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 $Z = \{Z_1, \dots, Z_m\}, Z_i \in \mathbb{R}$, 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.

$\square$

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.

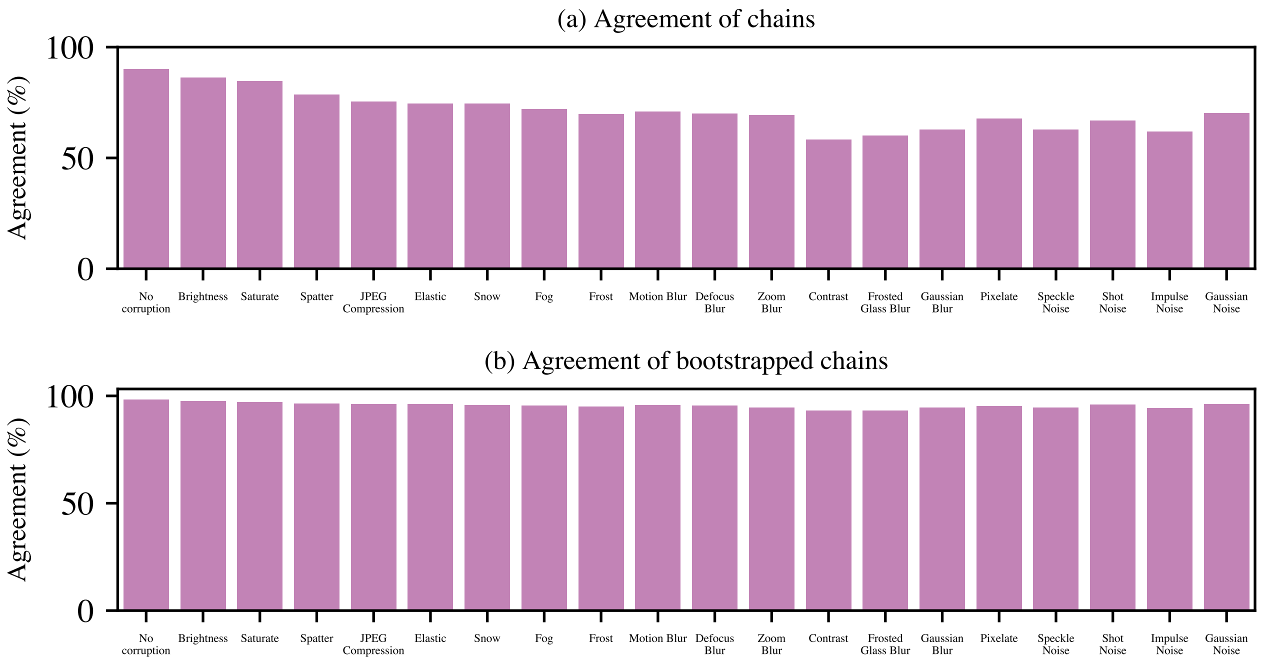

Figure 7: Assessment of function space mixing of ResNet-20-FRN full batch Hamiltonian Monte Carlo (HMC) samples trained on CIFAR-10. We measure the variability in predictions made across HMC chains released by Izmailov et al. [2021b]. To account for the finite sample size, we also measure the variability 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.

