Journal of Machine Learning Research 22 (2021) 1-64 Submitted 6/20; Revised 10/20; Published 2/21

Is SGD a Bayesian sampler? Well, almost.

Chris Mingard CHRISTOPHER.MINGARD@CHEM.OX.AC.UK
Department of Chemistry
University of Oxford

Guillermo Valle-Pérez GUILLERMO.VALLE@DTC.OX.AC.UK
Department of Physics
University of Oxford

Joar Skalse JOAR.SKALSE@CS.OX.AC.UK
Department of Computer Science
University of Oxford

Ard A. Louis ARD.LOUIS@PHYSICS.OX.AC.UK
Department of Physics
University of Oxford

Editor: Amos Storkey

Abstract

Deep neural networks (DNNs) generalise remarkably well in the overparameterised regime, suggesting a strong inductive bias towards functions with low generalisation error. We empirically investigate this bias by calculating, for a range of architectures and datasets, the probability $P_{\mathrm{SGD}}(f|S)$ that an overparameterised DNN, trained with stochastic gradient descent (SGD) or one of its variants, converges on a function f consistent with a training set S. We also use Gaussian processes to estimate the Bayesian posterior probability $P_{\mathrm{B}}(f|S)$ that the DNN expresses f upon random sampling of its parameters, conditioned on S.

Our main findings are that $P_{\mathrm{SGD}}(f|S)$ correlates remarkably well with $P_{\mathrm{B}}(f|S)$ and that $P_{\mathrm{B}}(f|S)$ is strongly biased towards low-error and low complexity functions. These results imply that strong inductive bias in the parameter-function map (which determines $P_{\mathrm{B}}(f|S)$), rather than a special property of SGD, is the primary explanation for why DNNs generalise so well in the overparameterised regime.

While our results suggest that the Bayesian posterior $P_{\mathrm{B}}(f|S)$ is the first order determinant of $P_{\mathrm{SGD}}(f|S)$, there remain second order differences that are sensitive to hyperparameter tuning. A function probability picture, based on $P_{\mathrm{SGD}}(f|S)$ and/or $P_{\mathrm{B}}(f|S)$, can shed light on the way that variations in architecture or hyperparameter settings such as batch size, learning rate, and optimiser choice, affect DNN performance.

Keywords: stochastic gradient descent, Bayesian neural networks, deep learning, Gaussian processes, generalisation.

1. Introduction

While deep neural networks (DNNs) have revolutionised modern machine learning (LeCun et al., 2015; Schmidhuber, 2015), a solid theoretical understanding of why they work so well is still lacking. One surprising property is that they typically perform best in the overparameterised regime, with many more parameters than data points. Standard learning theory approaches (Shalev-Shwartz and Ben-David, 2014), based for example on model capacity, suggest that such highly expressive (Cybenko, 1989; Hornik, 1991; Hanin, 2019) DNNs should heavily over-fit in this regime, and therefore not generalise at all.

Stochastic gradient descent (SGD) (Bottou et al., 2018) is one of the key technical innovations allowing large DNNs to be efficiently trained in the highly overparameterised regime. In supervised learning, SGD allows the user to efficiently find sets of parameters that lead to zero training error. The power of SGD as an optimiser for DNNs was demonstrated in an influential paper (Zhang et al., 2016), which showed that zero training error solutions for CIFAR-10 image data with randomised labels can be found with a relatively moderate increase in computational effort over that needed for a correctly labelled dataset. These experiments also framed the conundrum of generalisation in the overparameterised regime as follows: Given that DNNs can memorise randomly labelled image datasets, which leads to poor generalisation, why do they behave so differently on correctly labelled datasets and select for functions that generalise well? The solution to this conundrum must be that SGD-trained DNNs have an inductive bias towards functions that generalise well (on structured data).

The possibility that SGD is not just good for optimisation, but is also a key source of inductive bias, has generated an extensive literature. One major theme concerns the effect of SGD on the flatness of the minima found, typically expressed in terms of eigenvalues of a local Hessian or related measures. A link between better generalisation and flatter minima has been widely reported (Hochreiter and Schmidhuber, 1997a; Keskar et al., 2016; Jastrzebski et al., 2018; Wu et al., 2017; Zhang et al., 2018; Wei and Schwab, 2019), but see also (Dinh et al., 2017). Theoretical work on SGD has also generated a large and sophisticated literature. For example, in (Soudry et al., 2018) it was demonstrated that SGD finds the max-margin solution in unregularised logistic regression, whilst it was shown in (Brutzkus et al., 2017) that overparameterised DNNs trained with SGD avoid over-fitting on linearly separable data. Recently, (Allen-Zhu et al., 2019) proved agnostic generalisation bounds of SGD-trained neural networks. Other recent work (Poggio et al., 2020) suggests that gradient descent performs a hidden regularisation in normalised weights, but a different analysis suggests that such implicit regularisation may well be very hard to prove in a more general setting for SGD (Dauber et al., 2020). Overall, while SGD and its related algorithms are excellent optimisers, there is as yet no consensus on what inductive bias SGD provides for DNNs. For a more detailed discussion of this SGD-related literature see Section 7.2.

An alternative approach is to consider the inductive properties of *random neural networks*, that is *untrained* DNNs with weights sampled from a (typically i.i.d.) distribution. Recent theoretical and empirical work (Valle-Pérez et al., 2018; De Palma et al., 2018; Mingard et al., 2019) suggests that the (prior) probability $P(f)$ that an untrained DNN outputs a function f upon random sampling of its parameters (typically the weights and biases) is strongly biased towards "simple" functions with low Kolmogorov complexity (see also Section 7.3). A widely held assumption is that such simple hypotheses will generalise well – think Occam's razor. Indeed, many processes modelled by DNNs are simple (Lin et al., 2017; Goldt et al., 2019; Spigler et al., 2019). For more on these topics see Section 7.3 and Section 7.5.

If the inductive bias towards simplicity described above for untrained networks is preserved throughout training, then this could help explain the DNN generalisation conundrum. Again, there is an extensive literature relevant to this topic. For example, a number of papers (Poole et al., 2016; Lee et al., 2018; Valle-Pérez et al., 2018; Yang, 2019a; Mingard et al., 2019; Cohen et al., 2019; Wilson and Izmailov, 2020) employ arguments on heuristic grounds that the bias in untrained random neural networks could be used to study the inductive bias of optimiser-trained DNNs. Optimiser-trained DNNs have also been directly compared to their Bayesian counterparts (c.f. Sections 6 and 7.4 for more detailed discussions). In an important development, Lee et al. (2017); Matthews et al. (2018); Novak et al. (2018b) used the Gaussian process (GP) approximation to Bayesian DNNs, which is exact in the limit of infinite width, and found that the generalisation performance of Bayesian DNNs and SGD-trained DNNs was relatively similar for standard deep learning datasets such as CIFAR-10, though Wenzel et al. (2020) found more significant differences when using Monte Carlo to approximate finite-width Bayesian DNNs. Others have used either Monte Carlo methods (Mandt et al., 2017) or the GP approximation (Matthews et al., 2017; de G. Matthews et al., 2018; Lee et al., 2019; Wilson and Izmailov, 2020) to examine how similar the Bayesian posterior is to the sampling distribution of SGD (whether in parameter or function space), albeit on relatively low dimensional systems compared to conventional DNNs.

In this paper we perform extensive computations, for a series of standard DNNs and datasets, of the probability $P_{\mathrm{SGD}}(f|S)$ that a DNN trained with SGD (or one of its variants) to zero error on training set S, converges on a function f. We then compare these results to the Bayesian posterior probability $P_{\mathrm{B}}(f|S)$, for these same functions, conditioned on achieving zero training error on S.

The **main question** we explore here is: *How similar is $P_B(f|S)$ to $P_{SGD}(f|S)$*? If the two are significantly different, then we may conclude that SGD provides an important source of inductive bias. If the two are broadly similar over a wide range of architectures, datasets, and optimisers, then the inductive bias is primarily determined by the prior $P(f)$ of the untrained DNN.

1.1 Main results summary

We carried out extensive sampling experiments to estimate $P_{\mathrm{SGD}}(f|S)$. Functions are distinguished by the way they classify elements on a test set E. We use the Gaussian process (GP) approximation to estimate $P_{\mathrm{B}}(f|S)$ for the same systems. Our main findings are:

(1) $P_{\mathrm{SGD}}(f|S) \approx P_{\mathrm{B}}(f|S)$ for a range of architectures, including FCNs, CNNs and LSTMs, applied to datasets such as MNIST, Fashion-MNIST, an IMDb movie review database and an ionosphere dataset. This agreement also holds for variants of SGD, including Adam (Kingma and Ba, 2014), Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012) and RMSprop (Tieleman and Hinton, 2012).

(2) The $P_{\mathrm{B}}(f|S)$ of functions f that achieve zero-error on the training set S can vary over hundreds of orders of magnitude, with a strong bias towards a set of low generalisation/low complexity functions. This tiny fraction of high probability functions also dominate what is found by DNNs trained with SGD. It is striking that even within this subset of functions, $P_{\mathrm{SGD}}(f|S)$ and $P_{\mathrm{B}}(f|S)$ correlate so well. Our empirical results suggest that, *for DNNs with large bias in $P_B(f|S)$*, SGD behaves *to first order* like a Bayesian optimiser and is therefore exponentially biased towards simple functions with better generalisation. Thus, SGD is not itself the primary source of inductive bias for DNNs.

(3) A function-based picture can also be fruitful for illustrating *second order* effects where an optimiser-trained DNN differs from the Bayesian prediction. For example, training an FCN with different optimisers (OPT) such as Adam, Adagrad, Adadelta and RMSprop on MNIST generates slight but measurable variations in the distributions of $P_{\text{OPT}}(f|S)$. Such information can be used to analyse differences in performance. For instance, we find that changing batch size affects $P_{\text{Adam}}(f|S)$ but, as was found for generalisation error (Keskar et al., 2016; Goyal et al., 2017; Hoffer et al., 2017; Smith et al., 2017), this effect can be compensated by changes in learning rate. Architecture changes can also be examined in this picture. For example, adding max-pooling to a CNN trained with Adam on Fashion-MNIST increases both $P_{\text{B}}(f|S)$ and $P_{\text{Adam}}(f|S)$ for the lowest-error function f found.

2. Preliminaries

We first introduce a key definition from (Valle-Pérez et al., 2018) needed to specify $P(f)$ and $P_B(f|S)$.

Definition 1 (Parameter-function map) *Consider a parameterised supervised model, and let the input space be $\mathcal{X}$ and the output space be $\mathcal{Y}$. The space of functions the model can express is a set $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$. If the model has some number of parameters, taking values within a set $\Theta \subseteq \mathbb{R}^p$, then the parameter-function map $\mathcal{M}$ is defined by*

$$\begin{aligned} \mathcal{M} : \Theta &\to \mathcal{F} \\ \theta &\mapsto f_\theta \end{aligned}$$

where f_θ is the function corresponding to parameters $\theta \in \Theta$.

The function space $\mathcal{F}$ of a DNN $\mathcal{N}$ could in principle be considered to be the entire space of functions that $\mathcal{N}$ can express on the input vector space $\mathcal{X}$, but it could also be taken to be the set of partial functions $\mathcal{N}$ can express on some subset of $\mathcal{X}$. For example, $\mathcal{F}$ could be taken to be the set of possible classifications of images in MNIST. In this paper we always take $\mathcal{F}$ to be the set of possible outputs of $\mathcal{N}$ for the instances in some dataset.

2.1 The Bayesian prior probability, $P(f)$

Given a distribution $P_{\text{par}}(\theta)$ over the parameters, we define the $P(f)$ over functions as

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

where $\mathbb{1}$ is an indicator function (1 if its argument is true, and 0 otherwise). This is the probability that the model expresses f upon random sampling of parameters over a parameter initialisation distribution $P_{\text{par}}(\theta)$, which is typically taken to have a simple form such as a (truncated) Gaussian. $P(f)$ can also be interpreted as the probability that the DNN expresses f upon initialisation before an optimisation process. It was shown in (Valle-Pérez et al., 2018) that the exact form of $P_{\text{par}}(\theta)$ (for reasonable choices) does not affect $P(f)$ much (at least for ReLU networks). If we condition on functions that obtain zero generalisation error on a dataset S, then the procedure above can also be used to generate the posterior $P_B(f|S)$ which we describe next.

2.2 The Bayesian posterior probability, $P_{\mathbf{B}}(f|S)$

Here, we describe the Bayesian formalism we use, and show how bias in the prior affects the posterior. Consider a supervised learning problem with training data S corresponding to the exact values of the function which we wish to infer (i.e. no noise). This formulation corresponds to a 0-1 *likelihood* $P(S|f)$, indicating whether the data is consistent with the function. Formally, if $S = \{(x_i, y_i)\}_{i=1}^m$ corresponds to the set of training pairs, then we let

$$P(S|f) = \begin{cases} 1 \text{ if } \forall i, \quad f(x_i) = y_i \\ 0 \text{ otherwise .} \end{cases}$$

Note that in our calculations, this quantity is technically $P(S|f; \{x_i\})$, but we denote it as $P(S|f)$ to simplify notation. We will use a similar convention throughout, whereby the input points are (implicitly) conditioned over. We then assume the prior $P(f)$ corresponds to the one defined in Section 2.1. Bayesian inference then assigns a *Bayesian posterior probability* $P_{\mathrm{B}}(f|S)$ to each f by conditioning on the data according to Bayes rule

$$P_{\mathrm{B}}(f|S) := \frac{P(S|f)P(f)}{P(S)}, \tag{2}$$

where $P(S)$ is also called the *marginal likelihood* or *Bayesian evidence*. It is the total probability of all functions compatible with the training set. For discrete functions, $P(S) = \sum_f P(S|f)P(f) = \sum_{f \in C(S)} P(f)$, with $C(S)$ the set of all functions compatible with the training set. For a fixed training set, all the variation in $P_B(f|S)$ for $f \in C(S)$ comes from the prior $P(f)$ of the untrained network since $P(S)$ is constant. Thus, *the bias in the prior is essentially translated over to the posterior.*

Thus, $P_{\mathrm{B}}(f|S)$ is the distribution over functions that would be obtained by randomly sampling parameters according to $P_{\mathrm{par}}(\theta)$ and selecting only those that are compatible with S.

2.3 The optimiser probability, $P_{\mathbf{OPT}}(f|S)$

But DNNs are not normally trained by randomly sampling parameters: They are trained by an optimiser. The probability that the optimiser OPT (e.g. SGD) finds a function f with zero error on S can be defined as:

$$P_{\mathrm{OPT}}(f|S) := \int \mathbb{1}[\mathcal{M}(\theta_f) = f] P_{\mathrm{OPT}}(\theta_f|\theta_i, S) \tilde{P}_{\mathrm{par}}(\theta_i) d\theta_i d\theta_f \tag{3}$$

where $P_{\mathrm{OPT}}(\theta_t|\theta_i, S)$ denotes the probability that OPT, initialised with parameters θ_i on a DNN, converges to parameters θ_f when training is halted after the first epoch where zero classification error is achieved on S[1], if such a condition is achieved in a number of iterations less than the maximum number which we allow for the experiments. The initialisation distribution $\tilde{P}_{\mathrm{par}}(\theta_i)$ is defined analogously to $P_{\mathrm{par}}(\theta)$ in Equation (1) (though it need not be exactly the same). $P_{\mathrm{OPT}}(f|S)$ is, therefore, a measure of the 'size' of f's 'basin of attraction', which intuitively refers to the set of initial parameters that converge to f upon training.

[1] In the special case where we specify the experiment as 'overtraining', then we take the parameters after p epochs with 0 classification error.

3. Methodology, Datasets and DNNs

3.1 Methodology

3.1.1 Definition of functions

For a specific DNN, training set $S = \{(x_i, y_i)\}_{i=1}^{|S|}$ and test set $E = \{(x'_i, y'_i)\}_{i=1}^{|E|}$, we define a function f as a labelling of the inputs in S concatenated with the inputs in E.[2] We will only look at functions which have 0 error on S, so that, for a particular experiment with fixed S and E, the functions are distinguished only by the predictions they make on E. Furthermore we only consider binary classification tasks (c.f. Section 3.2), so that our output space[3] is $\mathcal{Y} = \{0, 1\}$. Therefore, we will represent functions by a binary string of length $|E|$ representing the labels on E; the ith character representing the label on the ith input of E, x'_i. For the sake of simplicity, we will not make a distinction between this *representation* of f and the function f itself, as they are related one-to-one for any particular experiment (with fixed S and E).

Restricting the input space where functions are defined can be thought of as a coarse-graining of the functions on the full input space (e.g. the space of all possible images for image classification), which allows us to estimate their probabilities from sampling. In the following subsections we explain how the main experimental quantities are computed. Further detail can be found in Appendix A.

3.1.2 Calculating $P_{\mathrm{OPT}}(f|S)$

For a given optimiser OPT (SGD or one of its variants), a DNN architecture, loss function (cross-entropy (CE) or mean-square error (MSE)), a training set S, and test set E, we repeat the following procedure n times: We sample initial parameters θ_i, from an i.i.d. truncated Gaussian distribution $\tilde{P}_{\mathrm{par}}(\theta_i)$, and train with the optimiser until the first epoch where the network has 100% training classification accuracy (except for experiments labelled "overtraining," where we halt training after p further epochs with 0 training error have occured, for some specified p) [4]. We then compute the function f found by evaluating the network on the inputs in E, as described before.

Note that during training, the network outputs are taken to be the pre-activations of the output layer, which are fed to either the MSE loss, or as logits for the CE loss. At evaluation (to compute f), the pre-activations are passed through a threshold function so that positive pre-activations output 1 and non-positive pre-activations output 0.

We chose sample sizes between $n = 10^4$ and $n = 10^6$. In other words, we typically sample over $n = 10^4$ to $n = 10^6$ different trained DNNS, and count how many times each function f appears in the sample to generate the estimates of $P_{\mathrm{OPT}}(f|S)$. We leave the dependence of $P_{\mathrm{OPT}}(f|S)$ on E implicit. This method of estimating $P_{\mathrm{OPT}}(f|S)$ is described more formally in Appendix A.1.1.

[2] Formally then, our space of functions is $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$, where $\mathcal{X} = \{x_i\}_{i=1}^{|S|} \cup \{x'_i\}^{|E|}$

[3] Note that for training with MSE loss, we centered the output so the loss is measured with respect to target values in $\{-1, 1\}$. This is so thresholding can occur at a value of the last layer preactivation of 0, which is the same as for cross-entropy loss on logits.

[4] If SGD fails to achieve 100% accuracy on S in a maximum number of iterations, we discard the run.

3.1.3 Calculating $P_{\mathrm{B}}(f|S)$ with Gaussian Processes

We use neural network Gaussian processes (NNGPs) (Lee et al., 2017; Matthews et al., 2018; Garriga-Alonso et al., 2019; Novak et al., 2018b) to approximate $P_{\mathrm{B}}(f|S)$, for some training set S and test set E. NNGPs have been shown to accurately approximate the prior over functions $P(f)$ of finite-width Bayesian DNNs (Valle-Pérez et al., 2018; de G. Matthews et al., 2018). We use DNNs with relatively wide intermediate layers, relative to the input dimension, to ensure that we are close to the infinite layer-width NNGP limit. Depending on the loss function, we estimate the posterior $P_{\mathrm{B}}(f|S)$ as follows:

- **Classifiation as regression with MSE loss**. As has been done in previous work on NNGPs, we consider the classification labels as regression targets with an MSE loss[5]. We compute the analytical posterior for the NNGP with Gaussian likelihood. This is the posterior over the real-valued outputs at the test points on E, which correspond to the pre-activations of the output layer of the DNN. We sample from this posterior, and threshold the real-values like we do for DNNs (positive becomes 1 and otherwise it becomes 0) to obtain labels on E, and thus a function f. We then **estimate $P_{\mathrm{B}}(f|S)$ by counting how many times each f is obtained from a set of n independent samples from the posterior**, similar to what we did for $P_{\mathrm{OPT}}(f|S)$. For more details on GP computations with MSE loss, see Appendix A.2.1. We describe this method more formally in Appendix A.1.2. We use this technique when comparing $P_{\mathrm{B}}(f|S)$ with $P_{\mathrm{OPT}}(f|S)$ for MSE loss (e.g. Figure 1a).

- **Classification with CE loss**. In several experiments, we approximate the NNGP posterior using a 0-1 misclassification loss, which is more justified for classification, and can be thought of as a "low temperature" version of the CE loss. Since, in contrast to the MSE case, this posterior is not analytically tractable, we use the expectation propagation (EP) approximation to estimate probabilities (Rasmussen, 2004). In particular, we **estimate $P_{\mathrm{B}}(f|S)$ via ratio of EP-approximated likelihoods**. The EP approximation can be used to estimate the marginal likelihood of any labelling over any set of inputs, given a GP prior. As shown in Equation (2), we can use Bayes theorem to express $P_{\mathrm{B}}(f|S)$ as a ratio of $P(f)$ and $P(S)$ (which is valid for functions with 0 error on S, and the $0-1$ likelihood), and then use EP approximation to obtain both of these probabilities. In the text, when we refer to the EP approximation for calculating $P_{\mathrm{B}}(f|S)$, we are using it as described above.

3.1.4 Comparing $P_{\mathrm{OPT}}(f|S)$ to $P_{\mathrm{B}}(f|S)$

We note that for MSE loss, we can sample to accurately estimate function probabilities, whereas for the CE loss, we must use the EP approximation to calculate the probability of functions.[6] When we compare $P_{\mathrm{B}}(f|S)$ to $P_{\mathrm{OPT}}(f|S)$ for CE loss, we take the functions found by the optimiser, which are obtained as described in Section 3.1.2, and calculate their $P_{\mathrm{B}}(f|S)$ using the EP approximation. For MSE loss, both the $P_{\mathrm{B}}(f|S)$ and the $P_{\mathrm{OPT}}(f|S)$ are sampled independently, and probabilities are compared for functions found by both methods.

[5] As for $P_{\mathrm{OPT}}(f|S)$ with MSE loss, we take the regression targets to be $\{-1, 1\}$, so thresholding occurs at 0

[6] See Appendix A.2.2 for more details.

3.1.5 Calculating $P_{\mathrm{B}}(f|S)$ for functions with generalisation error from 0% to 100%

For the zero training error case studied here, we define functions by their particular labelling on the test set E (as described in Section 3.1.1). A function can be generated by picking a certain labelling. Subsequently $P_{\mathrm{B}}(f|S)$ for CE loss can be calculated using the EP approximation as described above. To study how $P_{\mathrm{B}}(f|S)$ varies with generalisation error ϵ_G on E (the fraction of missclasified inputs on E), we perform the following procedure. For each value of ϵ_G chosen, typically 10 functions are uniformly sampled by randomly selecting $\epsilon_G|E|$ bits in E and flipping them. EP is then used to calculate $P_{\mathrm{B}}(f|S)$ for those functions. The probabilities $P_{\mathrm{B}}(f|S)$ can range over many orders of magnitude. The low probability functions cannot be obtained by direct sampling, so that a full comparison with $P_{\mathrm{OPT}}(f|S)$ is not feasible. This is more formally described in Appendix A.1.3.

3.1.6 CSR complexity

The critical sample ratio (CSR) is a measure of complexity of functions expressed by DNNs (Krueger et al., 2017). It is defined with respect to a sample of inputs as the fraction of those samples which are critical samples. A critical sample is defined to be an input such that there is another input within a box of side $2r$ centred around the input, producing a different output (for discrete classification outputs). See Appendix E for further details.

3.2 Data sets

To efficiently sample functions, we use relatively small test sets (typically $|E| = 100$) and, as is often done in the theoretical literature, binarise our classification datasets. We define the datasets used below:

MNIST: The MNIST database of handwritten numbers (LeCun et al., 1999) was binarised with even numbers classified as 0 and odd numbers as 1. Unless otherwise specified, we used $|S| = 10000$ and $|E| = 100$.

Fashion-MNIST: The Fashion-MNIST database (Xiao et al., 2017) was binarised with T-shirts, coats, pullovers, shirts and bags classified as 0 and trousers, dresses, sandals, trainers and ankle boots classified as 1. Unless otherwise specified, we used $|S| = 10000$ and $|E| = 100$.

IMDb movie review dataset: We take the IMDb movie review dataset from Keras. The task is to correctly classify each review as positive or negative given the text of the review. We preprocess the set by removing the most common words and normalising.[7] This procedure was employed to make sure there are functions with high enough probability to be sampled multiple times with Experiments 1 and 2 above. Used with $|S| = 45000$ and $|E| = 50$.

Ionosphere Dataset: This is a small non-image dataset with 34 features[8] aimed at identifying structure in the ionosphere (Sigillito et al., 1989). Used with $|S| = 301$ and $|E| = 50$.

