Do Bayesian Neural Networks Need To Be Fully Stochastic?

Mrinank Sharma[†]
University of Oxford

Sebastian Farquhar
University of Oxford

Eric Nalisnick
University of Amsterdam

Tom Rainforth
University of Oxford

Abstract

We investigate the efficacy of treating *all* the parameters in a Bayesian neural network stochastically and find compelling theoretical and empirical evidence that this standard construction may be unnecessary. To this end, we prove that expressive predictive distributions require only small amounts of stochasticity. In particular, partially stochastic networks with only n stochastic biases are universal probabilistic predictors for n-dimensional predictive problems. In empirical investigations, we find no systematic benefit of full stochasticity across four different inference modalities and eight datasets; partially stochastic networks can match and sometimes even outperform fully stochastic networks, despite their reduced memory costs.

1 Introduction

Bayesian neural networks (BNNs) are often considered to be the most principled approach for uncertainty quantification in deep learning [Abdar et al., 2021; Mackay, 1992; Neal, 1996; Wilson, 2020]. Indeed, they have a simple and compelling foundation: we use neural networks to define flexible hypotheses classes of predictive functions by defining a prior over *all* their weights and biases, then perform inference to produce posterior predictive distributions.

In practice, full posterior inference for BNNs is intractable and so practitioners must resort to approximate inference schemes [Blundell et al., 2015; Daxberger et al., 2021a; Neal, 1996; Welling and Teh, 2011]. This can lead to practical behaviour that is highly distinct from that of the true posterior [Coker et al., 2021; Foong et al., 2020], while still being extremely computationally expensive.

† Correspondance to Mrinank Sharma, <mrinank@robots.ox.ac.uk>.

Preprint. Under review.

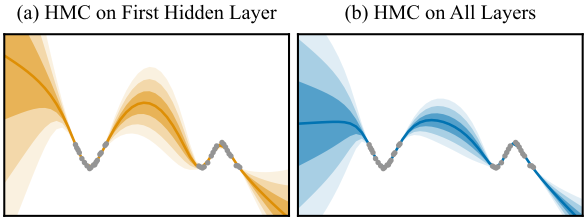

Figure 1: Perhaps surprisingly, inference over only the first hidden layer weights of a small multi-layer perceptron represents uncertainty as well as inference over all weights, whilst training c.a. 7 times faster. We first train a maximum-a-posterior network and then use Hamiltonian Monte Carlo inference over (a) the first hidden layer parameters only—other parameters are fixed—and (b) all network parameters. Lines: mean predictions. Shaded areas: predictive intervals.

To reduce these costs, the research community has recently considered partially stochastic networks [Daxberger et al., 2021a,b; Izmailov et al., 2020; Kristiadi et al., 2020; Lei et al., 2021; Ober and Rasmussen, 2019; Snoek et al., 2015]. Though promising, these approaches are usually seen as pragmatic cost-saving measures relative to more expensive but principled fully stochastic approaches. For example, Kristiadi et al. [2020] describe stochastic last-layer approaches as "approximation schemes," Daxberger et al. [2021b] see partial stochasticity as a tool for approximating the full posterior, and Ober and Rasmussen [2019] describe a compromise between "tractability and expressiveness."

In this work, we question this underlying assumption that full stochasticity is preferable to, and indeed more principled than, partial stochasticity. Despite the prevalence of this assumption, we uncover compelling theoretical and empirical evidence that suggests it may be misguided.

To begin, we first consider whether full stochasticity is necessary for our networks to be sufficiently expressive (§4). Although one may intuit that reducing the number of stochastic parameters would hamper expressivity, we prove this is not the case. In fact, many simple architectures using only a handful of stochastic parameters are universal conditional distribution approximators (UCDAs)—they can sample from any continuous conditional distribution arbitrarily well. Moreover, finite-width bounded-variance fully

stochastic layers can even destroy information about the input. These results demonstrate full stochasticity is certainly not necessary for expressive predictive distributions.

We then question whether full stochasticity can be justified by its original Bayesian formulation by examining whether approximate inference can faithfully capture the posterior. Here, we find even state-of-the-art inference schemes using impractical amounts of compute do *not* produce faithful representations (§5). Thus fully stochastic networks cannot be supported through their Bayesian formulation alone.

Of course, full stochasticity could still be a *practically* helpful construction for learning useful predictive distributions. Accordingly, we empirically investigate whether full stochasticity translates to improved predictive performance over partially stochastic networks (§6). In fact, across four inference modalities and eight datasets, we find no systematic benefit of full stochasticity; partially stochastic networks can match and sometimes even outperform fully stochastic networks, despite reduced memory costs and typically shorter training times (Fig. 1).

Overall, our work questions the prevalent assumption that full stochasticity is preferable to and more principled than partial stochasticity. We demonstrate that partially stochastic networks are no less principled than fully stochastic ones, challenging the *de facto* default model construction of full stochasticity. To summarise, our key contributions are:

(i) We show that there is no tradeoff between the number of stochastic parameters and network expressivity. In particular, we prove partially stochastic networks are universal conditional distribution approximators.

(ii) Across four inference modalities, ranging from high-fidelity Hamiltonian Monte Carlo to crude mean-field variational inference, we surprisingly demonstrate that there is no benefit for full stochasticity in terms of predictive performance. The best-performing partially stochastic network varies by inference modality.

2 Background

We focus on supervised learning problems. Let the training set be denoted as $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ with inputs $x_i \in \mathcal{X}$ and outputs $y_i \in \mathcal{Y}$. We assume the data is independently and identically drawn from an underlying distribution $P_{X,Y}$. Our task is to learn a conditional distribution $Y|X = x$.

Bayesian Neural Networks (BNNs) Let $f_\theta(x)$ be a deep neural network with parameters θ, which represent a set of weights and biases. Rather than employing empirical risk minimization to train θ, BNNs place a prior $p(\theta)$ over θ and define a likelihood, $p(y|f_\theta(x))$. By Bayes' rule, this now defines a *posterior*, $p(\theta|\mathcal{D}) \propto p(\theta)p(\mathcal{D}|\theta)$—where $p(\mathcal{D}|\theta) = \prod_i p(y_i|f_\theta(x_i))$—that represents the updated beliefs about θ given the data $\mathcal{D}$. Prediction is performed using the *posterior predictive*, $p(y|x, \mathcal{D}) = \mathbb{E}_{p(\theta|\mathcal{D})}\left[p(y|f_\theta(x))\right]$, which represents the push forward distribution of the posterior through the network for a given input x. Given that BNNs are explicitly algorithms for supervised prediction, one ultimately only cares about this posterior predictive distribution, rather than the posterior itself [Farquhar et al., 2020; Foong et al., 2020]. The properties of the posterior predictive distribution are often referred to as the "function space" properties of a BNN [Izmailov et al., 2021b].

Approximate Inference in BNNs Unfortunately, exact inference is generally intractable for BNNs. As such, practitioners resort to approximate inference, typically over all model parameters. Sampling-based approaches, such as Hamiltonian Monte Carlo (HMC) [Neal, 1996] or Stochastic Gradient Langevin Dynamics [Welling and Teh, 2011] attempt to sample from the posterior. Alternatively, traditional variational approaches [Blundell et al., 2015; Gal and Ghahramani, 2016; Mackay, 1992] learn an approximate posterior, $q(\theta; \phi) \approx p(\theta|\mathcal{D})$, for which existing methods usually make some kind of mean-field assumption over θ. Meanwhile, some modern approaches have instead looked directly to learn variational approximations of the posterior predictive [Rudner et al., 2020; Sun et al., 2019].

Partially Stochastic Networks Let $f_\Theta(x)$ be a deep neural network and define a likelihood $p(y|f_\Theta(x))$. In a partially stochastic network [Daxberger et al., 2021b; Dusenberry et al., 2020; Izmailov et al., 2020; Kristiadi et al., 2020, 2021; Lei et al., 2021; Snoek et al., 2015], we have $\Theta = \Theta_S \cap \Theta_D$. We learn point estimates for Θ_D and a distribution over Θ_S, which could be learnt jointly with the deterministic parameters or separately in a two-stage training procedure. To make predictions, we compute the *subset predictive distribution* by holding Θ_D fixed and pushing forward the distribution over Θ_S through the network.

3 Related Work

Limitations of BNNs Several works raise concerns around BNNs. Foong et al. [2020], Coker et al. [2021], and Trippe and Turner [2018] showed mean-field variational inference behaves pathologically. Others find deviating from posterior predictive, for example by sharpening the posterior [Wenzel et al., 2020] or degrading inference quality [Izmailov et al., 2021a], improves performance. Our work complements these observations. Our demonstration of inaccurate inference weakens the theoretical justification for BNNs (§5). Further, we find full stochasticity consistently does not improve predictive performance (§6), which similarly questions the value of the full network posterior predictive.

Existing Partially Stochastic Networks Partially stochastic networks are gaining popularity. Daxberger et al. [2021b] approximate full network inference by performing *expressive* inference over a carefully chosen subset of model weights. Further, Izmailov et al. [2020] perform expressive inference in an alternative probabilistic model, constructed

by projecting network parameters to a low-dimensional subspace. But we demonstrate that expressive inference is not necessary in theory (§4) and in practice (§6). Moreover, several works consider partial stochasticity as a pragmatic cost-saving measure relative to full stochasticity [Dusenberry et al., 2020; Kristiadi et al., 2020; Lei et al., 2021; Snoek et al., 2015]. We, however, question the value of full stochasticity and demonstrate partial stochasticity is no less justified than full stochasticity. Finally, we show stochastic output layers—the most popular approach—are typically not universal conditional distribution approximators (§4).

Alternative Uncertainty Quantification Approaches Other than BNNs, there are many approaches for uncertainty quantification in deep learning [Abdar et al., 2021]. Deep ensembles are popular and peformant [Lakshminarayanan et al., 2017]. Others use entirely deterministic methods [Mukhoti et al., 2021; Skafte et al., 2019; Van Amersfoort et al., 2020]. Further, Osband et al. [2021] suggest using neural networks to approximate inference in some other probabilistic model, rather than performing inference over a neural network's weights and biases. Our demonstration of inaccurate inference (§5) supports this perspective by highlighting the challenge of accurate posterior inference.

4 Expressivity of Partially Stochastic Networks

Fully stochastic networks are typically assumed to be preferable to partially stochastic networks. We now question this assumption by examining whether fully stochastic networks are necessary for theoretical *expressivity*. That is, can partially stochastic networks, in principle, approximate conditional distributions as well as fully stochastic ones? Our findings are emphatically in the affirmative: we will show that networks using only a number of random variables equal to the dimensionality of the output space are universal conditional distribution approximators.

Our theoretical results leverage the *Noise Outsourcing Lemma* [Austin, 2012; Kallenberg; Zhou et al., 2022] and the Universal Approximation Theorem (UAT) [Leshno et al., 1993]. We start by restating these results.

Lemma 1 (Noise Outsourcing Lemma [Austin, 2012; Kallenberg; Zhou et al., 2022])**.** *Let X and Y be random variables in Borel spaces $\mathcal{X}$ and $\mathcal{Y}$. For any given $m \geq 1$, there exists a random variable $\eta \sim \mathcal{N}(0, I_m)$ and a Borel-measurable function $\tilde{f} : \mathbb{R}^m \times \mathcal{X} \to \mathcal{Y}$ such that η is independent of X and*

$$(X, Y) = (X, \tilde{f}(\eta, X)) \tag{1}$$

almost surely. Thus, $\tilde{f}(\eta, x) \sim Y|X = x,\ \forall x \in \mathcal{X}$.