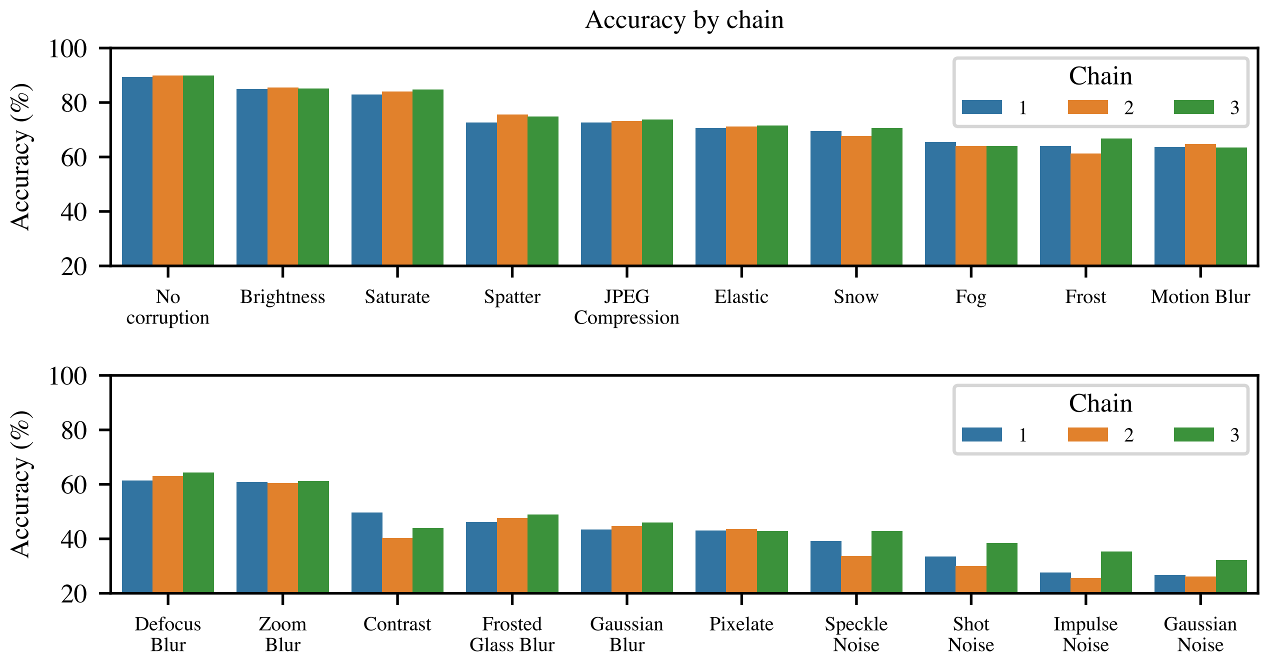

Figure 8: Assessment of function space mixing of ResNet-20-FRN full batch Hamiltonian Monte Carlo (HMC) samples trained on CIFAR-10. We measure the variability in predictions made across HMC chains released by Izmailov et al. [2021b]. Here, we present the accuracy of each chain on the CIFAR-10 test set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

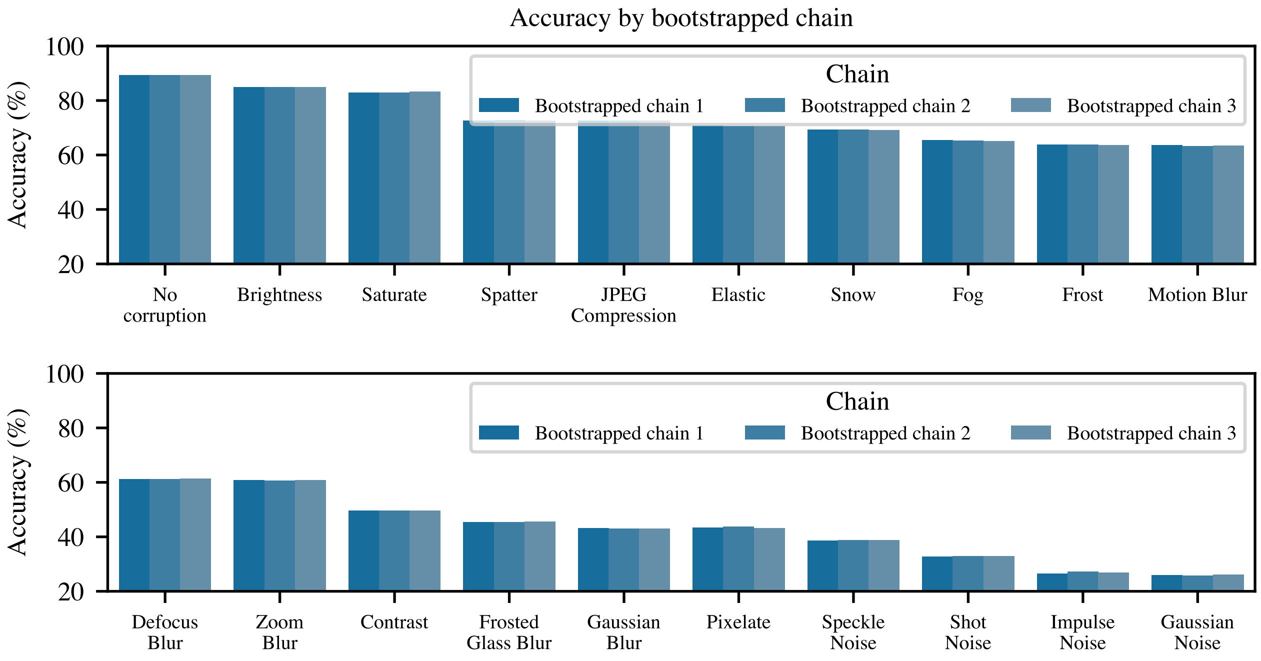

Figure 9: Assessment of within-chain function space variability of ResNet-20-FRN full batch Hamiltonian Monte Carlo (HMC) samples trained on CIFAR-10. We measure the variability in predictions made across simulated HMC chains, using released by Izmailov et al. [2021b]. Specifically, we generated multiple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

