DEEP LEARNING GENERALIZES BECAUSE THE PARAMETER-FUNCTION MAP IS BIASED TOWARDS SIMPLE FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks generalize remarkably well without explicit regularization even in the strongly over-parametrized regime. This success suggests that some form of implicit regularization must be at work. In this paper we argue that a strong intrinsic bias in the parameter-function map helps explain the success of deep neural networks. We provide evidence that the parameter-function map results in a heavily biased prior over functions, if we assume that the training algorithm samples parameters close to uniformly within the zero-error region. The PAC-Bayes theorem then guarantees good expected generalization for target functions producing high-likelihood training sets. We exploit connections between deep neural networks and Gaussian processes to estimate the marginal likelihood, finding remarkably good agreement between Gaussian processes and neural networks for small input sets. Using approximate marginal likelihood calculations we produce nontrivial generalization PAC-Bayes error bounds which correlate well with the true error on realistic datasets such as MNIST and CIFAR and for architectures including convolutional and fully connected networks. As predicted by recent arguments based on algorithmic information theory, we find that the prior probability drops exponentially with linear increases in several measures of descriptional complexity of the target function. As target functions in many real problems are expected to be highly structured, this simplicity bias offers an insight into why deep networks generalize well on real world problems, but badly on randomized data.

1 INTRODUCTION

Deep learning is a machine learning paradigm based on very large, expressive and composable models, which most often require similarly large data sets to train. The name comes from the main component in the models: deep neural networks, or artificial neural networks with many layers of representation. These models have been remarkably successful in domains ranging from image recognition and synthesis, to natural language processing, and reinforcement learning (Mnih et al. (2015); LeCun et al. (2015); Radford et al. (2015); Schmidhuber (2015)). There has been work on understanding the expressive power of certain classes of deep networks (Poggio et al. (2017)), their learning dynamics (Advani & Saxe (2017); Liao & Poggio (2017)), and generalization properties (Kawaguchi et al. (2017); Poggio et al. (2018)). However, a full theoretical understanding of many of these properties is still lacking.

Deep neural networks are typically over-parametrized, with many more parameters than training examples. The success of these highly-expressive models implies two things: 1) some form of inductive bias must be at work, to account for their successful generalization, and 2) classical learning theories based on worst-case[1] analyses such as those based on VC dimension, are insufficient to explain generalization in deep learning.

Regarding 1), it was originally thought that regularization methods such as Tikhonov regularization (Tikhonov (1943)), dropout (Srivastava et al. (2014)), or early stopping (Morgan & Bourlard

[1] Worst-case over all functions in the hypothesis class

(1990)) were key in providing this inductive bias. However, Zhang et al. (2016) demonstrated that highly-expressive deep neural networks still generalize successfully with no explicit regularization, reopening the question of the origin of the inductive bias. There is now more evidence that unregularized deep networks are biased towards simple functions (Arpit et al. (2017); Wu et al. (2017)). Stochastic gradient descent (SGD) has been conjectured as a possible cause of the bias (Soudry et al. (2017); Zhang et al. (2017)), and there is evidence that is may play a role, although there is no consensus in the field (Arpit et al. (2017)). Given that a large variety of algorithms have been used to train deep neural networks, from all the variants of SGD, to gradient-free methods like genetic algorithms (Such et al. (2017)), and given the fact that all of these perform well (the variance in generalization performance between them is relatively small) suggests that no strong assumptions are needed about the training algorithm to explain generalization.

The experiments by Zhang et al. (2016) also clearly demonstrated point 2), which spurred a wave of new work in learning theories tailored to deep learning (Kawaguchi et al. (2017); Arora et al. (2018); Morcos et al. (2018); Neyshabur et al. (2017b); Dziugaite & Roy (2017; 2018); Neyshabur et al. (2017a; 2018)). Although these works manage to find complexity measures and bounds that capture some of the observed behavior, none has yet managed to obtain bounds for realistic networks and training algorithms that fully explain the observed generalization performance.

The findings described above point to a different source for the remarkable generalization performance of deep neural networks. In this paper we claim that bias in the parameter-function map is the main reason why deep neural networks generalize.

1.1 Main contributions

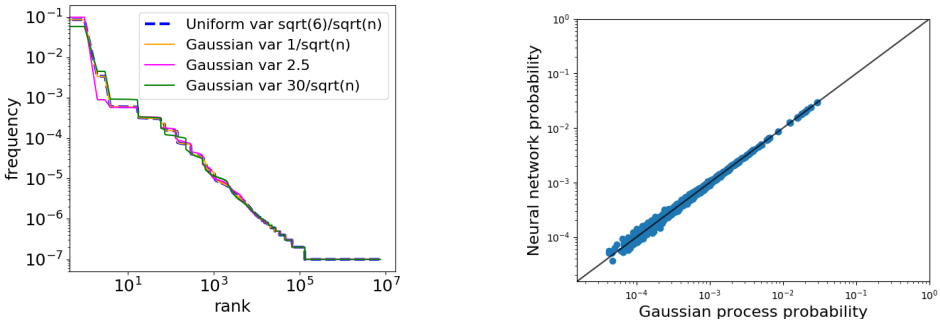
Our main contributions are:

- We show that the parameter-function map of deep neural networks is extremely biased towards simple functions, and therefore the prior over functions is expected to be extremely biased too. We claim this intrinsic bias is the fundamental source of inductive bias allowing neural networks generalize.
- We approximate the prior over functions using Gaussian processes, and present evidence that Gaussian processes reproduces neural network marginal likelihoods remarkably well even for finite width networks.
- Using the Gaussian process approximation of the prior, we compute PAC-Bayes expected generalization error bounds for a variety of common architectures and datasets. We obtain nonvacuous (less than 1) bounds, which follow the behaviour of the real generalization error.
- Finally, we show that the prior over functions correlates with measures of descriptional complexity of the functions –as predicted by recent results from algorithmic information theory (AIT) – hinting at why neural networks generalize for real-world problems.

2 The parameter-function map is highly biased

In order to explore the properties of the *parameter-function map*, we sample parameters using a Gaussian distribution or uniform within a hypercube, and measure the empirical frequencies by which different functions are obtained. This procedure is easiest for a discrete space of functions, and a small enough function space so that the probabilities of obtaining them more than once isn't negligible. We achieve this by using a ReLU neural network with a small number (7) of Boolean inputs, and a single Boolean output, so that the number of possible such (Boolean) functions is $2^{2^7} = 2^{128}$. We used a training set of 64 examples (half the size of the full space) and performed sampling for different distributions over parameters. As can be seen in Figure 1a the empirical (normalized) frequencies versus the rank exhibit a range of probabilities spanning as many orders of magnitude as the finite sample size allows (Note that the smallest probability must be less than $2^-128 \approx 3^{-39}$, so that the full range is at least 38 orders of magnitude) Using different distributions over parameters has a very small effect on the overall curve.

Although we used a realistic architecture, it is admittedly small by modern standards. Later in the paper we will show (indirect) evidence that a very strong bias is also present in the parameter-function



(a) (b)

Figure 1: (a) Probability versus rank of each of the functions (ranked by probability) from a sample of 10^8 parameters for a network with 7 Boolean inputs, two hidden layers of 40 neurons and one Boolean output. The labels are different parameter initialisations (n is the number of layers)
(b) Comparing the empirical frequency of different labelings for a sample of m MNIST images, obtained from randomly sampling parameters from a neural neural network, versus that obtained by sampling from the corresponding Gaussian process. The network has 2 fully connected hidden layers of 784 ReLU neurons each. The weight and bias variances are 1.0. The sample size is 10^7, and only points obtained in both samples are displayed. Note that this figure also implies significant bias.

maps of larger networks (including other architectures, like convolutional ones). But first we address the question of how the bias in the parameter-function map can lead to good generalization, under not-too-strong assumptions on the training algorithm used.

3 PAC-BAYES GENERALIZATION ERROR BOUNDS

The approach we develop to understand generalization starts by considering a given training set U (generated from some unknown target function), and a given stochastic learning algorithm, which in our experiments we will be stochastic gradient descent (SGD). For simplicity of analysis, we consider binary classification. In this set up, assuming realizability, one can ask about the probability of finding different functions (also called concepts, in binary classification) that fit the training data perfectly (empirical risk minimization). Of more interest, one can ask about the probability of different generalization errors, or the expected generalization error.

If the probability of finding a particular concept c , given a training set U, is $\frac{P(c)}{\sum_{c \in U} P(c)}$, where $c \in U$ means all concepts consistent with the training set[2], then Theorem 1 from the classic work by McAllester (1998) gives a bound on the expected generalization error:

Theorem 1. *(PAC-Bayes theorem (McAllester (1998))) For any measure P on any concept space and any measure on a space of instances we have, for $0 < \delta \leq 1$, that with probability at least $1 - \delta$ over the choice of sample of m instances all measurable subsets U of the concepts such that every element of U is consistent with the sample and with $P(U) > 0$ satisfies the following:*

$$\epsilon(U) \leq \frac{\ln \frac{1}{P(U)} + \ln \frac{1}{\delta} + 2 \ln m + 1}{m}$$

where $P(U) = \sum_{c \in U} P(c)$, and where $\epsilon(U) := E_{c \in U} \epsilon(c)$, i.e. the expected value of the generalization errors over concepts c in U with probability given by the posterior $\frac{P(c)}{P(U)}$. Here, $\epsilon(c)$ is the generalization error (probability of the concept c disagreeing with the target concept, when sampling inputs).

[2]This is just the posterior distribution, when the prior is $P(c)$ and likelihood equal to 1 if $c \in U$ and 0 otherwise

In order to apply the PAC-Bayes theorem, $P(c)$ must be independent of the training set U. Typically a neural network is trained by a stochastic algorithm such as SGD, which samples parameters. In order to apply the PAC-Bayes formalism, we thus make the following (informal) assumption:

stochastic gradient descent samples the zero-error region close to uniformly.

Given some distribution over parameters, the distribution over functions $P(c)$ is then determined by the parameter-function map, and if the parameter distribution is close to uniform, then $P(c)$ should be heavily biased as in Figure 1a. Stochastic gradient descent can be seen as a Markov chain with a stationary distribution, which under some assumptions can be shown to approximate the Gibbs distribution Mandt et al. (2017), although in general there is no a-priori reason to think this stationary distribution is uniform on the zero-error parameters. However, as we will see, for our theory to be useful, it suffices for the distribution to not be too far from uniform. Note that as the region of parameter space with zero-error may be unbounded, a uniform distribution is not well defined, so we will instead consider a Gaussian distribution with a sufficiently large variance[3]. One can interpret this as approximating the distribution obtained by early stopping, where SGD has not had time to equilibrate to its stationary distribution. We will discuss further the effect of the choice of variance in Section 5.