The noise outsourcing lemma states that conditional distribution estimation can always be reduced to learning an appropriate function $\tilde{f}$ that maps from the input and independent noise to the output. Thus, if we can learn a $\tilde{f}$, we can sample from $Y|X = x$ simply by sampling $\eta \sim N(0, I_m)$ and calculating $Y = \tilde{f}(\eta, x)$. We term $\tilde{f}$ a *generator function* of the conditional distribution $Y|X$ and note that it is not unique (e.g. we can always have $\eta' = -\eta$ and $\tilde{f}'(\eta', X) = \tilde{f}(-\eta', X)$).

Lemma 2 (Universal Approximation Theorem for Arbitrary Width Networks [Leshno et al., 1993])**.** *Let $\mathcal{X}$ be some compact subspace of $\mathbb{R}^d$ and let $\mathcal{Y} \subseteq \mathbb{R}^n$. Further, let $f_\theta : \mathcal{X} \to \mathcal{Y}$ be a fully connected neural network with one hidden layer of arbitrary width and a non-polynomial activation function, where $\theta \in \Theta$ represents the parameters of the network. Then for any arbitrary continuous function $g : \mathcal{X} \to \mathcal{Y}$ and all $\varepsilon > 0$,*

$$\exists \theta \in \Theta \ :\ \sup_{x \in \mathcal{X}} \|f_\theta(x) - g(x)\| < \varepsilon, \tag{2}$$

provided that the network is sufficiently wide.

Informally, Lemma 2 states that we can approximate any continuous function arbitrarily well with a sufficiently wide network, even if that network only has a single hidden layer.

We now combine these two ideas to present our main result below in Theorem 1, which shows that arbitrary-sized networks with a small fixed amount of stochasticity *before* their last layer are universal conditional distribution approximators. Specifically, we show that the following architectures with deterministic weights can approximate any continuous conditional distribution $Y|X = x$ arbitrarily well for all $x \in \mathcal{X} \subset \mathbb{R}^d$, where $Y \in \mathcal{Y} \subseteq \mathbb{R}^n$, using only a finite set of Gaussian random variables, $Z = \{Z_1, \ldots, Z_m\}$, $m \geq n$, that are independent of the input X and have finite mean and variance:

(i) A deterministic multi-layer perceptron (MLP) with a single hidden layer of arbitrary width; non-polynomial activation function; and which takes $[Z; X]$ as its input.

(ii) An MLP with $L = 2$ layers; continuous, invertible, and non-polynomial activation functions; d units with deterministic biases and m units with Gaussian biases in the first layer; and a second layer of arbitrary width.

(iii) An MLP with $L = 2$ layers; RELU activations; $2d$ units with deterministic biases and m units with Gaussian random biases in the first layer; and a second layer of arbitrary width.

(iv) An MLP with $L \geq 2$ layers; continuous and non-polynomial activation functions that are either invertible or RELUs; at least $2\max(d + m, n)$ units with deterministic biases in each hidden layer; finite weights and biases throughout; one non-final hidden layer with m additional units with Gaussian random biases (other layers may also have additional units with random biases, alongside their $2\max(d + m, n)$ deterministic ones), and; an arbitrary number of hidden units in one of the subsequent hidden layers.

We note that the above set of architectures is by no means exhaustive, as discussed later, but is chosen to be demonstrative of how simple architectures with universal approximation properties can be.

Theorem 1 (Universal Conditional Distribution with Finite Stochasticity)**.** *Let X be a random variable taking values in $\mathcal{X}$, where $\mathcal{X}$ is a compact subspace of $\mathbb{R}^d$, and let Y be a random variable taking values in $\mathcal{Y}$, where $\mathcal{Y} \subseteq \mathbb{R}^n$. Further, let $f_\theta : \mathbb{R}^m \times \mathcal{X} \to \mathcal{Y}$ represent one of the neural network architectures defined in (i-iv) with deterministic parameters $\theta \in \Theta$, such that, for input $X = x$, the network produces outputs $f_\theta(Z, x)$, where $Z = \{Z_1, \ldots, Z_m\}, Z_i \in \mathbb{R}$, are the random variables in the network, which are Gaussian, independent of X, and have finite mean and variance.*

If there exists a continuous generator function, $\tilde{f} : \mathbb{R}^m \times \mathcal{X} \to \mathcal{Y}$, for the conditional distribution $Y|X$, then f_θ can approximate $Y|X$ arbitrarily well. Formally, $\forall \varepsilon > 0, \lambda < \infty$,

$$\exists \theta \in \Theta, V \in \mathbb{R}^{m \times m}, u \in \mathbb{R}^m : \sup_{x \in \mathcal{X}, \eta \in \mathbb{R}^m, \|\eta\| \leq \lambda} \|f_\theta(V\eta + u, x) - \tilde{f}(\eta, x)\| < \varepsilon. \quad (3)$$

The proof is provided in the Supplement. At a high level, Theorem 1 shows that the collection of simple partially stochastic architectures (i-iv) are *Universal Conditional Distribution Approximators* (UCDAs). That is, they can form samplers which match *any continuous* target conditional distribution, $Y|X = x$, arbitrarily well: in principle, they can learn to do any probabilistic predictive task perfectly.

The high-level basis for the proof is to show a) that if our network can represent $[Z; x]$ exactly in one of its hidden layers and the downstream network is a universal deterministic approximator (as per Lemma 2), then it forms a UCDA, and then b) that each of the architectures (i-iv) satisfy these conditions. Note that the distribution over the random biases in these networks does not need to be learned: we only require the presence of some random noise that can be detached from the input, and the remainder of the network to be able to approximate the conditional generating function $\tilde{f}$.

Many other partially stochastic networks will also satisfy these conditions and thus form UCDAs, though it is difficult to exactly characterize this set. In practice, we expect *most* partially stochastic networks to form UCDAs, provided that they are sufficiently large, maintain some deterministic (or arbitrarily low variance) units in each layer, and have some stochasticity *before* the final layer. One could extend our results to more complex architectures, such as those that are not fully connected (e.g. CNNs [LeCun et al., 1995]) and/or which make use of skip connections (e.g. ResNets [He et al., 2016] and DenseNets [Iandola et al., 2014]). Meanwhile, Z being non-Gaussian should also be perfectly viable, provided it is measurable with respect to a m-dimensional Lebesgue measure with a continuous density function.

The following property is important to note in this generalization to other architectures.

Remark 1. *If a continuous generator function exists for independent random noise of dimension p, then one also exists for any higher noise dimension $q > p$.*

This follows directly from the fact that the generator can simply ignore some of the noise variables. As such, we can always add more units with stochastic biases and weights to a network without undermining universality. However, this does not necessarily mean we can *replace* the existing deterministic units with stochastic ones and still maintain universality. Our results thus explicitly *do not* ratify the standard BNN case, where all the weights and biases are stochastic with *bounded* means and variances: our construction relies on being able to perfectly reconstruct X, which is typically not possible when using a fully stochastic layer. In other words, finite-width fully stochastic layers can, in principle, destroy required information about the input.

Discussion of Assumptions Other than considerations about the architecture itself, the key assumption made by Theorem 1 is that a *continuous* generator function exists for the conditional distribution we are approximating, $Y|X$. Thankfully, this is generally a weak assumption; it is the analogue of the need for a continuous target in the UAT. One can think of it as a formalization of the need for the distribution $Y|X$ itself to be continuous.

Though not an explicit condition of the theorem itself, the architectures we consider further assume that the number of stochastic variables in the network m is greater than or equal to the output dimension n. This is because it is difficult, albeit not necessarily impossible, for a generator function to be continuous when mapping from lower-dimensional noise to a higher-dimensional output. However, if Y is measurable with respect to an n-dimensional Lebesgue measure, then a continuous generator function will usually exist for exactly $m = n$ dimensional noise (and thus all $m \geq n$ by Remark 1), if one exists at all. For example, we can consider sampling each dimension of Y autoregressively using the inverse cumulative density functions of the conditionals $Y_j|X, Y_{<j}$, whenever these all exist and are continuous.

Comparison to Previous Results Our results share some similarities to previous expressivity results on *fully* stochastic BNNs, most notably those of Farquhar et al. [2020] and Foong et al. [2020], who argued that deep, fully stochastic, mean-field BNNs are expressive. Their results rely on taking some weights in the network to the zero variance limit, so they are no longer fully stochastic. Thus, though their motivations, formulations, and conclusions are quite different to our own, their results are highly compatible with ours and can be viewed as indirectly hinting at the potential benefits of partially stochastic networks.

Classification Problems Classification problems have discrete $\mathcal{Y}$ that will clearly not satisfy our assumption of a

continuous generator function from $\mathbb{R}^m \times \mathcal{X}$. Thankfully, UCDA can be achieved even more easily here by simply regressing the class probabilities $P(Y = k|X = x)$ with a deterministic network, followed by making a simple draw of the class from this categorical distribution (which can be achieved with a single, one-dimensional, random draw).

Stochastic Last Layer Networks are *not* UCDAs As an aside, we also consider the expressivity of partially stochastic networks where only the last layer of the network is stochastic. Such approaches are used quite commonly in practice with notable success [Daxberger et al., 2021a; Kristiadi et al., 2020; Ober and Rasmussen, 2019; Snoek et al., 2015], often allowing tractable inference. However, such architectures will generally *not* be UCDAs (except for classification problems) because their distributional form of $Y|X = x$ is limited to a linear mapping of the weights and biases in the last layer. For example, if their distribution on weights and biases is Gaussian, this will induce a Gaussian distribution on $Y|X = x$ as well. Though this certainly does not undermine the usefulness of such approaches, it does highlight that care is required in their deployment.

5 Does Bayesian Reasoning Support Fully Stochastic Networks?

Although we have seen that fully stochastic networks are not necessary in terms of their theoretical expressivity, their use could alternatively be supported through their conformance to Bayesian principles. Indeed, following a strict Bayesian approach, one would place a prior distribution over all unknown parameters and then perform inference over each of them. This corresponds to a fully stochastic network. Such an approach could be justified through one or more of the following benefits: (a) the ability to naturally include prior beliefs through subjective prior distributions [Neal, 1996]; (b) improved uncertainty by averaging over different hypotheses consistent with observed data [Wilson, 2020]; and (c) coherent updates to uncertainty when observing data [Jaynes, 2003]. We now examine these benefits in turn.

First, with regard to (a), standard practice is to use vague parameter-space priors [Fortuin et al., 2021]. But these priors are chosen for convenience, not because they well capture our beliefs. Indeed, several studies have raised serious concerns about the suitability of current BNN prior distributions [Noci et al., 2021; Wenzel et al., 2020].

Similarly, (b) does not provide support for full stochasticity. We can average over multiple hypotheses consistent with the data with partially stochastic networks.

Finally, though (c) could still support full stochasticity, it is highly dependent on our ability to perform inference accurately. In particular, our approximations cannot be said to capture uncertainty in a "principled" Bayesian way if they vary significantly from true posterior. As such, it is natural to wonder: just how challenging is accurate inference in fully stochastic networks?

