 2^{-K(f)}$ which can counteract the $2^K$ growth of the number of functions with complexity. If this intrinsic inductive bias is slightly weaker, say $P(f) \propto 2^{-\alpha K(f)}$ with $\alpha < 1$, then it becomes much harder to overcome the $2^K$ growth and the learner will likely suffer from strong variance problems. While we were not able to significantly increase the bias towards simplicity in DNNs, too strong a bias towards simplicity can mean bias (instead of variance) problems because complex functions become hard to find[26,47].

An interesting direction to explore is the more formal arguments in AIT relating to the optimality of Solomonoff induction[27,48,49] (see also Supplementary Note 44). In particular, there may be fruitful links between simplicity bias, Solomonoff induction, and compression in deep learning, as recently discussed for example in the context of large language models[50,51].

Another promising direction for exploration is to connect our high-level theory here with the more detailed calculations of generalization in kernels[19–23,15]. These works can tell us *how* DNN inspired kernels need to align with data, but not so easily *why* this would be so. An extra challenge is that our work here has been in the context of discrete functions and classification, whereas the work on kernels is typically for continuous regression scenarios. However, the classification setting allows us to study the inductive bias of a model over the entire space of input functions and simultaneously make concrete generalization bounds. We distinguished the first-order question of why high-capacity DNNs (and related models) generalize at all from the second-order question of how to further improve DNN performance. Firstly, the fact that DNNs are strongly biased towards simplicity gives a basis upon which to look for further inductive biases in other directions that may be orthogonal to simplicity. Our results are derived in a Bayesian setting, but do not preclude the fact that SGD optimization itself can also introduce useful inductive biases (see also Supplementary Note 1). In particular, SGD likely enables feature learning. Our understanding of how this works, even in the infinite width kernel or GP limit[52–54], remains in early stages, but feature-learning is likely to play an important role in explaining why DNNs outperform other methods on large datasets.

Much of the exciting recent progress in foundation models such as GPT-4 likely arises from optimizer-induced inductive biases beyond the simplicity bias at initialization that may also play an important role there[26]. Direct inspiration from biology can also provide inspiration for alternate inductive biases, as recently shown for image recognition in CNNs[55]. Systematically studying and understanding how these inductive biases interact with data remains one of the key challenges in modern machine learning.

Finally, our observations about inductive bias can be inverted to identify characteristics—such as limits on the complexity – of the data that DNNs can successfully learn. In particular, the remarkable success of DNNs on a broad range of scientific problems[56–59] suggests that their inductive biases must recapitulate something deep about the structure of the natural world[29]. By understanding why DNNs select specific solutions, we may, in turn, gain profound insights into the structure of nature itself[60].

## Data availability

All code to generate the data used in this study is publicly available under the CC-BY 4.0 license in the following zenodo repository https://doi.org/10.5281/zenodo.13997032.

## Code availability

All code to generate the figures in this study is publicly available in the following zenodo repository https://doi.org/10.5281/zenodo.13997032.

## References


1. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436 (2015).
2. Schmidhuber, J. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Netw.* **10**, 857–873 (1997).
3. Breiman, L. Reflections after refereeing papers for nips. in *The Mathematics of Generalization* (Addison-Wesley, 1995).
4. Zdeborová, L. Understanding deep learning is also a job for physicists. *Nat. Phys.* **16**, 602–604 (2020).
5. Dyson, F. A meeting with enrico fermi. *Nature* **427**, 297–297 (2004).
6. Shalev-Shwartz, S. & Ben-David, S. *Understanding machine learning: From theory to algorithms* (Cambridge University Press, 2014).
7. Zhang, C., Bengio, S., Hardt, M., Recht, B. & Vinyals, O. Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530 (2016).
8. Cybenko, G. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **2**, 303–314 (1989).
9. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Netw.* **4**, 251–257 (1991).
10. Neal, R. M. Priors for infinite networks. in *Bayesian Learning for Neural Networks*. (Springer, 1994).
11. Lee, J. et al. Deep neural networks as Gaussian processes. In *International Conference on Learning Representations*. https://openreview.net/forum?id=B1EA-M-0Z (2018).
12. Matthews, A. G. D. G., Rowland, M., Hron, J., Turner, R. E. & Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. arXiv preprint arXiv:1804.11271 (2018).
13. Jacot, A., Gabriel, F. & Hongler, C. Neural tangent kernel: convergence and generalization in neural networks. *Adv. Neural Inform. Process. Syst.* 8571–8580 (2018).
14. Belkin, M., Ma, S. & Mandal, S. To understand deep learning we need to understand kernel learning. In *Proc. 35th International Conference on Machine Learning, vol. 80 of Proceedings of Machine Learning Research* (eds Dy, J. & Krause, A.), 541–549 (PMLR, Stockholmsmässan, Stockholm Sweden, 2018). http://proceedings.mlr.press/v80/belkin18a.html.
15. Simon, J. B., Dickens, M. & DeWeese, M. R. Neural tangent kernel eigenvalues accurately predict generalization. arXiv preprint arXiv:2110.03922 (2021).
16. Lee, J. et al. Finite versus infinite neural networks: an empirical study. *Adv. Neural Inf. Process. Syst.* **33**, 15156–15172 (2020).
17. Arora, S. et al. On exact computation with an infinitely wide neural net. *Adv. Neural Inform. Process. Syst.* 8139–8148 (2019).
18. Belkin, M., Hsu, D., Ma, S., & Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off.

