

Towards Efficient MCMC Sampling in Bayesian Neural Networks by Exploiting Symmetry

Jonas Gregor Wiese^{*1}, Lisa Wimmer^{*2,3}, Theodore Papamarkou⁴,
 Bernd Bischl^{2,3}, Stephan Günnemann^{1,3}, and David Rügamer^{2,3}

¹ Technical University of Munich

² Department of Statistics, LMU Munich

³ Munich Center for Machine Learning

⁴ Department of Mathematics, The University of Manchester

Abstract. Bayesian inference in deep neural networks is challenging due to the high-dimensional, strongly multi-modal parameter posterior density landscape. Markov chain Monte Carlo approaches asymptotically recover the true posterior but are considered prohibitively expensive for large modern architectures. Local methods, which have emerged as a popular alternative, focus on specific parameter regions that can be approximated by functions with tractable integrals. While these often yield satisfactory empirical results, they fail, by definition, to account for the multi-modality of the parameter posterior. In this work, we argue that the dilemma between exact-but-unaffordable and cheap-but-inexact approaches can be mitigated by exploiting symmetries in the posterior landscape. Such symmetries, induced by neuron interchangeability and certain activation functions, manifest in different parameter values leading to the same functional output value. We show theoretically that the posterior predictive density in Bayesian neural networks can be restricted to a symmetry-free parameter reference set. By further deriving an upper bound on the number of Monte Carlo chains required to capture the functional diversity, we propose a straightforward approach for feasible Bayesian inference. Our experiments suggest that efficient sampling is indeed possible, opening up a promising path to accurate uncertainty quantification in deep learning.

Keywords: Uncertainty quantification · Predictive uncertainty · Bayesian inference · Monte Carlo sampling · Posterior symmetry

1 Introduction

Despite big data being the dominant paradigm in deep learning, the lack of infinitely many observations makes uncertainty quantification (UQ) an important problem in the field. A key component of UQ in Bayesian learning is the parameter posterior density that assigns a posterior probability to each parameter value⁵ [23]. Between the extreme cases of all posterior probability mass

^{*} Equal contribution.

⁵ We assume the likelihood to be parameterized by a single parameter vector. In the case of neural networks (NNs), the parameter contains all weights and biases.

concentrating on a single value, indicating complete certainty about the model parameters, and being distributed uniformly over all possible values in a reflection of total ignorance, the shape of the parameter posterior density is central to the quantification of predictive uncertainty. However, the parameter posterior for Bayesian neural networks (BNNs) is typically highly multi-modal and rarely available in closed form. The classical Markov chain Monte Carlo (MCMC) approach asymptotically recovers the true posterior but is considered prohibitively expensive for BNNs, as the large number of posterior modes prevents a reasonable mixing of chains [25]. Popular approximation techniques, such as Laplace approximation (LA; [9, 29]) or deep ensembles (DE; [28]), therefore focus on local regions of the posterior landscape. While these methods are faster than traditional MCMC and perform well in many applications, they systematically omit regions of the parameter space that might be decisive for meaningful UQ [25] (also shown in Section 5.3).

In this work, we challenge the presumed infeasibility of MCMC for NNs and propose to exploit the – in this context, rarely considered – unidentifiability property of NNs, i.e., the existence of two or more equivalent parameter values that describe the same input-output mapping. We refer to these equivalent values as *equioutput parameter states*. Equioutput parameter states emerge from certain activation functions [6, 27, 37], as well as the free permutability of neuron parameters in hidden layers [20], and can be transformed into one another.

The functional redundancy arising from this phenomenon grows rapidly with the depth and width of a network (cf. Figure 1) and induces symmetries in the posterior density. For exact inference (up to a Monte Carlo error), we need to incorporate all non-equioutput parameter states that lead to distinct input-output mappings. Considering only these *functionally diverse* mappings means, in turn, that our effective parameter space makes up a comparatively small fraction of the network’s original parameter space. Since their numerous equioutput counterparts do not contribute any new information to predictive uncertainty, we need much fewer MCMC samples when approximating the posterior predictive density (PPD) via Monte Carlo integration. By explicitly removing symmetries from samples *post-hoc*, we can even expose the functionally relevant part of the posterior and provide an opportunity for interpretation and analytical approximation in the reduced effective parameter space.

Our Contributions. We analyze the role of posterior space redundancies in quantifying BNN uncertainty, making the following contributions: 1) We show that the full PPD can be obtained from a substantially smaller reference set containing uniquely identified parameter states in function space. 2) We propose an estimation procedure for the number of Monte Carlo chains required to discover functionally diverse modes, providing a practical guideline for sampling from the parameter space of multi-layer perceptrons (MLPs). 3) We supply experimental evidence that our approach yields superior predictive performance compared to standard MCMC and local approximation methods. 4) Lastly, we demonstrate the posterior interpretability and analytic approximation that can

be obtained from explicitly removing symmetries *post-hoc*, for which we propose an algorithmic proof-of-concept.

2 Related Work

In this section, we review the literature on the existence, removal, and utilization of symmetries in MLPs, as well as the relation between such symmetries and statistical inference.