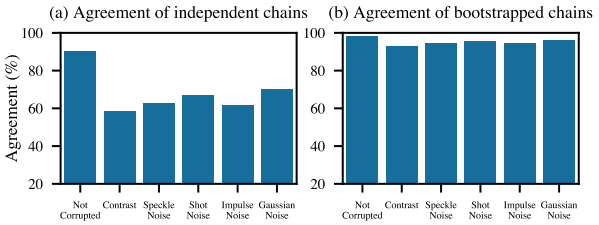

Figure 2: **Assessment of function space mixing of ResNet-20-FRN Hamiltonian Monte Carlo (HMC) samples trained on CIFAR-10.** We measure the variability in predictions across HMC chains released by Izmailov et al. [2021b]. We consider the CIFAR-10 test set and selected corruptions from the CIFAR-10-C dataset [Hendrycks and Dietterich, 2018]. (a) We compute the percentage of points that all three original chains make the same prediction on. (b) To account for the finite sample size, we measure the variability across simulated chains formed by resampling the first HMC chain (bootstrapping). The agreement of bootstrapped HMC chains is greater than 94% across all data considered.

To provide some insight, we revisit the posterior samples released by Izmailov et al. [2021b], who used full-batch HMC and 512 Tensor processing units—a deliberately extreme computing effort. As they do, we assess the variability of predictions across HMC chains. If each chain is well exploring the posterior predictive, the predictions made by each chain ought to agree. To assess the variability of predictions associated with the finite sample size, we resample the first HMC chain with replacement. Unlike Izmailov et al. [2021b], we focus on out-of-distribution (OOD) data, where poor function space mixing may manifest more strongly.

We compute the percentage of data points on which *all* chains produce the same prediction.[1] As shown in Fig. 2a, while the chains agree on 90% of the CIFAR-10 test set, the agreement falls to less than 60% on certain OOD corruptions. However, the agreement of the bootstrapped samples is consistently above 94% (Fig. 2b). The variability of predictions between chains far exceeds the variability of predictions within each chain, suggesting that each HMC chain is not well exploring the full posterior predictive distribution. Thus, additional chains would likely sample from previously unexplored regions of the posterior predictive.

Even with astronomical compute and a state-of-the-art unbiased inference scheme, we see that accurate posterior inference remains elusive. But practical methods tend to use biased and crude posterior approximations, aggravating these concerns and leading to pathological behaviour [Coker

[1] This is different to the agreement metric of Izmailov et al. [2021b], who report the percentage of data points on which one chain and the ensemble of the other two chains agree.

et al., 2021; Farquhar and Gal, 2019; Foong et al., 2020; Trippe and Turner, 2018; Wenzel et al., 2020].

Overall, we conclude that the use of fully stochastic methods can *not* be justified by their Bayesian formulation, at least not with current inference methods. Of course, this does not undermine the use of fully stochastic networks in and of itself. But, it does suggest adopting a holistic viewpoint, such as that of Osband et al. [2021], and focusing on developing methods that yield networks with the desired practical behaviours, rather than implicitly assuming that full approximate inference should be our ultimate aim.

6 Does Full Stochasticity Improve Predictions in Practice?

We saw that full stochasticity is unnecessary for theoretical expressivity (§4). Further, such networks cannot be supported through their Bayesian formulation alone (§5). Nevertheless, one could hypothesize that full stochasticity is *practically* useful for learning performant predictive distributions. We now examine this hypothesis: does full stochasticity improve predictive performance in practice?

Across four inference modalities and eight datasets, we find **no systematic benefit of full stochasticity**. In fact, there usually exist **partially stochastic networks that outperform fully stochastic ones**. Moreover, while previous work often argues that reducing stochasticity improves performance by enabling higher-fidelity inference [Daxberger et al., 2021b; Izmailov et al., 2020], we show partially stochastic networks can outperform full stochastic networks, even when both networks use the same posterior approximation families over their stochastic parameters.

Partially Stochastic Network Strategies Although there are many ways to train partially stochastic networks, here, we focus on the following relatively simple strategies:

(i) *Two-stage training*. All parameters of the network are trained deterministically e.g., using MAP inference with prior $p_1(\Theta) = p_1(\Theta_S, \Theta_D)$. We perform (approximate) inference over the stochastic subset, targeting $p(\Theta_S|\mathcal{D}; \Theta_D) \propto p_2(\Theta_S) \prod_i p(y_i|f_{\Theta_S \cap \Theta_D}(x_i))$. The stochastic subset could be chosen before or after deterministic training. We could also modify the prior over Θ_S i.e., have $p_2(\Theta_S) \neq \int p_1(\Theta_S, \Theta_D)\ \Theta_D$. Here, we consider two-stage partially stochastic variants of Hamiltonian Monte Carlo [Neal, 1996] (§6.1,6.2), Laplace Approximation [Mackay, 1992] (§6.3) and SWAG [Maddox et al., 2019] (§6.4).

(ii) *Joint training*. Alternatively, we can choose the stochastic subset *a priori*, and jointly train Θ_D and $q_\Phi(\Theta_S)$. Here, we use partially stochastic variational inference [Blundell et al., 2015; Graves, 2011; Hinton and Van Camp, 1993] (§6.1,6.5), where Θ_D and Φ are learnt by maximising the evidence lower bound.

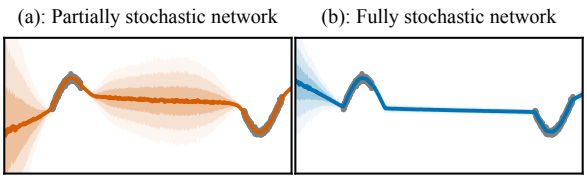

Figure 3: **1D regression with fully and partially stochastic mean-field variational inference.** The partially stochastic network has only a stochastic output layer. Lines: mean predictions. Shaded areas: $\pm\sigma, \pm 2\sigma, \pm 3\sigma$ predictive intervals.

We note that these strategies do not directly target the full network predictive. As such, these partially stochastic networks *do not* approximate the full network predictive distribution. In this section, we will examine whether their predictive distributions are useful in their own right.

6.1 1D Regression with Hamiltonian Monte Carlo and Variational Inference

To visually understand the effects of full and partial stochasticity, we first consider 1D regression. We consider both high-fidelity inference with Hamiltonian Monte Carlo (HMC) on a small dataset (c.a. 50 datapoints) and relatively crude approximate inference with mean-field variational inference (MFVI) on a larger dataset (c.a. 1000 datapoints). We use a two hidden layer MLP with independent $\mathcal{N}(0, \sigma^2)$ priors over the network's weights and biases.

First, on the smaller dataset, we train a deterministic MAP network. We then perform HMC over the first hidden layer weights (others fixed), and also over all weights. We follow Daxberger et al. [2021b] and increase the partially stochastic network's prior variance when performing HMC, also using $\sigma^2_{\text{PS}} = \sigma^2_{\text{FS}} \cdot |\Theta|/|\Theta_S|$. σ^2_{PS} and σ^2_{FS} represent the prior variance for the partially and fully stochastic network.

Examining the predictions (Fig. 1), we find that **both networks well capture in-between uncertainty**, but the partially stochastic network trains c.a. 7 times faster. Full stochasticity does not necessarily lead to substantially improved predictions, even under high-fidelity inference.

Second, on the larger dataset, we use MFVI to train a fully stochastic network and a partially stochastic network that uses only a stochastic output layer.

We find that the fully stochastic network does not well capture in-between uncertainty (Fig. 3b), even though the network is expressive enough to do so [Farquhar et al., 2020; Foong et al., 2020]. In contrast, **the partially stochastic network represents far more in-between uncertainty than the fully stochastic network** (Fig. 3a), whilst also using 200 times fewer stochastic parameters. Further, both networks use *the same* crude mean-field approximate posterior, showing that higher fidelity inference is not necessary

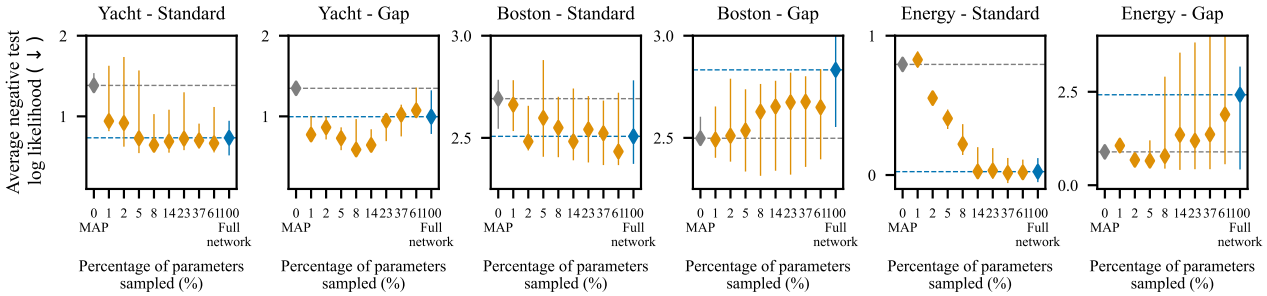

Figure 4: **UCI regression with Hamiltonian Monte Carlo (HMC).** We use a small MLP with high-fidelity HMC inference. The partially stochastic networks first train a deterministic MAP solution, and then sample only the weights that had the largest absolute value under that MAP solution; the remaining weights are fixed at their MAP value. We consider both standard splits and gap splits [Foong et al., 2019]. Diamonds: median across 15 train-test splits. Lines: interquartile range.

for partially stochastic networks to improve performance.

6.2 UCI Regression with Hamiltonian Monte Carlo

We next investigate the effect of increasing stochasticity under high-fidelity inference. That is, how does changing the number of stochastic parameters affect predictive performance? We thus use a small MLP and HMC inference on UCI regression datasets. Here, we consider partially stochastic networks with increasing numbers of stochastic parameters that are trained with two-stage HMC. That is, we first train a MAP network, and then form different stochastic networks by performing HMC over different subsets of parameters. We choose the stochastic subset by picking the weights and biases that had the maximum absolute value under the trained MAP solution. To understand the generalisation properties of these networks, we additionally consider the "gap" data splits from Foong et al. [2019]. To create these splits, we order the data by a chosen input feature, and use the central 10% as the test set. In contrast, the standard splits are created by uniformly sampling the dataset.

We first consider how increasing stochasticity affects predictive performance on the standard splits (Fig. 4). We find that increasing the number of sampled parameters first improves performance, but then **the benefits of further increasing stochasticity plateau**.

Furthermore, on the gap datasets, we find that **increasing stochasticity first improves and then degrades performance**. The underwhelming performance of high-fidelity inference with fully stochastic BNNs on out-of-distribution (OOD) data is reminiscent of observations by Izmailov et al. [2021a], who found that even MAP inference can outperform high-fidelity HMC on OOD data.

Together, these results demonstrate that partially stochastic networks can match and even outperform fully stochastic networks, *even when we can perform high-fidelity inference*.