*Proceedings of the National Academy of Sciences* **116**, 15849–15854 (2019).
19. Bordelon, B., Canatar, A. & Pehlevan, C. Spectrum dependent learning curves in kernel regression and wide neural networks. In *International Conference on Machine Learning*, 1024–1034 (PMLR, 2020).
20. Cohen, O., Malka, O. & Ringel, Z. Learning curves for overparametrized deep neural networks: a field theory perspective. *Phys. Rev. Res.* **3**, 023034 (2021).
21. Canatar, A., Bordelon, B. & Pehlevan, C. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nat. Commun.* **12**, 1–12 (2021).
22. El Harzli, O., Grau, B. C., Valle-Pérez, G. & Louis, A. A. Double-descent curves in neural networks: a new perspective using Gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38, No. 10, pp. 11856–11864 (2024).
23. Cui, H., Loureiro, B., Krzakala, F. & Zdeborová, L. Generalization error rates in kernel regression: the crossover from the noiseless to noisy regime. *Adv. Neural Inf. Process. Syst.* **34**, 10131–10143 (2021).
24. Valle-Pérez, G., Camargo, C. Q. & Louis, A. A Deep learning generalizes because the parameter-function map is biased towards simple functions. arXiv preprint arXiv:1805.08522 (2018).
25. Dingle, K., Camargo, C. Q. & Louis, A. A. Input–output maps are strongly biased towards simple outputs. *Nat. Commun.* **9**, 1–7 (2018).
26. Bhattamishra, S., Patel, A., Kanade, V. & Blunsom, P. Simplicity bias in transformers and their ability to learn sparse boolean functions. arXiv preprint arXiv:2211.12316 (2022).
27. Li, M. & Vitanyi, P. *An introduction to Kolmogorov complexity and its applications* (Springer-Verlag New York Inc, 2008).
28. Arpit, D. et al. A closer look at memorization in deep networks. arXiv preprint arXiv:1706.05394 (2017).
29. Lin, H. W., Tegmark, M. & Rolnick, D. Why does deep and cheap learning work so well? *J. Stat. Phys.* **168**, 1223–1247 (2017).
30. De Palma, G., Kiani, B. T. & Lloyd, S. Random deep neural networks are biased towards simple functions. arXiv preprint arXiv:1812.10156 (2018).
31. Mingard, C. et al. Neural networks are a priori biased towards boolean functions with low entropy. arXiv preprint arXiv:1909.11522 (2019).
32. Kalimeris, D. et al. Sgd on neural networks learns functions of increasing complexity. *Adv. Neural Inform. Process. Syst.* **32** (2019).
33. Huh, M. et al. The low-rank simplicity bias in deep networks. arXiv preprint arXiv:2103.10427 (2021).
34. Farhang, A. R., Bernstein, J. D., Tirumala, K., Liu, Y. & Yue, Y. Investigating generalization by controlling normalized margin. In *International Conference on Machine Learning*, 6324–6336 (PMLR, 2022).
35. Chiang, P.-y. et al. Loss landscapes are all you need: neural network generalization can be explained without the implicit bias of gradient descent. In *The Eleventh International Conference on Learning Representations* (ICLR, 2023).
36. Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J. & Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. *Adv. Neural Inform. Process. Syst.* 3360–3368 (2016).
37. Schoenholz, S. S., Gilmer, J., Ganguli, S. & Sohl-Dickstein, J. Deep information propagation. In *International Conference on Learning Representations* (2017). https://openreview.net/forum?id=H1W1UN9gg.
38. Yang, G. & Salman, H. A fine-grained spectral perspective on neural networks. arXiv preprint arXiv:1907.10599 (2019).
39. Schaffer, C. A conservation law for generalization performance. In *Machine Learning Proceedings 1994*, 259–265 (Elsevier, 1994).