One way to understand the bias observed in Fig 1 is that the regions of parameter space producing some functions is exponentially larger than that producing other functions. This is a huge effect which is likely to have a very significant effect on which functions SGD finds. Thus, even if the parameter distributions used here don't capture the exact behavior of SGD, the bias will probably still play a big role.

We also used the neural network from Section 2 trained on a Boolean function with Lempel-Ziv complexity of 38.5 and compared the generalization error with SGD to direct sampling of the parameters (keeping those with zero training error), finding (0.058 ± 0.01) and (0.058 ± 0.03) respectively. This good agreement is indirect evidence that, at least for this simple function, SGD performs similarly to i.i.d. sampling of parameters.

One disadvantage of the PAC-Bayes theorem is that it provides a bound on the expectation of the error, while generalization bounds in learning theory typically hold with high probability. To obtain better bounds holding w.h.p., one could bound the variance of the error, which we leave to explore in future work.

The advantage of the PAC-Bayes approach is that it allows for target function-dependent bounds. Target functions with large $P(c)$ will tend to also have a large $P(U)$, and so, according to PAC-Bayes, will generalise better than functions with small $P(c)$. Typically $\ln(P(U))$ dominates the numerator so that, to first order, a linear decrease in the bound corresponds to an exponential increase in the average $P(c)$. In order to use the PAC-Bayes approach, we need a method to calculate $P(U)$ for large systems, a problem we now turn to.

3.1 Gaussian process approximation to the prior over functions

In recent work (Lee et al. (2017); Matthews et al. (2018); Garriga-Alonso et al. (2018)), it was shown that infinitely-wide neural networks (including convolutional and residual networks) are equivalent to Gaussian processes. More precisely, this means that when the distribution over parameters is a Gaussian with a given variance, the (real-valued) output of the neural network over a given set of inputs is jointly Gaussian, with a covariance matrix K given by the kernel $k(x_1, x_2)$, where x_1 and x_2 are inputs. The kernel for fully connected ReLU networks has well known analytical form known as the arccosine kernel (Cho & Saul (2009)), while for convolutional and residual networks it can be efficiently computed[4]. Apart from the hyperparameters describing the architecture, the Gaussian process has an extra two parameters, the weight variance σ_w^2/n (where n is the size of the input to the layer) and the bias variance σ_b.

The main quantity in the PAC-Bayes theorem, $P(U)$, is precisely the probability of a given set of labels for the set of instances in the training set, also known as *marginal likelihood*, a connection

[3]Note that in high dimensions as a Gaussian distribution is very similar to a uniform distribution over a sphere.

[4]we use the code from Garriga-Alonso et al. (2018)

explored in recent work (Smith & Le (2018); Germain et al. (2016)). For binary classification, these labels are binary, and are related to the real-valued outputs of the network via a likelihood function like a step function or a sigmoid.

Neural networks are not infinitely-wide in practice, although the Gaussian approximation has been previously used to study them as a mean-field approximation (Schoenholz et al. (2016)). To test whether for common neural network architectures, this provides a good approximation for $P(U)$, we sampled functions (labellings for a particular set of inputs) from a fully connected neural network, and the corresponding Gaussian process, and compared the empirical frequencies of each function. We can obtain good estimates of $P(U)$ in this direct way, for very small set of inputs (here we use 10 random MNIST images, see also SI). The results are plotted in Figure 1, showing that the agreement between the neural network probabilities and the Gaussian probabilities is extremely good, even this far from the infinite width limit (and for input set sizes of this size).

For the case of classification, there is no analytic formula for $P(U)$ and as sampling becomes intractable for larger input set sizes, we need to use other approximations. In this paper we use the expectation-propagation (EP) approximation, implemented in GPy (since 2012), which is more accurate than the Laplacian approximation (Rasmussen (2004)). To see how good these approximations are, we compared them with the empirical frequencies obtained by directly sampling the neural network. The results are in Figure 1 in the SI. We find that the both the EP and Laplacian approximations correlate with the the empirical neural network likelihoods. In larger sets of inputs (1000), we also found that the relative difference between the log-likelihoods between the two approximations to be less than about 10%.

4 EXPERIMENTAL RESULTS

We tested the expected generalization error bounds described in the previous section in a variety of networks trained on binarized[5] versions of MNIST (LeCun et al. (1998)), fashion-MNIST (Xiao et al. (2017)), and CIFAR10 (Krizhevsky & Hinton (2009)). In particular, we will show we can obtain nonvacuous bounds[6] which are also better than random guessing, and that our bounds predict some of the behaviour of the generalization error as we change the learning task (for instance, by corrupting the data), and as we vary hyperparameters like the depth or architecture type.

Zhang et al. (2016) found that the generalization error increased continuously as the labels in CIFAR10 where randomized with an increasing probability. In Figure 2, we replicate these results for three datasets, and show that our bounds correctly predict the increase in generalization error. Furthermore, the bounds show that, for low corruption, MNIST and fashion-MNIST are similarly hard (although fashion-MIST is slightly harder), and CIFAR10 is considerably harder. This mirrors what is obtained from the true generalization errors. Also note that the bounds for MNIST and fashion-MNIST with little corruption are significantly below 0.5 (random guessing). For exmperimental details see SI 1.

In Table 1, we list the mean generalisation error and the bounds for the three datasets (at 0 label corruption), demonstrating that the PAC-Bayes bound closely follows the same trends.

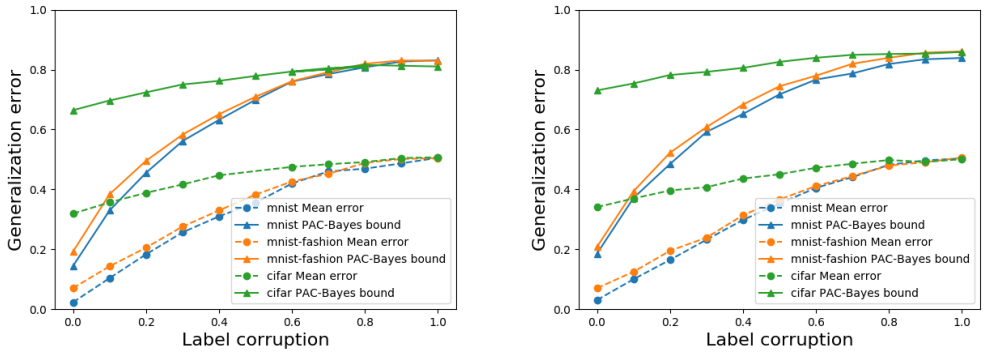
	MNIST		fashion-MNIST		CIFAR	
Network	Mean error	Bound	Mean error	Bound	Mean error	Bound
CNN	0.023	0.144	0.071	0.192	0.320	0.664
FC	0.031	0.190	0.070	0.208	0.341	0.730

Table 1: Mean generalization errors and PAC-Bayes bounds for the convolutional and fully connected network for 0 label corruption, for a sample of 10000 from different datasets.

In Figure 3 we show the generalization errors and PAC-Bayes bounds as we vary the size of the training set. We observe that the bounds show the same relative ordering as the true errors, with

[5]We label an image as 0 if to one of the first five classes and as 1 otherwise

[6]Here we use the term nonvacuous to refer to bounds which are less than 1.0 as in Dziugaite & Roy (2017)



(a) for a 4 hidden layers convolutional network (b) for a 1 hidden layer fully connected network

Figure 2: Mean generalization error and corresponding PAC-Bayes bound versus percentage of label corruption, for three datasets and a training set of size 10000. Note that the bounds follow the same trends as the true generalization errors. The empirical errors are averaged over 8 initializations. The Gaussian process parameters were $\sigma_w = 1.0$, $\sigma_b = 1.0$ for the CNN and $\sigma_w = 10.0$, $\sigma_b = 10.0$ for the FC.

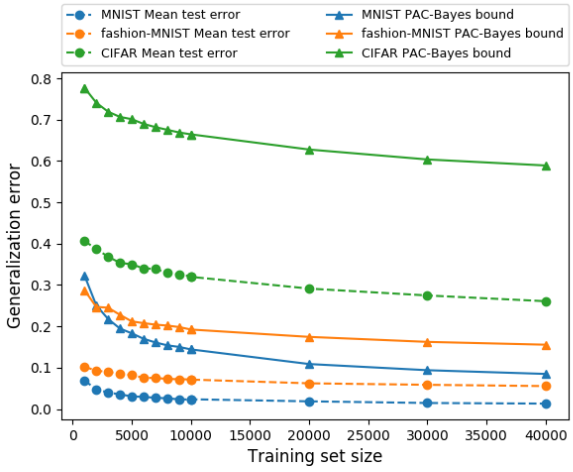


Figure 3: Mean generalization error and corresponding PAC-Bayes bound versus training set size, for three datasets, using a four layer convolutional network (details in SI 1). Note that the bounds follow the same trends as the true generalization errors. The empirical errors are averaged over 8 initializations. The Gaussian process parameters were $\sigma_w = 1.0$, $\sigma_b = 1.0$.

MNIST being lower than fashion-MNIST, which in turn is lower than CIFAR, in terms of generalization error. Furthermore, both the real errors and the bounds decrease with training set size. Note that if $P(U)$ didn't change, then the error would drop with $1/m$ It drops more slowly because increasing m also decreases $P(U)$ because fewer functions are compatible with a larger training set.

5 THE CHOICE OF VARIANCE HYPERPARAMETERS

One limitation of our approach is that it depends on the choice of the variances of the weights and biases used to define the equivalent Gaussian process. Most of the trends shown in the previous section were robust to this choice, but not all. For instance, the bound for MNIST was higher than that for fashion-MNIST for the fully connected network, if the variance was chosen to be 1.0.

In Figures 3 and 2 in SI, we show the effect of the variance hyperparameters on the bound. Note that for the fully connected network, the variance of the weights σ_w seems to have a much bigger role.

This is consistent with what is found in Lee et al. (2017). Furthermore, in Lee et al. (2017) they find, for smaller depths, that the neural network Gaussian process behaves best above $\sigma_w \approx 1.0$, which marks the transition between two phases characterized by the asymptotic behavior of the correlation of activations with depth. This also agrees with the behaviour of the PAC-Bayes bound. For CIFAR10, we find that the bound is best near the phase transition, which is also compatible with results in Lee et al. (2017). For convolutional networks, we found sharper transitions with weight variance, and an larger dependence on bias variance (see Fig. 3 in SI). For our experiments, we chose variances values above the phase transition, and which were fixed for each architecture.

The best choice of variance would correspond to the Gaussian distribution best approximates the behaviour of SGD. We measured the variance of the weights after training with SGD and early stopping (stop when 100% accuracy is reached) from a set of initializations, and obtained values an order of magnitude smaller than those used in the experiments above. Using these variances gave significantly worse bounds, above 50% for all levels of corruption.