6.3 Image Classification with Laplace Approximation

We now evaluate full and partial stochasticity in larger models. To do so, we consider Laplace Approximation networks

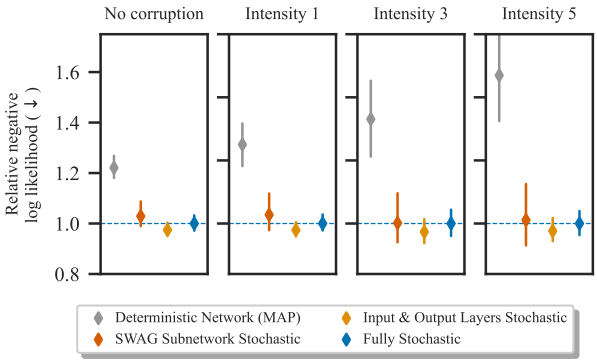

Figure 5: **Image classification with the Laplace Approximation**. We compute the average negative log-likelihood on CIFAR-10 and CIFAR-10-C relative to the fully stochastic network. Results are averaged across corruptions and shown for different corruption intensities. Markers and lines show mean and std. over 10 seeds.

on CIFAR-10 using a WideResNet-16-4. We use two-stage training, first training a MAP solution and then using post hoc Laplace approximations on subsets of model parameters. We primarily use KFAC covariance approximations [Ritter et al., 2018]. We also consider using a full covariance approximation using the stochastic subset selection strategy proposed by Daxberger et al. [2021b]—selecting parameters with the largest posterior variance under a diagonal SWAG approximation. To evaluate the networks, we compute the holdout likelihood for various networks on the CIFAR-10 and CIFAR-10-C corrupted datasets.

When comparing the relative performance between the fully stochastic network and a partially stochastic network where only the input and output layer is stochastic (Fig. 5), we find that **the partially stochastic network slightly outperforms the fully stochastic network**. This may be surprising since *both networks use the same KFAC posterior approximation* over their stochastic parameters, but the partially stochastic network has 900 times fewer of them and predicts faster.[2]

[2]Although the partially stochastic network has a stochastic

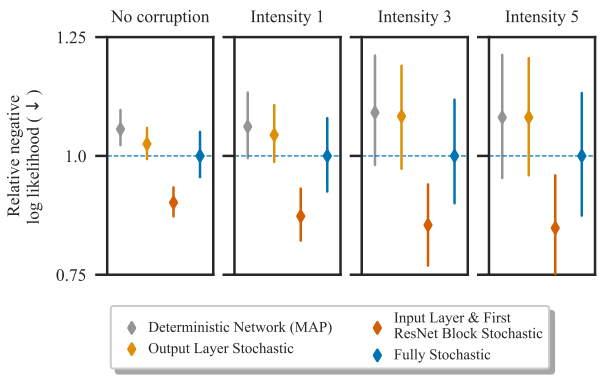

Figure 6: **Image classification with SWAG inference.** We compute the average negative log-likelihood on CIFAR-10 and CIFAR-10-C relative to the fully stochastic network. Results are additionally averaged across corruptions, and shown for different corruption intensities. Markers and lines show mean and std. over 10 seeds.

Moreover, despite the additional costs of subnetwork selection, the increased expressivity of the posterior approximation family, and increased numbers of stochastic parameters, the 'SWAG subnetwork stochastic' network actually *underperforms* the stochastic input and output layer network.

6.4 Image Classification with SWAG

We now investigate the effects of full and partial stochasticity under a different inference modality. We use SWA-Gaussian (SWAG, Maddox et al. [2019]), which runs high learning rate stochastic gradient descent (SGD) starting from a set of pre-trained weights. The approximate posterior is formed by fitting a low-rank Gaussian to the SGD iterates. For the partially stochastic networks, we perform SGD only on the stochastic subset i.e., particular subsets of model parameters. We use the default hyperparameters from Maddox et al. [2019] for SWAG with pre-trained weights, except that we tune the learning rate for each network separately. As before, we use a WideResNet-16-4 and evaluate the holdout likelihood on CIFAR-10 and CIFAR-10-C.

When comparing the relative performance across networks (Fig. 6), we find that the fully stochastic network outperforms the deterministic network, particularly on large corruption intensities. We further find **SWAG inference only over the input layer and the first ResNet block consistently outperforms the fully stochastic network**. Even though the fully stochastic network marginalises over more parameters, and thus over presumably more diverse functions, it surprisingly seems to perform worse than the partially stochastic network, despite 11x higher memory costs.

input layer, it is much faster than the fully stochastic network at prediction time because we use *linearised* predictive distributions.

Table 1: **Partially and fully stochastic networks trained with mean-field variational inference.** We report the accuracy and average negative log-likelihood (NLL) on the CIFAR test set when performing subset VI and learning the remaining parameters by maximising the (penalised) ELBO. Results are averaged across 3 random seeds.

	CIFAR10		CIFAR100	
Model	Acc (%)	NLL	Acc (%)	NLL
Deterministic	95.6	0.19	79.3	0.86
Fully stochastic	94.7	0.21	77.7	0.94
Input layer stochastic	**95.7**	0.19	**79.5**	0.86
Output layer stochastic	95.6	0.19	78.9	0.93
Output layer and last block stochastic	95.6	**0.17**	79.0	**0.83**

6.5 Image Classification with Variational Inference.

Finally, we investigate the effects of full and partial stochasticity on even larger networks. We apply MFVI on CIFAR-10 and CIFAR-100 with a Wide-ResNet-28-10, using the reference implementation from Nado et al. [2021]. We report the accuracy and negative log-likelihood. Strengthening our comparison, note that we re-used the tuned hyperparameters for the fully stochastic and deterministic networks from Nado et al. [2021], but did not tune the hyperparameters for the partially stochastic networks.

We find the fully stochastic network performed worse than the deterministic network, despite using twice as many parameters. In contrast, even without tuned hyperparameters, **the partially stochastic networks outperform the fully stochastic network**. The stochastic input layer performs best in terms of accuracy, and the network where the last block and output layer performs best in terms of NLL. In particular, we emphasise the potential of stochastic input layers rather than the more commonly considered stochastic output layers. In each case, the partially stochastic networks use only slightly more parameters than deterministic networks.

7 Discussion

We questioned the prevalent assumption that full stochasticity is preferable to and more principled than partial stochasticity. We found full stochasticity is not needed for theoretical expressivity (§4). Further, across four inference modalities, we did not find full stochasticity to yield consistent improvements in predictive performance (§6). In fact, there usually existed partially stochastic networks that outperformed their corresponding fully stochastic variants. Altogether, our results call into question full stochasticity as the *de facto* default model construction. We believe partially stochastic networks are a highly promising model class that are just as principled as fully stochastic networks. Furthermore, our observations around inaccurate inference in large

BNNs (§5) support holistic viewpoints such as those of Osband et al. [2021], which set aside posterior inference and instead focus on learning useful predictive distributions.

References

M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

T. Austin. Exchangeable random arrays. In *Notes for IAS workshop*, 2012.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

B. Coker, W. Pan, and F. Doshi-Velez. Wide Mean-Field Variational Bayesian Neural Networks Ignore the Data. *arXiv preprint arXiv:2106.07052*, 2021.

E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. Laplace Redux-Effortless Bayesian Deep Learning. *Advances in Neural Information Processing Systems*, 34, 2021a.

E. Daxberger, E. Nalisnick, J. U. Allingham, J. Antorán, and J. M. Hernández-Lobato. Bayesian deep learning via subnetwork inference. In *International Conference on Machine Learning*, pages 2510–2521. PMLR, 2021b.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

M. Dusenberry, G. Jerfel, Y. Wen, Y. Ma, J. Snoek, K. Heller, B. Lakshminarayanan, and D. Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In *International conference on machine learning*, pages 2782–2792. PMLR, 2020.

S. Farquhar and Y. Gal. A unifying bayesian view of continual learning. *arXiv preprint arXiv:1902.06494*, 2019.

S. Farquhar, L. Smith, and Y. Gal. Liberty or depth: Deep bayesian neural nets do not need complex weight posterior approximations. *Advances in Neural Information Processing Systems*, 33:4346–4357, 2020.

A. Foong, D. Burt, Y. Li, and R. Turner. On the expressiveness of approximate inference in bayesian neural networks. *Advances in Neural Information Processing Systems*, 33:15897–15908, 2020.

A. Y. Foong, Y. Li, J. M. Hernández-Lobato, and R. E. Turner. 'in-between'uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019.

V. Fortuin, A. Garriga-Alonso, F. Wenzel, G. Rätsch, R. Turner, M. van der Wilk, and L. Aitchison. Bayesian neural network priors revisited. *arXiv preprint arXiv:2102.06571*, 2021.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

A. Graves. Practical variational inference for neural networks. *Advances in neural information processing systems*, 24, 2011.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

D. Hendrycks and T. Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations*, 2018.

D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

D. Hendrycks, N. Carlini, J. Schulman, and J. Steinhardt. Unsolved problems in ml safety, 2021. URL https://arxiv.org/abs/2109.13916.

G. E. Hinton and D. Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

M. D. Hoffman and A. Gelman. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo, 2011. URL https://arxiv.org/abs/1111.4246.

F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

P. Izmailov, W. J. Maddox, P. Kirichenko, T. Garipov, D. Vetrov, and A. G. Wilson. Subspace inference for Bayesian deep learning. In *Uncertainty in Artificial Intelligence*, pages 1169–1179. PMLR, 2020.

P. Izmailov, P. Nicholson, S. Lotfi, and A. G. Wilson. Dangers of Bayesian model averaging under covariate shift. *Advances in Neural Information Processing Systems*, 34, 2021a.

P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson. What are Bayesian neural network posteriors really like? In *International Conference on Machine Learning*, pages 4629–4640. PMLR, 2021b.

A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

E. T. Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.

O. Kallenberg. *Foundations of modern probability*, volume 2. Springer.

R. Krishnan, P. Esposito, and M. Subedar. BayesianTorch: Bayesian neural network layers for uncertainty estimation. https://github.com/IntelLabs/bayesian-torch, Jan. 2022. URL https://doi.org/10.5281/zenodo.5908307.

A. Kristiadi, M. Hein, and P. Hennig. Being Bayesian, even just a bit, fixes overconfidence in relu networks. In *International conference on machine learning*, pages 5436–5446. PMLR, 2020.

A. Kristiadi, M. Hein, and P. Hennig. Learnable uncertainty under laplace approximations. In *Uncertainty in Artificial Intelligence*, pages 344–353. PMLR, 2021.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

S. Lei, Z. Tu, L. Rutkowski, F. Zhou, L. Shen, F. He, and D. Tao. Spatial-Temporal-Fusion BNN: Variational Bayesian Feature Layer. *arXiv preprint arXiv:2112.06281*, 2021.

M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

D. J. C. Mackay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.

W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

J. Mukhoti, A. Kirsch, J. van Amersfoort, P. H. Torr, and Y. Gal. Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. *arXiv e-prints*, pages arXiv–2102, 2021.

Z. Nado, N. Band, M. Collier, J. Djolonga, M. W. Dusenberry, S. Farquhar, Q. Feng, A. Filos, M. Havasi, R. Jenatton, et al. Uncertainty Baselines: Benchmarks for uncertainty & robustness in deep learning. *arXiv preprint arXiv:2106.04015*, 2021.

R. M. Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 1996.

L. Noci, K. Roth, G. Bachmann, S. Nowozin, and T. Hofmann. Disentangling the Roles of Curation, Data-Augmentation and the Prior in the Cold Posterior Effect. *Advances in Neural Information Processing Systems*, 34, 2021.

S. W. Ober and C. E. Rasmussen. Benchmarking the neural linear model for regression. *arXiv preprint arXiv:1912.08416*, 2019.

I. Osband, Z. Wen, M. Asghari, M. Ibrahimi, X. Lu, and B. Van Roy. Epistemic neural networks. *arXiv preprint arXiv:2107.08924*, 2021.

H. Ritter, A. Botev, and D. Barber. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6. International Conference on Representation Learning, 2018.