Existence of Parameter State Symmetries. Non-unique network parameter states have been considered in the literature before. [20] were among the first to note that equioutput states induce symmetries in the parameter space of MLPs. Focusing, within the general linear group of the parameter space, on the subgroup of transformations that leave the input-output mapping unchanged, they derived equivalence classes of equioutput parameter states and showed that, for every MLP, there exists a minimal and complete set of representatives covering all functionally different parameter states. [42] and [27] continued along this line of work to study single-hidden-layer MLPs with specific activation functions, advancing from tanh to more general self-affine activations. An extension to MLPs of arbitrary depth was studied by [6] in the context of tanh activations. More recently, [37] characterized equioutput parameter states for ReLU activations, again focusing on the case of a single hidden layer, and [1] classified all \mathbb{G} -invariant single-hidden-layer MLPs with ReLU activation for any finite orthogonal group \mathbb{G} . Lastly, [45] generalized much of the above in a framework addressing the identifiability of affine symmetries in arbitrary architectures.

Symmetry Removal. Symmetries in the parameter posterior density of Bayesian models can produce adverse effects that have been addressed in several research areas of statistics and machine learning. A prominent example is *label switching* in finite mixture models, where the permutability of label assignments to the mixture components induces symmetries similar to those in BNNs. To make mixture models identifiable, [3] introduced an adaptive Metropolis algorithm with online relabeling, effectively removing permutation symmetries by optimizing over discrete sets of permutation matrices. Such exhaustive-search approaches, however, scale poorly to modern NNs with many parameters, as the amount of equioutput states rises exponentially with the number of parameters.

In BNNs, symmetries have been known to slow down MCMC convergence to the stationary parameter posterior density due to budget wasted on visiting symmetric modes [31, 34]. [25], reporting results from extensive and large-scale experiments, indeed find that MCMC chains tend to mix better in function space than in parameter space. Consequently, reducing the effect of symmetries by imposing parameter constraints and defining anchoring points for subsets of the latent variables has been shown to improve mixing [41]. A proposal for constrained sampling can be found, for example, in [39], with application to ReLU-activated MLPs.

Utilizing Symmetries. Symmetries in the parameter posterior density are not, however, necessarily a nuisance. Quite on the contrary, they can be useful to enhance generalization and make inference affordable. An increasing body of work has been exploring the use of symmetry removal in the context of *mode connectivity*, an approach to find more robust NN solutions by retrieving connected areas of near-constant loss rather than isolated local optima [10, 17]. Focusing on equioutput permutations of hidden-layer neurons, [43] and [2], among others, propose to align the layer-wise embeddings of multiple networks and thus improve upon the performance of individual models. Following a similar idea, [38] apply a *post-hoc* standardization on parameter vectors in ReLU-activated NNs that draws from the notion of equioutput equivalence classes.

In the field of Bayesian deep learning, the idea of utilizing – exact or approximate – parameter symmetries, represented by permutation groups, has led to the development of *lifted MCMC* [5, 33]. Orbital Markov chains have been introduced to leverage parameter symmetries in order to reduce mixing times [5, 32]. Lifted MCMC has been considered mainly in the context of probabilistic graphical models. There is scope to harness lifted MCMC in the context of MLPs since these can be cast as graphical models [26, 45].

We believe that equioutput symmetries have the potential to facilitate MCMC inference, despite the apparent complexity they introduce in the parameter posterior density. From the insight that a vast part of the sampling space is made up of symmetric copies of some minimal search set, we conclude that running multiple short MCMC chains in parallel, each of which can sample a functionally different mode, represents a more efficient use of the available budget than collecting a large number of samples from a single chain. In Section 4, we propose an upper bound for MCMC chains necessary to observe all functionally diverse posterior modes, which is a key criterion for successful inference. The perspective of parameter posterior symmetries thus lends a new theoretical justification to previous efforts in multi-chain MCMC that are motivated mainly by the exploitation of parallel computing resources [30, 40]. Our experiments in Section 5 suggest that this approach is indeed more effective than single-chain MCMC in BNNs with many symmetries in their parameter posterior density. In agreement with [25], our findings advocate to focus on function-space instead of parameter-space mixing during MCMC sampling.

We thus view the existence of equioutput parameter states as a benign phenomenon. That said, there are still benefits to be gained from removing the symmetries: with a parameter posterior density reduced to the minimal parameter set sufficient to represent its full functional diversity, we get an opportunity for better interpretation, and possibly even analytical approximation. We demonstrate the potential of symmetry removal in Section 5.3 by means of a custom algorithm (Supplementary Material B). In the following section, we provide the mathematical background and introduce the characterization and formal notation of equioutput transformations.

3 Background and Notation

MLP Architectures. In this work, we consider NNs of the following form. Let $f : \mathcal{X} \rightarrow \mathcal{Y}$ represent an MLP with K layers, where layer $l \in \{1, \dots, K\}$ consists of M_l neurons, mapping a feature vector $\mathbf{x} = (x_1, \dots, x_n)^\top \in \mathcal{X} \subseteq \mathbb{R}^n, n \in \mathbb{N}$, to an outcome vector

$$f(\mathbf{x}) =: \hat{\mathbf{y}} = (\hat{y}_1, \dots, \hat{y}_m)^\top \in \mathcal{Y} \subseteq \mathbb{R}^m, \quad m \in \mathbb{N},$$

to estimate $\mathbf{y} = (y_1, \dots, y_m)^\top \in \mathcal{Y}$. The i -th neuron in the l -th layer of the MLP is associated with the weights $w_{lij}, j = 1, \dots, M_{l-1}$, and the bias b_{li} . We summarize all the MLP parameters in the vector

$$\boldsymbol{\theta} := (w_{211}, \dots, w_{KM_K M_{K-1}}, b_{21}, \dots, b_{KM_K})^\top \in \Theta \subseteq \mathbb{R}^d$$

and write $f_{\boldsymbol{\theta}}$ to make clear that the MLP is parameterized by $\boldsymbol{\theta}$. For each hidden layer $l \in \{2, \dots, K-1\}$, the inputs are linearly transformed and then activated by a function a . More specifically, we define the pre-activations of the i -th neuron in the l -th hidden layer as

$$o_{li} = \sum_{j=1}^{M_{l-1}} w_{lij} z_{(l-1)j} + b_{li}$$

with post-activations $z_{(l-1)i} = a(o_{(l-1)i})$ from the preceding layer. For the input layer, we have $z_{1i} = x_i, i = 1, \dots, n$, and for the output layer, $z_{Ki} = \hat{y}_i, i = 1, \dots, M_K$.