This measured variance doesn't necessarily measure the variance of the Gaussian prior that best models SGD, as it also depends on the shape of the zero-error surface (the likelihood function on parameter space). However, it might suggest that SGD is biased towards better solutions in paramater space, giving a stronger/better bias than that predicted only by the parameter-function map with Gaussian sampling of parameters. One way this could happen is if SGD is more likely to find flat (global) "minima"[7] than what is expected from near-uniform sampling of the region of zero-error (probability proportional to volume of minimum). This may be one of the main sources of error in our approach. A better understanding of SGD would be needed to progress in this front.

6 WHY DO NEURAL NETWORKS GENERALIZE IN REAL-WORLD PROBLEMS?

Neural networks are able to generalize for some target functions, and not for others (as demonstrated by experiments like those of Zhang et al. (2016)), so why do they generalize for real-world functions? Proposed answers to this question tend to start by saying that real-world problems are simple or have some structure (Lin et al. (2017); Schmidhuber (1997)), and that neural networks are perhaps biased towards simple functions (under the same notion of complexity).

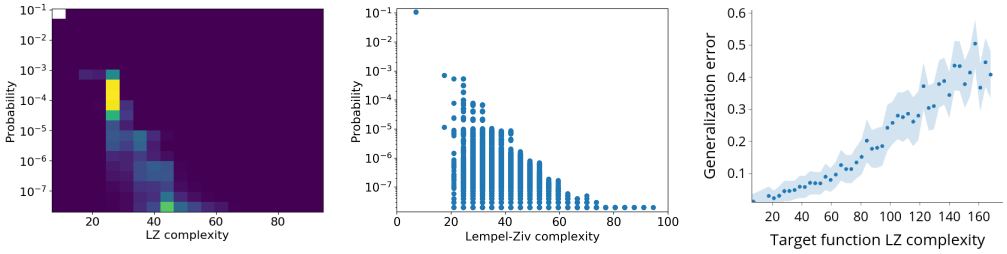
Algorithmic information theory (AIT) studies a notion of complexity (Kolmogorov complexity) that is asymptotically universal, but uncomputable. However, recent work Dingle et al. (2018) has shown that its predictions can still work for finite systems, using computable descriptional complexity measures such as Lempel-Ziv (LZ) complexity[8]. Given a number of criteria which are met by typical parameter-function maps (see SI 4) the prediction is that simple input-output maps which are biased, should be exponentially biased towards simple outputs.

If we apply this prediction to the parameter-function map of neural networks, we would expect that high-probability functions (which will typically have large $P(U)$ and thus result in good generalization) should be of low descriptional complexity. In Figure 4, we show that we indeed find this, with all high-probability functions having low LZ complexity, for a small neural network with 7 Boolean inputs and one Boolean output. In SI 5.3, we also show that similar results hold when using other complexity measures for Boolean functions. The correlation between complexity measures and the prior probability suggests that neural networks would generalize better for simple functions. Such a correlation has been observed before, but in Figure 4c we show it explicitly when comparing against the LZ complexity of the target function for a small neural network and find very clean correlations. The literature on complexity measures is vast. Here we simply note that there is nothing fundamental about LZ. Other approximate complexity measures that capture essential aspects of Kolmogorov complexity also show similar correlations (see SI 5.4).

There are many reasons to believe that real-world functions are simple or have some structure (Lin et al. (2017); Schmidhuber (1997)) and will therefore have low descriptional complexity. Putting this together with the above results means we expect the network to generalize well for real-world datasets and functions. As can also be seen in Figure 4c, it does not generalize well for complex (random) functions. By simple counting arguments, the number of high complexity functions is

[7] Note that the notion of minimum is not well defined given that the region of zero error seems to be mostly flat and connected (Sagun et al. (2017); Draxler et al. (2018))

[8] See SI 5, for definition of the Lempel-Ziv based complexity measure



(a) (b) (c)

Figure 4: (a) Histogram of functions in the probability versus Lempel-Ziv complexity plane, weighted according to their probability. (b) Probability versus Lempel-Ziv complexity. Probabilities are estimated from a sample of 10^8 parameters with a uniform distribution with variance of $1/\sqrt{n}$ with n the input size to the layer, for a network with 7 Boolean inputs, two hidden layers of 40 ReLU neurons each, and a single Boolean output. Points with a frequency of 10^{-8} are removed for clarity because these suffer from finite-size effects (see SI 6). (c) Generalization error versus Lempel-Ziv complexity of different target functions.

exponentially larger than the number of low complexity functions. Nevertheless, such functions may be less common in real-world applications.

Although here we have only shown that high-probability functions have low complexity for a small toy neural network, the generality of the AIT arguments from Dingle et al. (2018), where bias was observed for a wide range of different systems, suggests that an exponential probability-complexity bias may hold for larger neural networks as well.

Nevertheless, although this AIT argument and the above empirical results offer interesting hints, we are still far from a rigorous understanding of why neural networks generalize in real-world problems.

7 Conclusion and Future Work

In this paper, we show that the parameter-function map of deep neural networks is heavily biased, which allows one to make only mild assumptions about the behaviour of the training algorithm used, to explain the generalization. To give further support to this claim, we use Gaussian processes and PAC-Bayes to give quantitative generalization error bounds for common architectures and datasets. As far as we know, this is the first approach to offer nonvacuous (less than 1) generalization errors for realistic neural networks.

We also discuss the assumptions we make, which are possible sources of error for our bounds. These are:

1. The probability that the training algorithm (like SGD) finds a particular function in the zero-error region can be approximated by the probability that the function obtains upon i.i.d. sampling of parameters.
2. Gaussian processes model neural networks with i.i.d.-sampled parameters well even for finite widths.
3. Expectation-propagation gives a good approximation of the Gaussian process marginal likelihood.
4. PAC-Bayes offers tight bounds given the correct marginal likelihood $P(U)$.

We have shown evidence that number 2 is a very good approximation, and that number 3 is reasonably good. In addition, the fact that our bounds are able to correctly predict the behavior of the true error, offers evidence for the set of approximations as a whole, although further work in testing their validity is needed, specially that of number 1. Nevertheless, we think that the good agreement of our bounds constitutes good evidence for the approach we describe in the paper as well as for the claim

that bias in the parameter-function map is the main reason for generalization. We think that further work in understanding these assumptions can sharpen the results obtained here significantly.

Finally, we also tackled the question of why neural networks generalize in practice. We showed that the parameter-function map is biased towards functions with low descriptional complexity (using a variety of common complexity measures), and connect this with recent findings in applied algorithmic information theory. Because real-world problems tend to be far from random, using these same measures, this offers some insight into why neural networks generalize for real-world datasets and problems.

ACKNOWLEDGMENTS

REFERENCES

Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

Devansh Arpit, Stanisław Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.

Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pp. 342–350, 2009.

Kamaludin Dingle, Chico Q Camargo, and Ard A Louis. Input–output maps are strongly biased towards simple outputs. *Nature communications*, 9(1):761, 2018.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*, 2018.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Gintare Karolina Dziugaite and Daniel M Roy. Data-dependent pac-bayes priors via differential privacy. *arXiv preprint arXiv:1802.09583*, 2018.

Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow Gaussian processes. *arXiv preprint arXiv:1808.05587*, aug 2018. URL `https://arxiv.org/abs/1808.05587`.

Pascal Germain, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien. Pac-bayesian theory meets bayesian inference. In *Advances in Neural Information Processing Systems*, pp. 1884–1892, 2016.

GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

Qianli Liao and Tomaso Poggio. Theory of deep learning ii: Landscape of the empirical risk in deep learning. *arXiv preprint arXiv:1703.09833*, 2017.

Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168(6):1223–1247, 2017.

Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *arXiv preprint arXiv:1704.04289*, 2017.

Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

David A McAllester. Some pac-bayesian theorems. In *Proceedings of the eleventh annual conference on Computational learning theory*, pp. 230–234. ACM, 1998.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

Nelson Morgan and Hervé Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In *Advances in neural information processing systems*, pp. 630–637, 1990.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017a.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5949–5958, 2017b.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

T Poggio, K Kawaguchi, Q Liao, B Miranda, L Rosasco, X Boix, J Hidary, and HN Mhaskar. Theory of deep learning iii: the non-overfitting puzzle. Technical report, CBMM memo 073, 2018.

Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pp. 63–71. Springer, 2004.

Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Jürgen Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873, 1997.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. 2018.

Daniel Soudry, Elad Hoffer, and Nathan Srebro. The implicit bias of gradient descent on separable data. *arXiv preprint arXiv:1710.10345*, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

Andrey Nikolayevich Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pp. 195–198, 1943.

Lei Wu, Zhanxing Zhu, et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Chiyuan Zhang, Qianli Liao, Alexander Rakhlin, Brando Miranda, Noah Golowich, and Tomaso Poggio. Musings on deep learning: Properties of sgd. 2017.

SUPPLEMENTARY INFORMATION: DEEP LEARNING GENERALIZES BECAUSE THE PARAMETER-FUNCTION MAP IS BIASED TOWARDS SIMPLE FUNCTIONS

Anonymous authors
Paper under double-blind review

1 BASIC EXPERIMENTAL DETAILS

In the main experiments of the paper we used three classes of architectures. Here we describe them in more detail.

- Fully connected networks (FCs), with varying number of layers. The size of the hidden layers was the same as the input dimension, and the nonlinearity was ReLU. The last layer was a single Softmax neuron. We used default Keras settings for initialization (Glorot uniform)
- Convolutional neural networks (CNNs), with varying number of layers. The number of filters was 200, and the nonlinearity was ReLU. The last layer was a fully connected single Softmax neuron. The filter sizes alternated between $(2, 2)$ and $(5, 5)$, and the padding between SAME and VALID, the strides were 1 (same default settings as in the code for Garriga-Alonso et al. (2018)). We used default Keras settings for initialization (Glorot uniform)

In all experiments we trained with SGD with a learning rate of 0.01, and early stopping when the accuracy on the whole training set reaches 100%.