[7] We used the version of the dataset and preprocessing technique given here: `https://www.kaggle.com/drscarlat/imdb-sentiment-analysis-keras-and-tensorflow`

[8] `https://archive.ics.uci.edu/ml/datasets/Ionosphere`

For image datasets, we will typically use normalised data (pixel values in range [0,1]) for MSE loss, and unnormalised data for CE loss (pixel values in range [0,255]).

3.3 Architectures

We used the following standard architectures.
FCN: 2 hidden layer, 1024 node vanilla fully connected network (FCN) with ReLU activations.
CNN + (Max Pooling) + [BatchNorm]: Layer 1: Convolutional Layer with 32 features size 3 × 3. (Layer 1a: Max Pool 2 × 2). [Layer 1b: Batch Norm]. Layer 2: Flatten. Layer 3: FCN with width 1024. [Layer 3a: Batch Norm]. Layer 4: FCN, 1 output with ReLU activations.
LSTM: Layer 1: Embedding layer. Layer 2: LSTM, 256 outputs. Layer 3: FCN, 512 outputs. Layer 4: Fully-Connected, 1 output with ReLU activations for the fully connected layers.
Hyperparameters are, unless otherwise specified, the default values in Keras 2.3.0. See Appendix A.1.1 for details on the parameter initialisation.

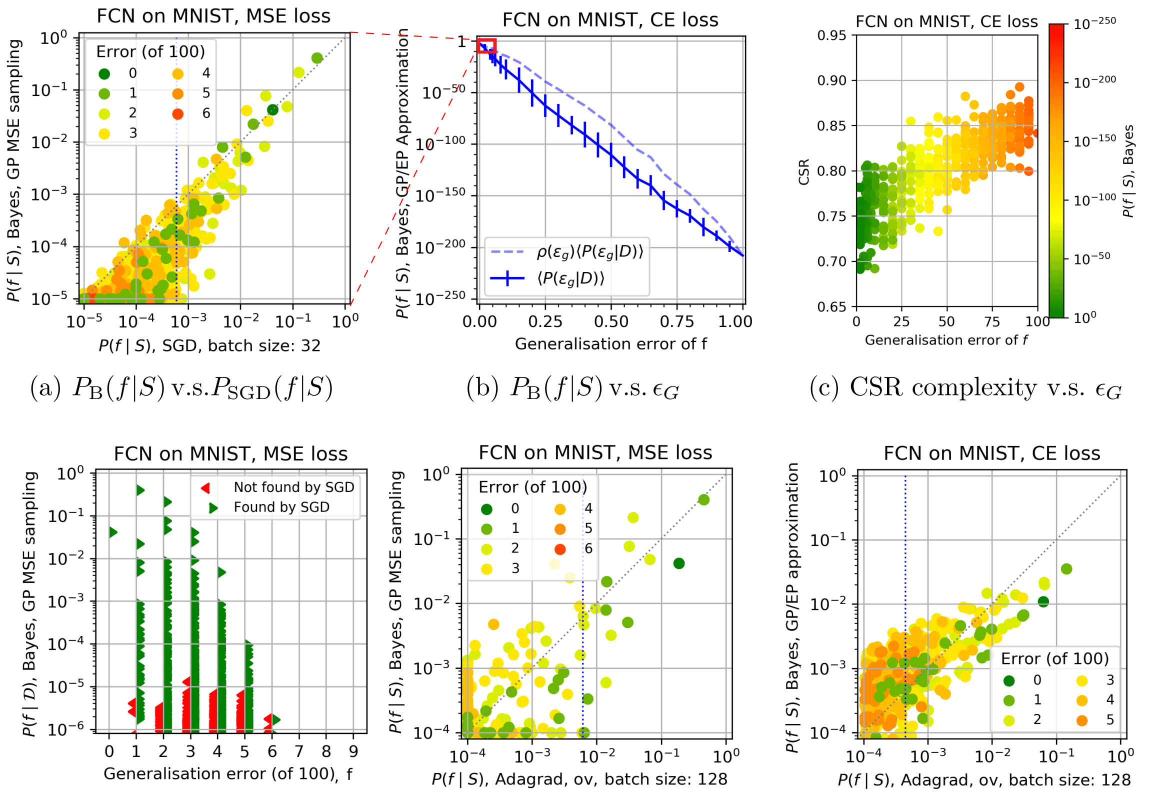
4. Empirical results for $P_{\mathbf{B}}(f|S)$ v.s. $P_{\mathbf{OPT}}(f|S)$ for different architectures and datasets

In this first of two main results sections, we focus on testing our hypothesis that $P_{\mathrm{B}}(f|S) \approx P_{\mathrm{OPT}}(f|S)$ for FCN, CNN and LSTM architectures on MNIST, Fashion-MNIST, the IMDb review, and the Ionosphere datasets, using several variants of the SGD optimiser. In the following subsection will describe the main results in detail for an FCN on MNIST. The experiments in the next sections will be the same except for the architecture, dataset, or optimiser which will be varied.

4.1 Comparing $P_{\mathbf{B}}(f|S)$ to $P_{\mathbf{OPT}}(f|S)$ for an FCN on MNIST

In Figure 1 we present a series of results for a standard DNN setup: an FCN (2 hidden layers, each 1024 node wide with ReLU activations), trained on (binarised) MNIST to zero training error with a training set size of $|S| = 10,000$ and a test set size of $|E| = 100$. Note that even for this small test set, there are $2^{100} \approx 1.3 \times 10^{30}$ functions with zero error on S, all of which an overparametrized DNN could express (Zhang et al., 2016)[9]. We chose standard values for batch size, learning rate, etc., if given by the default values in Keras 2.3.0 (e.g. batch size of 32 and learning rate of 0.01 for SGD). Our experiments in Section 5 and the appendices will show that our results are robust to the choice of these hyperparameters.
Figure 1a compares the value of $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{SGD}}(f|S)$ for the highest probability functions of each distribution, for MSE loss. Each data point in the plot corresponds to a unique function (a unique classification of images in the test set E). The functions are obtained by sampling both $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{SGD}}(f|S)$ and taking the union of the set of functions obtained. $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{SGD}}(f|S)$ were estimated as frequencies from the corresponding sample as explained in Sections 3.1.2 and 3.1.3. If a function does not appear in one of the samples,

[9] We also find in Figure 21a and Figure 21b that our 2-layer FCN is capable of expressing functions on MNIST with the full range of training and generalisation errors



(a) $P_{\mathrm{B}}(f|S)$ v.s.$P_{\mathrm{SGD}}(f|S)$ (b) $P_{\mathrm{B}}(f|S)$ v.s. ϵ_G (c) CSR complexity v.s. ϵ_G

(d) f found by NNGP in (a). (e) $P_{\mathrm{B}}(f|S)$ v.s.$P_{Adagrad}(f|S)$ (f) $P_{\mathrm{B}}(f|S)$ v.s.$P_{Adagrad}(f|S)$

Figure 1: **Comparing the Bayesian prediction $\mathbf{P_B(f|S)}$ to $\mathbf{P_{OPT}(f|S)}$ for SGD and Adagrad, for an FCN on MNIST** [We use training/test set size of 10,000/100; For (a,e,f), the vertical dotted blue lines are drawn at the highest value of $P_{\mathrm{OPT}}(f|S)$ such that the sum of $P_{\mathrm{OPT}}(f|S)$ for all functions above the line is $> 90\%$ (90% probability boundary); dashed grey line denotes $P_{\mathrm{B}}(f|S) = P_{\mathrm{OPT}}(f|S)$.]
(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{SGD}}(f|S)$ for MSE loss; Both $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{SGD}}(f|S)$ were sampled $n = 10^6$ times. The color shows the number of errors in the test set. The GP has average error $\langle\epsilon_G\rangle_{GP} = 1.61\%$, while SGD has average error $\langle\epsilon_G\rangle = 1.88\%$.
(b) $P_{\mathrm{B}}(f|S)$ (with CE loss) v.s. ϵ_G for the full range of possible errors on E. We use the methods from Section 3.1.5 with 20 random functions sampled per value of error. The solid blue line shows $\langle\log(P_{\mathrm{B}}(f|S))\rangle_{\epsilon_G}$, where the average is over the functions for a fixed ϵ_G; error bars are 2 standard deviations. The dashed blue line shows the weighted $\rho(\epsilon_G)\langle P_{\mathrm{B}}(f|S)\rangle_{\epsilon_G}$, where $\rho(\epsilon_G)$ is the number of functions with error ϵ_G. The small red box and dashed red lines illustrate the range of probability and error found in (a).
(c) CSR complexity versus generalisation error for the same functions as in fig (b). Color represents $P_{\mathrm{B}}(f|S)$, computed as in (b).
(d) Functions from (a) found by the sample of $P_{\mathrm{B}}(f|S)$, versus error. 913 functions of the functions are also found by SGD, taking up 97.70% of the probability for $P_{\mathrm{SGD}}(f|S)$, and 99.96% for $P_{\mathrm{B}}(f|S)$.
(e) $P_{\mathrm{B}}(f|S)$ v.s. $P_{Adagrad}(f|S)$ for MSE loss; $P_{Adagrad}(f|S)$ was sampled $n = 10^5$ times (while the GP sample was the same as in (a)). Adagrad was overtrained until 64 epochs had passed with zero error. The average error is $\langle\epsilon_G\rangle = 1.53\%$.
(f) is as (e) but with CE loss, so that the EP approximation was used for $P_{\mathrm{B}}(f|S)$, making the estimate of $P_{\mathrm{B}}(f|S)$ slightly less accurate. $\langle\epsilon_G\rangle = 2.63\%$.

we set its frequency to take the minimum value so that it would appear on top of one of the axes. For example, a function that appears in the SGD runs, but not in the sampling for $P_{\text{B}}(f|S)$, will appear on x-axis at the value obtained for $P_{\text{SGD}}(f|S)$. Here we used MSE loss rather than the more popular (and typically more computationally efficient) CE loss because for MSE, the posterior can be sampled from without further approximations, while for CE loss, the expectation propagation (EP) approximation needs to be used making $P_{\text{B}}(f|S)$ less accurate (see Appendix A.2 for further details).

Figure 1a also demonstrates that $P_{\text{SGD}}(f|S)$ and $P_{\text{B}}(f|S)$ are remarkably closely correlated for MSE loss, and that a remarkably small number of functions account for most of the probability mass for both $P_{\text{B}}(f|S)$ and $P_{\text{SGD}}(f|S)$. To appreciate how remarkably tight this agreement is, consider the full scale of probabilities for functions f that achieve zero error on the MNIST training set. The average $P_{\text{B}}(f|S)$ of all these functions is $2^{-100} \approx 10^{-30}$. Therefore the functions in Figure 1a have probabilities that are many orders of magnitude higher than average. At the same time, $P_{\text{B}}(f|S)$ and $P_{\text{SGD}}(f|S)$ for these functions typically agree within less than one order of magnitude. Another way of quantifying the agreement is that 90% of the cumulative probability weight from both $P_{\text{SGD}}(f|S)$ and $P_{\text{B}}(f|S)$ for the test set E in Figure 1a is made up from the contributions of only a few tens of functions with zero training error out of $\approx 10^{30}$ such possible functions (see vertical doted line in Figure 1a). Moreover, these particular functions are the same for both $P_{\text{B}}(f|S)$ and $P_{\text{SGD}}(f|S)$. The agreement between the two methods is remarkable. Overall, the observations in Figure 1a suggest that the main inductive bias of this DNN is present prior to training.

Figure 1b plots the mean probability for obtaining a generalisation error of ϵ_G in the training set E, which is estimated as $\rho(\epsilon_G)\langle P_B(f|S)\rangle_{\epsilon_G}$ where $\rho(\epsilon_G) = |E|!/((|E| - \epsilon_G|E|)!(\epsilon_G|E|)!)$ denotes the number of functions with $\epsilon_G|E|$ errors on E, and $\langle P_B(f|S)\rangle_{\epsilon_G}$ denotes the expected value of $P_{\text{B}}(f|S)$, where the expectation is with respect to uniformly sampling from the set of functions with fixed ϵ_G. As explained in Section 3.1.5, we estimate the average $\langle\cdot\rangle_{\epsilon_G}$ by sampling, and we estimated $P_{\text{B}}(f|S)$ for each f in the sample using the EP approximation.

Figure 1b can be interpreted as showing that the inductive bias encoded in $P_{\text{B}}(f|S)$ gives good generalisation. More precisely, we find that $P_{\text{B}}(f|S)$ is exponentially biased towards functions with low generalisation error. To illustrate how strong the bias is, we can look at $\rho(\epsilon_G)$. Over 50% of functions are in the range of $\epsilon_G = 50 \pm 3$ errors, while only 10^{-23}% have $\epsilon_G \leq 3$. Therefore for $P_{\text{B}}(f|S)$ to overcome the 'entropic' factor $\rho(\epsilon_G)$ and show the behaviour in Figure 1b, it must in average give a probability many orders of magnitude higher to low error functions than to high error functions. In Appendix A.1.3, we also observed that the probability p_i of misclassifying an image in the test set varies a lot between images, and that these probabilities are to first order independent. As a corollary, in Figure 8 we show for $P_{\text{B}}(f|S)$ and $P_{\text{SGD}}(f|S)$ that the probabilities of multiple images being misclassified can be accurately estimated from the products of the probabilities p_i for misclassifying individual images. Thus this system appears to behave like a Poisson-Binomial distribution with independent and non-identically distributed random p_i, which most likely also explains why $\langle \log P_B(f|S)\rangle_{\epsilon_G}$ scales nearly linearly with ϵ_G.

Although we cannot measure $P_{\text{SGD}}(f|S)$ for the high generalisation error functions, the agreement in Figure 1a (and elsewhere in this paper) implies that $P_{\text{SGD}}(f|S)$ must also be on average orders of magnitude lower for high error functions than low error functions.

However, at the moment we can only conjecture that $P_{\mathrm{SGD}}(f|S)$ follows the same exponential behaviour as $P_{\mathrm{B}}(f|S)$ over the whole range of ϵ_G. Finally, in Appendix A.1.3, we make some further remarks and caveats about this experiment, and other similar experiments.

Figure 1c shows the correlation between the complexity of the functions obtained to create Figure 1b, and their generalisation error, as well as their $P_{\mathrm{B}}(f|S)$ (from EP approximation) represented in their color. The complexity measure we used is the critical sample ratio (CSR) complexity (Krueger et al., 2017) computed on the inputs in E, which measures what fraction of inputs are near the decision boundary (see Section 3.1.6).

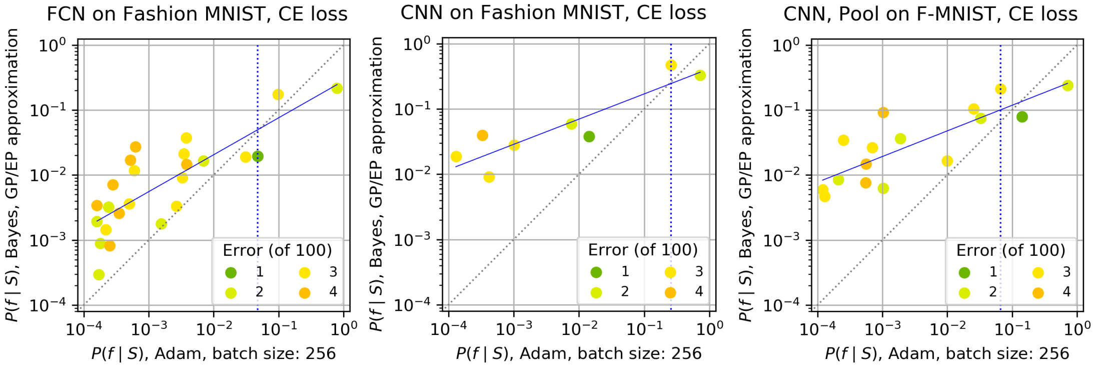
Figure 1c also shows that there is a inverse correlation between the generalisation of a function and its CSR complexity, as well as between $P_{\mathrm{B}}(f|S)$ and CSR. In Section 2.2, we showed that $P_{\mathrm{B}}(f|S)$ is proportional to the prior probability of a function $P(f)$ for functions that have zero error on the training set S. We can thus understand the inverse correlation between $P_{\mathrm{B}}(f|S)$ and CSR in the light of previous *simplicity bias* results showing that the prior $P(f)$ of Bayesian DNNs is exponentially biased towards functions with low Kolmogorov complexity (simple functions) (Valle-Pérez et al., 2018; Mingard et al., 2019). In (Valle-Pérez et al., 2018), it was further shown for an FCN on a subsample of MNIST that $P(f)$ correlated remarkably well with CSR[10], and our results are in agreement with that finding. The results in this figure extend those of Figure 1b to show that $P_{\mathrm{B}}(f|S)$ is biased both towards low error and simple functions, and that simple functions are the ones that tend to have good generalisation on MNIST.

Figure 1d shows the correlation between $P_{\mathrm{B}}(f|S)$ and ϵ_G for functions used for Figure 1a. We note that, as can also be observed in Figure 1a, values of $P_{\mathrm{B}}(f|S)$ are high for low error functions, and high error functions have relatively lower values of $P_{\mathrm{B}}(f|S)$. This figure also uses colour to show which functions were not found in the sampling of $P_{\mathrm{SGD}}(f|S)$. It shows clearly that SGD finds all the high $P_{\mathrm{B}}(f|S)$ functions.

Figure 1e shows the same type of experiment as in Figure 1b, but using a different SGD-based optimiser, Adagrad (Duchi et al., 2011) with overtraining (where training was halted after 64 epochs had passed with 100% training accuracy). We see that it exhibits similar correlation between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$ to vanilla SGD (and very similar agreement was observed without overtraining). We will see throughout the paper remarkably good correlations between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$ holds for a large range of optimisers and hyperaparameters

Figure 1f shows the same type of experiment as in Figure 1a, but using CE loss, the Adagrad optimiser, and overtraining (also to 64 epochs). See Figure 11b for the equivalent plot but without overtraining. As we are using CE loss (see Section 3.1.3 and Section 3.1.4), we sample functions from $P_{\mathrm{OPT}}(f|S)$, and then use the EP to estimate $P_{\mathrm{B}}(f|S)$ for the functions obtained. We find similar results to Figure 1e, where we used MSE loss (and direct sampling for $P_{\mathrm{B}}(f|S)$). The errors introduced by the EP approximation may explain why the correlation does not follow the x=y line as closely as it does for the MSE calculations. Nevertheless, the correlation between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{Adagrad}}(f|S)$ is strong, providing evidence that our results for an FCN on MNIST are not an artefact of the exact optimiser or loss function used.

[10] Furthermore in (Valle-Pérez et al., 2018), it was shown that this was not an exclusive property of CSR and that any measure that could approximate Kolmogorov complexity seems to also correlate well with $P(f)$.



(a) $P_{\mathrm{B}}(f|S)$ v.s.$P_{\mathrm{Adam}}(f|S)$ for FCN

(b) $P_{\mathrm{B}}(f|S)$ v.s.$P_{\mathrm{Adam}}(f|S)$ for CNN w/o pooling

(c) $P_{\mathrm{B}}(f|S)$ v.s.$P_{\mathrm{Adam}}(f|S)$ for CNN w/ pooling

Figure 2: **Comparing $\mathbf{P_B(f|S)}$ to $\mathbf{P_{Adam}(f|S)}$ for CNNs and the FCN on Fashion-MNIST** [We use a training/test set size of 10,000/100; vertical dotted blue lines denote 90% probability boundary; dashed grey line is $P_{\mathrm{B}}(f|S) = P_{\mathrm{OPT}}(f|S)$.] **(a)** FCN on Fashion-MNIST; $\langle \epsilon_G \rangle = 2.11\%$ for Adam with CE loss. **(b)** Vanilla CNN on Fashion-MNIST; $\langle \epsilon_G \rangle = 2.25\%$ for Adam with CE loss. **(c)** CNN with max-pooling on Fashion-MNIST; $\langle \epsilon_G \rangle = 1.96\%$ for Adam with CE loss. Note that when max-pooling is added, the probability of the lowest-error function increases notably for both $P_{\mathrm{Adam}}(f|S)$ and $P_{\mathrm{B}}(f|S)$. There is a strong correlation between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{SGD}}(f|S)$ in all three plots. See Figure 13 for related results, including $P_{\mathrm{B}}(f|S)$ vs ϵ_G, a CNN with batch normalisation, and a CNN with MSE loss.

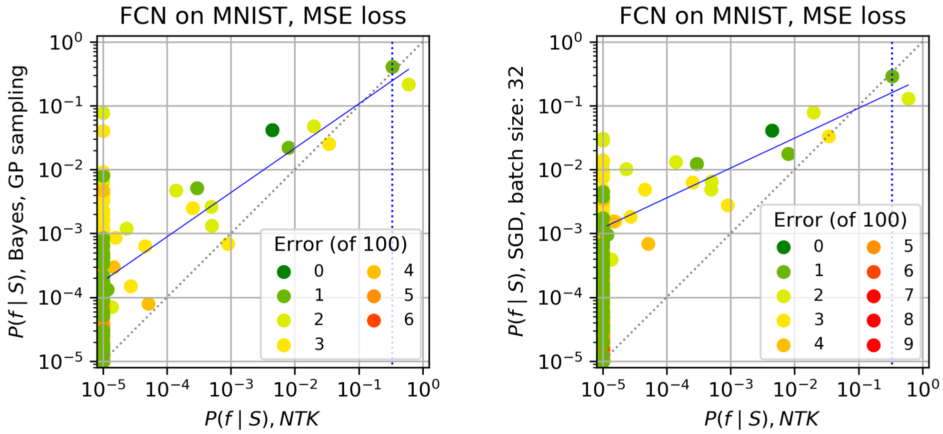
4.2 Comparing $P_{\mathrm{B}}(f|S)$ to $P_{\mathrm{Adam}}(f|S)$ for CNNs on Fashion-MNIST

We next turn to a more complex dataset, namely Fashion-MNIST (Xiao et al., 2017) which consists of images of clothing, as well as a more complex network architecture, the CNN (LeCun et al., 1999) which was designed in part to have a better inductive bias for images. See Section 3.2 and Section 3.3 for details on dataset and architecture. We can see in Figure 2 a strong correlation between $P_{\mathrm{B}}(f|S)$ and the probabilities found by the Adam optimiser (Kingma and Ba, 2014), a variant of SGD. Note that instead of MSE loss we used CE loss because it is more efficient. A downside of this choice is that we need to use the EP approximation for the GP calculations (see Appendix A.2.2). Although the correlation is strong, it does not follow x=y as closely as we generally find for MSE loss, which is quite possibly an effect of the EP approximation. See Figure 13 for an example with MSE loss where the correlation does follow x=y more closely. Both the FCN and the CNNs exhibit a strong bias towards low error functions on Fashion-MNIST as we can see in Figure 13c and Figure 13d.

For an example of how the effects of architecture modifications can be observed in the function probabilities, compare results in Figure 2b for the vanilla CNN to those in Figure 2c for a CNN with max-pooling (He et al., 2016), a method designed to improve the inductive bias of the CNN. As expected, the generalisation performance of the CNN improves, and an important contributor is the increase in the probability of the highest probability 1-error function in both $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{Adam}}(f|S)$, directly demonstrating an enhancement of the inductive bias. See Figure 13 for related results. This example demonstrates how a function based picture as well as analysis of the Bayesian $P_{\mathrm{B}}(f|S)$ sheds light on the inductive bias of a DNN. Such insights could help with architecture search, or more generally with developing new architectures with improved implicit bias toward desired low error functions.

4.3 Comparing $P_{\mathbf{B}}(f|S)$ and $P_{\mathbf{SGD}}(f|S)$ to Neural Tangent Kernel results

In Figure 3 we compare $P_{\mathrm{B}}(f|S)$ to the output of the neural tangent kernel (NTK) (Jacot et al., 2018), which approximates gradient descent in the limit of infinite width and infinitesimal learning rate. The generalisation error of NTK and NNGPs have been shown to be relatively close, and they produce similar functions on simple 1D regression (Lee et al., 2019; Novak et al., 2020). Here we show that this similarity also holds for the function probabilities for a more complex classification task. However, we also find the NTK misses many relatively high probability functions that both SGD and the GP find. We are currently investigating this surprising behaviour, which may arise from the infinitesimal learning rate. Their low probability may also be exacerbated by the fact that in Figure 3 the NTK is very highly biased towards one 2-error function, forcing other functions to have low cumulative probability. Again, this example demonstrates how a function based picture picks up rich details of the behaviour that would be missed when simply comparing generalisation error.