Predictive Uncertainty. In the Bayesian paradigm, a prior density $p(\boldsymbol{\theta})$ is imposed on the parameters, typically as part of a Bayesian model of the data. Using Bayes' rule, the parameter posterior density

$$p(\boldsymbol{\theta} | \mathcal{D}) = \frac{p(\mathcal{D} | \boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D})}$$

updates this prior belief based on the information provided by the data \mathcal{D} and encoded in the likelihood $p(\mathcal{D} | \boldsymbol{\theta})$. In supervised learning, the data are typically given by a set of N feature vectors $\mathbf{x} \in \mathcal{X}$ and outcome vectors $\mathbf{y} \in \mathcal{Y}$, forming the dataset $\mathcal{D} = \{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \dots, (\mathbf{x}^{(N)}, \mathbf{y}^{(N)})\}$. The PPD $p(\mathbf{y}^* | \mathbf{x}^*, \mathcal{D})$ quantifies the predictive or functional uncertainty of the model for a new observation $(\mathbf{x}^*, \mathbf{y}^*) \in \mathcal{X} \times \mathcal{Y}$. Since

$$p(\mathbf{y}^* | \mathbf{x}^*, \mathcal{D}) = \int_{\Theta} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta},$$

deriving this uncertainty requires access to the posterior density $p(\boldsymbol{\theta} | \mathcal{D})$, which can be estimated from MCMC sampling.

Equioutput Transformations. Let us now characterize the notion of equioutput parameter states, and the transformations to convert between them, more formally. Two parameter states $\boldsymbol{\theta}, \boldsymbol{\theta}'$ are considered *equioutput* if the maps $f_{\boldsymbol{\theta}}, f_{\boldsymbol{\theta}'}$ yield the same outputs for all possible inputs from \mathcal{X} . We denote this equivalence relation (see proof in Supplementary Material A.2) by \sim and write:

$$\boldsymbol{\theta} \sim \boldsymbol{\theta}' \iff f_{\boldsymbol{\theta}}(\mathbf{x}) = f_{\boldsymbol{\theta}'}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{X}, \quad \boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta.$$

The equioutput relation is always defined with respect to a particular MLP f , which we omit in our notation when it is clear from the context.

All MLPs with more than one neuron in at least one hidden layer exhibit such equioutput parameter states that arise from permutation invariances of the input-output mapping [20, 27]. Since the operations in the pre-activation of the i -th neuron in the l -th layer commute⁶, the $M_l > 1$ neurons of a hidden layer l can be freely interchanged by permuting their associated parameters. In addition, equioutput transformations can arise from the use of certain activation functions with inherent symmetry properties. For example, in the case of tanh, the signs of corresponding parameters can be flipped using $\tanh(x) = -\tanh(-x)$. For ReLU activations, a scaling transformation can be applied such that the mapping of the network remains unchanged, i.e., $\text{ReLU}(x) = c^{-1} \cdot \text{ReLU}(c \cdot x)$ for $|c| > 0$ (see also Supplementary Material A.1).

We consider transformation maps that are linear in $\boldsymbol{\theta}$ and induce a finite amount of equioutput transformation matrices, which includes, for example, the tanh activation function. The ReLU activation function allows for infinitely many possibilities of re-scaling the weights and is excluded from our findings. More specifically, let

$$\mathcal{F}_{\mathbf{T}} : \Theta \rightarrow \Theta, \boldsymbol{\theta} \mapsto \mathbf{T}\boldsymbol{\theta}, \quad \mathbf{T} \in \mathbb{R}^{d \times d},$$

be an activation-related transformation of a parameter vector that might, for instance, encode an output-preserving sign flip. $\mathcal{F}_{\mathbf{T}}$ constitutes an *equioutput* transformation if $f_{\boldsymbol{\theta}}(\cdot) = f_{\mathcal{F}_{\mathbf{T}}(\boldsymbol{\theta})}(\cdot)$. We collect all output-preserving transformation matrices \mathbf{T} in the set \mathcal{T} , i.e.,

$$\mathcal{T} = \left\{ \mathbf{T} \in \mathbb{R}^{d \times d} \mid f_{\boldsymbol{\theta}}(\cdot) = f_{\mathcal{F}_{\mathbf{T}}(\boldsymbol{\theta})}(\cdot) \right\}.$$

Similarly, let

$$\mathcal{F}_{\mathbf{P}} : \Theta \rightarrow \Theta, \boldsymbol{\theta} \mapsto \mathbf{P}\boldsymbol{\theta}, \quad \mathbf{P} \in \{0, 1\}^{d \times d},$$

be a transformation that permutes elements in the parameter vector. We define the set of permutation matrices that yield equioutput parameter states as

$$\mathcal{P} = \left\{ \mathbf{P} \in \mathbb{R}^{d \times d} \mid f_{\boldsymbol{\theta}}(\cdot) = f_{\mathcal{F}_{\mathbf{P}}(\boldsymbol{\theta})}(\cdot) \right\}.$$