T. G. Rudner, Z. Chen, and Y. Gal. Rethinking function-space variational inference in Bayesian neural networks. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2020.

S. Russell. *Human compatible: Artificial intelligence and the problem of control*. Penguin, 2019.

N. Skafte, M. Jørgensen, and S. Hauberg. Reliable training and estimation of variance networks. *Advances in Neural Information Processing Systems*, 32, 2019.

J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. Patwary, M. Prabhat, and R. Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

S. Sun, G. Zhang, J. Shi, and R. Grosse. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

B. Trippe and R. Turner. Overpruning in variational bayesian neural networks. *arXiv preprint arXiv:1801.06230*, 2018.

J. Van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal. Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning*, pages 9690–9700. PMLR, 2020.

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

Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*, 2018.

F. Wenzel, K. Roth, B. S. Veeling, J. Świątkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.

A. G. Wilson. The case for Bayesian deep learning. *arXiv preprint arXiv:2001.10995*, 2020.

S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

H. Zhang, Y. N. Dauphin, and T. Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

X. Zhou, Y. Jiao, J. Liu, and J. Huang. A deep generative approach to conditional sampling. *Journal of the American Statistical Association*, pages 1–12, 2022.

Do Bayesian Neural Networks Need To Be Fully Stochastic? Supplementary Materials

A Proofs

We provide a proof of Theorem 1, which states that a number of architectures are universal conditional distribution approximators (UCDAs). First, we restate the architectures that we consider and our theorem statement for convenience. The architectures that we consider are:

[a] A deterministic multi-layer perceptron (MLP) with a single hidden layer of arbitrary width; non-polynomial activation function; and which takes $[Z; X]$ as its input.

[b] An MLP with $L = 2$ layers; continuous, invertible, and non-polynomial activation functions; d units with deterministic biases and m units with Gaussian random biases in the first layer; and a second layer of arbitrary width.

[c] An MLP with $L = 2$ layers; RELU activations; $2d$ units with deterministic biases and m units with Gaussian random biases in the first layer; and a second layer of arbitrary width.

[d] An MLP with $L \geq 2$ layers; continuous and non-polynomial activation functions that are either invertible or RELUs; at least $2\max(d + m, n)$ units with deterministic biases in each hidden layer; finite weights and biases throughout; one non-final hidden layer with m additional units with Gaussian random biases (other layers may also have additional units with random biases, alongside their $2\max(d + m, n)$ deterministic ones), and; an arbitrary number of hidden units in one of the subsequent hidden layers.

We recall Theorem 1.

Theorem 1 (Universal Conditional Distribution with Finite Stochasticity)**.** *Let X be a random variable taking values in $\mathcal{X}$, where $\mathcal{X}$ is a compact subspace of $\mathbb{R}^d$, and let Y be a random variable taking values in $\mathcal{Y}$, where $\mathcal{Y} \subseteq \mathbb{R}^n$. Further, let $f_\theta : \mathbb{R}^m \times \mathcal{X} \to \mathcal{Y}$ represent one of the neural network architectures defined in (i-iv) with deterministic parameters $\theta \in \Theta$, such that, for input $X = x$, the network produces outputs $f_\theta(Z, x)$, where $Z = \{Z_1, \ldots, Z_m\}$, $Z_i \in \mathbb{R}$, are the random variables in the network, which are Gaussian, independent of X, and have finite mean and variance.*

If there exists a continuous generator function, $\tilde{f} : \mathbb{R}^m \times \mathcal{X} \to \mathcal{Y}$, for the conditional distribution $Y|X$, then f_θ can approximate $Y|X$ arbitrarily well. Formally, $\forall \varepsilon > 0, \lambda < \infty$,

$$\exists \theta \in \Theta, V \in \mathbb{R}^{m \times m}, u \in \mathbb{R}^m : \\ \sup_{x \in \mathcal{X}, \eta \in \mathbb{R}^m, \|\eta\| \leq \lambda} \|f_\theta(V\eta + u, x) - \tilde{f}(\eta, x)\| < \varepsilon. \tag{3}$$

Proof. We start by noting that for any Gaussian $Z \in \mathbb{R}^m$, there must be some invertible matrix $V \in \mathbb{R}^{m \times m}$ and vector $u \in \mathbb{R}^m$ such that $Z = V\eta + u$, where $\eta \sim \mathcal{N}(0, I_m)$ can be used as the noise input to our generator function. This is essentially a reparameterization, and it allows us to express $f_\theta(Z, x)$ as $f_\theta(V\eta + u, x)$.

We next show that if our network is able to represent the vector $[Z; x]$ exactly in one layer and the downstream subnetwork is a universal function approximator as per Lemma 2, this provides a sufficient condition for the result to hold.

More formally, assume that the all of the following hold for some hidden layer, $h_\ell \in \mathcal{H}_\ell \subset \mathbb{R}^\ell$,

1. Z and x are fully input into the network by this layer;
2. h_ℓ is compact provided $[Z; X]$ is itself is compact;
3. h_ℓ can exactly represent $[Z; x]$ in the sense that there is some deterministic, surjective, and continuous function, $g : \mathcal{H}_\ell \to \mathbb{R}^m \times \mathcal{X}$, such that $g(h_\ell)$ recovers $[Z; x]$ exactly for all h_ℓ.
4. The downstream network $f_\theta^{>\ell}(h_\ell)$ satisfies the assumptions of Lemma 2.

Invoking Lemma 2 for approximating the function $\tilde{f}\left([V^{-1}; \mathbf{0}](g(h_\ell) - [u; \mathbf{0}]), [\mathbf{0}; I_d]g(h_\ell)\right) = \tilde{f}(\eta, x)$ (noting that $\tilde{f}$ is continuous by assumption in the Theorem) gives

$$\forall \varepsilon > 0,\ \exists \theta\ :\ \sup_{h_\ell \in \mathcal{H}_\ell} \| f_\theta^{>\ell}(h_\ell) - \tilde{f}\left([V^{-1}; \mathbf{0}](g(h_\ell) - [u; \mathbf{0}]), [\mathbf{0}; I_d]g(h_\ell)\right) \| < \varepsilon. \tag{4}$$

Now by the first assumption, h_ℓ must itself be a function of $[Z; x] = [V\eta + u; x]$, so we can rewrite the above as

$$\forall \varepsilon > 0, \lambda < \infty\ \exists \theta\ :\ \sup_{x \in \mathcal{X}, \eta \in \mathbb{R}^m, \|\eta\| < \lambda} \| f_\theta(V\eta + u, x) - \tilde{f}(\eta, x) \| < \varepsilon,$$

which is the desired result, with V and u taking on the values required for $Z = V\eta + u$. Here λ and the assumption $\|\eta\| < \lambda$ have been introduced to ensure that $[Z; x]$ is itself compact, noting this further requires the assumption made in the theorem itself that Z has finite mean and variance.

To complete the proof, we now need to show that the provided architectures are capable of producing networks that satisfy the four assumptions above.

For architecture [a] they are all trivially satisfied as we have $h_0 = [Z; x]$, which directly ensures assumptions 1-3 hold, and $f_\theta^{>0}$ satisfies the assumptions of Lemma 2 and is a suitable universal approximator.

For architecture [b], we start by noting that the fourth assumption directly holds by the architecture construction. Now by using the weight matrix $W_1 = [\mathbf{0}; I_d]$ and the biases $b_1 = [Z; 0]$ for this first layer, we have that its pre-activations are exactly $[Z; x]$ for all Z and x. This ensures the first and second assumptions hold, noting that the continuity of the activation functions ensures that h_ℓ remains compact. Finally, we can show that the third assumption holds by using the fact that the architecture uses invertible activation functions to simply define the required g to be the corresponding inverse applied element-wise.

We can now view architecture [c] as an extension of architecture [b], wherein we no longer have an invertible activation function, but can exploit properties of the RELU and an increased number of hidden units instead. Here we will now use the weight matrix $W_1 = [\mathbf{0}; I_d; -I_d]$ and the biases $b_1 = [Z; \mathbf{0}; \mathbf{0}]$ for this first layer, so that its pre-activations are exactly $[Z; x; -x]$ for all Z and x. This again immediately ensure that the first two assumption holds, while the fourth assumption is again immediately ensured by downstream subnetwork construction. For the third assumption, we note that we have $h_\ell = [Z; \max(x, 0); -\min(x, 0)]$, and thus we already immediately have Z and simply need to substract the third set of hidden units from the second to recover x, that is the assumptions is satisfied by taking $g([a; b; c]) = [a; b - c]$.

Architecture [d] is now a generalization of those in [b] and [c] to allow additional layers and units in each layer. We can show that the result holds for this set of architectures by showing that any such architecture can replicate the behavior of one of the architectures in [b] or [c] exactly. For this, we first set all the weight matrices to the identity mapping and all the biases to zero for any layer which is not the specified layer with m random Gaussian biases, with an arbitrary number of hidden units, or the output layer. If the number of hidden units varies from one layer and the next, we simply pad the weight matrix with zeros, or truncate appropriately. Here the assumption that we have at least $2\max(d + m, n)$ deterministic units in each layer means we always have enough units to exactly propagate either $[Z; x; -Z; -x]$ or $[Y; -Y]$, as required depending on

the position in the network. For the weights coming into the layer with the m random biases, we use $W_\ell = [\mathbf{0}; I_d; -I_d; \mathbf{0}]$ and $b_\ell = [Z; \mathbf{0}; \mathbf{0}; \mathbf{0}]$, producing preactivations for h_ℓ that are always identical to the preactivations of h_1 in architecture [c], appended with zeros if necessary. The arguments for architectures [b] and [c] (depending on whether our activations are invertible or RELUs) can now be applied to show that we can always recover $[Z; x]$ from h_ℓ. From here we simply note that the downstream network will behave identically as if it only had one more hidden layer of arbitrary width. Thus, this architecture must always exactly emulate an architecture of type either [b] or [c], and is, therefore, a universal approximator as required.

□

B Ethical Considerations

We hope that our work will help pave the way for cheap, high-quality uncertainty estimates. Such estimates could help build safe and robust artificial intelligence Hendrycks et al. [2021]. Additionally, partially stochastic networks typically require less computation than fully stochastic networks and are therefore more environmentally friendly. However, strongly performing systems could lead to unintended consequences and pose societal costs Russell [2019], especially if humans place unwarranted credibility in the uncertainty estimates provided by deep learning systems.

C Computational Considerations

We now briefly discuss some of the computational considerations around partially stochasic networks. At deployment, the memory cost of partially stochastic networks scales with the number of stochastic parameters; the fewer stochastic parameters used, the lower the memory cost, with the exact savings depending on the specific implementation. However, the cost of computing the subset predictive depends on the particular stochastic subset. For example, a stochastic input layer would *not* reduce the number of forward passes required, whilst a stochastic output layer would.

D Additional results and experiment details

D.1 HMC Mixing Analysis (§5)

Here, we provide further results and details relating to the analysis in §5: Does Bayesian Reasoning Support Fully Stochastic Networks? In this section, we analysed the convergence of HMC samples provided by Izmailov et al. [2021b]. Table 2 contains details pertaining to this analysis.