2 TESTING THE APPROXIMATIONS TO THE GAUSSIAN PROCESS MARGINAL LIKELIHOOD

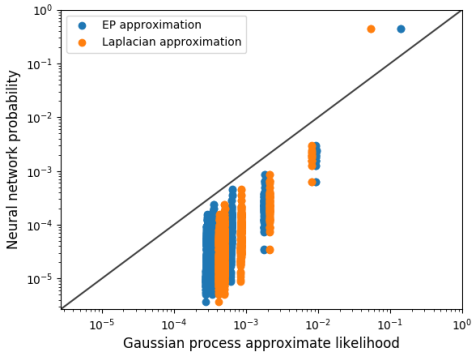
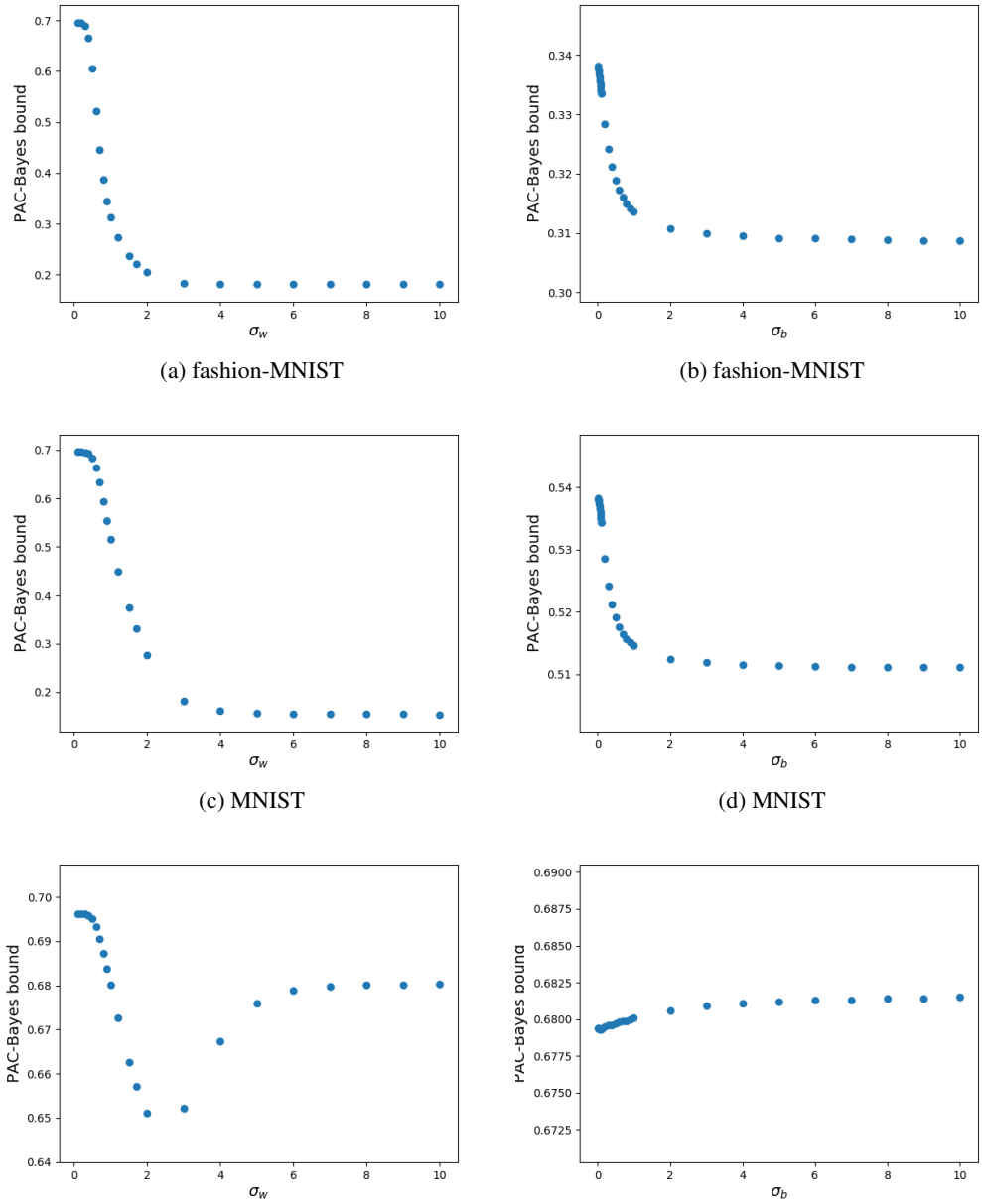


Figure 1: Comparing the empirical frequency of different labellings for a sample of 10 MNIST images obtained from randomly sampling parameters from a neural neural network, versus the approximate marginal likelihood from the corresponding Gaussian process. Blue dots correspond to the expectation-propagation approximation, and orange dots to the Laplace approximation. The network has 2 fully connected hidden layers of 784 ReLU neurons each. The weight and bias variances are 1.0.

3 DEPENDENCE OF PAC-BAYES BOUND ON VARIANCE HYPERPARAMETERS



(a) fashion-MNIST (b) fashion-MNIST

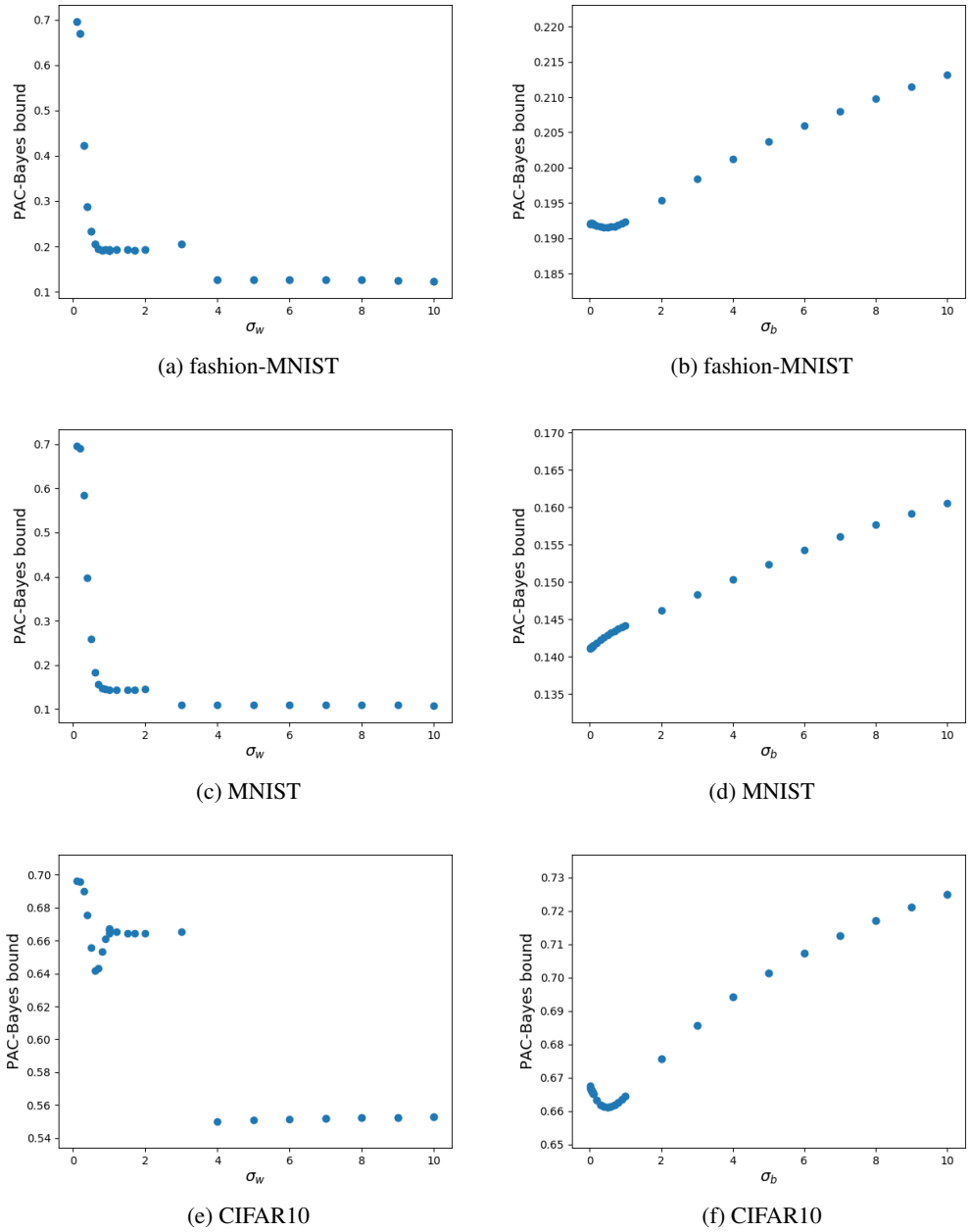
(c) MNIST (d) MNIST

(e) CIFAR10 (f) CIFAR10

Figure 2: PAC-Bayes bound versus the standard deviation parameter for the weights and biases, for a sample of 10000 from different datasets, and a two-layer fully connected network (with the layers of the same size as input). The fixed parameter is put to 1.0 in all cases.

4 SIMPLICITY BIAS AND THE PARAMETER-FUNCTION MAP

An important argument in Section 6 in the main text is that the parameter-function map of neural networks should exhibit the basic simplicity bias phenomenolgy recently described in Dingle et al. in Dingle et al. (2018). In this section we briely describe some key results of reference Dingle et al. (2018) relevant to this argument.



(a) fashion-MNIST

(b) fashion-MNIST

(c) MNIST

(d) MNIST

(e) CIFAR10

(f) CIFAR10

Figure 3: PAC-Bayes bound versus the standard deviation parameter for the weights and biases, for a sample of 10000 from different datasets, and a four-layer convolutional network. The fixed parameter is put to 1.0 in all cases.

A computable[1] input-output map $f : I \to O$, mapping N_I inputs from the set I to N_O outputs x from the set O[2] may exhibit simplicity bias if the following restrictions are satisfied Dingle et al. (2018):

1) *Map simplicity* The map should have limited complexity, that is its Kolmogorov complexity $K(f)$ should asymptotically satisfy $K(f)+K(n) \ll K(x)+O(1)$, for typical $x \in O$ where n is a measure of the size of the input set (e.g. for binary input sequences, $N_I = 2^n$.).

2) *Redundancy*: There should be many more inputs than outputs ($N_I \gg N_O$) so that the probability $P(x)$ that the map generates output x upon random selection of inputs $\in I$ can in principle vary significantly.

3) *Finite size* $N_O \gg 1$ to avoid potential finite size effects.

4) *Nonlinearity*: The map f must be be a nonlinear function since linear functions don't exhibit bias.

5) *Well behaved*: The map should not primarily produce pseudorandom outputs (such as the digits of π), because complexity approximators needed for practical applications will mistakenly label these as highly complex.

For the deep learning learning systems studied in this paper, the inputs of the map f are the parameters that fix the weights for the particular neural network architecture chosen, and the outputs are the functions that the system produces. Consider, for example, the configuration for Boolean functions studied in the main text. While the output functions rapidly grow in complexity with increasing size of the input layer, the map itself can be described with a low-complexity procedure, since it consists of reading the list of parameters, populating a given neural network architecture and evaluating it for all inputs. For reasonable architectures, the information needed to describe the map grows logarithmically with the input dimension n, so for large enough n, the amount of information required to describe the map will be much less than the information needed to describe a typical function, which requires 2^n bits. Thus the Kolmogorov complexity $K(f)$ of this map is asymptotically smaller than the the typical complexity of the output, as required by the map simplicity condition 1) above.