⁶ Recall that the pre-activation of neuron i in layer l is $o_{li} = \sum_{j=1}^{M_{l-1}} w_{lij} z_{(l-1)j} + b_{li}$. By the commutative property of sums, any permutation $\pi : J \rightarrow J$ of elements from the set $J = \{1, \dots, M_{l-1}\}$ will lead to the same pre-activation:

$$o_{li} = \sum_{j \in J} w_{lij} z_{(l-1)j} + b_{li} = \sum_{j \in \pi(J)} w_{lij} z_{(l-1)j} + b_{li}.$$

The cardinality of \mathcal{P} is at least $\prod_{l=2}^{K-1} M_l!$ [20] when traversing through the NN from the first layer in a sequential manner, applying to each layer permutations that compensate for permutations in its predecessor.

Since activation functions operate neuron-wise, activation- and permutation-related equioutput transformations do not interact (for instance, we could permute the associated weights of two neurons and later flip their sign). We can, therefore, define arbitrary combinations of activation and permutation transformations as

$$\mathcal{E} = \{\mathbf{E} = \mathbf{T}\mathbf{P} \in \mathbb{R}^{d \times d}, \mathbf{T} \in \mathcal{T}, \mathbf{P} \in \mathcal{P} \mid f_{\boldsymbol{\theta}}(\cdot) = f_{\mathcal{F}_{\mathbf{E}}(\boldsymbol{\theta})}(\cdot)\}.$$

The transformation matrices in \mathcal{E} will exhibit a block-diagonal structure with blocks corresponding to network layers. This is due to the permutations \mathbf{P} affecting both incoming and outgoing weights, but only in the sense that two incoming and two outgoing weights swap places, never changing layers. The activation-related sign flips or re-scalings occur neuron-wise, making \mathbf{T} a diagonal matrix that does not alter the block-diagonal structure of \mathbf{P} .

For the cardinality of the set \mathcal{E} of equioutput transformations, we can establish a lower bound that builds upon the minimum cardinality of \mathcal{P} :

$$|\mathcal{E}| \geq \prod_{l=2}^{K-1} M_l! \cdot |\mathcal{T}_l|,$$

where $|\mathcal{T}_l|$ denotes the number of activation-related transformations applicable to neurons in layer l . From this, it becomes immediately clear that the amount of functional redundancy increases rapidly with the network size (see also Figure 1). As a result of equioutput parameter states, the MLP parameter posterior density exhibits functional redundancy in the form of symmetries for commonly used priors (see Section 4):

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \stackrel{10,13}{=} \frac{p(\mathcal{D}|\mathbf{E}\boldsymbol{\theta})p(\mathbf{E}\boldsymbol{\theta})}{p(\mathcal{D})} = p(\mathbf{E}\boldsymbol{\theta}|\mathcal{D}), \quad \boldsymbol{\theta} \in \Theta, \mathbf{E} \in \mathcal{E}. \quad (1)$$

4 Efficient Sampling

These symmetric structures in the parameter posterior density suggest that sampling can be made more efficient. In the following section, we show that the PPD can theoretically be obtained from a small reference set of non-equivalent parameter states and propose an upper bound on Markov chains that suffice to sample all non-symmetric posterior modes.

4.1 Posterior Reference Set

As introduced in Section 3, for each parameter state $\boldsymbol{\theta}$ of an NN, there are functionally redundant counterparts $\boldsymbol{\theta}'$ related to $\boldsymbol{\theta}$ by an equioutput transformation, such that $f_{\boldsymbol{\theta}}(\cdot) = f_{\boldsymbol{\theta}'}(\cdot)$. We can use this equivalence relation to dissect the parameter space Θ into disjoint equivalence classes. For this, let the *reference set*

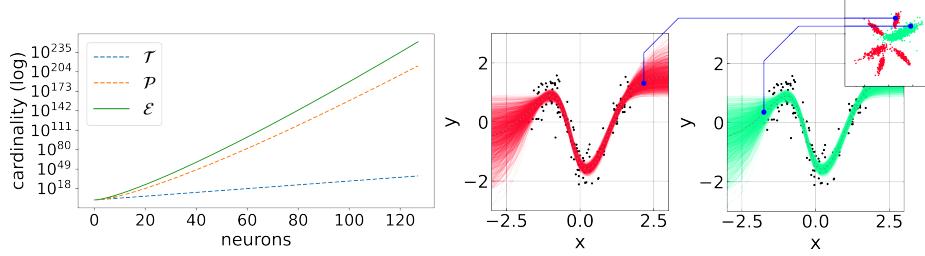


Fig. 1. Example of tanh-activated MLPs. *Left:* Cardinality lower bound of the equioutput transformation set for a single hidden layer with 1 to 128 neurons; the functional redundancy factor for a network with 128 neurons is $1.31 \cdot 10^{254}$. *Right:* A ten-dimensional MLP parameter posterior (top-right corner, depicted as bivariate marginal density) exhibits symmetries, such that all red sample clusters are equioutput-related to the green cluster. The associated function spaces are identical, i.e., many posterior modes are redundant.