Analysis Details To compute the prediction associated with each chain, we averaged the softmax probabilities produced by the samples associated with the chain, in accordance with:

$$p(y|x, \mathcal{D}) = \mathbb{E}_{p(\theta|\mathcal{D})}[p(y|x, \theta)]. \tag{5}$$

That is, for each chain, we computed a predictive distribution by averaging the prediction probabilities for each class across the samples from the relevant chain. The "prediction" for each datapoint associated with each chain is the class that has the highest predictive probability for that i.e., $\arg\max_y p(y|x, \mathcal{D})$.

The agreement metric that we report is the percentage of data-points from a given dataset on which *all three chains agree*. Note that this metric is different to the metric used by Izmailov et al. [2021b], who compute the percentage of points on which one chain and the *ensemble* of the remaining chains agree.

Additional Results Although we computed the agreement of each chain on all of the corruptions on the CIFAR-10-C dataset, we presented only a subset of corruptions in Fig. 2. Here, we additionally present results for the all corruptions below in Figure 7.

In an additional analysis, we compute the accuracy of each chain on different corruptions (Fig 8). We find differences in accuracy of up to 8% on certain corruptions, noticeably exceeding the within-chain variability (Fig 9). For example, the second HMC chain (orange) is less robust than the first and third HMC chain to all corruptions we consider. This further suggests that each HMC chain appears is exploring different regions of the posterior predictive.

Table 2: Additional details for analysis into whether full-batch HMC is converging, found in §5: Does Bayesian Reasoning Support Fully Stochastic Networks?

Hyper-parameter	Description
Dataset	CIFAR-10 [Krizhevsky et al., 2009] (MIT license) CIFAR-10-C [Hendrycks and Dietterich, 2018] (CC 4.0 license).
Use of existing assets	HMC samples from Izmailov et al. [2021b] (CC BY 4.0 license).
Architecture	ResNet-20-FRN, as in Izmailov et al. [2021b].
Compute Infrastructure	Google Colab
Hardware	Tesla T4 (or Tesla P100).
Runtime	ca. 12 hours.

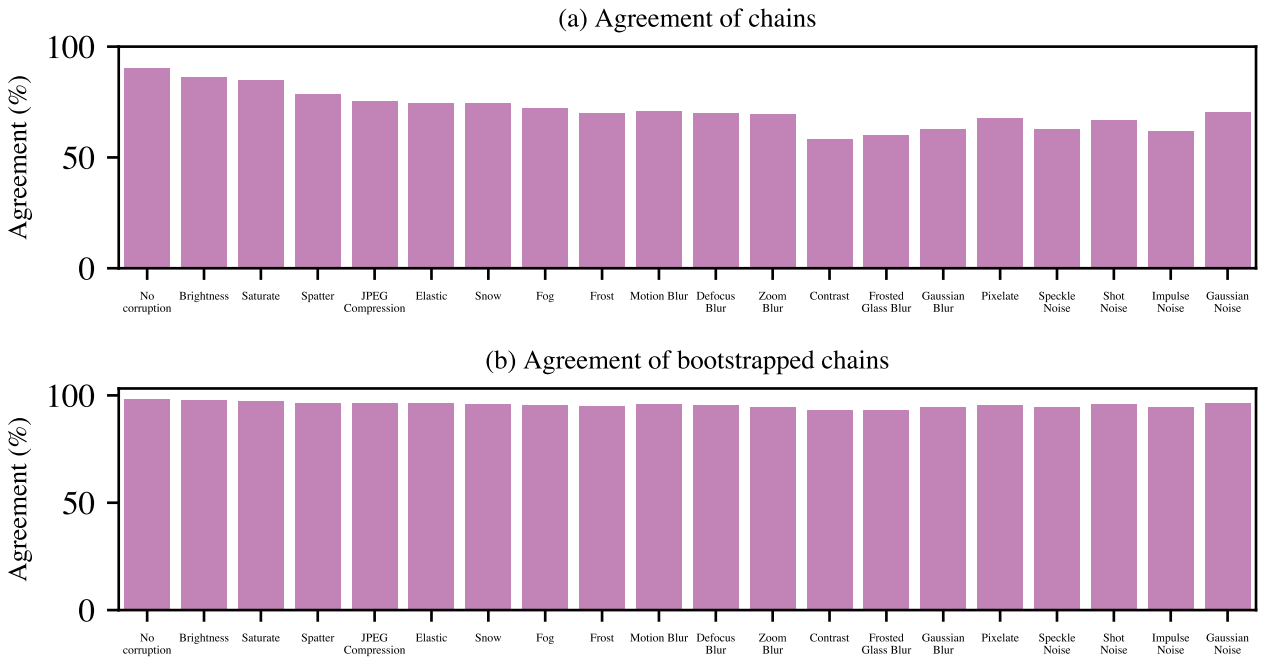

Figure 7: Assessment of function space mixing of ResNet-20-FRN full batch Hamiltonian Monte Carlo (HMC) samples trained on CIFAR-10. We measure the variability in predictions made across HMC chains released by Izmailov et al. [2021b]. To account for the finite sample size, we also measure the variability across simulated chains formed by resampling the first HMC chain i.e., bootstrapping. (a) We compute the percentage of points across different corruptions that all three chains make the same prediction on. While the agreement is 90% on the CIFAR-10 test set, the agreement decreases to <60% on certain datasets. (b) The agreement of bootstrapped HMC chains is greater than 94% across all data considered.

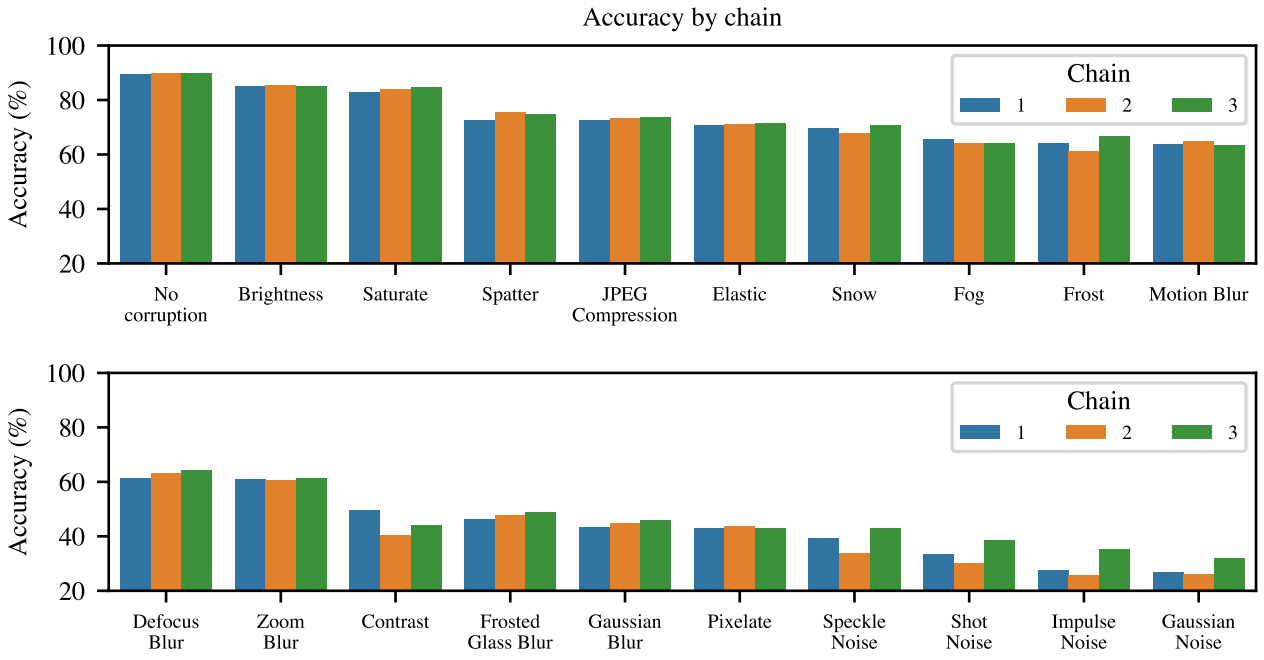

Figure 8: Assessment of function space mixing of ResNet-20-FRN full batch Hamiltonian Monte Carlo (HMC) samples trained on CIFAR-10. We measure the variability in predictions made across HMC chains released by Izmailov et al. [2021b]. Here, we present the accuracy of each chain on the CIFAR-10 test set and all corruptions of the CIFAR-10-C Hendrycks and Dietterich [2018] dataset with corruption intensity 5.

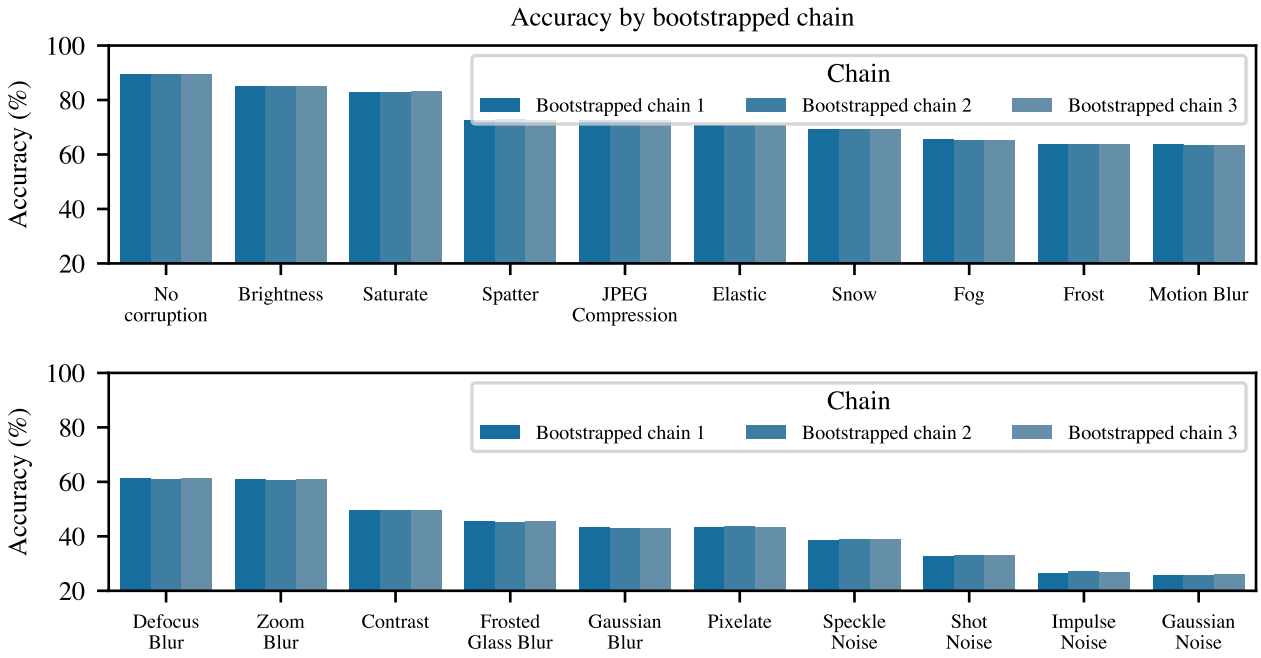

Figure 9: Assessment of within-chain function space variability of ResNet-20-FRN full batch Hamiltonian Monte Carlo (HMC) samples trained on CIFAR-10. We measure the variability in predictions made across simulated HMC chains, using released by Izmailov et al. [2021b]. Specifically, we generated multiple simulated chains by sampling from the first chain with replacement.