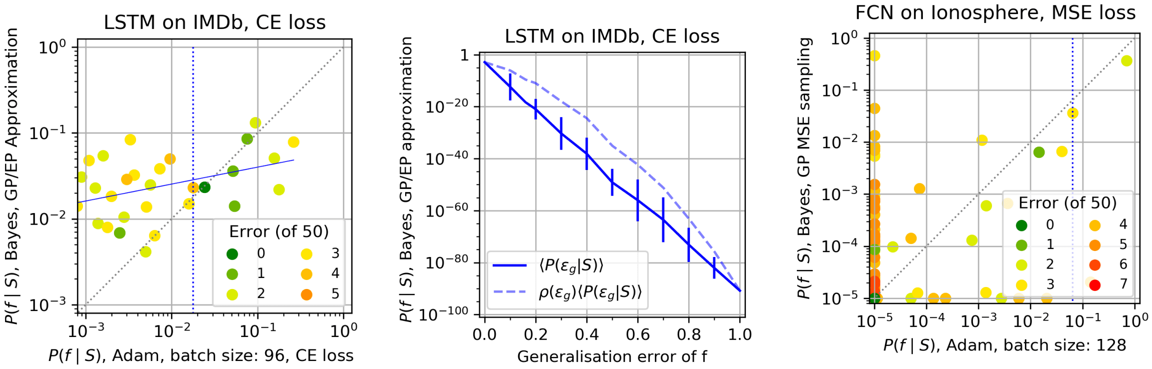
(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{NTK}}(f|S)$

(b) $P_{\mathrm{SGD}}(f|S)$ v.s. $P_{\mathrm{NTK}}(f|S)$

Figure 3: **Comparing $\mathbf{P_B(f|S)}$ and $\mathbf{P_{SGD}(f|S)}$ to $\mathbf{P_{NTK}(f|S)}$ for an FCN on MNIST.** [The functions to the right of the blue dotted lines make up 90% of the total probability. We sampled 10^7 times for NTK and GP, and 10^6 for SGD]. In (a) we show the correlation between $P_{NTK}(f|\mathcal{D})$ and $P_{\mathrm{B}}(f|S)$. Weighted by probability, 77.5% of functions found by sampling from the GP are found by NTK; all functions found by NTK are found by sampling from the GP. In (b), we show the correlation between the $P_{NTK}(f|\mathcal{D})$ and $P_{\mathrm{SGD}}(f|S)$. Weighted by probability, 65.8% of functions found by SGD are found by NTK; all functions found by NTK are found by SGD. $\langle\epsilon_G\rangle = 1.69\%$ (NTK), $\langle\epsilon_G\rangle = 1.61\%$ (GP), $\langle\epsilon_G\rangle = 1.88\%$ (SGD).

4.4 Comparing $P_{\mathbf{B}}(f|S)$ to $P_{\mathbf{Adam}}(f|S)$ for LSTM on IMDb sentiment analysis

We test a more complex DNN with a LSTM layer (Hochreiter and Schmidhuber, 1997b), applied to a problem of sentiment analysis on the IMDb movie database. We used a smaller test set $|E| = 50$ and a larger training set $|S|$ =45,000 to ensure that generalisation was good enough to ensure that functions are found with sufficient frequency to be able to extract probabilities. As can be seen in Figure 4a we again observe a reasonable correlation between the functions found by Bayesian sampling, and those found by the optimiser. Figure 4b also shows that, as observed for other datasets, this system is highly biased towards low error functions. We show some further experiments with the LSTM in Figure 14 in Appendix D, including an experiment with MSE loss to avoid the EP approximation.



(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$ (b) $P_{\mathrm{B}}(f|S)$ v.s. $\langle\epsilon_G\rangle$ (c) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$

Figure 4: **Comparing $\mathbf{P_B(f|S)}$ to $\mathbf{P_{Adam}(f|S)}$ for a LSTM on the IMDb movie review dataset, and an FCN on the ionosphere dataset.** (a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$ for LSTM on IMDb dataset, ($\langle\epsilon_G\rangle$ =4.28%, 10^4 samples). Because of the computational cost of the problem, we used a training set size of 45000 and a test set of size 50. (b) $P_{\mathrm{B}}(f|S)$ v.s. $\langle\epsilon_G\rangle$ for the LSTM on IMDb shows that the functions found by the Adam optimiser are in the small fraction of high $P_{\mathrm{B}}(f|S)$ probability/low error functions. (c) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$ for an FCN with 3 hidden layers of width 256 on the Ionosphere dataset. Training set size is 301 and the test set size is 50. ($\langle\epsilon_G\rangle = 4.59\%$ for Adam, $\langle\epsilon_G\rangle = 5.41\%$ for the GP). See Figures 14 and 15 for further results for these systems.

4.5 Comparing $P_{\mathbf{B}}(f|S)$ to $P_{\mathbf{Adam}}(f|S)$ for FCN on Ionosphere dataset

As another non-image classification example, we use the small non-image Ionosphere dataset (with a training set of size 301), using an FCN with 3 hidden layers of width 256. As can be seen in Figure 4c, for MSE loss we find a fairly good correlation. Further details and an example with CE loss can be found in Figure 15.

4.6 Effects of training set size

We performed experiments comparing $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$ for different training set sizes for the FCN on MINST. We observe that increasing the amount of training data from $|S| = 1000$ to $|S| = 20000$ increases the bias towards low error functions. This increase has the following effects: 1) An increase in the value of $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{SGD}}(f|S)$ for functions with low $\langle\epsilon_G\rangle$ by several orders of magnitude, 2) an increase by several orders of magnitude of $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$ for the mode functions (the ones with highest probability), 3) A decrease in the number of functions that cumulatively take up 90% of the observed probability weight, and 4) a significant increase in the tightness of correlation between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$. See Appendix C in Appendix C for detailed results and plots.

4.7 Results for other test sets

For the experiments shown in this section, sampling efficiency considerations means that we have limited ourselves to a relatively small test sets ($|E| \leq 100$). In Figure 12, we have checked that other test sets also show close agreement between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{SGD}}(f|S)$. For larger $|E|$, $P_{\mathrm{OPT}}(f|S)$ quickly becomes impossible to directly measure empirically – doubling the test set roughly means squaring the number of samples to obtain qualitatively similar results, as the values for $P_{\mathrm{B}}(f|S)$ decrease exponentially with test set size. However, if we assume that the images are approximately independently distributed throughout the larger test set, as Appendix B suggests, then we can estimate the highest probabilities from products of $P_{\mathrm{B}}(f|S)$ or $P_{\mathrm{SGD}}(f|S)$ on the smaller sets.

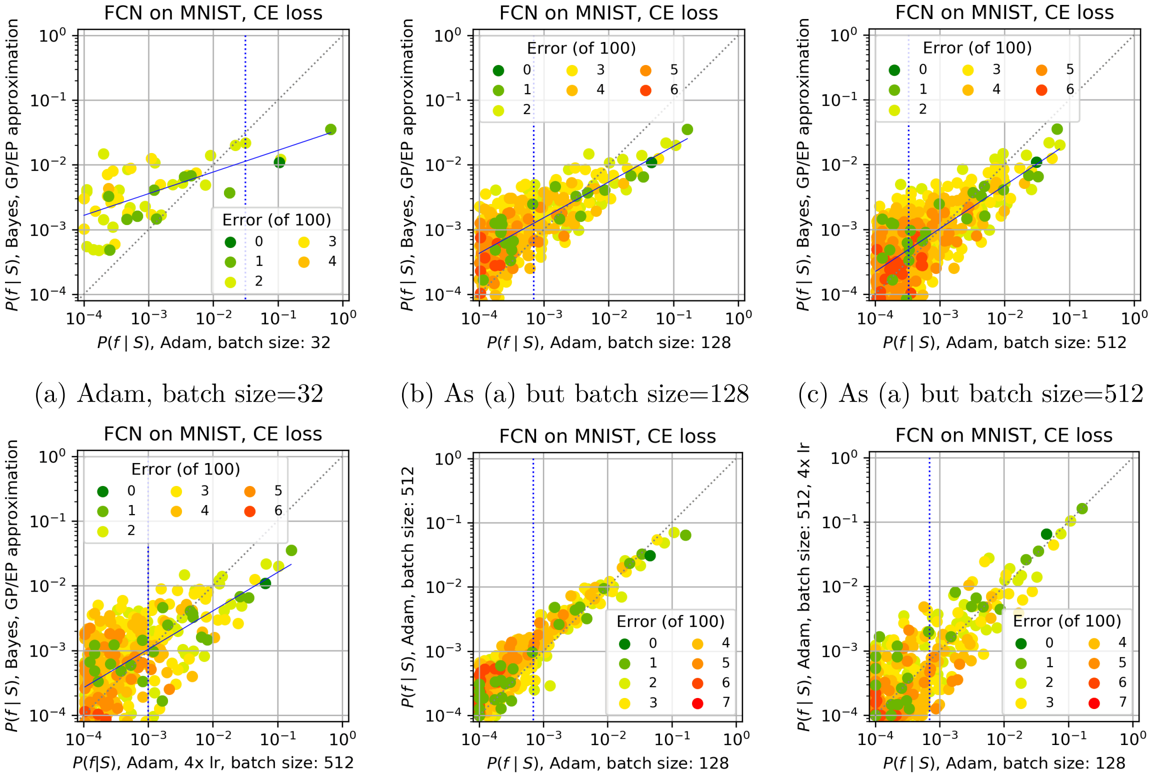
5. The effect of hyperparameter changes and optimisers on $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$

In the first section we focussed on the first-order similarity between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$. In this second main results section, we focus on second-order effects that affect $P_{\mathrm{OPT}}(f|S)$ differently from $P_{\mathrm{B}}(f|S)$. These include the effects of hyperparameter settings and optimiser choice.

5.1 Changing batch size and learning rate

In a well-known study, (Keskar et al., 2016) showed that, for a fixed learning rate, using smaller batch sizes could lead to better generalisation. In Figure 5 (a)-(c) we observe this same effect but reflected in the more finely grained spectrum of function probabilities. For batch size 512, we also reproduce in Figure 5d the effect observed in (Goyal et al., 2017; Hoffer et al., 2017; Smith et al., 2017), that speeding up the learning rate for a fixed batch size can mimic the improvement in $\langle\epsilon_G\rangle$ for smaller batches. Interestingly, as can be seen by comparing Figures 5d to 5f, the overall correlation of the function probability spectrum appears tighter for the 128 and 512 batch size with the same learning rates, even though the generalisation errors are different. However, if the learning rate is increased 4× for the the 512 batch size system, then there is a closer correlation with batch size 128 for the higher probability functions. It is these latter functions that dominate the average for $\langle\epsilon_G\rangle$ and so the closer correlation for those functions, rather than the less good correlation for low probability functions, explains the better agreement seen in generalisation error for the two systems.

In Figure 18 of Appendix D, we vary batch size for MSE, finding different trends to CE loss. For MSE, increasing batch size leads to better generalisation due to second order effects where $P_{\mathrm{SGD}}(f|S)$ preferentially converges on a few key higher probability/lower error functions. The batch size can be correlated with the noise spectrum of the underlying Langevin equation that describes SGD (Bottou et al., 2018; Jastrzebski et al., 2018; Zhang et al., 2018). What our function based results demonstrate is that the behaviour of the optimiser on the loss-landscape is affected in subtle ways by the form of the loss function, as well as the amount noise, and possibly also by correlations in the noise.



(a) Adam, batch size=32

(b) As (a) but batch size=128

(c) As (a) but batch size=512

(d) As (c) but 4x learning rate

(e) Batch size 128 v.s. 512

(f) Batch size 128 v.s. 512 4x lr

Figure 5: **Effects of changing batch size and learning rate on $P_{\mathbf{B}}(f|S)$ and $P_{\mathbf{Adam}}(f|S)$ for FCN on MNIST with CE loss** [We use training/test set size 10,000/100. Vertical dotted blue lines denote 90% probability boundary; dashed grey line is $x = y$.]
(a) Batch size = 32, $\langle \epsilon_G \rangle = 1.13\%$.
(b) Batch size= 128, $\langle \epsilon_G \rangle = 2.20\%$.
(c) Batch size = 512, $\langle \epsilon_G \rangle = 2.67\%$.
(d) Batch size =512 and faster learning rate (4x the others), $\langle \epsilon_G \rangle = 2.14\%$.
(e) Direct comparison of $P_{\text{Adam}}(f|S)$ for batch size 128 and 512.
(f) Direct comparison of $P_{\text{Adam}}(f|S)$ for batch size 128 and 512 with a 4× faster learning rate.
The $P_{\text{Adam}}(f|S)$ probabilities for the dominant functions in (d) and (b) are remarkably similar, as can be seen by comparing (e) and (f). It is these higher probability functions that explain the similarity in $\langle \epsilon_G \rangle$ for batch size 128 and batch size 512 with a faster learning rate. See Figure 18 for related batch size results for MSE loss.

Finally, see Figure 20 for experiments with a selection of learning rates (from $0.01-8.0\times$ default) and batch sizes (from 32 to 512), which demonstrate that $P_{\mathrm{OPT}}(f|S)$ remains strongly correlated with $P_{\mathrm{B}}(f|S)$ for the FCN on MNIST, so the correlation is not strongly affected by learning rates within this range.

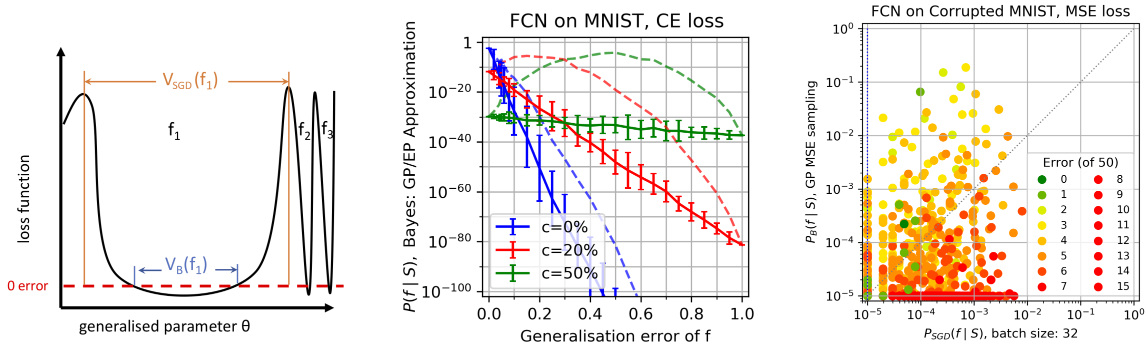
5.2 Changing optimisers

We trained the FCN on MNIST with different optimisers (Adam, Adagrad, RMSprop, Adadelta), and found that to first order $P_{\mathrm{B}}(f|S)$ correlated well with $P_{\mathrm{OPT}}(f|S)$ for all four optimisers. We also observed some second order effects, including that the distribution of $P_{\mathrm{Adam}}(f|S)$ and $P_{\mathrm{Adagrad}}(f|S)$ were very similar to one another, as were $P_{\mathrm{RMSprop}}(f|S)$ and $P_{\mathrm{Adadelta}}(f|S)$, but there was noticeable variation between the two groups. We find that $P_{\mathrm{Adam}}(f|S)$ with batch size of 32 is very similar to $P_{\mathrm{RMSprop}}(f|S)$ with a batch size of 128. The effect of optimiser choice, batch size, learning rate, and other hyperameters is complex, and the parameter space is large. Analysing optimisers in function-space could be a way to better understand the interaction of these choices with the loss landscape, and understanding the effects of hyperparameter tuning. See Appendix C.1 for further detail and the plots.

6. Heuristic arguments for the correlation between $\mathbf{P_B(f|S)}$ and $\mathbf{P_{SGD}(f|S)}$

At first sight it may seem rather surprising that SGD, which follows gradients down a complex loss-landscape, should converge on a function f with a probability anything like the Bayesian posterior $P_{\mathrm{B}}(f|S)$ that upon random sampling of parameters, a DNN generates functions f conditioned on S. Indeed, in the general case of an arbitrary learner we don't expect this correspondence to hold. However, as shown e.g. in Fig 1, $P_{\mathrm{B}}(f|S)$ is orders of magnitude larger for functions with small generalisation error than it is for functions with poor generalisation. As explained in Sections 7.3 and 7.5, such an exponential bias towards low complexity/low error functions can be expected on fairly general grounds (Valle-Pérez et al., 2018; Mingard et al., 2019; Dingle et al., 2018, 2020). If our null expectation is of a large variation in the prior probabilities, then the good correlation can be heuristically justified by a landscape picture (Wales et al., 2003), where $P_{\mathrm{B}}(f|S)$ is interpreted as the "basin volume" $V_B(f)$ (with measure $p_{\mathrm{par}}(\theta)$) of function f), while $P_{\mathrm{SGD}}(f|S)$ is interpreted as the "basin of attraction" $V_{SGD}(f)$, which is loosely defined as a measure of the set of initial parameters θ_i for which the optimiser converges to f with high probability (this concept also found in related form in the dynamical systems literature (Strogatz, 2018)). If $V_B(f)$ varies over many orders of magnitude, then it seems reasonable to expect that $V_{SGD}(f)$ should correlate with $V_B(f)$, as illustrated schematically in Figure 6a. Such general intuitions about landscapes are widely held (Wales et al., 2003; Massen and Doye, 2007; Ballard et al., 2017), and have also been put forward for the particular landscapes of deep learning; see in particular Wu et al. (2017) who also argue that functions with good generalisation have larger basins of attraction.

Another source of intuition follows form a well trodden path linking basic concepts from statistical mechanics to optimisation and learning theory. For example, simple gradient descent (GD) with a small amount of white noise can be described by an over-damped Langevin equation (Welling and Teh, 2011; Smith and Le, 2017; Naveh et al., 2020) that



(a) Schematic loss landscape

(b) Corrupted data, CE loss

(c) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{SGD}}(f|S)$

Figure 6: **Schematic landscape and effects of randomising training labels.** (a) Cartoon of a biased loss-landscape. The three functions f_1, f_2 and f_3 all reach zero classification error (dashed red line), but due to bias in the parameter-function map, the "basin size" $V_B(f_1) \gg V_B(f_2), V_B(f_3)$, which typically implies that for the "basins of attraction" $V_{SGD}(f_1) \gg V_{SGD}(f_2), V_{SGD}(f_3)$. $P_{\mathrm{B}}(f|S)$ is proportional to $V_B(f)$, and $P_{\mathrm{SGD}}(f|S)$ is proportional to $V_{SGD}(f)$. (b) $P_{\mathrm{B}}(f|S)$ (solid) and $\rho(\epsilon_G)P_{\mathrm{B}}(f|S)$ (dashed) v.s. ϵ_G, for test set of size 100 and CE loss (as in Figure 1b) but including label corruption c. (b) $P_{\mathrm{SGD}}(f|S)$ v.s. $P_{\mathrm{B}}(f|S)$ on MNIST with a 2-layer 1024 node wide FCN with MSE loss, test set size 50, and 20% of the training labels randomised ($\langle\epsilon_G\rangle_{SGD} = 13.4\%$ and $\langle\epsilon_G\rangle_{GP} = 5.80\%$). Here functions with frequency < 10 are also shown on the plot. The correlation is much less pronounced than for the unrandomised case shown in Figure 1a. Dots on the axes denote functions found by just one of the two methods. Let F be the set of functions found by both the optimiser and under GP sampling. Then $\sum_{f\in F} P_{\mathrm{B}}(f|S) = 99.3\%$, and $\sum_{f\in F} P_{\mathrm{SGD}}(f|S) = 24.3\%$. In other words, while the Adam optimiser finds almost all functions with high $P_{\mathrm{B}}(f|S)$, it also finds many functions with low $P_{\mathrm{B}}(f|S)$. The much weaker bias under label corruption observed in (b) likely explains the weaker correlation between the Bayesian results and that of the optimiser found here.

converges (under some light further conditions) to the Boltzmann distribution The Boltzmann distribution can, in turn, be interpreted as being equivalent to a Bayesian posterior $P_B(f|S) \propto e^{S(f)-\beta E(f)}$ (MacKay, 2003) where $S(f)$ is configurational "entropy" that counts the number of states that generate f and encodes the prior, and $E(f)$ represents the energy, encoding the log likelihood or loss function. For SGD the equivalent coarse-grained differential equation reduces to Langevin equation with anisotropic noise (Smith and Le, 2017; Zhang et al., 2018) and doesn't exactly converge to the Bayesian posterior (Mandt et al., 2017; Brosse et al., 2018). Nevertheless, it has been conjectured that with small step size, SGD may approximate the Bayesian posterior (Naveh et al., 2020; Cohen et al., 2019), as we empirically find in our experiments. These connections are rich and worth exploring further in this context. Nevertheless, some caution is needed with these analogies to statistical mechanics because they depend on assumptions which may only to hold on prohibitively long time-scales.

A better analogy may be to the "arrival of the frequent" phenomenon in evolutionary dynamics (Schaper and Louis, 2014), which, like the "basin of attraction" arguments, does not require steady state. Instead it predicts which structures are likely to be *reached first* by an evolutionary process. For RNA secondary structures, for example, it predicts that a stochastic evolutionary process will reach structures with a probability that to first order is proportional to the likelihood that uniform random sampling of genotypes produces the structure. Indeed, this phenomenon – where the probability upon random sampling predicts the outcomes of a complex search process – can be observed in naturally occurring RNA (Dingle et al., 2015), the result of evolutionary dynamics. This type of non-equilibrium analysis may be more relevant for the way we train most of the DNNs in this paper, since we stop the first time 0 training error is reached. The analogy between these evolutionary results with what we observe for SGD is intriguing, but needs further exploration.

To illustrate the effect of the amount of bias in the posterior, we randomise labels for MNIST and calculate the $P_{\mathrm{B}}(f|S)$. As we can see in Figure 6b, this results in a less strongly biased posterior. The mean log-probability $\langle\log(P_{\mathrm{B}}(f|S))\rangle$ v.s. ϵ_G curve becomes less steep with increasing corruption For a relatively small fraction of low error functions to dominate, as they do for zero corruptions in Figure 1a, the bias must be strong enough here to overcome the "entropic" factor $\rho(\epsilon_G)$. For the 20% and 50% corruption this is clearly not the case, and a huge number of functions with larger error will dominate $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{SGD}}(f|S)$. As can be seen in Figure 6c, one effect of weaker bias is that the correlation between the optimiser and the Bayesian sampling is much less strong. This behaviour is consistent with the heuristic arguments above, which should only work if the differences in basin volumes are large enough to overcome the myriad other factors that can affect $P_{\mathrm{OPT}}(f|S)$.

7. Related work on inductive bias on neural networks

In this section we summarise some key aspects of the literature related to why DNNs exhibit good generalisation while overparameterised, expanding on some briefer remarks in Section 1.

7.1 The link between inductive bias and generalisation

Much of the work on inductive biases in stochastic gradient descent (SGD) is framed as a discussion about generalisation. The two concepts are of course intimately related. Before discussing related work on inductive bias DNNs, it may be helpful to distinguish two different questions about generalisation:

1) **Question of over-parameterised generalisation**: Why do DNNs generalise at all in the overparameterised regime, where classical learning theory doesn't guarantee generalisation?

2) **Question of fine-tuned generalisation**: Given that vanilla DNNs already generalise reasonably well, how can architecture choice and hyperparameter tuning further improve generalisation?

The first question arises because among the functions that an overparameterised DNN can express, the number that can fit a training data set S, but generalise poorly, is typically many orders of magnitude larger than the number that achieve good generalisation. From classical learning theory we would therefore expect extremely poor generalisation. However, in practice it is often found that many DNN architectures, as long as they are expressive enough to fit the data, generalise sufficiently well to imply a significant inductive bias towards a small fraction of functions that generalise well.

This question is also related to the conundrum of why DNNs avoid the "curse of dimensionality", which relates to the poor generalisation that certain highly expressive non-parametric models have in high dimensions (Donoho et al., 2000). Valle-Pérez et al. (2018) argue that the curse of dimensionality is linked to a prior which is not sufficiently biased and that DNNs may avoid this problem by virtue of the strong bias in the prior.

The second question arises from two common experiences in DNN research. Firstly, changes in architecture can lead to important improvements in generalisation. For example, a CNN with max-pooling typically performs better than a vanilla FCN on image data. Secondly, hyperparameter tuning within a fixed architecture can lead to further improvements of generalisation. While these methods of improving generalisation are important in practice, the starting point is normally a DNN that already has enough inductive bias to raise question 1) above. It is therefore important not to conflate the study of question 2) – as vital as this may be to successful practical implementations — with the more general question of why DNNs generalise in the first place.