\mathcal{S}_1 be a minimal set of representatives of each equivalence class (cf. *open minimal sufficient search sets* in [6]). All parameter states in \mathcal{S}_1 are functionally diverse, i.e., $\boldsymbol{\theta}, \bar{\boldsymbol{\theta}} \in \mathcal{S}_1 \Rightarrow \boldsymbol{\theta} \not\sim \bar{\boldsymbol{\theta}}$, and each element in Θ is equivalent to exactly one element in \mathcal{S}_1 . For a finite amount of equioutput transformations, as in the case of tanh-activated MLPs (finite possibilities of sign-flip combinations of hidden neurons), the NN parameter space can then be dissected into $|\mathcal{E}|$ disjoint *representative sets*, which contain equioutput transformations of the elements of the reference set, in the following way.

Proposition 1 (Parameter space dissection). *Let \mathcal{S}_1 be the reference set of uniquely identified network parameter states. Then, for a finite number of equioutput transformations, it holds that the parameter space can be dissected into $|\mathcal{E}|$ disjoint, non-empty representative sets up to a set $\mathcal{S}^0 \subset \Theta$, i.e.,*

$$\Theta = \left(\dot{\cup}_{j=1}^{|\mathcal{E}|} \mathcal{S}_j \right) \dot{\cup} \mathcal{S}^0, \text{ where } \mathcal{S}_j \cong \{ \boldsymbol{\theta} \mid \boldsymbol{\theta} = \mathbf{E}_j \boldsymbol{\theta}' \quad \forall \boldsymbol{\theta}' \in \mathcal{S}_1, \mathbf{E}_j \in \mathcal{E} \}, \quad (2)$$

where $\dot{\cup}$ denotes the union over disjoint sets. We use \mathcal{S}^0 as a residual quantity to account for cases that cannot be assigned unambiguously to one of the sets \mathcal{S}_j because they remain unchanged even under a transformation with non-identity matrices $\mathbf{E}_j \in \mathcal{E}$.

The edge cases that make up \mathcal{S}^0 exist, e.g., on the boundary of two classes [6] or in degenerated cases such as the zero vector [42, 45]. For a characterization of the involved sets, as well as a proof sketch, see Supplementary Material A.4.

Equioutput parameter states have the same posterior probabilities $p(\boldsymbol{\theta}|\mathcal{D}) = p(\mathbf{E}\boldsymbol{\theta}|\mathcal{D})$ if the prior is transformation-invariant; see Supplementary Material A.3, Equations (10)-(13). Moreover, equioutput parameter states produce by definition the same predictions $p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\theta}) = p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{E}\boldsymbol{\theta})$ for any $\mathbf{E} \in \mathcal{E}$. Thus, the following corollary holds.

Corollary 1 (Reformulated posterior predictive density). *Let \mathcal{E} be finite. As in Proposition 1, consider the disjoint non-empty sets $\mathcal{S}_j, j \in \{1, \dots, |\mathcal{E}|\}$, and residual space \mathcal{S}^0 . If the prior density $p(\boldsymbol{\theta})$ is transformation-invariant, then the posterior predictive density expresses as*

$$p(\mathbf{y}^* | \mathbf{x}^*, \mathcal{D}) = \int_{\Theta} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} \quad (3)$$

$$\begin{aligned} &= |\mathcal{E}| \int_{\mathcal{S}_j} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} + \int_{\mathcal{S}^0} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} \\ &\approx |\mathcal{E}| \int_{\mathcal{S}_j} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta}. \end{aligned} \quad (4)$$

The proof of Corollary 1 is given in Supplementary Material A.3. It follows from Proposition 1 and the assumption of transformation-invariant prior densities, which is often satisfied in practice (e.g., for widely-applied isotropic Gaussian priors). We can further approximate (3) by (4) as the set $\mathcal{S}^0 \subset \mathbb{R}^d$ is of negligible size (depending on Θ , potentially even with zero Lebesgue measure).

As a consequence of Corollary 1, the PPD can be obtained up to the residual set by only integrating over uniquely identified parameter states from one of the sets \mathcal{S}_j , with a multiplicative factor $|\mathcal{E}|$ that corrects the probability values by the amount of redundancy in the posterior. In other words, only a fraction $1/|\mathcal{E}|$ of the posterior must be sampled in order to infer a set of uniquely identified parameter states of the NN, and thus, to obtain the full PPD. This reduces the target sampling space drastically, as illustrated in Figure 1. For example, it allows the posterior space of a single-layer, tanh-activated network with 128 neurons to be effectively reduced to a 10^{254} -th of its original size.

In the case of an infinite amount of equioutput transformations, such as in ReLU-activated MLPs (the scaling factor $|c| > 0$ can be chosen arbitrarily), we can use similar reasoning. Only one representative set of the posterior density needs to be observed in order to capture the full functional diversity of a network because the integrals over two representative sets are identical. For a more in-depth discussion of ReLU symmetries, see, for example, [4].

How to Obtain a Representative Set? In practice, when using Monte Carlo to approximate Equation (4), it is not necessary to actually constrain the sampling procedure to a specific set \mathcal{S}_j , which might indeed not be straightforward⁷. Since any equioutput transformation is known *a priori*, we just need to be aware of the fact that each sample can theoretically be mapped to different representative sets after running the sampling procedure. Hence, for the calculation of the PPD integral, the samples can remain scattered across the various representative sets as long as they cover all functionally diverse parameter states. For

⁷ [12] demonstrate that finding invariant representations for groups acting on the input space is an NP-hard problem. While we are not aware of such a result for the parameter space, the NP-hardness in [12] for permutations of the inputs only suggests a similar property in our case.

the purpose of providing better interpretability and analytic approximation of the posterior, it may still be worthwhile to explicitly remove the symmetries. In Section 5.3, we demonstrate such symmetry removal using a custom algorithm for tanh-activated networks (Supplementary Material B).