The redundancy condition 2) depends on the network architecture and discretization. For overparameterised networks, this condition is typically satisfied. In our specific case, where we use floating point numbers for the parameters (input set I), and Boolean functions (output set O), this condition is clearly satisfied. Neural nets can represent very large numbers of potential functions (see for example estimates of VC dimension Bartlett et al. (2017b); Baum & Haussler (1989)), so that condition 3) is also generally satisfied. Neural network parameter-function maps are evidently non-linear, satisfying condition 4). Condition 5) is perhaps the least understood condition within simplicity bias. However, the lack of any function with high probability and high complexity (at least when using LZ complexity), provides some empirical validation. This condition also agrees with the expectation that neural networks won't predict the outputs of a good pseudorandom number generator. One of the implicit assumptions in the simplicity bias framework is that, although true Kolmogorov complexity is always uncomputable, approximations based on well chosen complexity measures perform well for most relevant outputs x. Nevertheless, where and when this assumptions holds is a deep problem for which further research is needed.

5 Other complexity measures

One of the key steps to practical application of the simplicity bias framework of Dingle et al. in Dingle et al. (2018) is the identification of a suitable complexity measure $\tilde{K}(x)$ which mimics aspects of the (uncomputable) Kolmogorov complexity $K(x)$ for the problem being studied. It was shown for the maps in Dingle et al. (2018) that several different complexity measures all generated the same qualitative simplicity bias behaviour:

$$P(x) \leq 2^{-(a\tilde{K}(x)+b)} \tag{1}$$

[1]Here computable simply means that all inputs lead to outputs, in other words there is no halting problem.

[2]This language of finite input and outputs sets assumes discrete inputs and outputs, either because they are intrinsically discrete, or because they can be made discrete by a coarse-graining procedure. For the parameter-function maps studied in this paper the set of outputs (the full hypothesis class) is typically naturally discrete, but the inputs are continuous. However, the input parameters can always be discretised without any loss of generality.

but with different values of a and b depending on the complexity measure and of course depending on the map, but independent of output x. Showing that the same qualitative results obtain for different complexity measures is sign of robustness for simplicity bias.

Below we list a number of different descriptional complexity measures which we used, to extend the experiments in Section 6 in the main text.

5.1 COMPLEXTY MEASURES

Lempel-Ziv complexity (*LZ complexity* for short). The Boolean functions studied in the main text can be written as binary strings, which makes it possible to use measures of complexity based on finding regularities in binary strings. One of the best is Lempel-Ziv complexity, based on the Lempel-Ziv compression algorithm. It has many nice properties, like asymptotic optimality, and being asymptotically equal to the Kolmogorov complexity for an ergodic source. We use the variation of Lempel-Ziv complexity from Dingle et al. (2018) which is based on the 1976 Lempel Ziv algorithm Lempel & Ziv (1976):

$$K_{LZ}(x) = \begin{cases} \log_2(n), & x = 0^n \text{ or } 1^n \\ \log_2(n)[N_w(x_1...x_n) + N_w(x_n...x_1)]/2, & \text{otherwise} \end{cases} \tag{2}$$

where n is the length of the binary string, and $N_w(x_1...x_n)$ is the number of words in the Lempel-Ziv "dictionary" when it compresses output x. The symmetrization makes the measure more fine-grained, and the value for the simplest strings ensures that they scale as expeted for Kolmogorov complexity. This complexity measure is the primary one used in the main text.

We note that the binary string representation depends on the order in which inputs are listed to construct it, which is not a feature of the function itself. This may affect the LZ complexity, although for simple input orderings, it will typically have a negligible effect.

Entropy. A fundamental, though weak, measure of complexity is the entropy. For a given binary string this is defined as $S = -\frac{n_0}{N}\log_2\frac{n_0}{N} - \frac{n_1}{N}\log_2\frac{n_1}{N}$, where n_0 is the number of zeros in the string, and n_1 is the number of ones, and $N = n_0 + n_1$. This measure is close to 1 when the number of ones and zeros is similar, and is close to 0 when the string is mostly ones, or mostly zeros. Entropy and $K_{LZ}(x)$ are compared in fig. 4, and in more detail in supplementary note 7 (and supplementary information figure 1) of reference Dingle et al. (2018). They correlate, in the sense that low entropy $S(x)$ means low $K_{LZ}(x)$, but it is also possible to have Large entropy but low $K_{LZ}(x)$, for example for a string such as $10101010....$

Boolean expression complexity. Boolean functions can be compressed by finding simpler ways to represent them. We used the standard SciPy implementation of the Quine-McCluskey algorithm to minimize the Boolean function into a small sum of products form, and then defined the number of operations in the resulting Boolean expression as a *Boolean complexity* measure.

Generalization complexity. L. Franco et al. have introduced a complexity measure for Boolean functions, designed to capture how difficult the function is to learn and generalize Franco & Anthony (2004), which was used to empirically find that simple functions generalize better in a neural network Franco (2006). The measure consists of a sum of terms, each measuring the average over all inputs fraction of neighbours which change the output. The first term considers neighbours at Hamming distance of 1, the second at Hamming distance of 2 and so on. The first term is also known (up to a normalization constant) as average sensitivity Friedgut (1998). The terms in the series have also been called “generalized robustness” in the evolutionary theory literature Greenbury et al. (2016). Here we use the first two terms, so the measure is:

$$C(f) = C_1(f) + C_2(f),$$
$$C_1(f) = \frac{1}{2^n n}\sum_{x\in X}\sum_{y\in \text{Nei}_1(x)} |f(x) - f(y)|,$$
$$C_1(f) = \frac{2}{2^n n(n-1)}\sum_{x\in X}\sum_{y\in \text{Nei}_2(x)} |f(x) - f(y)|,$$

where $\text{Nei}_i(x)$ is all neighbours of x at Hamming distance i.

Critical sample ratio. A measure of the complexity of a function was introduced in Arpit et al. (2017) to explore the dependence of generalization with complexity. In general, it is defined with respect to a sample of inputs as the fraction of those samples which are *critical samples*, defined to be an input such that there is another input within a ball of radius r, producing a different output (for discrete outputs). Here, we define it as the fraction of all inputs, that have another input at Hamming distance 1, producing a different output.

5.2 Correlation between complexities

In Fig. 4, we compare the different complexity measures against one another. We also plot the frequency of each complexity; generally more functions are found with higher complexity.

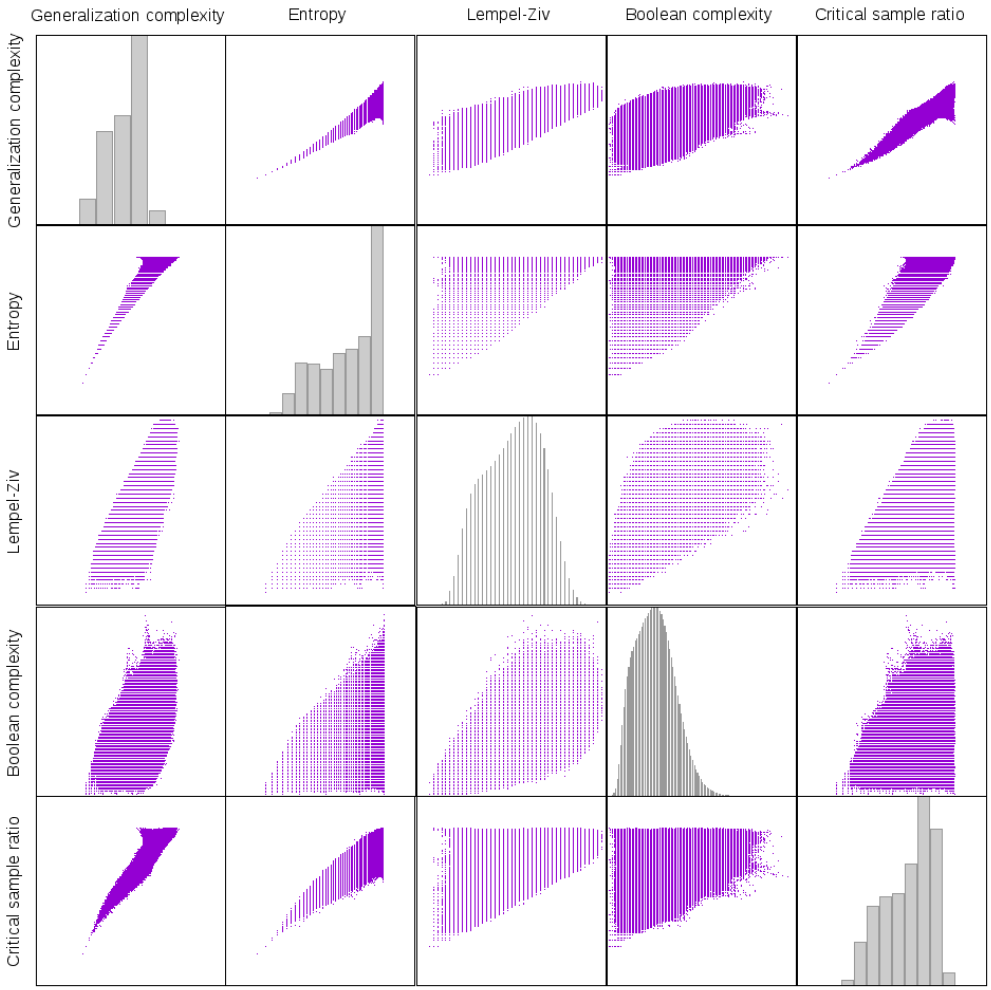
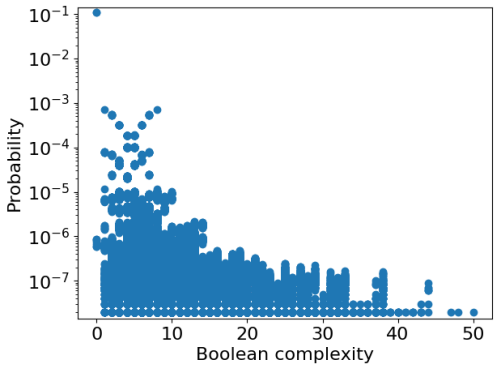


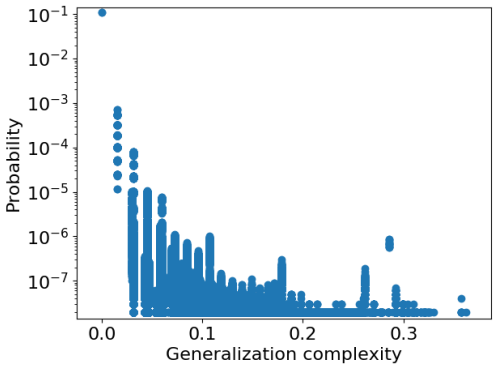
Figure 4: Scatter matrix showing the correlation between the different complexity measures used in this paper On the diagonal, a histogram (in grey) of frequency versus complexity is depicted. The functions are from the sample of 10^8 parameters for the $(7, 40, 40, 1)$ network.