40. Wolpert, D. H. The lack of a priori distinctions between learning algorithms. *Neural Comput.* **8**, 1341–1390 (1996).
41. McAllester, D. A. Some pac-bayesian theorems. *Mach. Learn.* **37**, 355–363 (1999).
42. Valle-Pérez, G. & Louis, A. A. Generalization bounds for deep learning. arXiv preprint arXiv:2012.04115 (2020).
43. Mingard, C., Valle-Pérez, G., Skalse, J. & Louis, A. A. Is sgd a bayesian sampler? well, almost. *J. Mach. Learn. Res.* **22**, 1–64 (2021).
44. Spigler, S., Geiger, M. & Wyart, M. Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm. *J. Stat. Mech. Theory Exp.* **2020**, 124001 (2020).
45. Goldt, S., Mézard, M., Krzakala, F. & Zdeborová, L. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X* **10**, 041044 (2020).
46. Mitchell, T. M. The need for biases in learning generalizations (rutgers computer science tech. rept. cbm-tr-117). *Rutgers University* (1980).
47. Shah, H., Tamuly, K., Raghunathan, A., Jain, P. & Netrapalli, P. The pitfalls of simplicity bias in neural networks. *Adv. Neural Inf. Process. Syst.* **33**, 9573–9585 (2020).
48. Rathmanner, S. & Hutter, M. A philosophical treatise of universal induction. *Entropy* **13**, 1076–1136 (2011).
49. Grau-Moya, J. et al. Learning universal predictors. arXiv preprint arXiv:2401.14953 (2024).
50. Delétang, G. et al. Language modeling is compression. arXiv preprint arXiv:2309.10668 (2023).
51. Li, Z. et al. Understanding is compression. arXiv preprint arXiv:2407.07723 (2024).
52. Seroussi, I., Naveh, G. & Ringel, Z. Separation of scales and a thermodynamic description of feature learning in some cnns. *Nat. Commun.* **14**, 908 (2023).
53. Pacelli, R., Ariosto, S. & Ginelli, F. A statistical mechanics framework for bayesian deep neural networks beyond the infinite-width limit. *Nat. Mach. Intell.* **5**, 1497–1507 (2023).
54. Yang, G., Simon, J. B. & Bernstein, J. A spectral condition for feature learning. arXiv preprint arXiv:2310.17813 (2023).
55. Evans, B. D., Malhotra, G. & Bowers, J. S. Biological convolutions improve dnn robustness to noise and generalisation. *Neural Netw.* **148**, 96–110 (2022).
56. Ching, T. et al. Opportunities and obstacles for deep learning in biology and medicine. *J. R. Soc. Interface* **15**, 20170387 (2018).
57. Mehta, P. et al. A high-bias, low-variance introduction to machine learning for physicists. *Phys. Rep.* **810**, 1–124 (2019).
58. Carleo, G. et al. Machine learning and the physical sciences. *Rev. Mod. Phys.* **91**, 045002 (2019).
59. Karniadakis, G. E. et al. Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
60. Belkin, M. Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numerica* **30**, 203–248 (2021).

## Acknowledgements

We thank Satwik Bhattamishra, Kamal Dingle, Nayara Fonseca, and Yoonsoo Nam for helpful discussions. C.M. acknowledges funding from an EPSRC iCASE grant with IBM (grant number EP/S513842/1)

## Author contributions

C.M. and H.R. designed, conducted, and analyzed the experiments, and contributed to writing the manuscript. G.V.P. conducted and analyzed experiments. A.A.L. designed the project, analyzed the experiments, and contributed to writing the manuscript. 

## Competing interests

The authors declare no competing interests.

## Additional information

**Supplementary information** The online version contains supplementary material available at https://doi.org/10.1038/s41467-024-54813-x.

**Correspondence** and requests for materials should be addressed to Ard A. Louis.

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