4.2 An Upper Bound for Markov Chains

The question remains how many samples are needed to approximate a set of uniquely identified parameter states sufficiently well. Even in a symmetry-free setting, BNN posteriors can exhibit multiple functionally diverse modes representing structurally different hypotheses, depending on the network architecture and the underlying data-generating process. For example, in Section 5.3, we discuss the case of an under-parameterized network that preserves three distinctive modes caused by its restricted capacity.

In the following, we assume $\nu \in \mathbb{N}$ functionally diverse modes with the goal of visiting every mode or its local proximity at least once when running MCMC. As the ability to switch from one mode to another within a chain depends on various factors, such as the acceptance probability and the current state of other parameters, increasing the number of samples per chain does not necessarily correlate with the number of visited modes. We, therefore, propose to focus on the number of independent chains, rather than the number of samples per chain, to effectively control the number of visited modes.

This further allows us to derive an upper bound for the number of independent chains that are required to visit every mode at least once. The number of samples from each chain will then ultimately determine the approximation quality. In the computation of the PPD, we formulate the Monte Carlo integration over all samples from all chains simultaneously [30]. In practice, given a user-defined number of maximal resources ρ (e.g., CPU cores), the following proposition provides a lower bound on the probability that the number of chains \mathcal{G} necessary to visit every mode remains below the resource limit of the user (i.e., $\mathcal{G} < \rho$).

Proposition 2 (Probabilistic bound for sufficient number of Markov chains). *Let π_1, \dots, π_ν be the respective probabilities of the ν functionally diverse modes to be visited by an independently started Markov chain and $\Pi_J := \sum_{j \in J} \pi_j$. Then, given ρ chains,*

$$\mathbb{P}(\mathcal{G} < \rho) \geq 1 - \rho^{-1} \left\{ \sum_{q=0}^{\nu-1} (-1)^{\nu-1-q} \sum_{J:|J|=q} (1 - \Pi_J)^{-1} \right\}. \quad (5)$$

The proof can be found in Supplementary Material A.6. Note that this bound is independent of the NN architecture and only depends on the assumptions about the number and probabilities of functionally diverse modes ν , disregarding symmetric copies. Proposition 2 can be used to calculate the number of MCMC chains given certain assumptions – for example, from domain knowledge, or in a worst-case scenario calculation – and thus provides practical guidance for MCMC sampling of MLPs. Judging by the comparably high predictive performance of

Table 1. Mean log pointwise predictive density (LPPD) values on test sets (larger is better; one standard error in parentheses). The highest performance per dataset and network is highlighted in bold.

	Smaller network f_1			Larger network f_2		
	MCMC (ours)	MCMC (s.c.)	DE	MCMC (ours)	MCMC (s.c.)	DE
\mathcal{D}_S	-0.53 (± 0.09)	-0.56 (± 0.11)	-0.58 (± 0.11)	-0.59 (± 0.12)	-0.59 (± 0.12)	-2.13 (± 0.03)
\mathcal{D}_I	0.79 (± 0.06)	0.65 (± 0.07)	0.56 (± 0.06)	0.91 (± 0.09)	0.91 (± 0.09)	-2.02 (± 0.02)
\mathcal{D}_R	0.64 (± 0.10)	0.75 (± 0.11)	-1.46 (± 0.06)	0.95 (± 0.08)	0.95 (± 0.08)	-2.20 (± 0.02)
Airfoil	-0.74 (± 0.04)	-0.80 (± 0.05)	-1.62 (± 0.03)	0.92 (± 0.05)	0.72 (± 0.10)	-2.17 (± 0.01)
Concrete	-0.41 (± 0.05)	-0.44 (± 0.06)	-1.59 (± 0.03)	0.26 (± 0.07)	0.25 (± 0.07)	-2.03 (± 0.01)
Diabetes	-1.20 (± 0.07)	-1.20 (± 0.07)	-1.47 (± 0.07)	-1.18 (± 0.08)	-1.22 (± 0.09)	-2.09 (± 0.04)
Energy	0.92 (± 0.04)	0.69 (± 0.12)	-1.76 (± 0.02)	2.07 (± 0.46)	2.38 (± 0.11)	-1.99 (± 0.02)
ForestF	-1.37 (± 0.07)	-1.37 (± 0.07)	-1.60 (± 0.06)	-1.43 (± 0.45)	-1.69 (± 0.49)	-2.20 (± 0.02)
Yacht	1.90 (± 0.16)	1.29 (± 0.56)	-1.14 (± 0.14)	3.31 (± 0.21)	0.15 (± 0.09)	-2.18 (± 0.03)

local approximations such as LA and DE [28, 29], we conclude that a small amount of functional modes is reasonable to assume in practice. Our qualitative experiments in Section 5 support this supposition.

As an example of applying Proposition 2, assume $\nu = 3$ functionally diverse modes in a reference set with $\pi_1 = 0.57, \pi_2 = 0.35, \pi_3 = 0.08$ (chosen to represent a rather diverse functional mode set). An upper bound of $\rho = 1274$ chains ensures that we observe all functionally diverse modes with probability $\mathbb{P}(\mathcal{G} < \rho) \geq 0.99$.