5.3 PROBABILITY-COMPLEXITY PLOTS

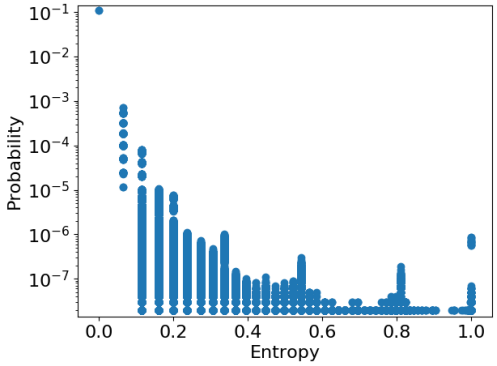
In Fig. 5 we show how the probability versus complexity plots look for other complexity measures. The behaviour is similar to that seen for the LZ complexity measure in Fig 1(b) of the main text. In Fig. 6 we show probability versus LZ complexity plots for other choices of parameter distributions.



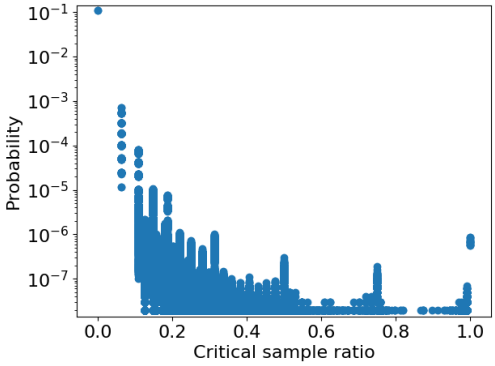
(a) Probability versus Boolean complexity



(b) Probability versus generalization complexity

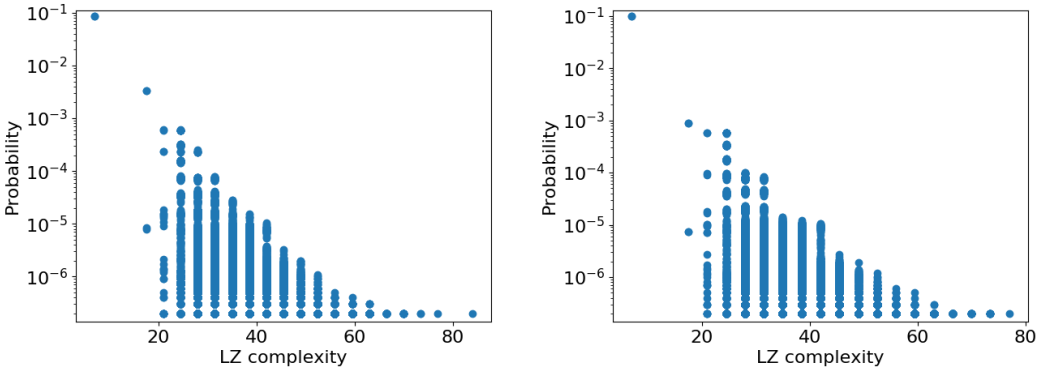


(c) Probability versus entropy



(d) Probability versus critical sample ratio

Figure 5: Probability versus different measures of complexity (see main text for Lempel-Ziv), estimated from a sample of 10^8 parameters, for a network of shape $(7, 40, 40, 1)$. Points with a frequency of 10^{-8} are removed for clarity because these suffer from finite-size effects (see SI 6). The measures of complexity are described in SI 5.



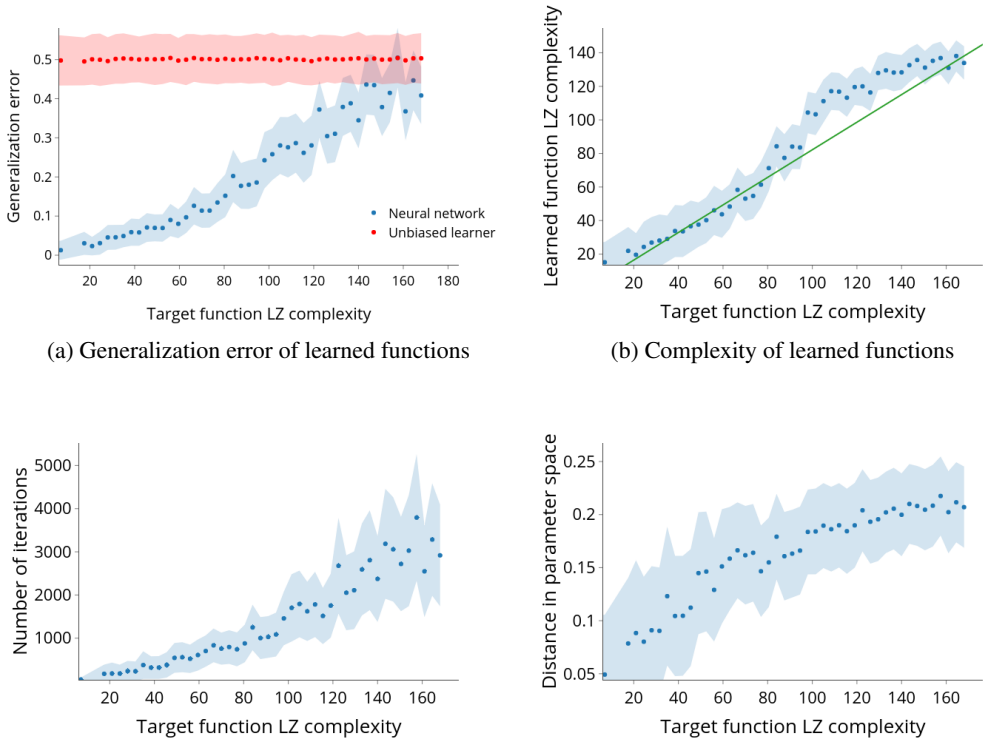
(a) (b)

Figure 6: Probability versus LZ complexity for network of shape $(7, 40, 40, 1)$ and varying sampling distributions. Samples are of size 10^7. (a) Weights are sampled from a Gaussian with variance $1/\sqrt{n}$ where n is the input dimension of each layer. (b) Weights are sampled from a Gaussian with variance 2.5

5.4 EFFECTS OF TARGET FUNCTION COMPLEXITY ON LEARNING FOR DIFFERENT COMPLEXITY MEASURES

Here we show the effect of the complexity of the target function on learning, as well as other complementary results. Here we compare neural network learning to random guessing, which we call "unbiased learner". Note that both probably have the same hypothesis class as we tested that the neural network used here can fit random functions.

The functions in these experiments were chosen by randomly sampling parameters, and so even the highest complexity ones are probably not fully random[3]. In fact, when training the network on truly random functions, we obtain generalization errors equal or above those of the unbiased learner. This is expected from the No Free Lunch theorem, which says that no algorithm can generalize better (for off-training error) uniformly over all functions than any other algorithm (Wolpert & Waters (1994)).



(a) Generalization error of learned functions

(b) Complexity of learned functions

(c) Number of iterations to perfectly fit training set

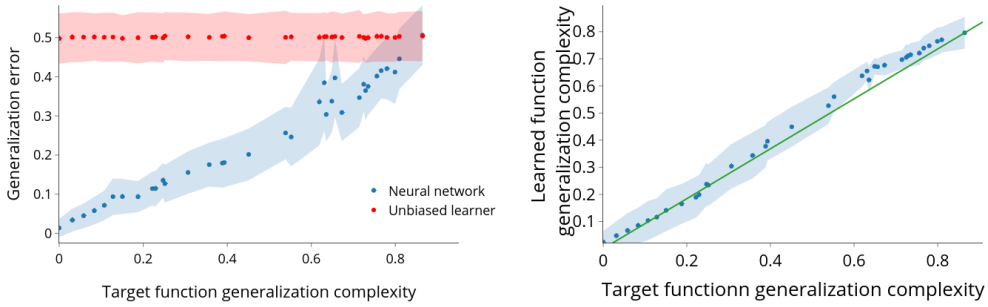
(d) Net Euclidean distance traveled in parameter space to fit training set

Figure 7: Different learning metrics versus the LZ complexity of the target function, when learning with a network of shape $(7, 40, 40, 1)$. Dots represent the means, while the shaded envelope corresponds to piecewise linear interpolation of the standard deviation, over 500 random initializations and training sets.

5.5 LEMPEL-ZIV VERSUS ENTROPY

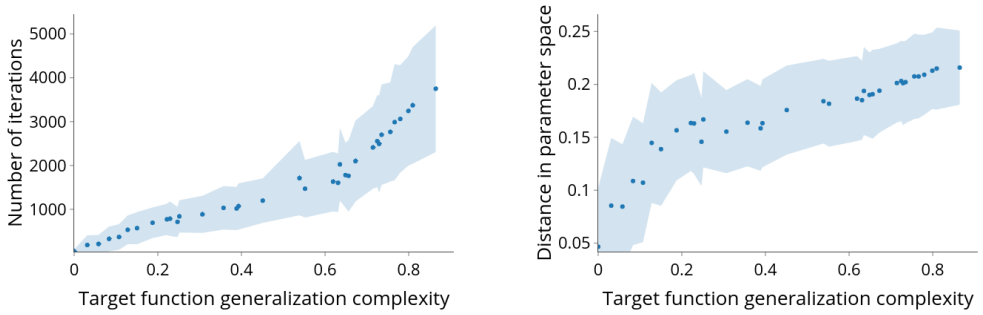
To check that the correlation between LZ complexity and generalization isn't only because of a correlation with function entropy (which is just a measure of the fraction of inputs mapping to 1 or 0, see Section 5), we observed that for some target functions with maximum entropy (but

[3]The fact that non-random strings can have maximum LZ complexity is a consequence of LZ complexity being a less powerful complexity measure than Kolmogorov complexity, see e.g. Estevez-Rams et al. (2013). The fact that neural networks do well for non-random functions, even if they have maximum LZ, suggests that their simplicity bias captures a notion of complexity stronger than LZ.