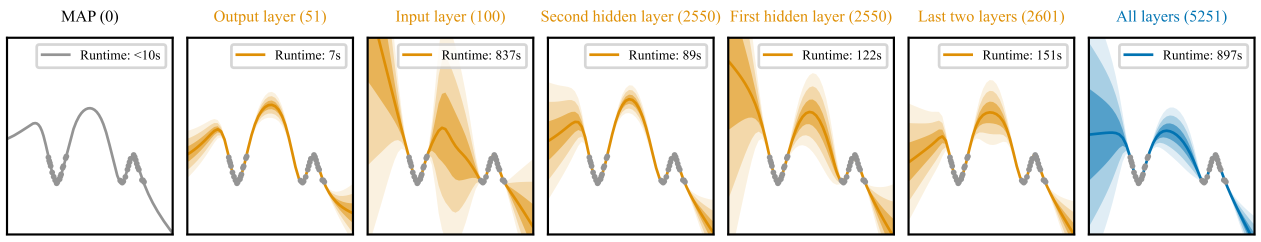

Figure 10: Additional partially stochastic network configurations using HMC inference over subsets of model parameters.

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

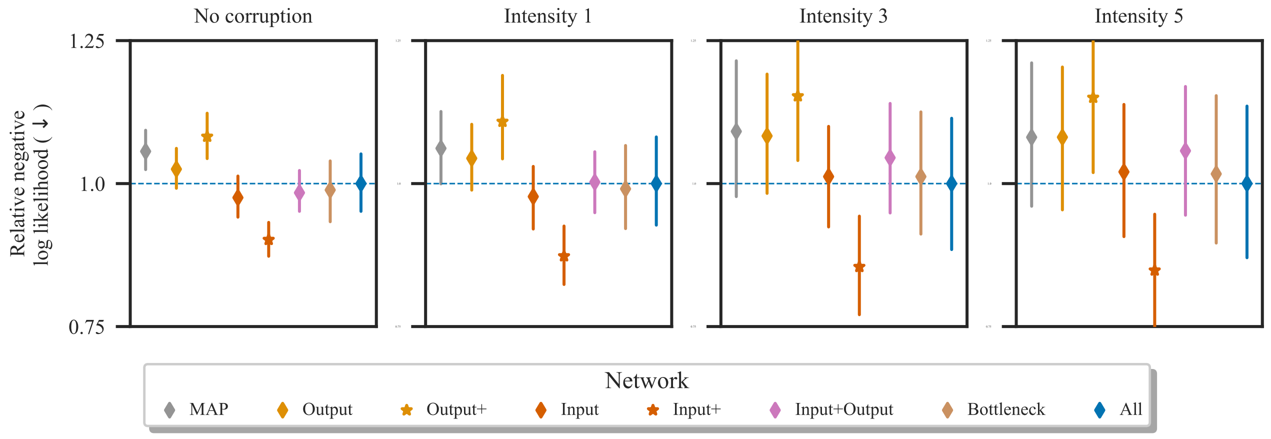

Figure 11: Relative NLL for various SWAG networks on CIFAR-10 and CIFAR-10-C Hendrycks and Dietterich [2018]. Results averaged across 10 random seeds. We show many more configurations here—see Table 8 for correspondence between model name and the stochastic units.

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.