5 Experiments

We now investigate our theoretical findings and compare the resulting approach to single-chain MCMC and DE. In all experiments⁸, we employ a Bayesian regression model with a normal likelihood function, standard normal prior for parameters θ , and a truncated standard normal prior restricted to the positive real line for the variance of the normal likelihood, which we treat as a nuisance parameter. Depending on the task, we either use a No-U-Turn sampler [22] with 2^{10} warmup steps to collect a single sample from the posterior or derive the maximum-a-posteriori estimator using a gradient-based method (details are given in Supplementary Material C.2, C.3).

5.1 Performance Comparison

In our first experiment, we demonstrate the predictive performance of BNNs, where the PPD is calculated based on MCMC sampling, using the derived upper bound for the number of chains (ours). In this case, we collect one sample per chain for G chains, and thus G samples in total. This is compared to MCMC sampling collecting G samples from a single chain (s.c.), and DE with ten ensemble members on three synthetic datasets (\mathcal{D}_S , \mathcal{D}_I , and \mathcal{D}_R) as well as benchmark data from [11] (for dataset details and additional results on LA,

⁸ https://anonymous.4open.science/r/efficient_mcmc_sampling_in_bnn_symmetry-6FAF/

see Supplementary Material C.1 and E.1, respectively). We use a smaller NN f_1 with a single hidden layer containing three neurons and a larger network f_2 with three hidden layers having 16 neurons each, both with tanh activation. As in Section 4.2, we assume three functionally diverse modes $\nu = 3$ and mode probabilities $\pi_1 = 0.57, \pi_2 = 0.35, \pi_3 = 0.08$ as in the given example. To demonstrate the performance of our MCMC-based PPD approximation, we measure the goodness-of-fit on the test data using the log point-wise predictive density (LPPD; [18])

$$\text{LPPD} = \log \int_{\Theta} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} \approx \log \left(\frac{1}{G} \sum_{g=1}^G p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}^{(g)}) \right), \quad (6)$$

where $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(G)}$ are G samples obtained across all chains via MCMC sampling from the parameter posterior density $p(\boldsymbol{\theta} | \mathcal{D})$. Equation (6) is evaluated at each test point $(\mathbf{x}^*, \mathbf{y}^*)$. Table 1 reports the mean LPPD across N^* independent test points for each combination of dataset and sampling scheme (see Appendix C.1 for details). Our results clearly indicate that using only a moderate amount of Markov chains, following our approach, yields equal or even better performance than single-chain MCMC and DE in all but two experiments.

5.2 Practical Evaluation of Corollary 1

Next, we investigate the property derived in Corollary 1 using our proposed upper bound of chains, again with the assumption from the example in Section 4.2. To this end, we analyze the PPD for dataset \mathcal{D}_I , using network f_2 . For every newly collected sample in the MCMC run, the updated PPD is computed approximately on a two-dimensional (input/output) grid. Then, the Kullback-Leibler (KL) divergence between consecutive densities is averaged over the grid of input values of f_2 (details in Supplementary Material D.1). As shown in Figure 2, despite the size of the network f_2 and the high amount of equioutput parameter states $|\mathcal{E}| = (16! \cdot 2^{16})^3 \approx 2.58 \cdot 10^{54}$, the PPD converges after notably fewer than $|\mathcal{E}|$ samples and plots of the function space indicate the saturation of functional diversity already after 1274 samples from as many chains.

5.3 Posterior Symmetry Removal

So far we were mainly concerned with the predictive performance of MCMC sampling. Yet, mapping all samples to a joint representative set, as characterized in Section 4.1, has the potential to reduce the effective weight space enormously, facilitating interpretability and possibly even analytical approximation. For this, we propose a custom algorithm for tanh-activated MLPs as a proof-of-concept. Our algorithm removes symmetries in a data-dependent manner and thus minimizes the number of remaining modes in the representative set (details in Supplementary Material B).

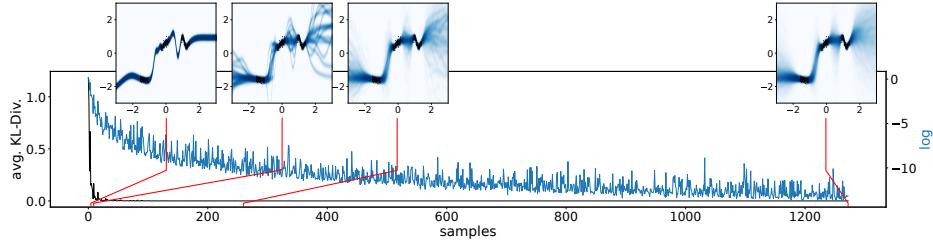


Fig. 2. Convergence of MCMC depicted as the change in KL-divergence on original (black) and log-scale (blue) when consecutively adding another sample from a new and independent chain and re-estimating the posterior density. Small overlaying plots: approximated PPD of the network after 2^0 , 2^4 , 2^8 , and $G = 1274$ samples; darker colors correspond to higher probabilities.