D.2 1D Regression with Hamiltonian Monte Carlo (§6.1)

We now provide further details relating to §6.1: 1D Regression with Hamiltonian Monte Carlo and Variational Inference. In this section, we focus on the experiment details related to the experiments that used Hamiltonian Monte Carlo. Please see Table 3 for relevant experiment details.

Data We generate synthetic data as follows. We draw 25 points from $\mathcal{U}(-3, -1.7)$ and 25 points from $\mathcal{U}(2.2, 4)$ to generate a set of 50 input points, $\{x_i\}$. We generate the output using $y_i = \sin(4 \cdot (x_i - 4.3)) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, 0.05)^2$.

Additional Results In Fig. 10, we show the predictive distributions of additional partially stochastic networks that use two-stage training. We note that for the the No-U-Turn Sampler (NUTS), the number of steps is chosen adaptively.

Table 3: Additional experiment details for 1D Regression using Hamiltonian Monte Carlo, found in §6.1: 1D Regression with Hamiltonian Monte Carlo and Variational Inference.

Hyper-parameter	Description
Architecture	Multi-layer perceptron
Number of Hidden Layers	2
Layer Width	50
Activation Function	SiLU [Hendrycks and Gimpel, 2016]
Prior Mean	0
Prior Variance	$\frac{\lvert\Theta\rvert}{\lvert\Theta_S\rvert}$, following [Daxberger et al., 2021b].
Network Parameterization	Neural Tangent Kernel Parameterization [Jacot et al., 2018]
Inference Algorithm	Hamiltonian Monte Carlo [Neal, 1996] with NUTS [Hoffman and Gelman, 2011]
MCMC chains	8
Warmup samples per chain	1000
Samples per chain	500
Maximum Tree Depth	15
Likelihood Function	Gaussian
Output Noise Variance	0.05^2 (As generated)
Dataset	Synthetic
Dataset Split	70% train, 20% val, 10% test.
Preprocessing	None
Computing Infrastructure	Macbook Pro
Runtime	ca. 15 minutes (Fully stochastic network).

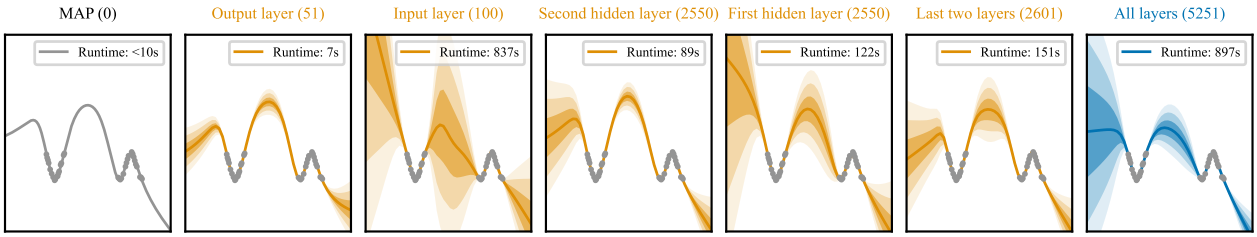

Figure 10: Additional partially stochastic network configurations using HMC inference over subsets of model parameters.

D.3 1D Regression with Variational Inference (§6.1)

We now provide further details relating to §6.1: 1D Regression with Hamiltonian Monte Carlo and Variational Inference. In this section, we focus on the experiment details related to the experiments that used variational inference. Please see Table 4 for relevant experiment details.

Data We generate synthetic data as follows. We draw 700 points from $\mathcal{U}(-2,-1.4)$ and 700 points from $\mathcal{U}(2,2.8)$ to generate a set of 1400 input points, $\{x_i\}$. We generate the output using $y_i = \sin(4\cdot(x_i-4.3))+\epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0,0.05)^2$.

Table 4: Additional experiment details for 1d regression using variational inference, found in §6.1: 1D Regression with Hamiltonian Monte Carlo and Variational Inference.

Hyper-parameter	Description
Architecture	Multi-layer perceptron
Number of Hidden Layers	3
Layer Width	100
Activation Function	Leaky ReLU
Prior	$\mathcal{N}(0,1)$
Training Monte Carlo Samples	1
Inference Algorithm	Flipout Mean-Field Variational Inference [Wen et al., 2018]
Posterior Mean Initialisation	$\mu \sim \mathcal{N}(0,0.1^2)$
Posterior Standard Deviation Initialistion	$\sigma = \log(1+\exp(\rho))$, with $\rho \sim \mathcal{N}(-3,0.1)$
Stochastic Layers	All, or output layer only.
Likelihood Function	Gaussian
Output Noise Variance	0.05^2 (As generated)
Dataset	Synthetic
Dataset Split	70% train, 20% val, 10% test.
Preprocessing	None
Optimizer	AdamW [Loshchilov and Hutter, 2017]
Learning Rate	0.001
Weight Decay	0.0001 only on deterministic weights and biases
Batch Zize	350
Epochs	12000
Plotting Epoch	Maximum validation set likelihood
Computing Infrastructure	Nvidia Tesla V100-PCIE-32GB
Runtime	ca. 15 minutes.
Use of existing assets	Bayesian Torch (BSD-3-Clause License) [Krishnan et al., 2022]

D.4 UCI Regression with Hamiltonian Monte Carlo (§6.2)

We now provide further details relating to §6.2: UCI Regression with Hamiltonian Monte Carlo. Please see Table 5 for relevant experiment details.

Additional Details. We note the additional details used in these experiments. (i) We used a homoscedastic noise model $p(y_i|x_i,\theta) = \mathcal{N}(y_i|f_\theta(x_i),\sigma_o^2)$, where $f_\theta(x_i)$ represents the neural network predictions. (ii) We tuned the prior variance so that the deterministic MAP network does not overfit. (iii) For the energy dataset, we predict only the first outcome variance, such that all the tasks we consider have one dimensional targets. (iv) All stochastic networks use a tempered posterior, where the sampler targets the density $\lambda \cdot \log p(\mathcal{D}|\theta) + \log p(\theta)$. We tuned λ for each dataset by maximising the likelihood of a validation set. (v) We place a prior over the output noise precision, $\lambda_o = 1/\sigma_o^2$.

Table 5: Additional experiment details for UCI regression using Hamiltonian Monte Carlo, found in §6.2: UCI Regression with Hamiltonian Monte Carlo.

Hyper-parameter	Description
Architecture	Multi-layer perceptron
Number of Hidden Layers	2
Layer Width	50
Activation Function	Leaky ReLU
Prior	$\mathcal{N}(0,\sigma^2)$
Prior Variance	$\sigma^2 \in [0.1, 0.01, 0.01]$ for UCI Yacht, Boston and Energy respectively.
Likelihood Scale	$\lambda \in [6.0, 1.0, 8.0]$ for UCI Yacht, Boston and Energy respectively.
Inference Algorithm	Hamiltonian Monte Carlo [Neal, 1996] with NUTS [Hoffman and Gelman, 2011]
MCMC chains	8
Warmup samples per chain	325
Samples per chain	75
Maximum Tree Depth	15
Output Precision Prior	$\mathrm{Gamma}(3.0, 1.0)$
Likelihood Function	Gaussian
Datasets	UCI Yacht, Boston, Energy [Dua and Graff, 2017]
Dataset Split	90% train, 10% test. Standard and "gap" splits [Foong et al., 2019]
Preprocessing	Feature normalisation
Computing Infrastructure	Internal CPU Cluster
Runtime	$\leq$30 minutes; exact time depends on network.

D.5 Image Classification with Laplace Approximation (§6.3)

We now provide further results and details relating to §6.3: Image Classification with Laplace Approximation. In this section, we considered the use of the Laplace approximation for fully stochastic and partially stochastic networks on an image classification task. Please see Table 6 for relevant experiment details.

Note that the experiments in this section build heavily on the `Laplace` library, released by Daxberger et al. [2021a].

Table 6: Additional experiment details for image classification experiments using the Laplace approximation, found in §6.3: Image Classification with Laplace Approximation.

Hyper-parameter	Description
Architecture	FixUp [Zhang et al., 2019] WideResNet-16-4 [Zagoruyko and Komodakis, 2016] following [Daxberger et al., 2021a]
Dataset	CIFAR-10 [Krizhevsky et al., 2009] (MIT License), CIFAR-10-C [Hendrycks et al., 2021] (CC 4.0 License).
Use of Existing Assets	`Laplace` Library [Daxberger et al., 2021a] (MIT License)
Computing Infrastructure	4x Nvidia A100 GPU.
Preprocessing	Per-channel normalisation $\mu = 0$, $\sigma = 1$
Number of Seeds	10
MAP Training	
Data Augmentation	Random crop and horizontal flip
Runtime	ca. 2 hours.
Epochs	350
Batch Size	1024
Optimizer	AdamW [Loshchilov and Hutter, 2017]
Learning Rate	0.001
Weight Decay	0.0001
Laplace Approximation	
Hessian Structure	Kronecker Factorised (KFAC)
Validation Set	10% of CIFAR-10 test set.
Prior Precision Tuning	Min val NLL (log-sweep in $(10^{-2}, 10^{5})$ with 125 increments)
Batch Size	32
Predictive	Linearized GLM Predictive
Temperature	1.0
Runtime	ca. 5 hours for fully stochastic networks less for partially stochastic networks

D.6 Image Classification with SWAG (§6.4)

We now provide further results and details relating to §6.4: Image Classification with SWAG. In this section, we considered the use of the SWAG inference for fully stochastic and partially stochastic networks on an image classification task. Please see Table 7 for relevant experiment details. We mostly followed Maddox et al. [2019] in the choice of hyperparameters, using the hyperparameters they used for their ImageNet experiments from a pre-trained solution. We, however, tuned the learning rate per architecture using a validation set.

Additional Partially Stochastic Network Configurations We present selected partially stochastic network configurations in Fig. 6. Fig. 11 shows more configurations. Several configurations outperform the fully stochastic network in distribution, but only the input and first ResNet block stochastic network outperforms the fully stochastic network on large corruption intensities. Nevertheless, the partially stochastic networks have lower memory cost.