7.2 Related work on implicit bias in optimiser-trained networks

As mentioned in the introduction, there is an extensive literature on inductive biases in SGD. Much of this literature is empirical: improvements are observed when using particular tuned hyperparameters with variants of SGD. One of the most common rationalisation is in terms of "flatness" which is inspired by early work (Hochreiter and Schmidhuber, 1997a) who predicted that flatter minima would generalise better. Flatness is often measured using some combination of the eigenvalues of the Hessian matrix for a trained DNN. (Keskar et al., 2016) showed that DNNs trained with small batch SGD generalise better than identical models trained with large batch SGD (by up to 5%), and also found a correlation between small batch size and minima that are less "sharp" (using not the eigenvalues of the Hessian but a more computationally tractable sensitivity measure). While these results are genuinely interesting, they are mainly relevant to issues raised by question 2 above. For example in (Keskar et al., 2016) the authors explicitly point out that their results are not about “overfitting” (e.g. question 1 above).

The effects of changing hyperparameters can be subtle. For example, another series of recent papers (Goyal et al., 2017; Hoffer et al., 2017; Smith et al., 2017) suggest that better generalisation with small batch SGD may be caused by the fact that the number of optimisation steps per epoch decreases when the batch size increases. These studies showed that a similar improvement in generalisation performance to that found by reducing batch size can be created by increasing the learning rate, or by overtraining (i.e. by continuing to train after 100% accuracy has been reached). In particular, in (Hoffer et al., 2017) it was argued that overtraining does not generally negatively impact generalisation, as naive expectations based on overfitting might suggest. These results also challenge some theoretical studies that suggested that SGD may control the capacity of the models by limiting the number of parameter updates (Brutzkus et al., 2017).

In another interesting paper, Zhang et al. (2018) derive a Langevin type equation for both SGD. And argue that in contrast to GD, the noise is anisotropic, and that this may explain why SGD is more likely to find “flatter minima”. Similarly, Jastrzebski et al. (2018) argue that isotropic SGD-induced noise also helps push the optimiser away from sharper minima. An important caveat to the work on sharpness can be found in the work of Dinh *et al.* (Dinh et al., 2017) who use the non-negative homogeneity of the ReLU activation function to show that for a number of the measures used in the papers cited above, the “flatness” can be made arbitrarily large (or sharp) without changing the function (and therefore the generalisation performance) that the DNN expresses. This result suggests that care must be used when interpreting local measures of flatness. Finally in this vein, generalisation has also been linked to related concepts including low frequency (Rahaman et al., 2018), and to sensitivity to changes in the inputs (Arpit et al., 2017; Novak et al., 2018a).

There is much more literature on SGD induced inductive bias, but the upshot is that while fine-tuning optimiser hyperparameters can be very important for improving generalisation, and by implication, the inductive bias of a DNN, a complete understanding remains elusive. Moreover, where improvements are found, these tend to be in the class of answers to question 2) above. An important example of a paper on flatness that does explicitly address question 1 above is (Wu et al., 2017), who show that generalisation trends for data with different levels of corruption correlates with the log of the product of the top 50 eigenvalues of the Hessian both for SGD and for GD trained networks. By heuristically linking their local flatness measure to the global basin volume, they make a very similar argument to the one we flesh out in more detail here, namely that the basin of attraction volume of “good” solutions is much larger than that of “bad” solutions that do not generalise well.

Significant theoretical effort has been spent on extracting properties of a trained neural network that could be used to explain generalisation. By implication, these investigations should also help illuminate the nature of the implicit bias of trained networks. For example, investigators have attempted to use sensitivity to perturbations (whether in inputs or weights) to explain the generalisation performance either using a PAC-Bayesian analysis (Bartlett et al., 2017; Dziugaite and Roy, 2017; Neyshabur et al., 2018), or a compression approach (Arora et al., 2018; Zhou et al., 2019). In contrast to the work described above that studies the specific effect of hyperparameter tuning on SGD, much of the work listed in this paragraph is directly applicable to question 1. A very comprehensive review of this line of work empirically finds that the PAC-Bayesian sensitivity approaches seem the most promising (Jiang et al., 2019), but no clear answer to the question 1 has emerged.

The more theoretical side of the study of SGD has also seen recent progress. For example, (Soudry et al., 2018) showed that SGD finds the max-margin solution in unregularised logistic regression, whilst it was shown in (Brutzkus et al., 2017) that overparameterised DNNs trained with SGD avoid over-fitting on linearly separable data. More recently, (Allen-Zhu et al., 2019) proved agnostic generalisation bounds for SGD-trained DNNs (up to three layers), which impose less restrictive assumptions (on the data, architecture, and optimiser) than previous works. Such theoretical analyses may be a potentially fruitful source of new ideas to explain generalisation.

Another interesting direction is to investigate properties of the loss-landscape itself. Several studies have shown interesting parallels between the loss landscape of DNNs and the energy landscape of spin glasses (Choromanska et al., 2015; Baity-Jesi et al., 2019; Becker et al., 2020). While such insights may help explain why SGD works so well as an optimiser in these high dimensional spaces, it is at present less clear how these studies help explain question 1) above.

A completely different theme builds on the concept of an information bottleneck (Tishby and Zaslavsky, 2015; Shwartz-Ziv and Tishby, 2017) which suggest that generalisation arises from information compression in deeper layers, aided by SGD. However, recent work (Saxe et al., 2019) suggests that the compression is strongly affected by activation functions used, suggesting again that this approach is not general enough to capture the implicit bias needed to answer question 1. We note that the debate about this theme is ongoing.

Finally, it is important to note that simple vanilla gradient descent (GD), when it can be made to converge, does not differ that much (on the scale of question 1 above) from SGD and its variants in generalisation performance (Keskar et al., 2016; Wu et al., 2017; Zhang et al., 2018; Choi et al., 2019). Therefore if training with an optimiser itself generates the inductive bias needed to answer question 1, that bias must already largely be present in simple GD.

7.3 Related work on implicit bias in random neural networks

We briefly review work inspired by a powerful result from algorithmic information theory (AIT) called the coding theorem (Li and Vitanyi, 2008). First derived by Levin (Levin, 1974), and building on concepts pioneered by Solomonoff (Solomonoff, 1964), it is closely related to more recent bound applicable to a wider range of input-output maps (Dingle et al., 2018, 2020). This bound predicts (under certain fairly general conditions that the maps must fulfil) that upon randomly sampling the parameters of an input-output map M, the probability $P(f)$ of obtaining output f can be bounded as

$$P(f) \leq 2^{-K(f|M)+\mathcal{O}(1)} \approx 2^{-a\tilde{K}(f)+b} \tag{4}$$

where $K(f)$ is the Kolmogorov complexity of f, the $\mathcal{O}(1)$ terms do not depend on the outputs (at least asymptotically), $\tilde{K}(f)$ is a suitable approximation to $K(f)$ and a and b are parameters that depend on the map, but not on f. The computable bound was empirically shown to work remarkably well for a wide range of input-output maps from across science and engineering (Dingle et al., 2018), giving confidence that it should be widely applicable, at least for maps that satisfy the conditions needed for it to apply. In addition, a statistical lower-bound can be derived that predicts that most of the probability weight will lie relatively close to the bound (Dingle et al., 2020).

The application of this bound to DNNs was first shown in (Valle-Pérez et al., 2018). We note that the input-output map of interest is not the map from inputs to DNN outputs, but rather the map from the network parameters to the function f it produces on inputs $\mathcal{X}$ which was described in Theorem 1. The prediction of Equation (4) for a DNN with parameters sampled randomly (from, for example, truncated i.i.d. Gaussians) is that, if the parameter-function map is sufficiently biased, then the probability of the DNN producing a function f on input data $x_{i=0}^{n}$ drops exponentially with increasing complexity of the function f. Note that technically we should write f as $f|\mathcal{X}$ to indicate the dependence of the function modelled by the DNN on the inputs $\mathcal{X}$. We also note that the AIT bound of Equation (4) on its own does not force a map to be biased. It still holds for a uniform distribution. But if the map is biased, then it will be biased according to Equation (4).

In (Valle-Pérez et al., 2018) it was shown empirically that this very general prediction of Equation (4) holds for the $P(f)$ of a number of different DNNs. This testing was achieved both via direct sampling of the parameters of a small DNN on Boolean inputs and with NNGP calculations for more complex systems. In a complementary approach (Mingard et al., 2019) some exact results were proven for simplified networks, that are also consistent with the bound of Equation (4). In particular, they proved that for a perceptron with no bias term, upon randomly sampling the parameters (with a distribution satisfying certain weak assumptions), any value of class-imbalance was equally likely. There are many fewer functions with high class imbalance (low “entropy”) than low class imbalance. Low entropy implies low $K(f)$ (but not the other way around). Thus, these results imply a bias of $P(f)$ towards certain simple functions. They also proved that for infinite-width ReLU DNNs, this bias becomes monotonically stronger as the number of layers grows. A different direction was pursued in (De Palma et al., 2018), who showed that, upon randomly sampling the parameters of a ReLU DNN acting on Boolean inputs, the functions obtained had an average sensitivity to inputs which is much lower than if randomly sampling functions. Functions with low input sensitivity are also simple, thus proving another manifestation of simplicity bias present in these systems.

On the other hand, in a recent paper (Yang and Salman, 2019), it was shown that for DNNs with activation functions such as *Erf* and *Tanh*, the bias starts to disappear as the system enters the “chaotic regime”, which happens for weight variances above a certain threshold, as the depth grows (Poole et al., 2016) (note that ReLU networks don’t have such a chaotic regime). While these hyperparameters are not typically used for DNNs, they do show that there exist regimes where there is no simplicity bias. Note that the AIT coding theorem bound Equation (4) still holds, but $P(f)$ is simply approaching a uniform distribution, and the bound becomes loose for small complexity. These results are also interesting because, if the bias becomes weaker, then it may also be the case that the correlation between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{SGD}}(f|S)$ starts to disappear, an effect we are currently investigating.

7.4 Related work comparing optimiser-trained and Bayesian neural networks

Another set of investigations studying random neural networks use important recent extensions of Neal's seminal proof (Neal, 1994, 2012) – that a single-layer DNN with random i.i.d. weights is equivalent to a Gaussian process (GP) (Mackay, 1998) in the infinite width limit – to multiple layers and architectures (Lee et al., 2017; Matthews et al., 2018; Novak et al., 2018b; Garriga-Alonso et al., 2019; Yang, 2019b). These studies have used this correspondence to effectively perform a very good approximation to exact Bayesian inference in DNNs. When they have compared them to SGD-trained DNNs (Lee et al., 2017; Matthews et al., 2018; Novak et al., 2018b), the results have generally shown a close agreement between the generalisation performance of optimiser-trained DNNs and their corresponding Bayesian neural network Gaussian process (NNGP).

In this context another significant development is the introduction of the neural tangent kernel (NTK) (Jacot et al., 2018) which approximates the dynamics of an infinite width DNN with parameters that are trained by gradient descent in the limit of an infinitesimal learning rate. Recent comparisons to NNGPs show relatively similar performance of the NTK, see for example (Arora et al., 2019; Lee et al., 2019; Novak et al., 2020). While there are small performance differences, the overall agreement between NNGPs and the NTK or optimiser trained DNNs is close enough to suggest that the primary source of inductive bias needed for question 1 above is already present in the untrained network, and is essentially maintained under training dynamics.

The linearisation of DNNs offered by NTK can also be used to prove that, in this regime, GD samples from the Bayesian posterior in a sample-then-optimise fashion. For linear regression models, Matthews et al. (2017) showed that solutions after training GD with a Gaussian initialisation correspond to exact posterior samples. This idea is also related to Deep Ensembles which has been proposed to be "approximately Bayesian" in Wilson and Izmailov (2020).

In this context, further indirect evidence comes from Valle-Pérez et al. (2018) who used a simple PAC-Bayesian bound (McAllester, 1999) that applies to exact Bayesian inference, to predict the generalisation error of SGD-trained DNNs. The bound was shown to provide relatively tight predictions for optimiser-trained DNNs for an FCN and CNNs on MNIST, Fashion-MNIST and CIFAR-10. Moreover, this bound, which takes the Bayesian marginal likelihood as input, reproduced trends such as the increase in the generalisation error upon an increased fraction of randomised labels.

These lines of work serve as independent evidence to suggest that optimiser-trained DNNs behave very similarly to the same DNNs trained with Bayesian inference, and helped inspire the work in this paper, where we directly tackle this question. These studies also suggest that the infinite-width limit may be enough to answer question 1, as the number of parameters in a DNN typically doesn't have a drastic effect on generalisation (as long as the network is expressive enough to fit the data).

7.5 Related work on complexity of data, simplicity bias and generalisation

In Section 7.3, we discussed work showing that DNNs may have an inductive bias towards simple functions in their parameter-function map. Here, we briefly discuss how this "simplicity bias" concept may connect to generalisation. As implied by the no free lunch theorem (Wolpert and Waters, 1994), a bias towards simplicity does not automatically imply good generalisation. Instead certain key hypotheses about the data are needed, in particular that it is described by functions that are simple (in a similar sense to the inductive bias). Now the assumption that a more parsimonious hypothesis is more likely to be true has been influential since antiquity and is often articulated by invoking Occam's razor. However, the fundamental justification for this heuristic is disputed, see e.g. (Sober, 2015) for an overview of the philosophical literature, e.g. (MacKay, 1992; Blumer et al., 1987; Rasmussen and Ghahramani, 2001; Domingos, 1999) for a set of different perspectives from the machine learning literature, and e.g. (Rathmanner and Hutter, 2011; Sterkenburg, 2016) for a spirited discussion of the links between the razor and concepts from AIT (pioneered in particular by Solomonoff).

Studies which imply that data typically studied with DNNs is somehow "simple" include an influential paper (Lin et al., 2017) invoking arguments, mainly from statistical mechanics, to argue that deep learning works well because the laws of physics typically select for function classes that are "mathematically simple", and so easy to learn. More direct studies have also demonstrated certain types of simplicity. For example, following on previous work in this vein, (Spigler et al., 2019) calculated an effective dimension $d_{eff} \approx 15$ for MNIST, which is much lower than the $28^2 = 784$ dimensional manifold in which the data is embedded. Individual numbers can have effective dimensions that are even lower, ranging from 7 to 13 (Hein and Audibert, 2005). So the functions that fit MNIST data are much simpler than those that fit random data (Goldt et al., 2019). An implicit bias towards simplicity may therefore improve generalisation for structured data, but it will likely have the opposite effect for more random data.

8. Discussion

We argue here that the inductive bias found in DNNs trained by SGD or related optimisers, is, to first order, determined by the parameter-function map of an untrained DNN. While on a log scale we find $P_{\mathrm{SGD}}(f|S) \approx P_{\mathrm{B}}(f|S)$ there are also measurable second order deviations that are sensitive to hyperparameter tuning and optimiser choice.

For the conundrum of why DNNs generalise at all in the overparameterised regime, our results strongly suggest that the solution must be found in the properties of $P_{\mathrm{B}}(f|S)$, and not in further biases introduced by SGD. Arguments that DNN priors are exponentially biased towards simple functions (Valle-Pérez et al., 2018; Mingard et al., 2019; De Palma et al., 2018) may help explain the inductive bias of $P_{\mathrm{B}}(f|S)$, but more work needs to be done to explore the complex interplay between bias in the prior, the data, and generalisation. While they may not explain the fundamental conundrum above, second order deviations from $P_{\mathrm{B}}(f|S)$ are important in practice for further fine-tuning the generalisation performance.

Our function probability perspective also provides more fine-grained tools for the analysis of DNNs than simply comparing the average test error. This picture can facilitate the investigation of hyperparameter changes, or potentially also the study of techniques such as batch normalisation or dropout. It could assist in the design of new architectures or optimisers.

It is not obvious how to determine the uncertainty in a prediction of a DNN model. However, if, as we argue here, SGD behaves like a Bayesian sampler, then this offers additional justification for using Deep Ensembles to measure this uncertainty in the case of DNNs (Wilson and Izmailov, 2020). Our results could therefore make it easier to use neural networks in applications where it is important to be able to quantify prediction uncertainty

Most of our examples are for image classification. It would be interesting to study the related problem of using DNNs for regression. Sampling considerations means that it is easier to study $P_{\text{SGD}}(f|S)$ for smaller generalisation errors. It would be interesting to study systems with intrinsically larger $\langle \epsilon_G \rangle$ within this picture as well. There the biasing effect of the optimiser may be larger.

Finally, to study the correlation between $P_{\text{B}}(f|S)$ and $P_{\text{SGD}}(f|S)$, we mainly used a fixed test and training set. While we did examine other test and training sets (see Appendices), this was mainly to confirm that our results were not an artefact of our particular choices. A promising future direction would be a Bayesian approach that includes averaging over training sets.

Acknowledgements

We would like to thank two anonymous referees for constructive comments that helped up improve the paper.

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6158–6169, 2019.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 254–263. PMLR, 10–15 Jul 2018. URL `http://proceedings.mlr.press/v80/arora18b.html`.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8139–8148, 2019.

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.

Marco Baity-Jesi, Levent Sagun, Mario Geiger, Stefano Spigler, Gérard Ben Arous, Chiara Cammarota, Yann LeCun, Matthieu Wyart, and Giulio Biroli. Comparing dynamics: Deep neural networks versus glassy systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124013, 2019.

Andrew J Ballard, Ritankar Das, Stefano Martiniani, Dhagash Mehta, Levent Sagun, Jacob D Stevenson, and David J Wales. Energy landscapes for machine learning. *Physical Chemistry Chemical Physics*, 19(20):12585–12603, 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*, pages 6240–6249, 2017.

Simon Becker, Yao Zhang, et al. Geometry of energy landscapes and the optimizability of deep neural networks. *Physical Review Letters*, 124(10):108301, 2020.

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

Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Nicolas Brosse, Alain Durmus, and Eric Moulines. The promises and pitfalls of stochastic gradient langevin dynamics. In *Advances in Neural Information Processing Systems*, pages 8268–8278, 2018.

Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*, 2017.

Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison, and George E Dahl. On empirical comparisons of optimizers for deep learning. *arXiv preprint arXiv:1910.05446*, 2019.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204, 2015.

Omry Cohen, Or Malka, and Zohar Ringel. Learning curves for deep neural networks: a gaussian field theory perspective. *arXiv preprint arXiv:1906.05301*, 2019.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Assaf Dauber, Meir Feder, Tomer Koren, and Roi Livni. Can implicit bias explain generalization? stochastic convex optimization as a case study. *arXiv preprint arXiv:2003.06152*, 2020.

Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=H1-nGgWC-`.

Giacomo De Palma, Bobak Toussi Kiani, and Seth Lloyd. Random deep neural networks are biased towards simple functions. *arXiv preprint arXiv:1812.10156*, 2018.

Kamaludin Dingle, Steffen Schaper, and Ard A Louis. The structure of the genotype–phenotype map strongly constrains the evolution of non-coding rna. *Interface focus*, 5(6): 20150053, 2015.

Kamaludin Dingle, Chico Q Camargo, and Ard A Louis. Input–output maps are strongly biased towards simple outputs. *Nature Communications*, 9(1):1–7, 2018.

Kamaludin Dingle, Guillermo Valle Pérez, and Ard A Louis. Generic predictions of output probability based on complexities of inputs and outputs. *Scientific Reports*, 10(1):1–9, 2020.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org, 2017.

Pedro Domingos. The role of occam's razor in knowledge discovery. *Data mining and knowledge discovery*, 3(4):409–425, 1999.

David L Donoho et al. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS math challenges lecture*, 1(2000):32, 2000.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul): 2121–2159, 2011.

Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*, 2017. URL http://auai.org/uai2017/proceedings/papers/173.pdf.

Adrià Garriga-Alonso, Carl Edward Rasmussen, and Laurence Aitchison. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bklfsi0cKm.

Sebastian Goldt, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. Modelling the influence of data structure on learning in neural networks. *arXiv preprint arXiv:1909.11500*, 2019.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Boris Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 7(10):992, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Matthias Hein and Jean-Yves Audibert. Intrinsic dimensionality estimation of submanifolds in rd. In *Proceedings of the 22nd international conference on Machine learning*, pages 289–296, 2005.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

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

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997b.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pages 1731–1741, 2017.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos J Storkey. Finding flatter minima with sgd. In *ICLR (Workshop)*, 2018.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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. *CoRR*, abs/1609.04836, 2016. URL `http://arxiv.org/abs/1609.04836`.

Diederik P Kingma and Jimmy Lei Ba. Adam: Amethod for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*, 2014.

David Krueger, Nicolas Ballas, Stanislaw Jastrzebski, Devansh Arpit, Maxinder S Kanwal, Tegan Maharaj, Emmanuel Bengio, Asja Fischer, and Aaron Courville. Deep nets don't learn via memorization. *International Conference on Learning Representations Workshop*, 2017. URL `https://openreview.net/forum?id=HJC2SzZCW`.

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.

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.

Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1EA-M-0Z`.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pages 8572–8583, 2019.

L.A. Levin. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Problemy Peredachi Informatsii*, 10(3):30–35, 1974.

M. Li and P.M.B. Vitanyi. *An introduction to Kolmogorov complexity and its applications*. Springer-Verlag New York Inc, 2008.

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.

David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

David JC Mackay. Introduction to gaussian processes. *NATO ASI series. Series F: computer and system sciences*, pages 133–165, 1998.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

Claire P Massen and Jonathan PK Doye. Power-law distributions for the areas of the basins of attraction on a potential energy landscape. *Physical Review E*, 75(3):037101, 2007.

Alexander G de G Matthews, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Sample-then-optimize posterior sampling for bayesian linear models. In *NIPS Workshop on Advances in Approximate Bayesian Inference*, 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. Pac-bayesian model averaging. In *Proceedings of the twelfth annual conference on Computational learning theory*, pages 164–170, 1999.

Chris Mingard, Joar Skalse, Guillermo Valle-Pérez, David Martínez-Rubio, Vladimir Mikulik, and Ard A Louis. Neural networks are a priori biased towards boolean functions with low entropy. *arXiv preprint arXiv:1909.11522*, 2019.

Gadi Naveh, Oded Ben-David, Haim Sompolinsky, and Zohar Ringel. Predicting the outputs of finite networks trained with noisy gradients. *arXiv preprint arXiv:2004.01190*, 2020.

Radford M Neal. Priors for infinite networks (tech. rep. no. crg-tr-94-1). *University of Toronto*, 1994.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Skz_WfbCZ`.

Roman Novak, Yasaman Bahri, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *International Conference on Learning Representations*, 2018a. URL `https://openreview.net/forum?id=HJC2SzZCW`.

Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Bayesian convolutional neural networks with many channels are gaussian processes. *arXiv preprint arXiv:1810.05148*, 2018b.

Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SklD9yrFPS`.

Tomaso Poggio, Qianli Liao, and Andrzej Banburski. Complexity control by gradient descent in deep networks. *Nature Communications*, 11(1):1–5, 2020.

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*, pages 3360–3368, 2016.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. *arXiv preprint arXiv:1806.08734*, 2018.

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

Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. In *Advances in neural information processing systems*, pages 294–300, 2001.

Samuel Rathmanner and Marcus Hutter. A philosophical treatise of universal induction. *Entropy*, 13(6):1076–1136, 2011.

Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. *arXiv preprint arXiv:1909.12673*, 2019.

Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

Steffen Schaper and Ard A Louis. The arrival of the frequent: how bias in genotype-phenotype maps can steer populations to local optima. *PloS one*, 9(2):e86635, 2014.

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

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

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Vincent G Sigillito, Simon P Wing, Larrie V Hutton, and Kile B Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10(3):262–266, 1989.

Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. *CoRR*, abs/1710.06451, 2017. URL `http://arxiv.org/abs/1710.06451`.

Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

Elliott Sober. *Ockham's razors*. Cambridge University Press, 2015.

Ray J Solomonoff. A formal theory of inductive inference. part i. *Information and control*, 7 (1):1–22, 1964.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Stefano Spigler, Mario Geiger, and Matthieu Wyart. Asymptotic learning curves of kernel methods: empirical data vs teacher-student paradigm. *arXiv preprint arXiv:1905.10843*, 2019.

Tom F Sterkenburg. Solomonoff prediction and occam's razor. *Philosophy of Science*, 83(4): 459–479, 2016.

Steven H Strogatz. *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015.

Guillermo Valle-Pérez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *arXiv preprint arXiv:1805.08522*, 2018.

David Wales et al. *Energy landscapes: Applications to clusters, biomolecules and glasses*. Cambridge University Press, 2003.

Mingwei Wei and David J Schwab. How noise affects the hessian spectrum in overparameterized neural networks. *arXiv preprint arXiv:1910.00195*, 2019.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *arXiv preprint arXiv:2002.08791*, 2020.

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.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL `http://arxiv.org/abs/1708.07747`.

Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019a.

Greg Yang. Wide feedforward or recurrent neural networks of any architecture are gaussian processes. In *Advances in Neural Information Processing Systems*, pages 9951–9960, 2019b.