(a) Generalization error of learned functions

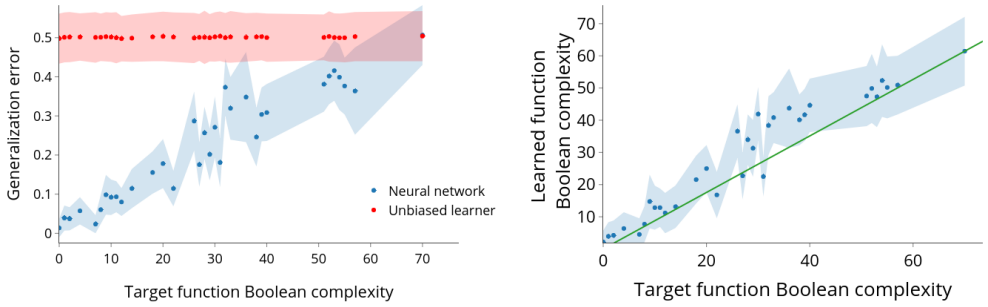
(b) Complexity of learned functions



(c) Number of iterations to perfectly fit training set

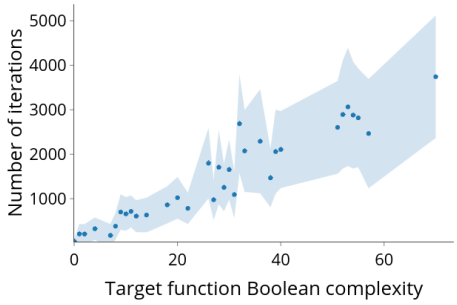
(d) Net Euclidean distance traveled in parameter space to fit training set

Figure 8: Different learning metrics versus the generalization complexity of the target function, when learning with a network of shape $(7, 40, 40, 1)$. Dots represent the means, while the shaded envelope corresponds to piecewise linear interpolation of the standard deviation, over 500 random initializations and training sets.

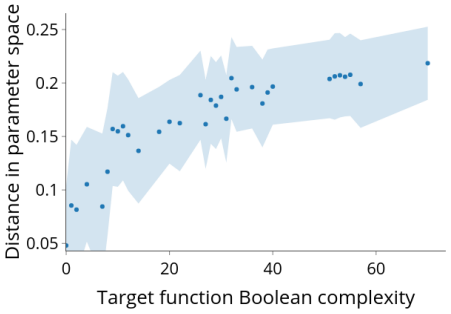


(a) Generalization error of learned functions

(b) Complexity of learned functions

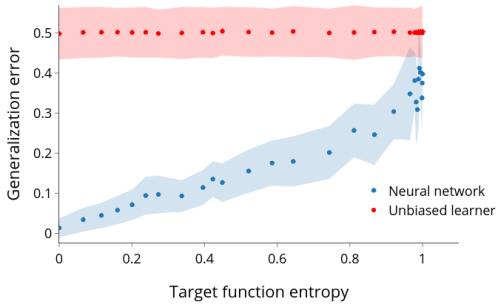


(c) Number of iterations to perfectly fit training set

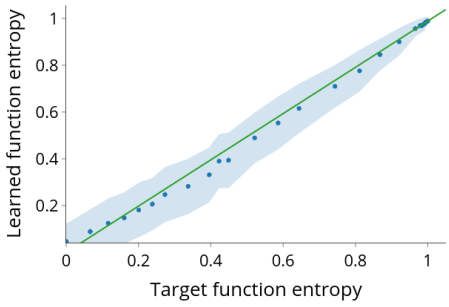


(d) Net Euclidean distance traveled in parameter space to fit training set

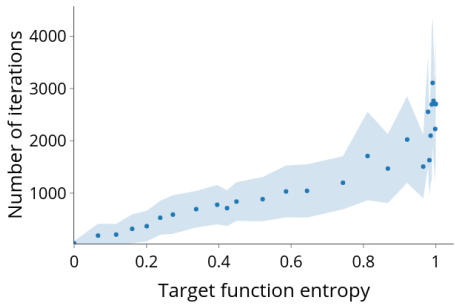
Figure 9: Different learning metrics versus the Boolean complexity of the target function, when learning with a network of shape $(7, 40, 40, 1)$. Dots represent the means, while the shaded envelope corresponds to piecewise linear interpolation of the standard deviation, over 500 random initializations and training sets.



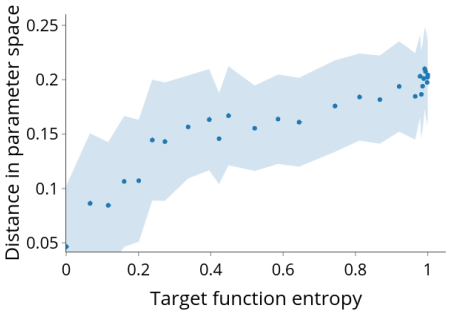
(a) Generalization error of learned functions



(b) Complexity of learned functions



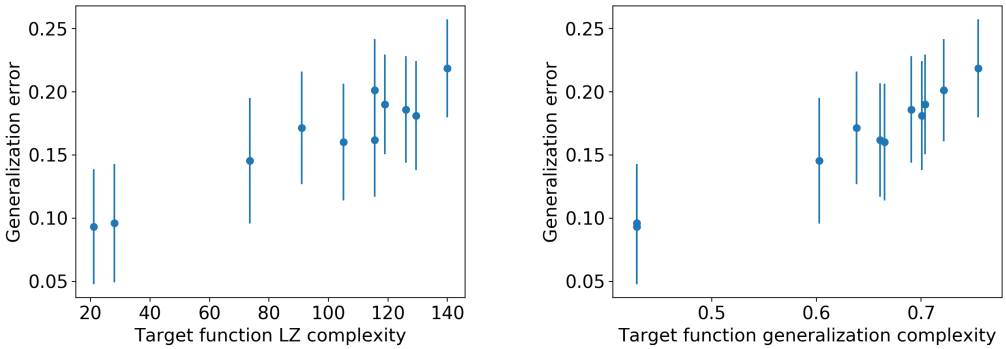
(c) Number of iterations to perfectly fit training set



(d) Net Euclidean distance traveled in parameter space to fit training set

Figure 10: Different learning metrics versus the entropy of the target function, when learning with a network of shape $(7, 40, 40, 1)$. Dots represent the means, while the shaded envelope corresponds to piecewise linear interpolation of the standard deviation, over 500 random initializations and training sets.

which are simple when measured using LZ complexity), the network still generalizes better than the unbiased learner, showing that the bias towards simpler functions is better captured by more powerful complexity measures than entropy[4]. This is confirmed by the results in Fig. 11 where we fix the target function entropy (to 1.0), and observe that the generalization error still exhibits considerable variation, as well as a positive correlation with complexity



(a) (b)

Figure 11: Generalization error of learned function versus the complexity of the target function for target functions with fixed entropy 1.0, for a network of shape $(7, 20, 20, 1)$. Complexity measures are (a) LZ and (b) generalisation complexity. Here the training set size was of size 64, but sampled with replacement, and the generalization error is over the whole input space. Note that despite the fixed entropy there is still variation in generalization error, which correlates with the complexity of the function. These figures demonstrate that entropy is a less accurate complexity measure than LZ or generalisation complexity, for predicting generalization performance.

6 Finite-size effects for sampling probability

Since for a sample of size N the minimum estimated probability is $1/N$, many of the low-probability samples that arise just once may in fact have a much lower probability than suggested. See Figure 12), for an illustration of how this finite-size sampling effect manifests with changing sample size N. For this reason, these points are typically removed from plots.

7 Effect of number of layers on simplicity bias

In Figure 13 we show the effect of the number of layers on the bias (for feedforward neural networks with 40 neurons per layer). We can see that between the 0 layer perceptron and the 2 layer network there is an increased number of higher complexity functions. This is most likely because of the increasing expressivity of the network. For 2 layers and above, the expressivity doesn't significantly change, and instead, we observe a shift of the distribution towards lower complexity.

[4]LZ is a better approximation to Kolmogorov complexity than entropy Cover & Thomas (2012), but of course LZ can still fail, for example when measuring the complexity of the digits of π.

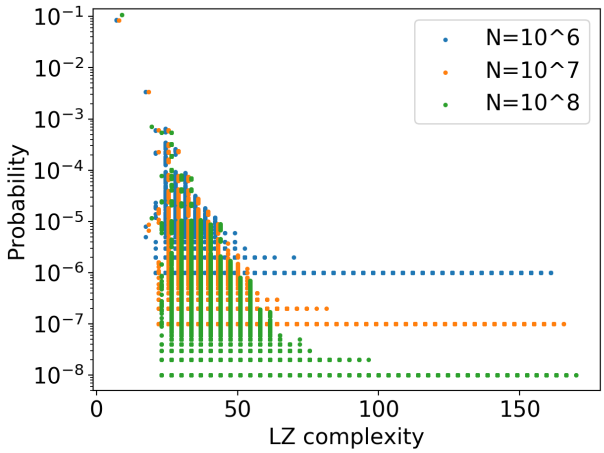
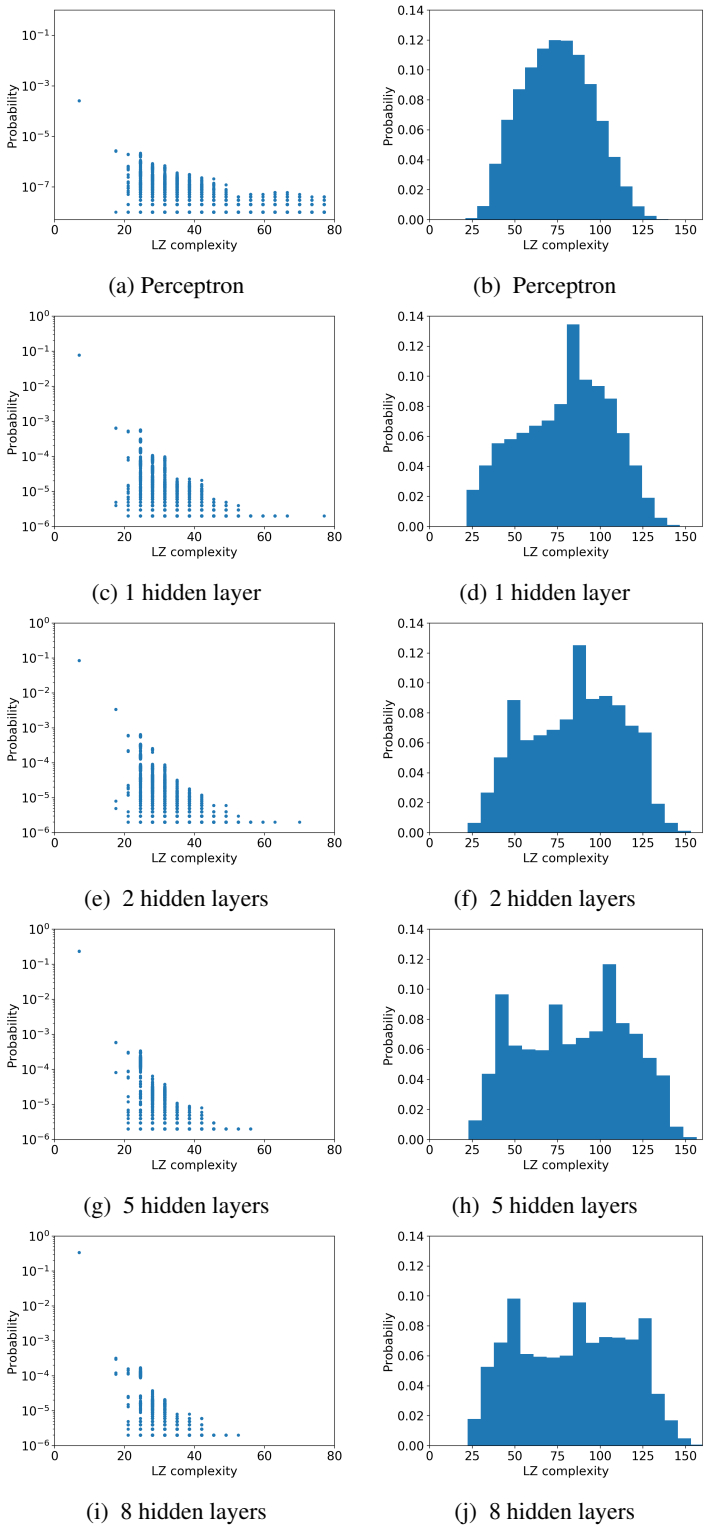


Figure 12: Probability (calculated from frequency) versus Lempel-Ziv complexity for a neural network of shape $(7, 40, 40, 1)$, and sample sizes $N = 10^6, 10^7, 10^8$. The lowest frequency functions for a given sample size can be seen to suffer from finite-size effects, causing them to have a higher frequency than their true probability.