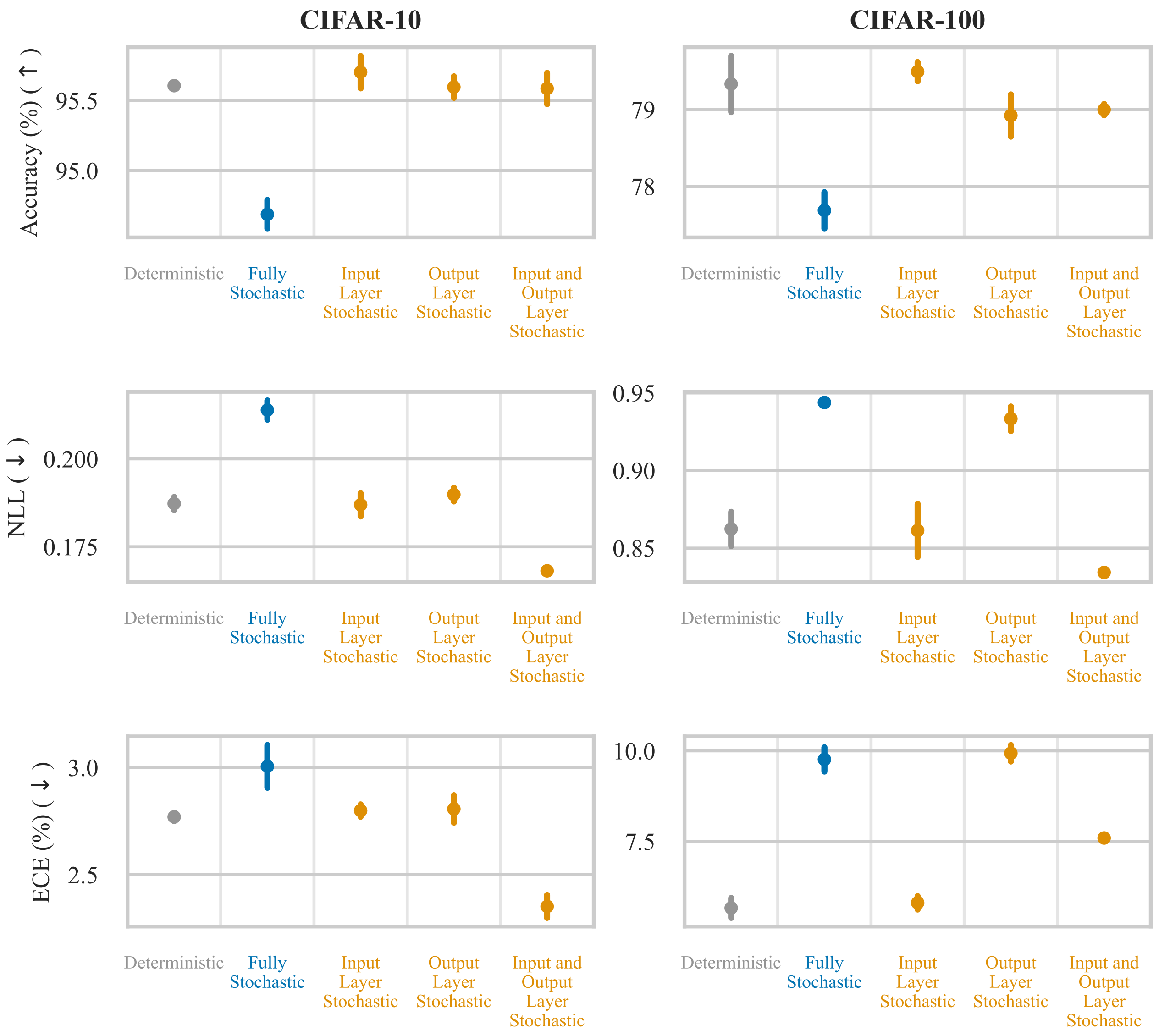

Figure 12: We report the accuracy, expected calibration error (ECE) and NLL on the standard CIFAR test sets when performing VI for subsets of parameters and learning the remaining parameters by maximising the (penalised) ELBO. Dots indicate the mean across 3 random seeds, bars indicate the standard deviation.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

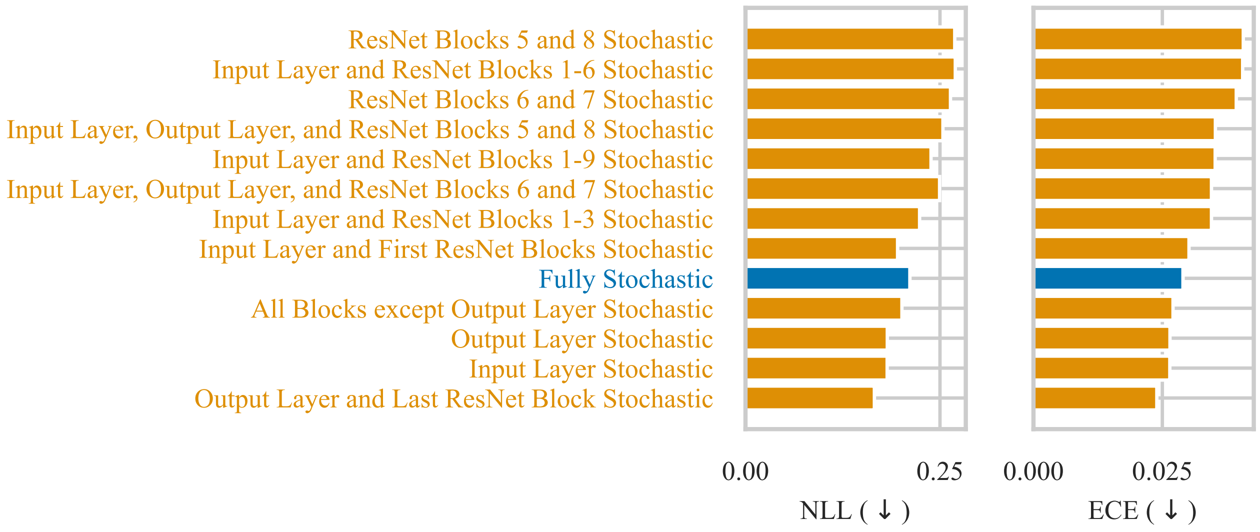

Figure 13: NLL and expected calibration error (ECE) on the CIFAR-10 test set for different network configurations. These results produced using only 1 random seed. Though every partially stochastic network does not perform well, there are performant partially stochastic networks.

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. Bayesian-Torch: Bayesian neural network layers for uncertainty estimation, 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.