Table 7: Additional experiment details for image classification experiments using SWAG, found in §6.4: Image Classification with SWAG

Hyper-parameter	Description
Architecture	FixUp [Zhang et al., 2019] WideResNet-16-4 [Zagoruyko and Komodakis, 2016] following [Daxberger et al., 2021a]
Dataset	CIFAR-10 [Krizhevsky et al., 2009] (MIT License), CIFAR-10-C [Hendrycks et al., 2021] (CC 4.0 License).
Use of Existing Assets	`Laplace` Library [Daxberger et al., 2021a] (MIT License)
Computing Infrastructure	4x Nvidia A100 GPU.
Preprocessing	Per-channel normalisation $\mu = 0$, $\sigma = 1$
Number of Seeds	10
MAP Training	
Data Augmentation	Random crop and horizontal flip
Runtime	ca. 2 hours.
Epochs	350
Batch Size	1024
Optimizer	AdamW [Loshchilov and Hutter, 2017]
Learning Rate	0.001
Weight Decay	0.0001
SWAG	
Rank of Covariance Matrix (K)	20
Evaluation Monte Carlo Samples	30
SWAG Epochs	10
SWAG Snapshots per Epoch	4
Weight decay	3e-4
Validation Set	10% of CIFAR-10 test set.
Learning Rate	Tuned: log-sweep in $(10^{-5}, 10^{-2})$ with 25 increments)
Batch Size	1024
Runtime	ca. 3 hours

Table 8: Correspondence between network name and stochastic blocks for additional configurations for SWAG experiments (Fig. 11). Note that ResNet block 1 is the ResNet block immediately after the input layer, and as the block number increases, the block is closer to the network output

Name	Stochastic Units
MAP	None
All (Fully Stochastic)	All layers
Input Layer	Input Layer
Input+	Input Layer and ResNet Block 1
Output Layer	Output Layer
Output+	Output Layer and ResNet Block 3
Input and Output Layer	Input and Output Layer
Bottleneck	ResNet Block 2

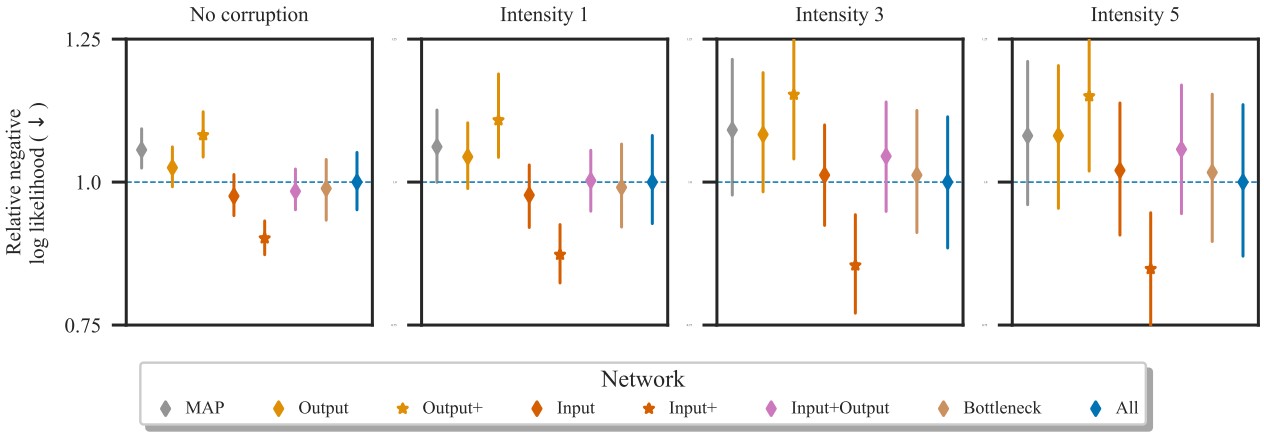

Figure 11: Relative NLL for various SWAG networks on CIFAR-10 and CIFAR-10-C Hendrycks and Dietterich [2018]. Results averaged across 10 random seeds. We show many more configurations here—see Table 8 for correspondence between model name and the stochastic units.

D.7 Image Classification with Variational Inference

We now provide further results and details relating to §6.5: Image Classification with Variational Inference. In this section, we considered the use of variational inference for fully stochastic and partially stochastic networks on an image classification task. Please see Table 9 for relevant experiment details.

Note that the experiments in this section build heavily on the `uncertainty-baselines` library, released by Nado et al. [2021].

Table 9: Additional experiment details for image classification experiments using variational inference,found in §6.5:Image Classification with Variational Inference.

Hyper-parameter	Description
Architecture	WideResNet-28-10 Zagoruyko and Komodakis [2016]
Dataset	CIFAR-10, CIFAR-100 Krizhevsky et al. [2009] (MIT License)
Use of Existing Assets	`uncertainty-baselines` Nado et al. [2021] (Apache 2.0 license)
Computing Infrastructure	4x Nvidia A100 GPU.
Inference Algorithm	Flipout Mean-Field Variational Inference Wen et al. [2018].
KL Annealing Epochs	200
Prior σ	0.1
Posterior Standard Deviation Initialisation	0.001
Training Monte Carlo Samples	1
Evaluation Monte Carlo Samples	5
Training Epochs	250
Dataset Split	95% train, 5% validation.
ℓ_2 Weight Decay	$4 \cdot 10^4$
Batch Size	256
Learning Rate	0.2
Learning Rate Warmup Epochs	1
Momentum	0.9
Learning Rate Decay Ratio	0.2
Learning Rate Decay Epochs	60, 120, 160
Optimizer	SGD
Preprocessing	Per-channel normalisation $\mu = 0$, $\sigma = 1$
Runtime	ca. 8 hours (fully stochastic)

Variability across random seeds. Fig. 12 shows the mean and standard deviation of across different random seeds for large scale image classification with variational inference on the CIFAR test sets. The conclusions in §6.5: Image Classification with Variational Inference are consistent across random seeds—partially stochastic networks can perform well, while fully stochastic networks do not appear to be well-performing despite their large computational cost.

Additional network configurations. We considered several partially stochastic network considerations—see Fig. 13—and presented a selection of the results in §6.5: Image Classification with Variational Inference. Though every partially stochastic network does not perform well, there are performant partially stochastic networks. One exciting area for future work is investigating and establishing best practices for the configuration and training of such partially stochastic networks.

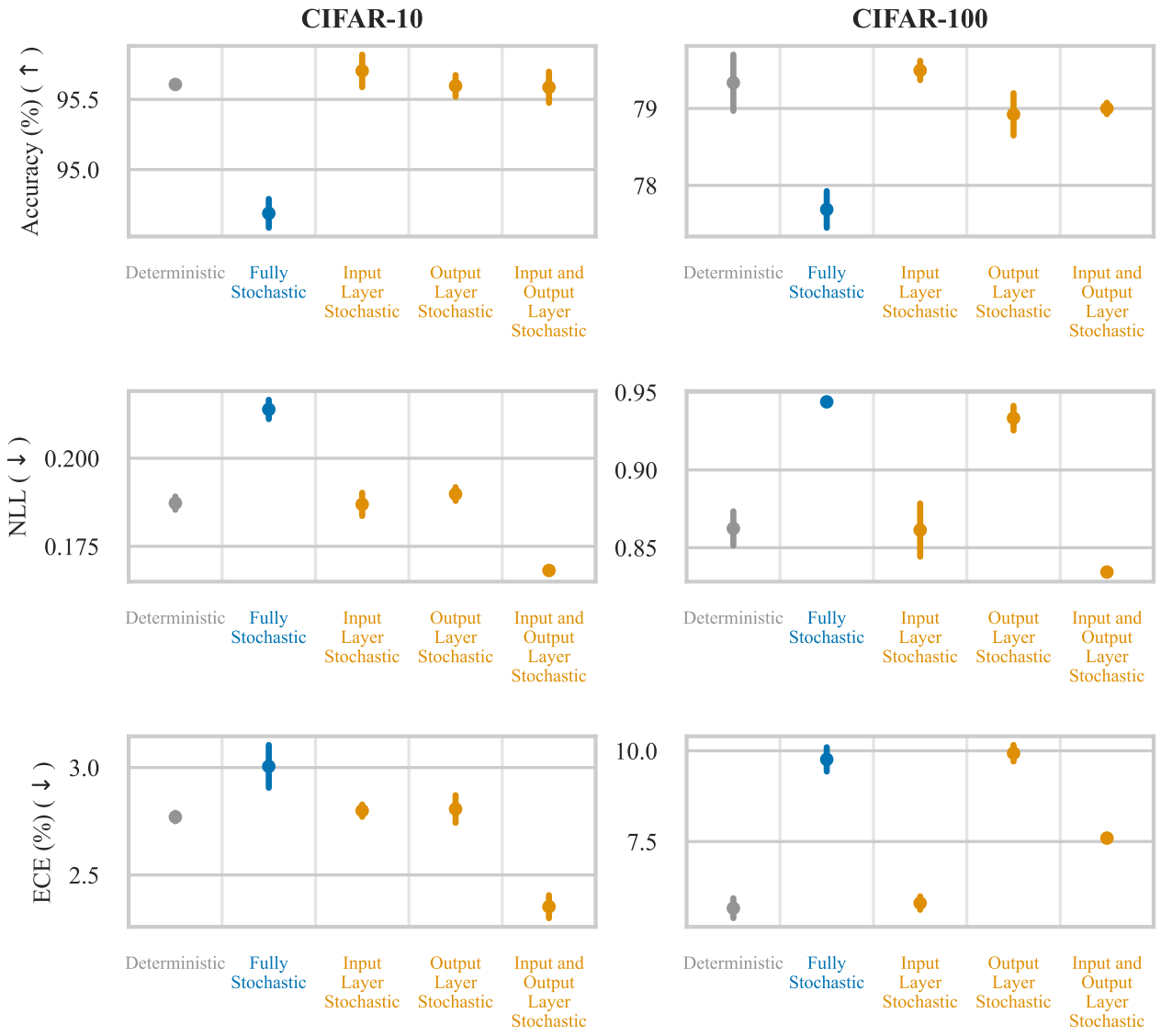

Figure 12: We report the accuracy, expected calibration error (ECE) and NLL on the standard CIFAR test sets when performing VI for subsets of parameters and learning the remaining parameters by maximising the (penalised) ELBO. Dots indicate the mean across 3 random seeds, bars indicate the standard deviation. This results are a graphical display of Table 1, found in §6.5: Image Classification with Variational Inference.

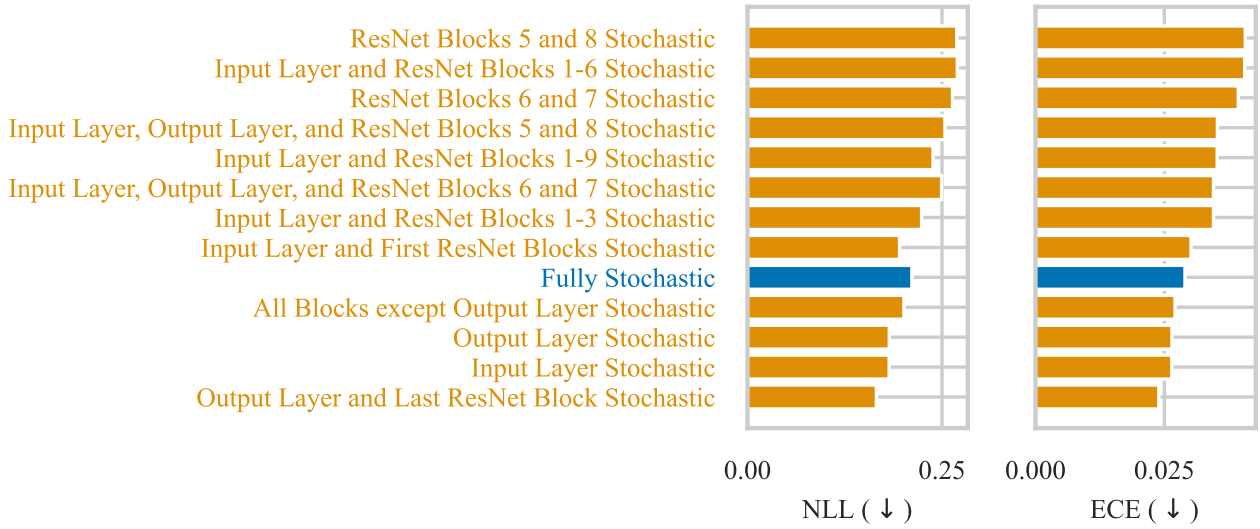

Figure 13: NLL and expected calibration error (ECE) on the CIFAR-10 test set for different network configurations. These results produced using only 1 random seed. Though every partially stochastic network does not perform well, there are performant partially stochastic networks.