Greg Yang and Hadi Salman. A fine-grained spectral perspective on neural networks. *arXiv preprint arXiv:1907.10599*, 2019.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

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

Yao Zhang, Andrew M Saxe, Madhu S Advani, and Alpha A Lee. Energy–entropy competition and the effectiveness of stochastic gradient descent in machine learning. *Molecular Physics*, 116(21-22):3214–3223, 2018.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P. Adams, and Peter Orbanz. Non-vacuous generalization bounds at the imagenet scale: a PAC-bayesian compression approach. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJgqqsAct7`.

Appendix A. Further detail for methods (Section 3)

In this Appendix we provide further details and explanation of the methodology outlined in Section 3. In particular, we discuss our experiments in Appendix A.1 and the GP approximation for $P_{\mathrm{B}}(f|S)$ in Appendix A.2.

A.1 Methodology in detail

For each experiment performed in the main text, we pick a DNN $\mathcal{N}$ (either FCN, CNN, or an LSTM) and a dataset $\mathcal{D}$ (MNIST, Fashion-MNIST or the IMDb dataset). We also pick a fixed training set $S \subset \mathcal{D}$, and a fixed test set $E \subset \mathcal{D}$. Training sets are typically of size $10,000$ for FCN and CNN and $45,000$ for the LSTM. Test sets are typically small, 100 for the FCN and CNN, and 50 for the LSTM.

A.1.1 Using an optimiser to calculate $\mathbf{P}_{\mathrm{OPT}}(\mathbf{f}|\mathbf{S})$

When calculating $P_{\mathrm{OPT}}(f|S)$ we first pick an optimiser OPT which is either plain SGD, or one of its derivatives: Adam, Adagrad, RMSprop, or Adadelta. Next we pick a loss-function, either mean-square error (MSE) or cross-entropy (CE). We also need to pick an initial parameter distribution $\tilde{P}_{par}(\theta)$ which we take to be from a truncated i.i.d. Gaussian distribution (the distribution from which the DNN $\mathcal{N}$ is randomly initialised, see Equation (3)).

Algorithm 1 Calculating $P_{\mathrm{OPT}}(f|S)$

```
input: DNN $\mathcal{N}$, training data $S$, test data $E$, optimiser $OPT$.
$F \leftarrow \langle\rangle$                {the ‘functions’ found during training}
do $n$ times:
    re-initialise the weights of $\mathcal{N}$ from an i.i.d. Gaussian distribution
    train $\mathcal{N}$ on $S$ until it reaches 100 % training accuracy
    record the classification of $\mathcal{N}$ on $E$ and save it to $F$
$A \leftarrow \varnothing$                {the frequency and ‘volume’ of each ‘function’}
for each distinct $f \in F$ do
    let $\rho_f$ be the frequency of $f$ in $F$
    calculate the probability $P_{\mathrm{OPT}}(f|S) = \rho_f/n$ of $f$ in $F$
    save $P_{\mathrm{OPT}}(f|S)$ to $A$
end for
return $A$
```

For fully-connected layers we used $\sigma_b = 0$ and $\sigma_w = 1/\sqrt{w}$ where w is the width of the layer. For convolutional and LSTM layers we used the default initialisation provided by Keras 2.3.0[11]. This specifies $\tilde{P}_{par}(\theta)$. As we see in Algorithm 1, we then sample n times from $\tilde{P}_{par}(\theta)$, training each time with OPT until the training error on S is zero, at which point we record the function by what errors it makes on E. Note that if for some reason SGD does not converge, we don't count that run in order to have normalised distributions over functions. We typically run between $n = 10^4$ and $n = 10^7$ runs (depending on the system).

[11] CNN: `https://keras.io/api/layers/convolution_layers/convolution2d/`
LSTM: `https://keras.io/api/layers/recurrent_layers/lstm/`

Once the runs are finished, we compile all the empirical frequencies for the functions that are found.

A.1.2 Bayesian sampling for $\mathbf{P}_{\mathrm{B}}(\mathbf{f}|\mathbf{S})$

We use the GP approximation to estimate the Bayesian posterior $P_{\mathrm{B}}(f|S)$. Here we follow (Valle-Pérez et al., 2018) where this technique is explained (see also Appendix A.2 for more details). We need to define a distribution $P_{par}(\theta)$ (the definition of the prior from which we calculate $P_{\mathrm{B}}(f|S)$, see Equation (1)). While there are some subtleties in how the prior distribution $P_{par}(\theta)$ relates to the initialisation distribution $\tilde{P}_{par}(\theta)$, we took a simple approach and defined P_{par} to be the same as the corresponding $\tilde{P}_{par}(\theta)$, except we set σ_b to be a small constant, typically $0.1 \times \sigma_w$.

To estimate $P_{\mathrm{B}}(f|S)$, there is a small compromise that must be made here. For a number of reasons, MSE loss is less popular for the kinds of classification problems we mainly study in this paper. We also find that it typically takes significantly longer to train using an optimiser so that $P_{\mathrm{OPT}}(f|S)$ is more expensive to evaluate for MSE loss on the problems we study. On the other hand, $P_{\mathrm{B}}(f|S)$ can be directly sampled n times from the exact posterior (described in Appendix A.2.1) using Algorithm 2), and so is relatively accurate and simple to evaluate.

For CE loss, which is more frequently used for classification, and is also typically quicker to train than MSE for SGD and its variants, we need to use a further approximation. Here we follow (Valle-Pérez et al., 2018) and use the expectation-propagation (EP) approximation for $P_{\mathrm{B}}(f|S)$. We then estimate the posterior log probabilities using the estimations of the log marginal likelihoods $\log P(S)$ (see Appendix A.2.2 for explanation) in Algorithm 3, or we sample from the approximate EP posterior and use Algorithm 2. These two methods give very similar answers (Figure 7c).

Algorithm 2 Calculating $P_{\mathrm{B}}(f|S)$ (via sampling)

```
input: DNN $\mathcal{N}$, training data $S$, test data $E$.
$F \leftarrow \varnothing$          {functions sampled from the GP or GP/EP posterior}
do $n$ times:
    sample a function $f$ from the GP or GP/EP posterior when conditioning on $S$
    find $f$ on $E$
    save $f$ to $F$
$\mathfrak{R} \leftarrow \varnothing$          {function probabilities}
for each distinct $f \in F$ do
    let $\rho_f$ be the frequency of $f$ in $F$
    calculate $P_{\mathrm{B}}(f|S) = \rho_f/n$
    save $P_{\mathrm{B}}(f|S)$ to $\mathfrak{R}$
end for
return $\mathfrak{R}$
```

Algorithm 3 Calculating $P_{\mathrm{B}}(f|S)$ for specific f via the ratio of likelihoods approximation

```
input: DNN $\mathcal{N}$, training data $S$, test data $E$, optimiser $OPT$, set of functions $F$ (from Algorithm 1)
for each distinct $f \in F$ do
    let $\rho_f$ be the frequency of $f$ in $F$
    use GP or GP/EP approximation to estimate $P_{\mathrm{B}}(f|S)$ of $f$ using the ratio of likelihoods approximation.
    save $P_{\mathrm{B}}(f|S)$ to $A$
end for
return $A$
```

A.1.3 Calculating $P_{\mathrm{B}}(f|S)$ for functions with a wider range of ϵ_G

Given a training dataset S and test dataset E we can generate a random sample of different partial functions with varying levels of error on E (by taking the test set classification and corrupting some percentage of labels). We can then use the GP/EP approximation to estimate $P_{\mathrm{B}}(f|S)$. We typically sample 20 examples for each number of errors. The averages are taken on the logs of the probabilities, and error bars on plots are 2σ, where σ is the standard deviation. Note that the vast majority of functions have such small probabilities, that it is not feasible to estimate their $P_{\mathrm{OPT}}(f|S)$ (nor use Algorithm 2)

While the experiments will be informative for how the space is biased, it does not guarantee that all high-probability functions will be found. Since these functions affect generalisation the most, we rely on the results from Algorithm 2 to check that there are no high-probability functions that are missed.

Algorithm 4 Calculating $P_{\mathrm{B}}(f|S)$ for larger range of ϵ_G

```
input: DNN $\mathcal{N}$, training data $S$, test data $E$.
for $\epsilon \in \{0.0, 0.5, \ldots 1.0\}$ do
    $V_\epsilon \leftarrow \langle\rangle$
    generate classification $c$ with error $\epsilon$ on $E$ (by randomly choosing $|E| \times \epsilon$ distinct labels in the correct function (restricted to the test set) to switch to incorrect).
    use GP/EP and "ratio of likelihoods" approximation to estimate the $P_{\mathrm{B}}(f|S)$ of $c$
    save the relative volume $P_{\mathrm{B}}(f|S)$ of $f$ to $V_\epsilon$
end for
return $V_{0.0} \ldots V_{1.0}$
```

We here make a few more remarks for interpreting the results in experiments where we generate functions for a wide range of errors, such as those in Figure 1b. Firstly, as shown in Figure 8a in Appendix B, there can be a wide variation in the probabilities p_i of misclassifying each of the 100 images in this test set. The generalisation error is therefore dominated by a small number of harder to classify images. This means that the probabilities of functions for a fixed ϵ_G can vary a lot. This explains why we find that even though the highest probability function in Figure 1a is a 1-error function, on average the probability for 1-error functions in Figure 1b is lower than that of the 0-error function. The high variance in $P_{\mathrm{B}}(f|S)$ within the functions of fixed ϵ_G, as well as the EP approximation also means that the estimates of $\langle P_B(f|S)\rangle_{\epsilon_G}$ may be less accurate. For Figure 1b, $\sum_{\epsilon}\rho(\epsilon)\langle P_B(f|S)\rangle_{\epsilon_G} \approx 0.1$, which is not far off the correct value of 1. Keeping in mind that we may be missing some higher probability outliers in the average due to finite sampling, this agreement is encouraging. In short, although the quantitative values may not be fully accurate, the results in Figure 1b are indicative of the strong exponential trend towards low probability with increasing error.

A third point of clarification is that in Figure 1b we used CE loss rather than MSE loss. We made this choice because we need to estimate very small $P_{\mathrm{B}}(f|S)$ values, for which we need to use the "ratio of likelihoods" approximation. While we only described how to use EP for CE loss, we could also use the EP approximation to estimate $P_{\mathrm{B}}(f|S)$ for the analytical posterior of the GP with MSE loss. However, from some preliminary tests, the EP approximation introduced significant systematic errors and in particular it didn't show better results than the EP/"ratio of likelihoods" method for CE loss. In Appendix A.2.3, we can also see that the differences between $P_{\mathrm{B}}(f|S)$ for MSE and CE loss are probably of less than a few orders of magnitude, and would not significantly affect the results in Figure 1b.

A.1.4 Further notes on methods

When $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$ are obtained by sampling, we typically sample between 10^5 and 10^7 times. To avoid finite sampling effects, we place any functions found with frequencies < 10 on the axes of our graphs. However, those functions are included in calculations involving generalisation errors, although typically they contribute very little because they are by definition low probability.

We will also regularly provide values for $\sum_{f\in F} P_{\mathrm{B}}(f|S)$ and $\sum_{f\in F} P_{\mathrm{OPT}}(f|S)$ in our figures comparing $P_{\mathrm{OPT}}(f|S)$ with $P_{\mathrm{B}}(f|S)$ (only when $P_{\mathrm{B}}(f|S)$ is obtained by direct sampling), where F is the set of functions found by both GP sampling and by the optimiser (within a finite number of samples from the GP and optimiser-trained DNNs, typically $10^4 - 10^6$). A value of $\sum P_{\mathrm{B}}(f|S)$ close to 1 indicates that almost all functions with high $P_{\mathrm{B}}(f|S)$ are found by the optimiser. Similarly, a high value of $P_{\mathrm{OPT}}(f|S)$ implies that GP sampling finds almost all functions with high $P_{\mathrm{OPT}}(f|S)$ found by the optimiser. Note that for the EP approximation needed for CE loss, we directly calculate the $P_{\mathrm{B}}(f|S)$ for functions found by the optimiser, and so a $\sum P_{\mathrm{OPT}}(f|S)$ is not defined.

Finally, when values for the generalisation error $\langle\epsilon_G\rangle$ are depicted in the graphs showing $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{OPT}}(f|S)$, $\langle\epsilon_G\rangle$ refers to the generalisation error of the optimiser. When $\langle\epsilon_G\rangle$ is presented in graphs with only $P_{\mathrm{B}}(f|S)$ it refers to the $\langle\epsilon_G\rangle$ from GP sampling, or alternatively we use the symbol $\langle\epsilon_G\rangle_{GP}$.

A.2 Description of Gaussian process calculations

In this section, we provide more details on the Gaussian process (GP) methods used in the paper. We follow the general Bayesian formalism introduced in Section 2.

A.2.1 GP with Mean Squared-Error (MSE) loss

For this formulation, we consider the space of functions to be the space of real-valued functions on $\mathcal{X}$. These functions correspond to the real-valued pre-activations of the last layer of the neural network, before a final non-linearity (like softmax or a step function) is applied. In the GP limit of DNNs, these functions have a GP prior (the NNGP). We then use a Gaussian likelihood defined as

$$P(S|f) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{1}{2\sigma^2}(f(x_i) - y_i)^2\right), \tag{5}$$

where σ^2 is the variance.

This likelihood allows us to analytically compute the exact posterior (Rasmussen, 2004). In the experiments in the paper we therefore sampled from this exact posterior, to get values of $f(x) \in \mathbb{R}$ at the test points, which were then thresholded at 0 to find the predicted class label. We have chosen a small value of the variance $\sigma^2 = 0.002$, to simulate SGD achieving a small value of the MSE loss.

Note that under the standard assumption that training and test instances come from the same distribution, this algorithm may be considered to be not fully Bayesian in the sense that the training and test labels are treated differently (Gaussian likelihood in training points versus Bernoulli likelihood at test points). Nevertheless, we believe that the differences are small.

A.2.2 GP with 0-1 likelihood, EP, and ratio of likelihoods approximation

While MSE loss has the advantage that the GP is analytically tractable, it has the disadvantage that it is less principled and less commonly used for classification than the much more popular cross-entropy (CE) loss function. Unfortunately the GP approximation to CE loss is more complex. Here our formulation uses a Gaussian process prior over a space of real-valued "latent" functions $\tilde{f}$ on $\mathcal{X}$. We call these functions "latent" because the true space of functions we are interested in is the space of Boolean functions on $\mathcal{X}$, obtained after applying the threshold nonlinearity to the last layer pre-activations. Formally, f is related to $\tilde{f}$ via a Bernoulli distribution with Heaviside linking function, as

$$P(f(x) = 1) = \begin{cases} 1 \text{ if } \tilde{f}(x) > 0 \\ 0 \text{ otherwise } . \end{cases}$$

We then use the previously defined 0-1 likelihood (Section 2.2) applied to the a binary-valued function f (taking values in $\{0, 1\}$). We use this likelihood, arguing that it best approximates the behaviour of an optimiser which is trained (using cross-entropy loss) until it *first* reaches 0 training error, which is the case in all the experiments in the paper (except those labelled as "overtraining"). Because of this we informally refer to this method as "using the CE loss" throughout the paper. Unfortunately, this likelihood makes the posterior of the GP analytically intractable. We therefore use a standard approximation technique known as expectation propagation (EP) (Rasmussen, 2004), which approximates the posterior over the latent function as a Gaussian, which we can sample from and then use the Heaviside function to predict the binary labels at the test points. We use this technique to approximate posterior probabilities of the GP with 0-1 likelihood, by sampling.

The marginal likelihood can also be estimated with the EP algorithm (Rasmussen, 2004). This gives the probability of a labelling of a set of points. Remember that in the Bayesian formalism S is essentially defined as the event that the input points x_i in the training set have labels y_i. We can similarly identify a function f with the event that the set of input points x in the whole domain $\mathcal{X}$ have labels $f(x)$, which is analogous, and can thus be computed in the same manner as the marginal likelihood! The posterior in Equation (2) for a function f which is compatible with S, can then be simply expressed as

$$P(f|S) = \frac{P(f)}{P(S)},$$

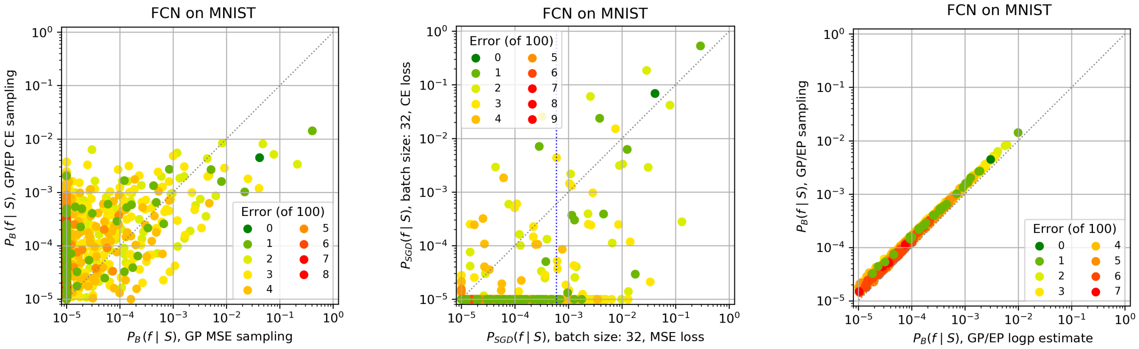
where both $P(f)$ and $P(S)$ are readily computed using the EP algorithm to approximate marginal likelihood. This will be referred to as the "ratio of likelihoods" approximation in the appendices. We have found that this method gives very similar results to the estimates using sampling from the approximate posterior obtained from EP (see Figure 7c). When we refer to the EP approximation, we imply that we are using the ratio of $P(f)$ and $P(S)$, each determined using the EP approximation, unless stated otherwise.

For the LSTM experiments, we used a smooth version of the 0-1 likelihood, analogous to using standard cross-entropy loss rather than miss-classification error. This was because the EP approximation was numerically unstable with the 0-1 likelihood for this system. The smooth version is described in Appendix A.2.3, and we empirically found that the two gave very similar estimates of probability.

A.2.3 Empirical results concerning the GP approximations

In this section we compare the behaviour of the GP approximations (and SGD) with different loss functions. We have argued that the GP/EP approximation underestimates probabilities by a power law (that is approximately linear in log-log; see (Valle-Pérez et al., 2018) for more details). Our results with the EP approximation are consistent with this expectation. As detailed in Appendices A.2.1 and A.2.2, there are subtle differences in the way the GP approximation with MSE loss and the GP/EP approximation with CE loss calculate their respective estimates for $P_{\mathrm{B}}(f|S)$. However, the (latent) function has the same prior in both cases, so we may expect the posterior $P_{\mathrm{B}}(f|S)$ to correlate. And it is clear from Figure 7a that they do indeed correlate. We believe that the correlation not being centred around $y = x$ is predominately due to the EP approximation because apart from scatter, the behaviour of SGD with the two loss functions is centred around $y = x$. In Appendix F, we also compare the EP and MSE approximations with a estimation via direct sampling (and thus with controlled error) of the posterior probabilities for 0-1 likelihood for small Boolean function datasets, where these computations are feasible. We indeed find that EP tends to underestimate posterior probabilities, specially for complex target functions. Overall what we find is that the EP approximation does reasonably well on relative probabilities, but less well on absolute probabilities.

To mitigate the effect that the EP approximation underestimates the probabilities, we perform a simple empirical regularisation. For systems where we find that $\sum P_{\mathrm{B}}(f|S) \approx 1$ for the MSE approximation, we renormalise the $P_{\mathrm{B}}(f|S)$ from the EP approximation by a constant factor such that $\sum P_{\mathrm{B}}(f|S) = 1$. For most systems we study the effect of this regularisation procedure is relatively small on a log scale. This method facilitates the comparison with $P_{\mathrm{SGD}}(f|S)$ because the errors in the absolute values are regularised in the same way for all systems. Note also that because we sample to obtain $P_{\mathrm{SGD}}(f|S)$ its empirical frequencies automatically sum to 1. If it were the case that the Bayesian sampling found a significant number of different high probability functions, then this would be observed in a lack of correlation in the comparison with $P_{\mathrm{SGD}}(f|S)$. Instead, we find a strong correlation between $P_{\mathrm{SGD}}(f|S)$ and $P_{\mathrm{B}}(f|S)$ calculated with the EP approximation, suggesting that the functions found are the dominant ones found by both methods, as is explicitly found to be the case for most instances of MSE loss that we studied. This regularisation is applied to all experiments (for ease of comparison), unless otherwise specified.



(a) $P_{\mathrm{B}}(f|S)$(CE/EP) versus $P_{\mathrm{B}}(f|S)$ (MSE)

(b) $P_{\mathrm{SGD}}(f|S)$(CE) versus $P_{\mathrm{SGD}}(f|S)$ (MSE)

(c) $P_{\mathrm{B}}(f|S)$(CE/EP) sampled v.s. ratio of likelihoods estimate

Figure 7: **Comparing GP approximations with CE and MSE loss for FCN on MNIST** In (a) we compare the behaviour of the GP approximations for $P_{\mathrm{B}}(f|S)$ with MSE loss to the GP/EP approximation with CE loss. We sampled from the GP MSE posterior, and the GP/EP CE posterior distribution 10^6 times. It is expected that the two measures should diverge somewhat due to details of the loss function on the training data. (b) compares $P_{\mathrm{SGD}}(f|S)$ with MSE loss and $P_{\mathrm{SGD}}(f|S)$ with CE loss. Functions with high $P_{\mathrm{SGD}}(f|S)$ are scattered around $y = x$. This implies that the loss function does not substantially affect $P_{\mathrm{SGD}}(f|S)$ on average. Note that in (a) the two methods correlate, but that 1) the GP-EP is systematically lower than the MSE, and 2) that the slope is below $x = y$. These two trends are, we believe, more general for the GP-EP approximation on CE loss. (c) Here we compare the GP/EP $\log(p)$ approximation with GP/EP sampling. For this figure, we use only functions found by Adam in 10^6 samples, and compare probabilities found by the GP/EP $\log(p)$ approximation to those found by GP/EP sampling. We use both methods in this paper, but as is clear from the above figure, there is not much difference between them for functions with high $P_{\mathrm{B}}(f|S)$.

Specifically, for Figure 9 the renormalisation constant was calculated for Adam without overtraining and batch size 128 (as it had the highest raw value for $\sum P_{\mathrm{B}}(f|S)$), and the probabilities in the other plots with FCN on MNIST are all adjusted by the multiplicative constant of 3.59, which is modest on the full log scale of the graphs. For other plots, the probabilities were normalised. The two systems for which this renormalisation has a larger effect are the LSTM and the ionosphere dataset. While for both systems the MSE sampling looks relatively close to $y = x$, the raw EP approximation has significantly lower probabilities. The renormalisation factors were 1.15×10^5 and 97 respectively. Smaller MSE experiments verify that there is a strong correlation between $P_{\mathrm{SGD}}(f|S)$ and $P_{\mathrm{B}}(f|S)$ for these systems.

In Appendix A.2.2, we described using a Heaviside linking function when using the 0-1 loss function in the EP approximation. To test this, we have also sampled f following a Bernoulli distribution with a Probit linking functionto the latent $\tilde{f}$ (which is analogous to using a cross-entropy loss). To test the differences between the results (which we assume to be small), we tested 100 randomly selected functions (on MNIST) with $\langle \epsilon_G \rangle$ ranging from 0% to 100%. Of these, the average difference between the results as a percentage of the magnitude of the log probabilities was 0.013%, and the maximum was 0.58% for the FCN architecture. We also compare the GP/EP sampling with GP/EP $\log P(S)$ approximation in Figure 7c. The $\log P(S)$ approximation is a further approximation that allows $\log P(f)$ to be extracted without requiring sampling. Unless otherwise specified, all results in the main text use the GP/EP $\log P(S)$ approximation rather than sampling.

Appendix B. Notes on the distribution of MNIST data

Here, we show that p_i, the mean probability, for an FCN on MNIST with MSE loss function, that either SGD or the NNGP classifies the i'th image in the test set E (used throughout this paper, with $|E| = 100$) incorrectly, varies by many orders of magnitude. We also show that images are classified in an approximately independent manner.

The values of p_i for the MNIST test set which is used in the majority of this paper are given in Figure 8a. Note that $(1/n)\sum_i p_i = \langle \epsilon_G \rangle)$. As can be seen, there are a small number of images that have a much higher probability of being miss-classified. It should be kept in mind that the exact spectrum of p_i will depend on which images are in the test set. For example, in Figure 12 we compare two test sets (albeit with CE-loss and the EP approximation). As can be observed, the zero error function has significantly higher probability in the second test set, and the highest 1-error function has lower probability than for the test set we use in the paper. For test sets of this size such fluctuations are not unexpected.