(a) Perceptron (b) Perceptron

(c) 1 hidden layer (d) 1 hidden layer

(e) 2 hidden layers (f) 2 hidden layers

(g) 5 hidden layers (h) 5 hidden layers

(i) 8 hidden layers (j) 8 hidden layers

Figure 13: Probability versus LZ complexity for networks with different number of layers. Samples are of size 10^6, except for the 1 hidden layer case, where it is 50000. (a) & (b) A perceptron with 7 input neurons (complexity is capped at 80 to aid comparison with the other figures). (c) & (d) A network with 1 hidden layer of 40 neurons (e) & (f) A network with 2 hidden layer of 40 neurons (g) & (h) A network with 5 hidden layers of 40 neurons each. (i) & (j) A network with 8 hidden layers of 40 neurons each

8 OTHER RELATED WORK

The topic of generalization in neural networks has been extensively studied both in theory and experiment, and the literature is vast. Theoretical approaches to generalization include classical notions like VC dimension Baum & Haussler (1989); Bartlett et al. (2017b) and Rademacher complexity Sun et al. (2016), but also more modern concepts such as robustness Xu & Mannor (2012), compression Arora et al. (2018) as well as studies on the relation between generalization and properties of stochastic gradient descent (SGD) algorithms Zhang et al. (2017); Soudry et al. (2017); Advani & Saxe (2017).

Empirical studies have also pushed the boundaries proposed by theory, In particular, in recent work by Zhang et al. Zhang et al. (2016), it is shown that while deep neural networks are expressive enough to fit randomly labeled data, they can still generalize for data with structure. The generalization error correlates with the amount of randomization in the labels. A similar result was found much earlier in experiments with smaller neural networks Franco (2006), where the authors defined a complexity measure for Boolean functions, called generalization complexity (see SI 5), which appears to correlate well with the generalization error.

Inspired by the results of Zhang et al. Zhang et al. (2016), Arpit et al. Arpit et al. (2017) propose that the data dependence of generalization for neural networks can be explained because they tend to prioritize learning simple patterns first. The authors show some experimental evidence supporting this hypothesis, and suggest that SGD might be the origin of this implicit regularization. This argument is inspired by the fact that SGD converges to minimum norm solutions for linear models Yao et al. (2007), but only suggestive empirical results are available for the case of nonlinear models, so that the question remains open Soudry et al. (2017). Wu et al. Wu et al. (2017) argue that full-batch gradient descent also generalizes well, suggesting that SGD is not the main cause behind generalization. It may be that SGD provides some form of implicit regularisation, but here we argue that the exponential bias towards simplicity is so strong that it is likely the main origin of the implicit regularization in the parameter-function map.

The idea of having a bias towards simple patterns has a long history, going back to the philosophical principle of Occam's razor, but having been formalized much more recently in several ways in learning theory. For instance, the concepts of minimum description length (MDL) Rissanen (1978), Blumer algorithms Blumer et al. (1987); Wolpert & Waters (1994), and universal induction Ming & Vitányi (2014) all rely on a bias towards simple hypotheses. Interestingly, these approaches go hand in hand with non-uniform learnability, which is an area of learning theory which tries to predict data-dependent generalization. For example, MDL tends to be analyzed using structural risk minimization or the related PAC-Bayes approach Vapnik (2013); Shalev-Shwartz & Ben-David (2014).

Hutter et al. Lattimore & Hutter (2013) have shown that the generalization error grows with the target function complexity for a perfect Occam algorithm[5] which uses Kolmogorov complexity to choose between hypotheses. Schmidhuber applied variants of universal induction to learn neural networks Schmidhuber (1997). The simplicity bias from Dingle et al. Dingle et al. (2018) arises from a simpler version of the coding theorem of Solomonoff and Levin Ming & Vitányi (2014). More theoretical work is needed to make these connections rigorous, but it may be that neural networks intrinsically approximate universal induction because the parameter-function map results in a prior which approximates the universal distribution.

Other approaches that have been explored for neural networks try to bound generalization by bounding capacity measures like different types of norms of the weights (Neyshabur et al. (2015); Keskar et al. (2016); Neyshabur et al. (2017b;a); Bartlett et al. (2017a); Golowich et al. (2017); Arora et al. (2018)), or unit capacity Neyshabur et al. (2018). These capture the behaviour of the real test error (like its improvement with overparametrization (Neyshabur et al. (2018)), or with training epoch (Arora et al. (2018))). However, these approaches haven't been able to obtain nonvacuous bounds yet.

Another popular approach to explaining generalisation is based around the idea of flat minima Keskar et al. (2016); Wu et al. (2017). In Hochreiter & Schmidhuber (1997), Hochreiter and Schmidhuber

[5]Here what we call a 'perfect Occam algorithm' is an algorithm which returns the simplest hypothesis which is consistent with the training data, as measured using some complexity measure, such as Kolmogorov complexity.

argue that flatness could be linked to generalization via the MDL principle. Several experiments also suggest that flatness correlates with generalization. However, it has also been pointed out that flatness is not enough to understand generalization, as sharp minima can also generalize Dinh et al. (2017). We show in Section 2 in the main text that simple functions have much larger regions of parameter space producing them, so that they likely give rise to flat minima, even though the same function might also be produced by other sharp regions of parameter space.

Other papers discussing properties of the parameter-function map in neural networks include Montufar et al. Montufar et al. (2014), who suggested that looking at the size of parameter space producing functions of certain complexity (measured by the number of linear regions) would be interesting, but left it for future work. In Poole et al. (2016), Poole et al. briefly look at the sensitivity to small perturbations of the parameter-function map. In spite of these previous works, there is clearly still much scope to study the properties of the parameter-function map for neural networks.

Finally, our work follows the growing line of work exploring random neural networks Schoenholz et al. (2016); Giryes et al. (2016); Poole et al. (2016); Schoenholz et al. (2017), as a way to understand fundamental properties of neural networks, robust to other choices like initialization, objective function, and training algorithm.

REFERENCES

Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *arXiv preprint arXiv:1706.05394*, 2017.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017a.

Peter L Bartlett, Nick Harvey, Chris Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. arxiv preprint. *arXiv*, 1703, 2017b.

Eric B Baum and David Haussler. What size net gives valid generalization? In *Advances in neural information processing systems*, pp. 81–90, 1989.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Occam's razor. *Information processing letters*, 24(6):377–380, 1987.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Kamaludin Dingle, Chico Q Camargo, and Ard A Louis. Input–output maps are strongly biased towards simple outputs. *Nature communications*, 9(1):761, 2018.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

E Estevez-Rams, R Lora Serrano, B Aragón Fernández, and I Brito Reyes. On the non-randomness of maximum lempel ziv complexity sequences of finite size. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 23 (2):023118, 2013.

Leonardo Franco. Generalization ability of boolean functions implemented in feedforward neural networks. *Neurocomputing*, 70(1):351–361, 2006.

Leonardo Franco and Martin Anthony. On a generalization complexity measure for boolean functions. In *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, volume 2, pp. 973–978. IEEE, 2004.

Ehud Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18 (1):27–35, 1998.

Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow Gaussian processes. *arXiv preprint arXiv:1808.05587*, aug 2018. URL `https://arxiv.org/abs/1808.05587`.

Raja Giryes, Guillermo Sapiro, and Alexander M Bronstein. Deep neural networks with random gaussian weights: a universal classification strategy? *IEEE Trans. Signal Processing*, 64(13):3444–3457, 2016.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. *arXiv preprint arXiv:1712.06541*, 2017.

Sam F Greenbury, Steffen Schaper, Sebastian E Ahnert, and Ard A Louis. Genetic correlations greatly increase mutational robustness and can both reduce and enhance evolvability. *PLoS computational biology*, 12(3): e1004773, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Tor Lattimore and Marcus Hutter. No free lunch versus occam's razor in supervised learning. In *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence*, pp. 223–235. Springer, 2013.

Abraham Lempel and Jacob Ziv. On the complexity of finite sequences. *IEEE Transactions on information theory*, 22(1):75–81, 1976.

LI Ming and Paul MB Vitányi. Kolmogorov complexity and its applications. *Algorithms and Complexity*, 1:187, 2014.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pp. 1376–1401, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017a.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5949–5958, 2017b.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pp. 3360–3368, 2016.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Jürgen Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873, 1997.

Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. A correspondence between random neural networks and statistical field theory. *arXiv preprint arXiv:1710.06570*, 2017.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Daniel Soudry, Elad Hoffer, and Nathan Srebro. The implicit bias of gradient descent on separable data. *arXiv preprint arXiv:1710.10345*, 2017.

Shizhao Sun, Wei Chen, Liwei Wang, Xiaoguang Liu, and Tie-Yan Liu. On the depth of deep neural networks: A theoretical view. In *AAAI*, pp. 2066–2072, 2016.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

David H Wolpert and R Waters. The relationship between pac, the statistical physics framework, the bayesian framework, and the vc framework. In *In*. Citeseer, 1994.

Lei Wu, Zhanxing Zhu, et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

Huan Xu and Shie Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Chiyuan Zhang, Qianli Liao, Alexander Rakhlin, Brando Miranda, Noah Golowich, and Tomaso Poggio. Musings on deep learning: Properties of sgd. 2017.