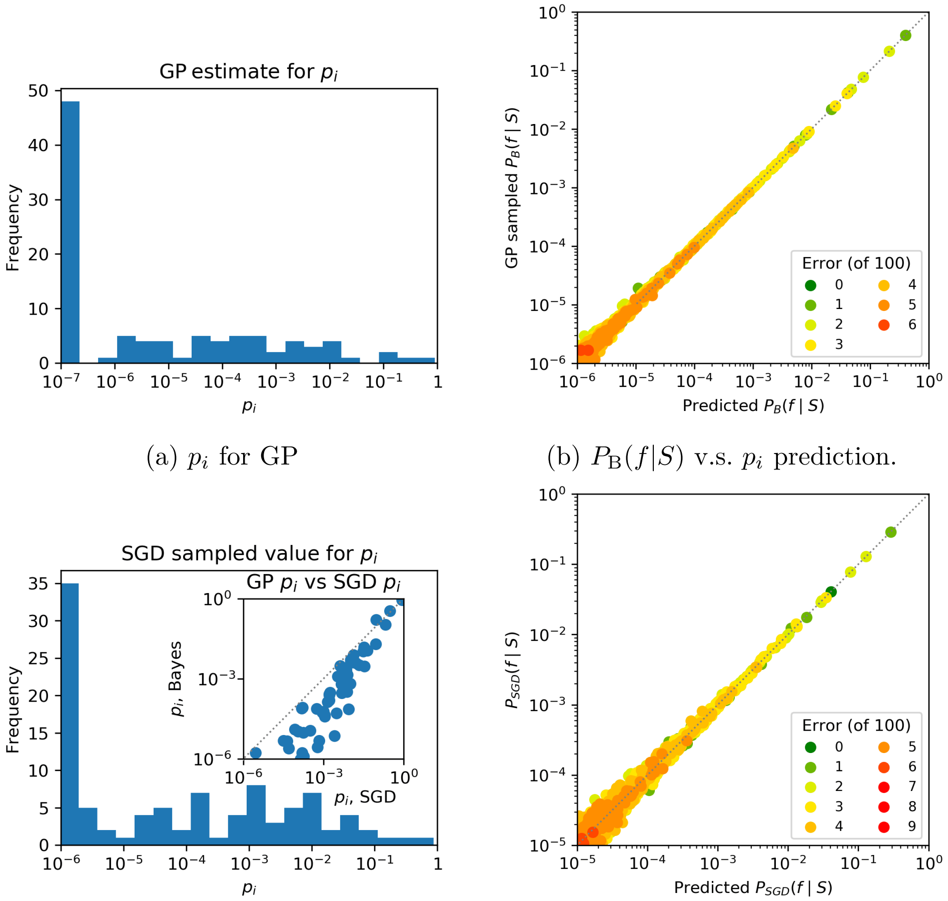
Next we use the p_i to calculate probabilities for functions with other sets of errors. More specifically, consider test sets E_i containing only one image I_i (where i takes values in $\mathbb{N}$). Then calculating the probability of a function on $E = \{E_i\}_{i=0}^{i=|E|}$ for a test set of size $|E|$ can be done by multiplying the appropriate probabilities for functions on E_i. Applying the above method to this data is shown in Figure 8b. All the functions shown are very close to the $y = x$ line, indicating that the images are classified in an approximately independent fashion.

In all our examples we observe a clear linear decay of the mean of $\log(P_{\mathrm{B}}(f|S))$ v.s. ϵ_G. If the p_i were identical, then this would simply be what is expected for a Binomial distribution. For independent but different p_i over images in a finite test set the distribution is called the Poisson Bionomial distribution. Obviously its values depend on the exact distribution of p_i. However, there exists a Chernoff bound

$$p(\epsilon) \leq P[\mathcal{E} > \epsilon] \leq \exp(-\epsilon(\log(\epsilon/\langle \epsilon_G \rangle) - 1) - \langle \epsilon_G \rangle), \tag{6}$$

where $p(\epsilon)$ is the pmf, $P[\mathcal{E} > \epsilon]$ is the cmf, and $\langle \epsilon_G \rangle = (1/n)\sum_i p_i$ is the mean. On a log scale, this means

$$\log_{10}(p(\epsilon)) \leq [-\epsilon(\log(\epsilon/\langle \epsilon_G \rangle) - 1) - \langle \epsilon_G \rangle] \log_{10}(e), \tag{7}$$



(a) p_i for GP

(b) $P_{\mathrm{B}}(f|S)$ v.s. p_i prediction.

(c) p_i for SGD

(d) $P_{\mathrm{B}}(f|S)$ v.s. p_i prediction.

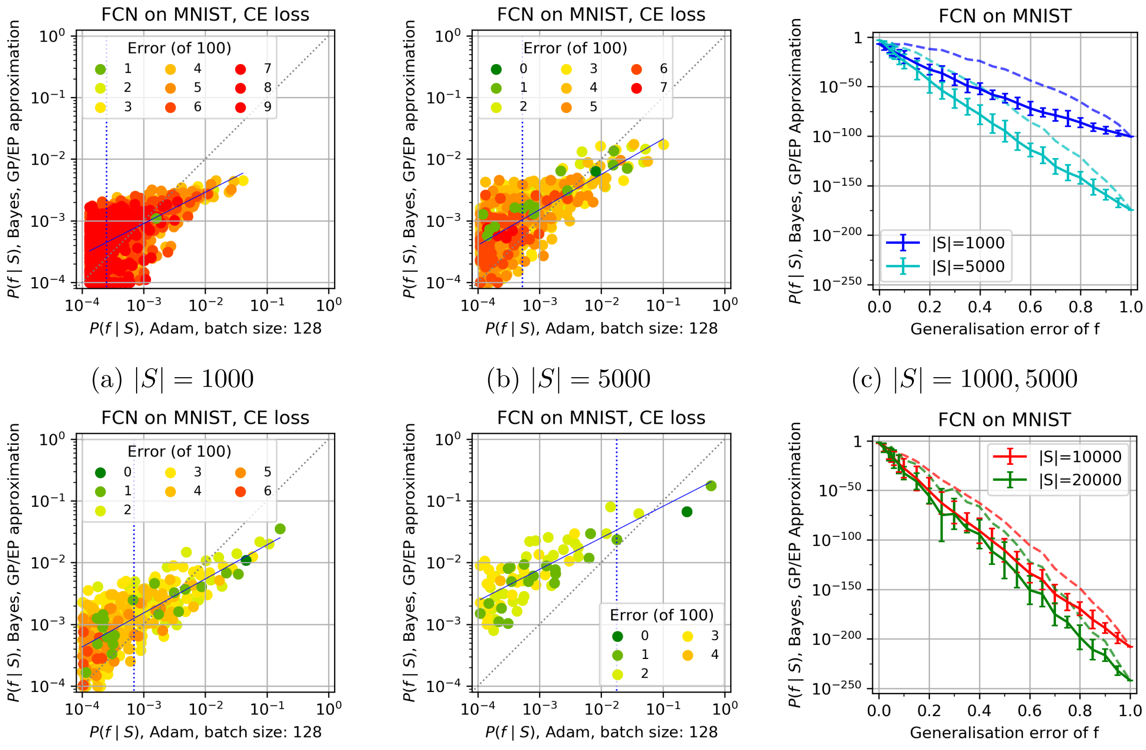
Figure 8: (a) shows the NNGP estimate for the values of p_i for an FCN with MSE loss on MNIST, for the training set of size 10000 and test set of size 100 that we use in the main text. Clearly these vary over many orders of magnitude. The sample size is 10^7, so frequencies were cut off at 10^{-7} (so functions in that bin have $p_i \leq 10^{-7}$). 20 bins in total. (b) calculates the probability of other error functions using the values of p_i from Figure 1d using the assumption that the images from (a) are independently distributed, and compares this prediction for $P_{\mathrm{B}}(f|S)$ to the value of $P_{\mathrm{B}}(f|S)$ obtained by direct GP MSE sampling (see Figure 1d for the data). Each datapoint is for a specific function f, and clearly they are close to the $y = x$ line, implying that, at least for these higher probability functions found by direct sampling, the images in this small test set are classified by the GP in a (close to) independent fashion. Figures (c) and (d) are the equivalent of (a) and (b), but for SGD (see Figure 1a for the data). There were only 10^6 samples, so (c) and (d) are cut off at one order of magnitude lower than (a) and (b). (c) includes an inset comparing the values for p_i with GP MSE sampling from (a) and p_i with SGD from the main part of (c). The correlation is fairly tight for the highest p_i which dominate the total probability mass, and this correlation helps explain the strong correlation seen for other numbers of errors throughout this paper.

which indicates an exponential like drop-off for $\epsilon \geq \langle \epsilon_G \rangle$ (similar to what is observed in Figure 1a). Of course this is an upper bound and the actual distribution can strongly depend on the full spectrum of p_i values. We are currently exploring these issues in more detail.

Appendix C. Effects of training set size

It is also instructive to study the correlation between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$ for different training set sizes $|S|$. As can be seen clearly in Figure 9, as the training set increases in size, the functions with zero training error are more strongly biased towards low generalisation error, as expected. Figures 9c and 9f also illustrate how the stronger bias with increasing $|S|$ means that the entropic factor $\rho(\epsilon_G)$ plays a smaller role. Thus, for larger training set size, but for the same amount of sampling n, fewer functions are found, but on average they have higher probability.

An important question in deep learning is: How does the error reduce with increasing the training set size? There is intriguing evidence that such "learning curves" follow a power law that depends on data complexity, and only weakly on the architecture (Hestness et al., 2017; Spigler et al., 2019; Rosenfeld et al., 2019; Kaplan et al., 2020). Figure 9 shows how the spectrum of function probabilities changes with increasing $|S|$. Investigations based on this more fine-grained picture may help improve our understanding of learning curves.



(a) $|S| = 1000$

(b) $|S| = 5000$

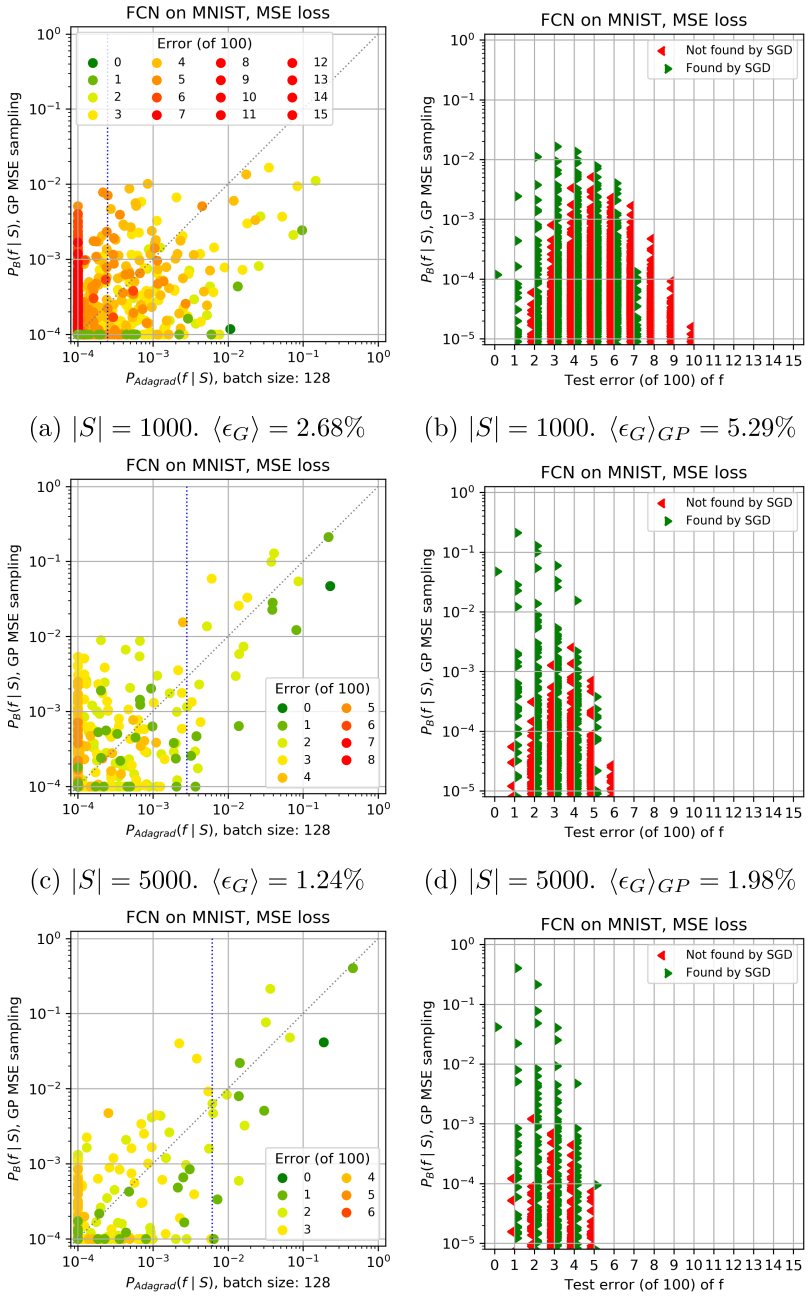
(c) $|S| = 1000, 5000$

(d) $|S| = 10000$

(e) $|S| = 20000$

(f) $|S| = 10000, 20000$

Figure 9: **Comparing $\mathbf{P_B(f|S)}$ to $\mathbf{P_{Adam}(f|S)}$ for an FCN on MNIST with CE loss for different training set sizes.** [We use test set size of $|E| = 100$; vertical dotted blue lines denote 90% probability boundary; solid blue line is a guide to the eye, dashed grey line is $x = y$.]
(a) 1000 training examples. $\langle \epsilon_G \rangle = 6.65\%$ for Adam.
(b) 5000 training examples. $\langle \epsilon_G \rangle = 3.33\%$ for Adam.
(c) $P_{\mathrm{B}}(f|S)$ v.s. ϵ_G for 1000 and 5000 training examples.
(d) 10000 training examples. $\langle \epsilon_G \rangle = 2.20\%$ for Adam.
(e) 20000 training examples. $\langle \epsilon_G \rangle = 0.89\%$ for Adam.
(f) $P_{\mathrm{B}}(f|S)$ v.s. ϵ_G for 5000 and 10,000 training examples.
A trend of increasing bias towards lower error functions with increasing training set size can be clearly observed. See Figure 10 for related results with MSE loss.



(a) $|S| = 1000$. $\langle \epsilon_G \rangle = 2.68\%$

(b) $|S| = 1000$. $\langle \epsilon_G \rangle_{GP} = 5.29\%$

(c) $|S| = 5000$. $\langle \epsilon_G \rangle = 1.24\%$

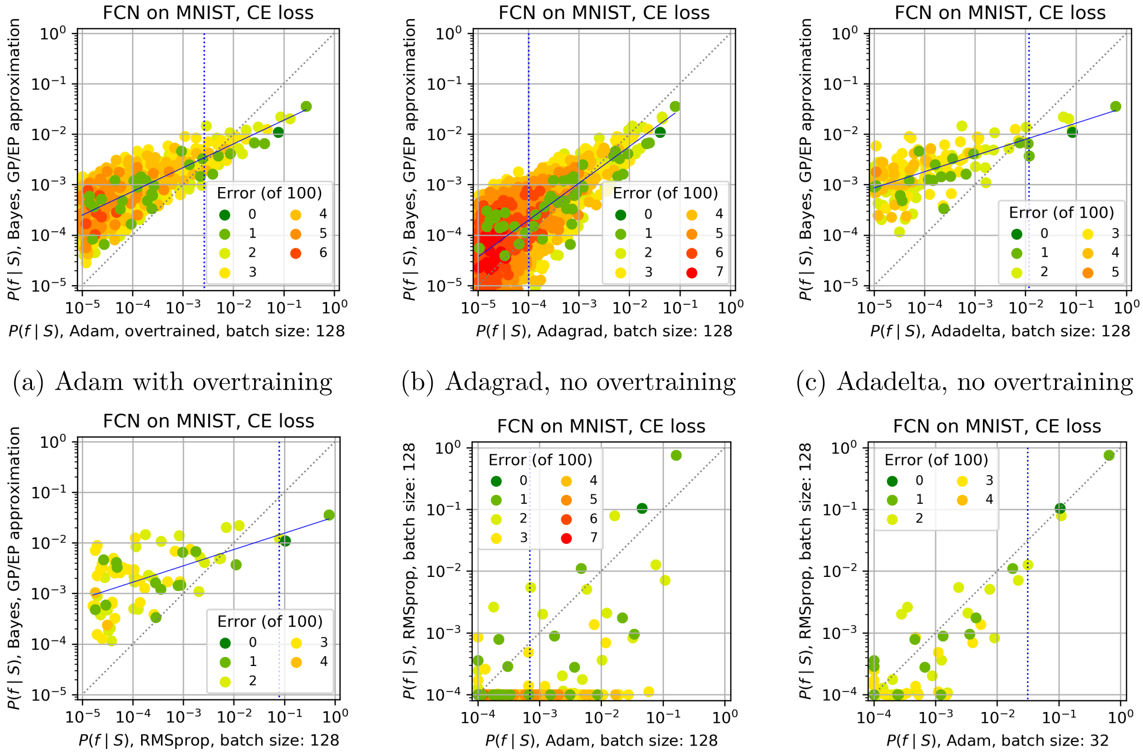
(d) $|S| = 5000$. $\langle \epsilon_G \rangle_{GP} = 1.98\%$

(e) $|S| = 10000$. $\langle \epsilon_G \rangle = 1.14\%$

(f) $|S| = 10000$. $\langle \epsilon_G \rangle GP = 1.61\%$

Figure 10: **$\mathbf{P_B(f|S)}$ v.s. $\mathbf{P_{Adagrad}(f|S)}$ for an FCN on MNIST with MSE loss, for different training set sizes. Further results for Figure 9.** [Test set size $|E| = 100$ and batch size=128. Vertical dotted blue lines denote 90% probability boundary; solid blue lines are fit to guide the eye; dashed grey line is $x = y$.] (a) and (b) show 1000 training examples, and (c) and (d) show 5000 training examples, and (e) and (f) show 10000. As expected, more training examples reduce the generalisation error and increase the correlation between $P_{\mathrm{Adagrad}}(f|S)$ and $P_{\mathrm{B}}(f|S)$. For all the larger training sets, the $\sum P_{\mathrm{B}}(f|S) \approx 1$, but for $|S = 1000|$ functions found by GP sampling only make up about 40% of the probability of all functions found by Adagrad.

C.1 Changing optimisers



(a) Adam with overtraining

(b) Adagrad, no overtraining

(c) Adadelta, no overtraining

(d) RMSprop, no overtraining

(e) RMSprop vs Adam 128

(f) RMSprop vs Adam 32

Figure 11: **Comparing $\mathbf{P_B(f|S)}$ to $\mathbf{P_{OPT}(f|S)}$ for an FCN on MNIST with CE loss for different optimisers.** [We use training/test set size 10,000/100 and batch size=128. Vertical dotted blue lines denote 90% probability boundary; dashed grey line is $x = y$.]
(a) Adam with overtraining for 64 epochs. $\langle \epsilon_G \rangle = 1.73\%$.
(b) Adagrad, $\langle \epsilon_G \rangle = 2.63\%$.
(c) Adadelta, $\langle \epsilon_G \rangle = 1.23\%$.
(d) RMSprop, $\langle \epsilon_G \rangle = 1.02\%$.
(e) $P_{\text{RMSprop}}(f|S)$ v.s. $P_{\text{Adam}}(f|S)$ both with batch size 128.
(f) $P_{\text{RMSprop}}(f|S)$ with batch size 128 v.s. $P_{\text{Adam}}(f|S)$ with batch size 32.
For Adam without overtraining, see Figure 5b. For Adagrad with overtraining see Figure 1f. For others with overtraining see Figure 16. Overtraining has a milder effect than changing the optimiser does. (e) and (f) show that there is a surprisingly close correspondence between RMSprop with batch size 128 and Adam with batch size 32. By contrast, for the same batch size of 128, Adam and Adagrad are similar to one another, as are Adadelta and RMSprop (see Figure 17 for a direct comparison). The latter two have better generalisation performance, which is reflected in higher probabilities for the lowest error functions. See Figures 16 and 17 for further results.
In all these figures, while $P_{\text{OPT}}(f|S)$ is to first order determined by $P_{\text{B}}(f|S)$, there are noticeable second order differences.

Much research effort has gone into adaptations of SGD. One goal is to achieve more efficient optimisation, but another is to achieve better generalisation. Figure 11 illustrates the effect of changing the optimiser on the correlation between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$. (See also Figures 1f, 5b and 16 to complete the set of optimisers with and without overtraining). To first order this figure shows that $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{OPT}}(f|S)$ are remarkably closely correlated for all these optimisers, even taking into account that the EP approximation introduces errors, and likely leads to a slightly too small slope in $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{OPT}}(f|S)$.

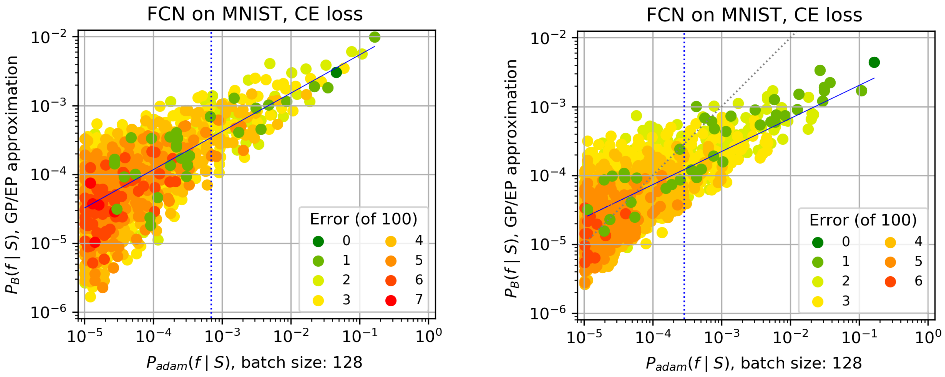
What is perhaps more interesting here are second-order effects, since $P_{\mathrm{B}}(f|S)$ is identical in each plot. For example, RMSprop has the best generalisation performance, which is reflected in a stronger inductive bias towards a few key low error functions. Note also the similarity of batch size 128 RMSprop to Adam with smaller batch size of 32. We emphasise that this performance here doesn't mean that RMSprop is in general superior to the other SGD variants for FCNs on MNIST. To investigate that question, we would need to study other test and training sets, and need to do further hyperparameter tuning (Choi et al., 2019).

We can also compare the effect of overtraining. In each case shown in Figures 1f, 5b and 16, overtraining brings a modest improvement in generalisation error on this test set (Adam from $\epsilon_G = 2.2\%$ to $\epsilon_G = 1.74\%$ and Adagrad from $\epsilon_G = 2.63\%$ to $\epsilon_G = 2.19\%$), for example. From the graphs one can see a slight increase in the optimiser probability of the lowest error function with overtraining, but also a clear reduction in the scatter of the data for the whole range of probabilities, suggesting that for CE loss, on average, overtraining brings $P_{\mathrm{OPT}}(f|S)$ closer to the Bayesian prediction. This behaviour can possibly be rationalised in that overtraining allows the optimiser to sample functions with probabilities closer to the steady-state average (see also Section 6).

Finally, in Figures 11e, 11f and 17 we directly compare the $P_{\mathrm{OPT}}(f|S)$ to one another, in other words, without using $P_{\mathrm{B}}(f|S)$. A number of clear trends are visible, for example, with batch size 128, Adam and Adagrad are very similar to one another, as are RMSprop and Adadelta. However, as can be seen in Figure 11f, Adam with a smaller batch size of 32 is very similar to RMSprop with batch size of 128. What these correlation plots show is that the behaviour of the different optimisers can depend on batch size in subtle ways that may not necessarily be picked up by the generalisation error. Further work (and significant computational resources) would be needed to completely compare these methods.

These examples show that studying the spectrum of function probabilities provides more fine-grained data than simply comparing generalisation error does. Future studies on problems such as optimiser choice or hyperparameter tuning could exploit this fuller set of information to increase understanding and to improve DNN performance.

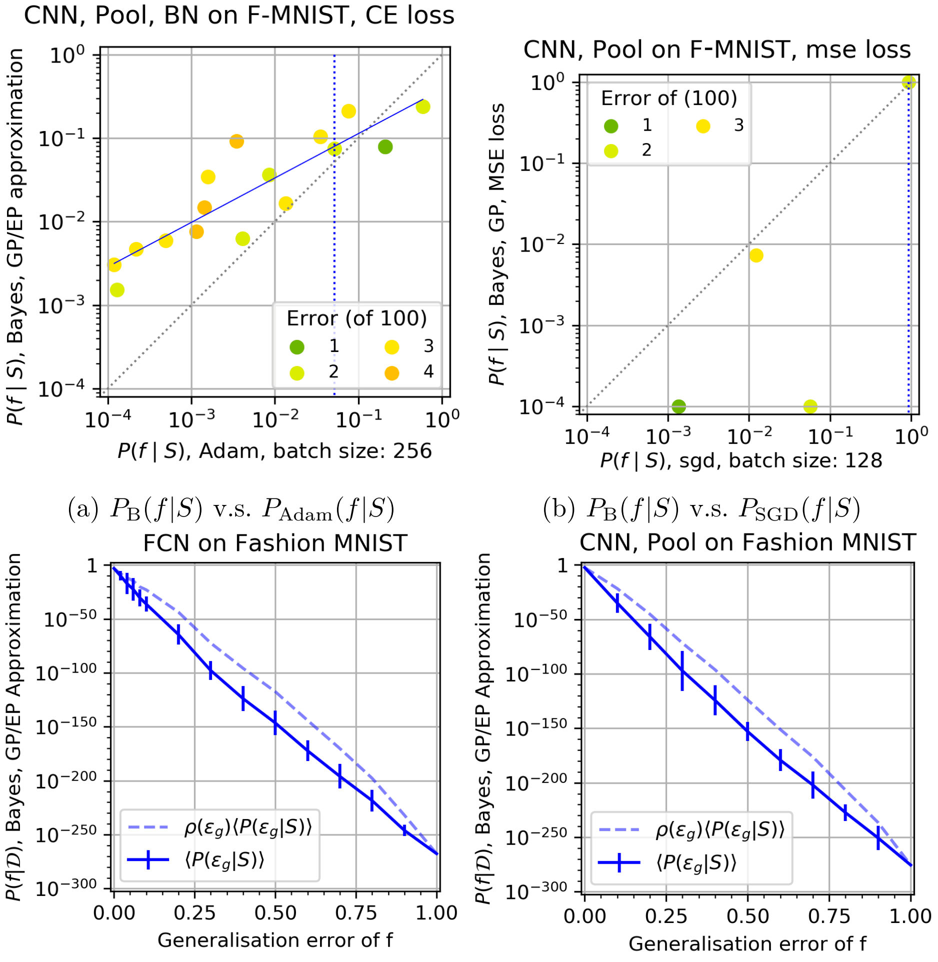
Appendix D. Further results comparing $P_{\mathbf{OPT}}(f|S)$ to $P_{\mathbf{B}}(f|S)$.



(a) Test set 1: $\langle \epsilon_G \rangle$ =2.20%.

(b) Test set 2: $\langle \epsilon_G \rangle$ =1.62%

Figure 12: **Comparing $P_{\mathbf{B}}(f|S)$ to $P_{\mathbf{Adam}}(f|S)$ for an FCN on MNIST with CE loss for two different test sets** [We use training/test set size 10,000/100 and batch size=128. Vertical dotted blue lines denote 90% probability boundary; solid blue lines are fit to guide the eye; dashed grey line is $x = y$.] (a) is the test set used throughout the paper and (b) is another test set (disjoint from both the first test set and the training set), chosen at random from MNIST. To first order the two look very similar, but to second order small differences can be seen, not just in the function probabilities, but also in the slope of $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$, which may be affected by the EP approximation used here. No normalisation was applied to this figure so it depicts the raw EP approximation (see Appendix A.2.2).



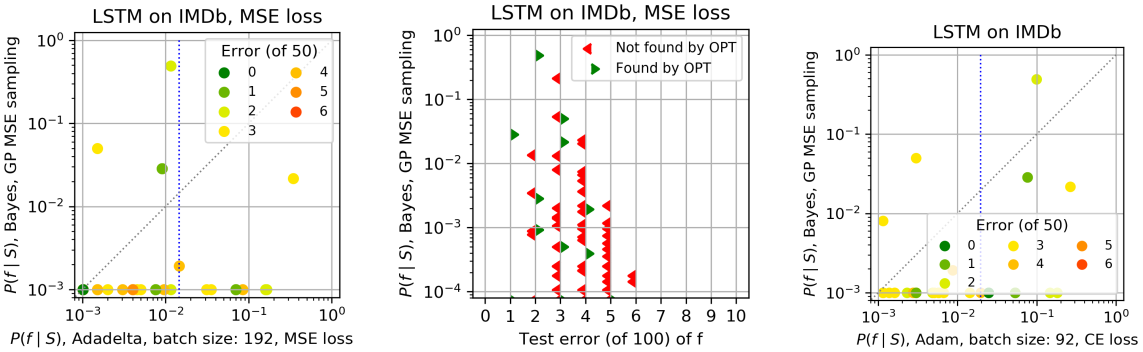
(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$

(b) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{SGD}}(f|S)$

(c) 15 samples per $\langle \epsilon_G \rangle$

(d) 15 samples per $\langle \epsilon_G \rangle$

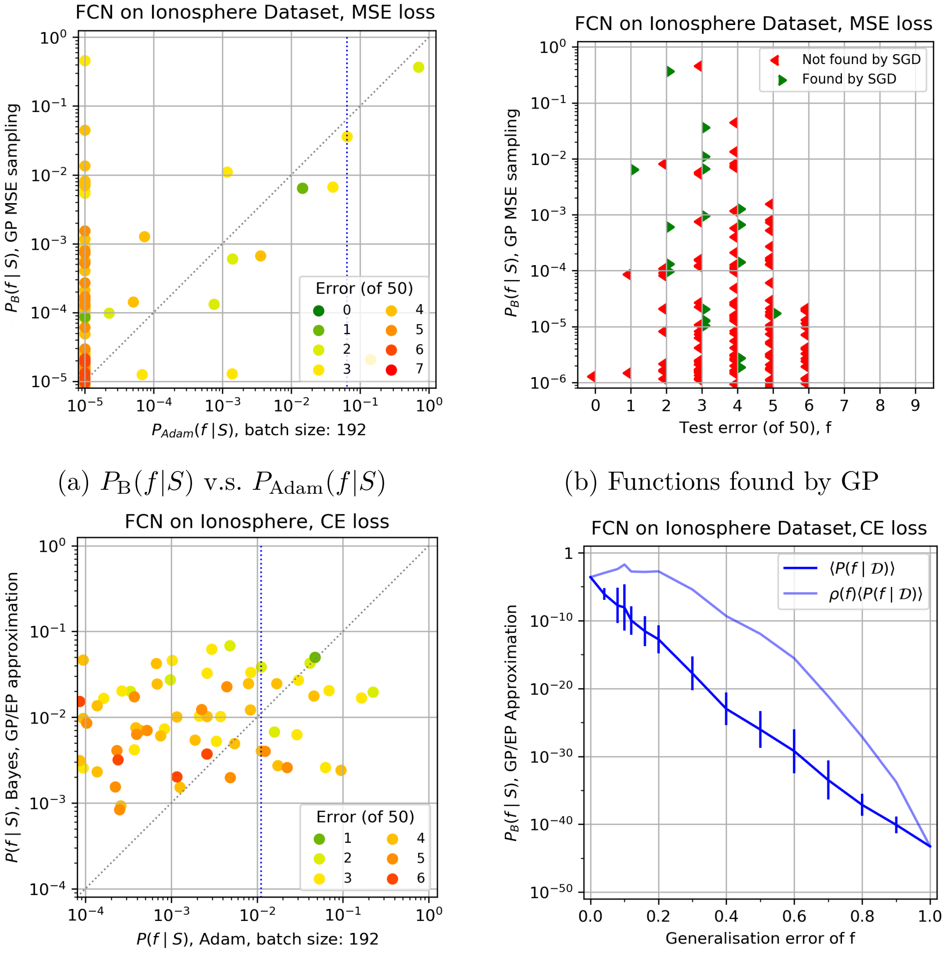
Figure 13: **Comparing the Bayesian prediction $\mathbf{P_B(f|S)}$ to $\mathbf{P_{Adam}(f|S)}$ for an FCN and CNNs on Fashion-MNIST. Further results for Figure 2.** [We use train/test set size of 10,000/100; vertical dotted blue lines denote 90% probability boundary; solid blue lines are fit to guide the eye; dashed grey line is $x = y$.] (a) CNN with max-pooling and batch normalisation on Fashion-MNIST; $\langle \epsilon_G \rangle = 2.11\%$ for Adam with CE loss. Note that the GP kernel used is the same as in Figure 2c, so with pooling but without batch normalisation. The effect of batch normalisation is relatively small on this system. (b) CNN with max-pooling on Fashion-MNIST and MSE loss. $\langle \epsilon_G \rangle = 2.01$ for SGD and $\langle \epsilon_G \rangle = 2.00$ for the GP MSE sampling. 2000 samples for SGD, 10^5 samples from the GP. The same function is found over 99% of the time by both the GP and SGD. (c) and (d) show an average of $P_{\mathrm{B}}(f|S)$ versus error for an FCN and CNN with max pooling respectively.



(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adadelta}}(f|S)$ (b) f found by GP in (a) (c) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$

Figure 14: **Comparing $P_{\mathbf{B}}(f|S)$ to $P_{\mathbf{OPT}}(f|S)$ for a LSTM on the IMDb movie review dataset. Further results for Figure 4.** [We use training/test set size 45,000/50 and batch size=192. Vertical dotted blue lines denote 90% probability boundary; solid blue lines are fit to guide the eye; dashed grey line is $x = y$.]
(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adadelta}}(f|S)$ for MSE loss. $n = 2200$ for the optimiser and $n = 2.6 * 10^4$ for the GP. $\langle \epsilon_G \rangle = 5.22\%$ and $\langle \epsilon_G \rangle_{GP} = 5.04\%$
(b) Probability of all functions found by NNGP compared to those also found by the optimiser. Green points are the set jointly found functions F. Red denotes functions found only by the GP, $\sum_{f \in F} P_{\mathrm{B}}(f|S) = 54.1\%$ and $\sum_{f \in F} P_{\mathrm{OPT}}(f|S) = 77.0\%$.
(c) Compares the $P_{\mathrm{Adam}}(f|S)$ with CE loss from Figure 4a, to the sampled $P_{\mathrm{B}}(f|S)$ using MSE loss for $n = 10^4$ samples.

While these results are for very limited sample numbers, they provide evidence that for the LSTM, $P_{\mathrm{B}}(f|S)$ has values on the same order of magnitude as $P_{\mathrm{SGD}}(f|S)$. The low values we find for the raw EP approximation estimates of $P_{\mathrm{B}}(f|S)$ are likely to be due to errors in the EP absolute values. The fact that we still see correlations for the CE-trained LSTM with the renormalised EP approximations for $P_{\mathrm{B}}(f|S)$ suggests that the EP still does reasonably on relative errors. CE has the advantage that it is much faster to use than MSE.



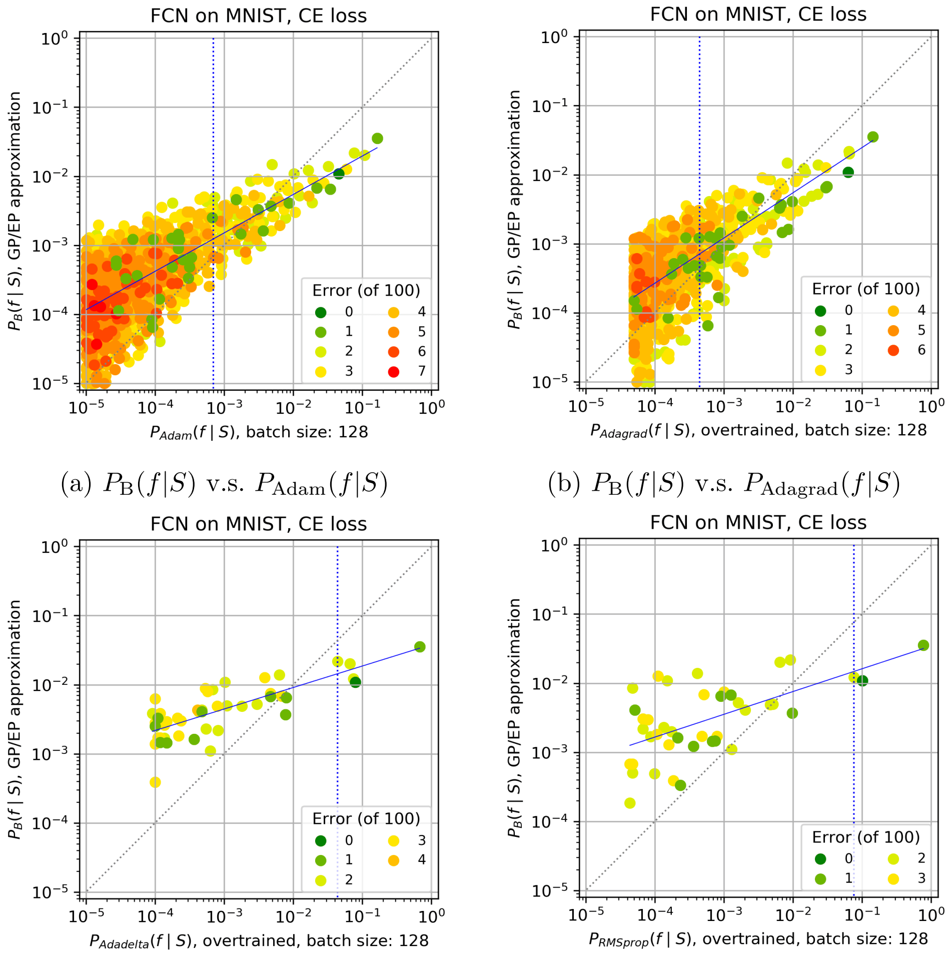
(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$

(b) Functions found by GP

(c) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$

(d) $P_{\mathrm{B}}(f|S)$ v.s. ϵ_G

Figure 15: **Comparing $P_{\mathbf{B}}(f|S)$ to $P_{\mathbf{Adam}}(f|S)$ for an FCN on the Ionosphere dataset with CE loss. Further results for Figure 4.** [We use training/test set size 301/50 and batch size=192. Vertical dotted blue lines denote 90% probability boundary; solid blue lines are fit to guide the eye; dashed grey line is $x = y$.]
(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$ for MSE (same as in Figure 4), put here ease of comparison).
(b) Probability of all functions found by NNGP compared to those also found by the optimiser. Green points are the set jointly found functions F. $\sum_{f\in F} P_{\mathrm{B}}(f|S) = 43.1\%$ and $\sum_{f\in F} P_{\mathrm{Adam}}(f|S) = 99.8\%$ (in other words nearly all functions found by Adam are also found by the GP, but the GP also finds functions that Adam doesn't for this level of sampling). For the optimiser, $\langle\epsilon_G\rangle = 4.59\%$ and for the GP MSE sampling, $\langle\epsilon_G\rangle_{GP} = 5.41\%$.
(c) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$ for CE. $\langle\epsilon_G\rangle = 5.88\%$.
(d) $< P_{\mathrm{B}}(f|S) >$ versus $\langle\epsilon_G\rangle$ with CE, 20 samples per ϵ_G. The bias towards low error functions is less strong than what is found for MNIST or Fashion-MNIST.



(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adam}}(f|S)$

(b) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adagrad}}(f|S)$

(c) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{Adadelta}}(f|S)$

(d) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{RMSprop}}(f|S)$

Figure 16: **Comparing $\mathbf{P_B(f|S)}$ to $\mathbf{P_{OPT}(f|S)}$ for an FCN on MNIST with CE loss for different optimisers. Further results for Figure 11.** [We use training/test set size 10,000/100; batch size=128. Vertical dotted blue lines denote 90% probability boundary; solid blue lines are fit to guide the eye; dashed grey line is $x = y$.]
(a) Adam, no overtraining, $\langle \epsilon_G \rangle = 2.20\%$. Sample size: $n = 10^6$.
(b) Adagrad with overtraining, $\langle \epsilon_G \rangle = 2.19\%$. Sample size: $n = 2.1 \times 10^5$.
(c) Adadelta with overtraining, $\langle \epsilon_G \rangle = 1.17\%$. Sample size: $n = 10^5$.
(d) RMSprop with overtraining, $\langle \epsilon_G \rangle = 1.01\%$. Sample size: $n = 2.5 \times 10^5$.

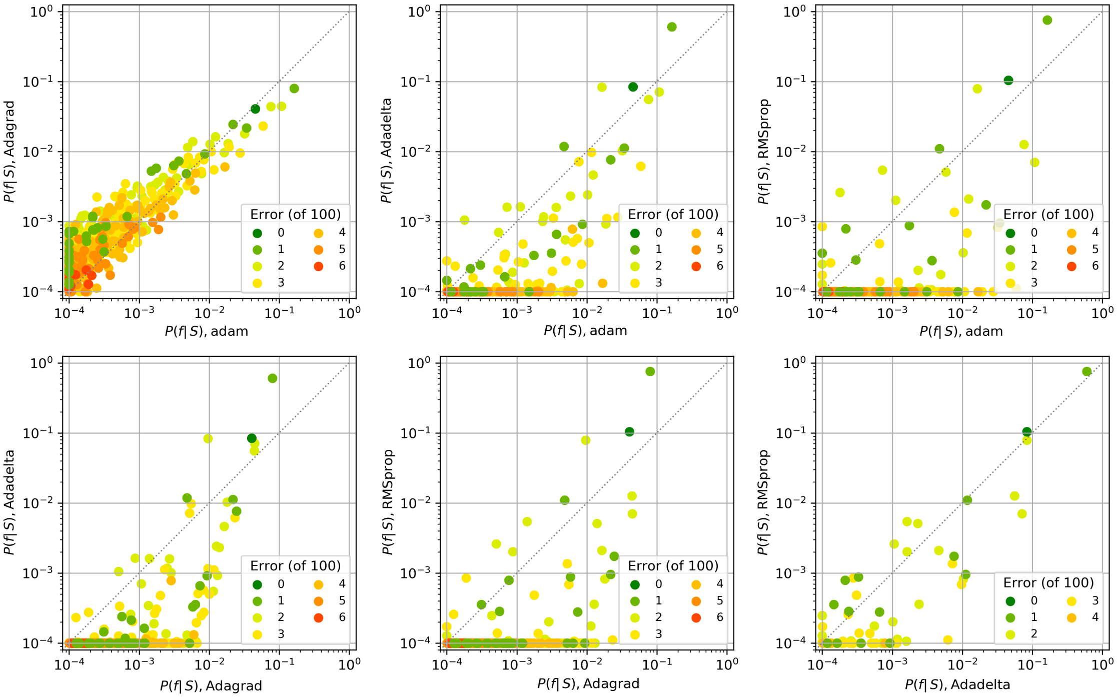
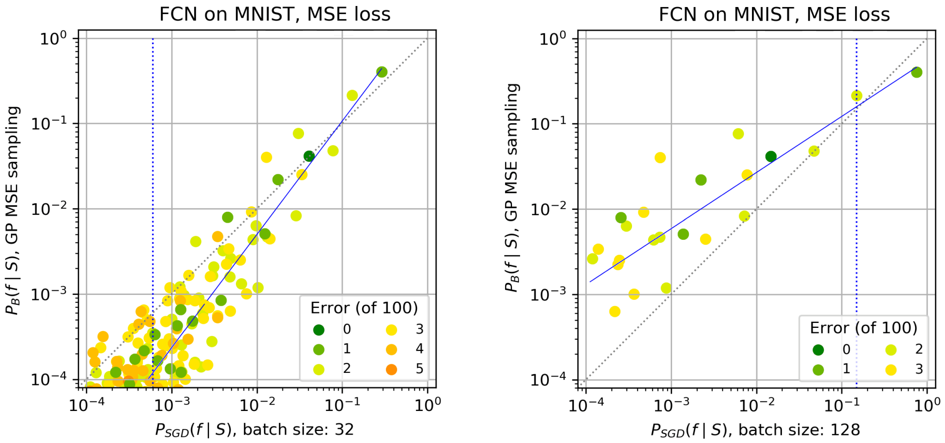


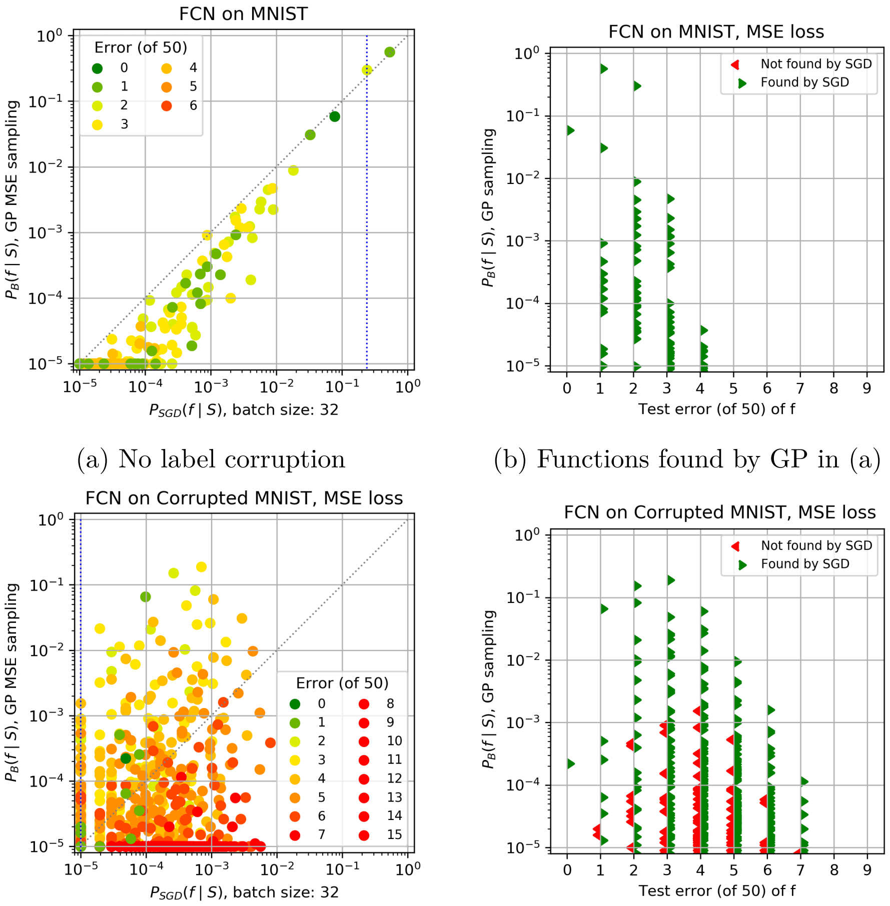
Figure 17: **Comparing $P_{\mathbf{OPT}}(f|S)$ with $P_{\mathbf{OPT}}(f|S)$ for different optimisers. Further results for Figure 11.** [We use the FCN architecture on MNIST, CE loss and a batch size of 128, training/test set size = 10,000/100, $y = x$ is denoted by a dashed line]
Adagrad and Adam correlate very well as do Adadelta and RMSprop for these hyperparameters. The other combinations do not correlate as well, suggesting that they sample the loss-function differently from one another. These plots do not rely on the GP or GP/EP approximation. We believe that this sort of experiment may prove useful for understanding differences in the behaviour of the optimisers.



(a) $\langle \epsilon_G \rangle$ =1.88%, 0 Test error opt/gp:

(b) $\langle \epsilon_G \rangle$ =1.22%, 0 Test error opt/gp:

Figure 18: **$\mathbf{P_B(f|S)}$ v.s. $\mathbf{P_{Adam}(f|S)}$ for an FCN on MNIST with the Adam optimiser and MSE loss with different batch sizes. Further results for Section 5.1.** Batch sizes: (a) 32, and (b) 128. For this MSE loss function, we observe that increasing the batch size leads to better generalisation, and also to a shallower best fit because the highest probability functions are sampled with enhanced probability by SGD. This trend with batch size is the opposite of what was observed for CE loss in Figure 5.



(a) No label corruption

(b) Functions found by GP in (a)

(c) 20% label corruption.

(d) Functions found by GP in (c)

Figure 19: **Comparing $P_{\mathbf{SGD}}(f|S)$ to $P_{\mathbf{B}}(f|S)$ for an FCN on MNIST with label corruption and MSE loss. Further results for Figure 6.** [Training/test set size 10,000/50; batch size=128; vertical dotted blue lines denote 90% probability boundary; solid blue lines are fit to guide the eye; dashed grey line is $x = y$; points on the axes are found by one technique only.]

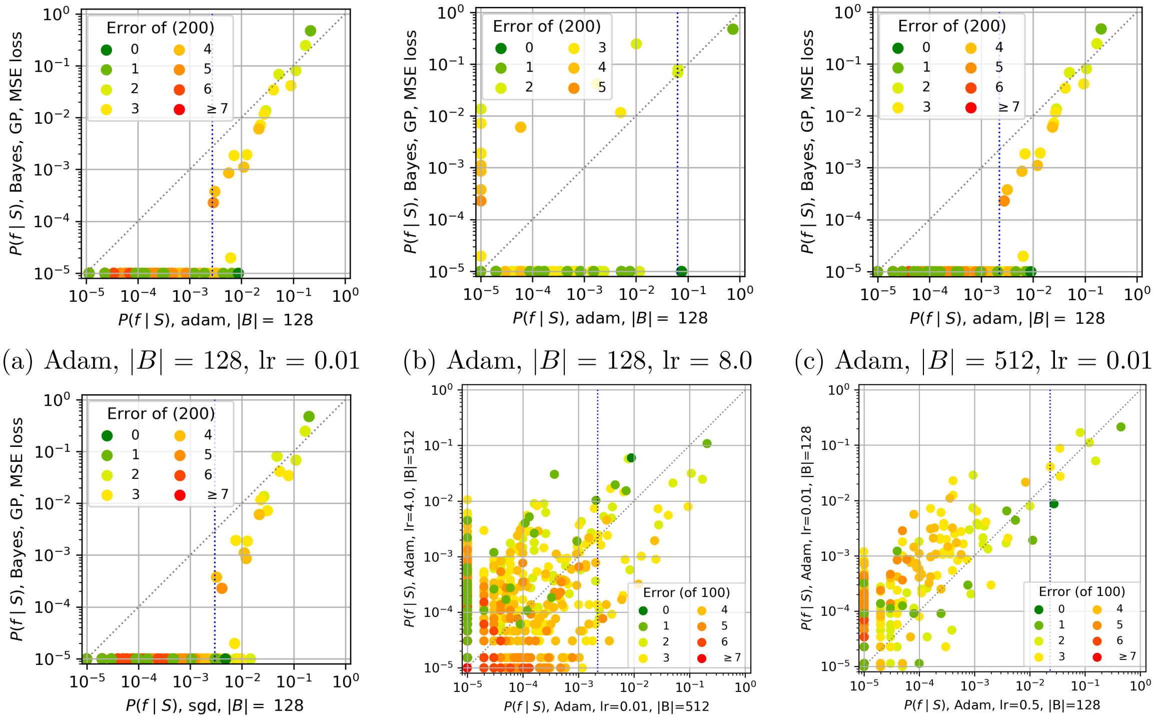
(a) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{SGD}}(f|S)$ for no corruption, $n = 10^6$, $\langle\epsilon_G\rangle = 2.64\%$ for SGD and $\langle\epsilon_G\rangle_{GP} = 3.22\%$ for the GP. (Note this relative error is larger than for the $|E| = 100$ test set because it includes a hard to classify image. Such fluctuations are expected for small test sets.

(b) Green dots denote functions in the set F, which are found by both SGD and the GP sampling. Red dots are found by GP, but not by SGD. On this scale all GP functions are found by SGD.

(c) $P_{\mathrm{B}}(f|S)$ v.s. $P_{\mathrm{SGD}}(f|S)$ for 20% corruption (as Figure 6(c), but included for ease of comparison). ($\langle\epsilon_G\rangle = 13.4\%$ for SGD and $\langle\epsilon_G\rangle_{GP} = 5.80\%$ for the GP). Here we included functions with frequency < 10. $n = 10^5$

(d) In contrast to (b), a considerable number of low probability functions are not found by SGD. Here $\sum_{f\in F} P_{\mathrm{B}}(f|S) = 99.3\%$, and $\sum_{f\in F} P_{\mathrm{SGD}}(f|S) = 24.3\%$, indicating that while SGD finds almost all functions with high $P_{\mathrm{B}}(f|S)$, it also finds many functions with low $P_{\mathrm{B}}(f|S)$.

Comparing the 0% and 20% label corruption shows that the weaker bias in the latter leads to less strong correlation between $P_{\mathrm{B}}(f|S)$ and $P_{\mathrm{SGD}}(f|S)$.



(a) Adam, $|B| = 128$, lr $= 0.01$ (b) Adam, $|B| = 128$, lr $= 8.0$ (c) Adam, $|B| = 512$, lr $= 0.01$

(d) SGD, $|B| = 32$, lr $= 0.1$ (e) $|B| = 512$, lr$= 0.01$ and 4.0 (f) $|B| = 128$, lr$= 0.5$ and 0.01

Figure 20: **The effect of the learning rate and batch size on $P_{\mathbf{OPT}}(f|S)$ v.s. $P_{\mathbf{B}}(f|S)$.** [Training/test set size: 10000/200. Variable batch sizes ($|B|$) and learning rates. 10^5 samples for all optimisers; 10^6 samples from the GP.] (a) - (d) shows $P_{\mathrm{OPT}}(f|S)$ v.s. $P_{\mathrm{B}}(f|S)$ for a selection of different optimisers (adam and SGD), learning rates and batch sizes. Adjusting the learning rate does have a noticeable effect on the functions found by the optimisers, but $P_{\mathrm{OPT}}(f|S)$ remains approximately proportional to $P_{\mathrm{B}}(f|S)$ in each example. (e) and (f) compare $P_{\mathrm{Adam}}(f|S)$ with different learning rates for fixed batch sizes (512 and 128 respectively). In (e), the learning rates compared differ by two orders of magnitude, and still show fairly strong correlation.

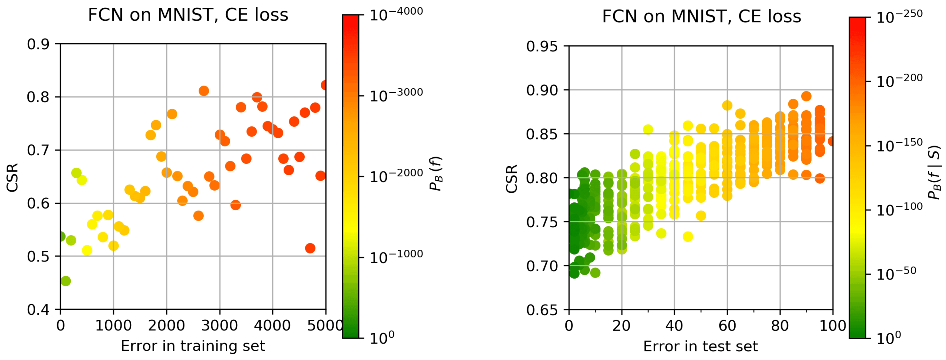
Appendix E. Critical Sample Ratio

A measure of the complexity of a function, the critical sample ratio (CSR), was introduced in (Krueger et al., 2017). 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 box of side $2r$ centred around the input, producing a different output (for discrete outputs).

Following (Valle-Pérez et al., 2018), we use CSR as a rough estimate of the complexity of functions found in the posterior (conditioning on S), and the prior (i.e. functions on S). In our experiments, we used binarised MNIST with a training set size of 10000 and a test set of size 100 (analogously to the majority of our other experiments). For the prior, Figure 21a, we randomly generated 100 functions with errors ranging between 0 and 10000 on S. For each function, we recorded the error, $P_{\mathrm{B}}(f|S)$ and the CSR. For the posterior, we generated

500 functions with a range of errors on the test set and concatenated them with the function correct on the training set. We then proceeded as with the prior.

To calculate the CSR, we trained a DNN to model the function in question. Clearly this induces effects not purely due to the parameter-function map – although as we have seen in Appendix A, the functions found by SGD are likely to be similar to those that would be found by training by random sampling of parameters. Therefore, we may expect this process to approximately give the average CSR of parameters producing the function of interest. In Figure 2a. of (Valle-Pérez et al., 2018), an example of a very similar experiment with CIFAR10 can be found, where the network was not trained. This produces very similar results to our experiments with network training.



(a) 10 samples per $\langle\epsilon\rangle$

(b) 20 samples per $\langle\epsilon\rangle$

Figure 21: (a) shows the result of our experiment on the functions in the prior; (b) shows the result out our experiment on functions in the posterior. Clearly there is a strong correlation between the CSR complexity and the error of the function; between the CSR complexity and $P_{\mathrm{B}}(f|S)$; and between the error and $P_{\mathrm{B}}(f|S)$. Different hyperparameters were used in the experiment to obtain a suitable range of complexity.

Appendix F. Bayesian inference by direct sampling on Boolean system.

In this section we consider a much simpler system, with a smaller input space, and thus a much smaller space of functions. This allows us to approximate Bayesian inference in the case of 0-1 likelihood (see Appendix A.2.2) much more accurately, via direct sampling. We use the same DNN architecture and synthetic data studied in (Valle-Pérez et al., 2018). It consists of a two layer neural network with 7 Boolean inputs, two hidden layers of 40 ReLU-activated neurons, and a single Boolean output. The space of functions is thus the space of $2^{128} \approx 3.4 \times 10^{38}$ Boolean functions of 7 inputs.

We perform ‘approximate Bayesian inference’ (ABI) by sampling the parameters of the neural network i.i.d. from a Gaussian distribution with distribution parameters[12] $\sigma_w = \sigma_b = 1.0$, evaluating the neural network on the training set, and saving the samples for which the neural network achieves 100% training accuracy. Each of these samples

[12] Remember that, following standard convention, the actual weight variance is dividing by the number of input neurons

corresponds to a function, **sampled from the *exact* Bayesian posterior**. We estimate the posterior probabilities by the empirical frequencies of individual functions (defined on all $2^7 = 128$ inputs), after sampling the parameters 10^{10} times. We used a random but fixed training set consisting of 32 out of the 128 inputs for a given target function. Target functions were chosen among functions that appeared with reasonably high frequency, in a large sample obtained by randomly sampling the weights of the neural network, so as to ensure ABI would give enough samples. They were chosen to have a range of values of Lempel-Ziv complexity. See (Valle-Pérez et al., 2018) for the definition of Lempel-Ziv complexity of Boolean functions used here.

Representative results are shown in Figures 22,23,24 (results for the other 4 functions we tested look qualitatively similar). We empirically found that the ABI probabilities correlated and are of a similar order of magnitude to the SGD probabilities over the whole range of Boolean functions tried. SGD is consistently more biased towards the most likely functions for CE loss, although it only increased their probability by about a factor of two (a small amount relative to the whole range of probabilities, but because for this system this is the dominant function, this secondary effect can still have a significant effect on the average generalisation error). We also performed sampling using the EP approximation to the posterior with 0-1 loss, and the exact posterior using MSE loss to directly see the effects coming from the EP approximation. We found that for some functions (the simplest ones) GP/EP gave probabilities which were close to those found by ABI. However, for more complex functions[13] GP/EP highly underestimates the probabilities In fact, for the most complex functions we studied, GP/EP didn't find a single function more than once in our sampling. This is in contrast to the GP/MSE sampling probabilities, which shows reasonable correlation with the ABI probabilities, as well as with the probabilities of SGD trained with MSE loss.

These results support the main hypothesis of the paper that the Bayesian posterior probabilities correlate with the SGD probabilities. They also suggest that the EP approximation can sometimes heavily underestimate the probabilities. This agrees with the conjecture that the EP approximation is the main cause of the discrepancy in the magnitudes of the probabilities between GP/EP and SGD observed in the rest of the experiments in this paper, and that for CE loss, the true Bayesian prior probabilities may in fact match the SGD probabilities a lot more closely. Furthermore, the MSE results suggest that MSE may give a good approximation to the Bayesian posterior probabilities (even the ones based on 0-1 likelihood). However, we note that this is a small toy model, and so our analysis leaves as an open question how to understand the error induced by the EP approximation in more realistic systems. One approach could be to cross-validate some of our results with state-of-the-art Monte Carlo sampling techniques for Bayesian inference (Rasmussen, 2004).

Description of figures. In Figures 22,23 and 24 in this section, we show, for three representative target Boolean functions, data comparing $P_{\mathrm{SGD}}(f|S)$ for CE or MSE loss versus different ways to estimate $P_{\mathrm{B}}(f|S)$, namely ABI, GP/EP for CE loss and GP/MSE for MSE loss.

[13] These are still rather simple w.r.t. the full range of possibilities, necessary to ensure that ABI sampling is feasible.

In the **first column**, we show scatter plots comparing sampled probabilities (where functions not found in the sample are shown as if having a frequency of one). The colours denote the number of errors for each function on the test set.

The **second column** shows probability versus rank of the different test-set functions (when ranked by probability) for the two sampling methods.

The **third column** shows test accuracy histograms for the two sampling methods.

In the **first row**, we use sampling of parameters (and 0-1 loss) to estimate $P_{\mathrm{B}}(f|S)$, which we use as the gold standard method as it has controlled small errors.

In the **second row**, we estimate $P_{\mathrm{B}}(f|S)$ with the GP/EP approximation introduced in Appendix A.2.2.

In the **third row**, we compare $P_{\mathrm{B}}(f|S)$ estimated from sampling of the exact MSE posterior (explained in Appendix A.2.1) versus the ABI sampling (for 0-1 loss).

In the **fourth row**, we compare $P_{\mathrm{SGD}}(f|S)$ when training with MSE loss versus ABI sampling (for 0-1 loss).

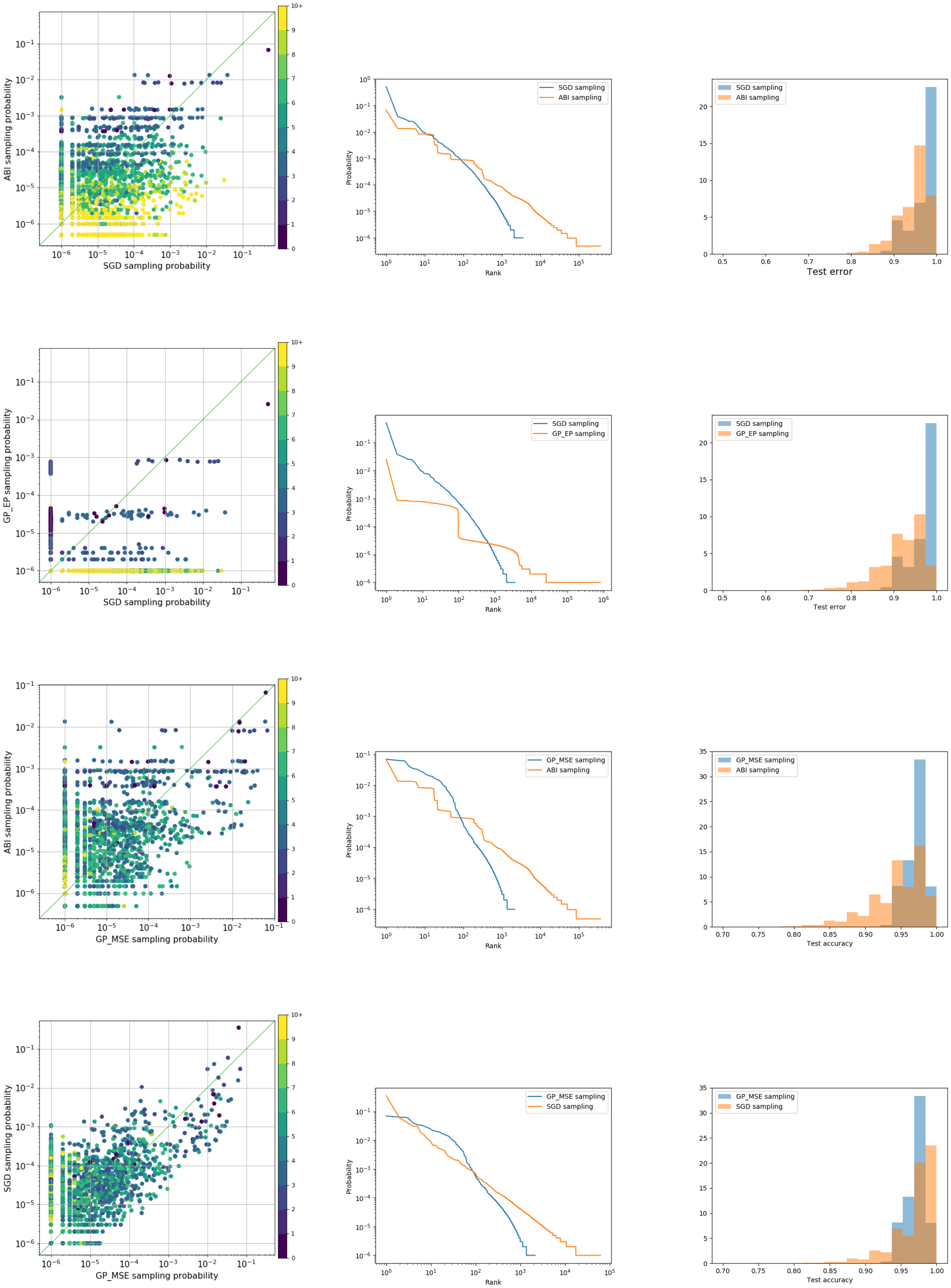


Figure 22: Results for Appendix F for target function with LZ complexity 35.0. Refer to text for detailed description.

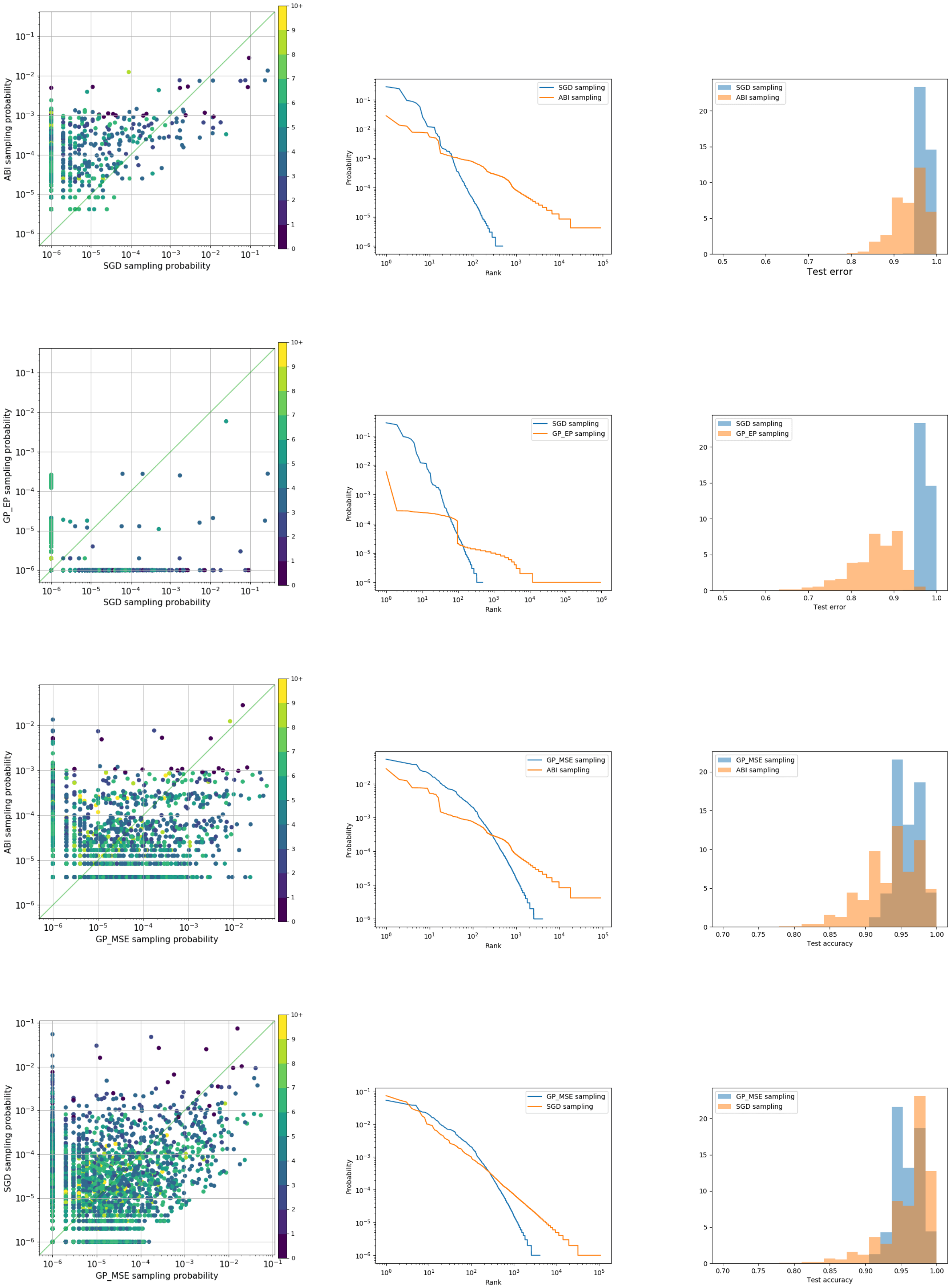


Figure 23: Results for Appendix F for target function with LZ complexity 28.0. Refer to text for detailed description.

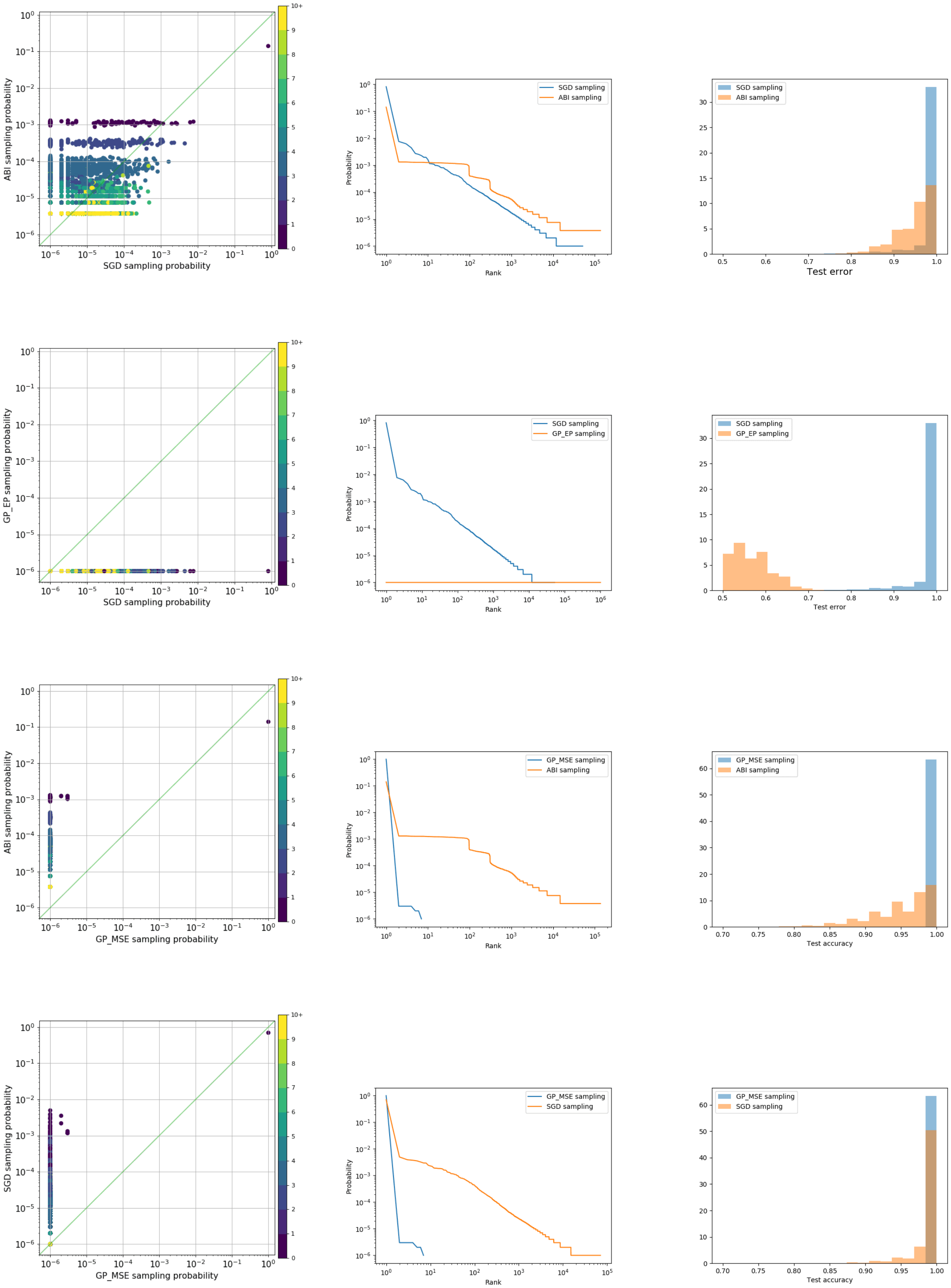


Figure 24: Results for Appendix F for a (different) target function of LZ complexity 28.0. Refer to text for detailed description.