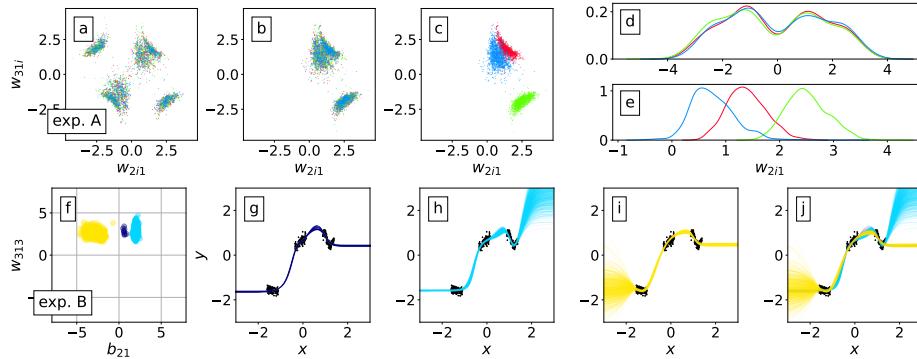


Fig. 3. For experiment A (top row), the parameter subspaces of hidden neurons are visualized in their initial state as obtained when running MCMC (a), and after their transformation and reassignment during the steps of Algorithm 4 (b, c). Different colors encode the current neuron index in the hidden layer. By reassigning neurons, Algorithm 4 effectively finds an optimal reference set and allows to separate the multi-modal complex univariate marginal density (d) into three univariate densities. For experiment B (bottom row), the MLP parameter posterior density after symmetry removal results in a tri-modal system, here illustrated as a bivariate plot (f). Investigating these modes, we find that all are functionally diverse, i.e., represent a different hypothesis of the dataset (g-i). Combined, they form the full function space (j).

We demonstrate the efficacy of the approach for f_1 in two experiments A and B for datasets \mathcal{D}_S and \mathcal{D}_I , respectively. For experiment A (Figure 3, top) we visualize the neuron parameter space along the steps of the proposed algorithm (for details, see Algorithms 1-4 in Supplementary Material B). Different colors encode the current neuron index (i.e., one of $\{1, 2, 3\}$) in the hidden layer of the respective neuron parameter vector. Initially, symmetries of the posterior densities are clearly noticeable (Figure 3a), and the neuron parameter vectors

are distributed identically (Figure 3d). Upon the first step of the algorithm (Figure 3b), the parameter space is effectively halved as a consequence of removing the tanh-induced sign-flip symmetry, and three clusters remain. The second (clustering) step removes all permutation symmetries from the full posterior density by assigning areas of this parameter subspace to the neurons of the model. This is depicted in Figure 3c and clearly shows the separation of states by different cluster colors. In Figures 3d and 3e, the univariate marginal density of each neuron’s incoming weight w_{2i1} reveals their reassignment to the parameter space. After running our algorithm, each neuron exhibits a distinct unimodal density, resulting in a unimodal density in the full parameter space (visualization in Supplementary Material E.3). We can conclude that only one functionally diverse mode exists in this case, which should also be recovered by local approximations like LA.

For experiment B (Figure 3, bottom), we focus on the visualization of the full parameter posterior density obtained after the application of the symmetry removal algorithm (Figure 3f). Three functionally diverse modes, represented by different colors, remain in the BNN posterior. We can now interpret these modes in function space by further clustering the transformed samples using a spectral clustering approach (details in Supplementary Material D.2). Figures 3g-j visualize the network parameter states in the function space, revealing three functionally diverse hypotheses the network can potentially learn from the given dataset. Such knowledge allows for a better-suited approximation by, e.g., a mixture of Laplace approximations (MoLA; [13]), as shown in Supplementary Material E.2. Note that approaches focusing on a single mode, such as standard LA, would have captured only a third of the functional diversity. For meaningful UQ, it is thus imperative that all functionally diverse modes are accounted for.

6 Discussion

We showed that the PPD for Bayesian MLPs can be obtained from just a fraction of the parameter space due to the existence of equioutput parameter states. Together with an upper bound on the number of MCMC chains to guarantee the recovery of every functionally diverse mode, our approach paves the way towards exact uncertainty quantification (up to a Monte Carlo error) in deep learning. Furthermore, we demonstrate the use of symmetry removal and present a proof-of-concept approach to map samples of tanh-activated MLPs to a representative set. This *post-hoc* procedure improves the interpretability of the symmetry-free posterior density drastically and facilitates analytical approximations. As a future research direction, we plan to investigate whether our MCMC sampling approach can be improved by initializing the sampling states in an informative way via ensemble training, building upon insights in [19, 36, 46].

Acknowledgments. LW is supported by the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the German Federal Ministry of Education and Research.

References

1. Agrawal, D., Ostrowski, J.: A classification of G-invariant shallow neural networks. In: Advances in Neural Information Processing Systems (2022)
2. Ainsworth, S., Hayase, J., Srinivasa, S.: Git re-basin: Merging models modulo permutation symmetries. In: The Eleventh International Conference on Learning Representations (2023), <https://openreview.net/forum?id=CQsmMYm1P5T>
3. Bardenet, R., Kégl, B.: An adaptive Monte-Carlo Markov chain algorithm for inference from mixture signals. Journal of Physics: Conference Series **368** (2012)
4. Bona-Pellissier, J., Bachoc, F., Malgouyres, F.: Parameter identifiability of a deep feedforward ReLU neural network (2021)
5. Van den Broeck, G., Kersting, K., Natarajan, S., Poole, D.: An Introduction to Lifted Probabilistic Inference. MIT Press (2021), <https://mitpress.mit.edu/books/introduction-lifted-probabilistic-inference>
6. Chen, A.M., Lu, H.m., Hecht-Nielsen, R.: On the Geometry of Feedforward Neural Network Error Surfaces. Neural Computation **5**(6), 910–927 (1993)
7. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3), 273–297 (1995)
8. Cortez, P., Morais, A.d.J.R.: A data mining approach to predict forest fires using meteorological data (2007)
9. Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., Hennig, P.: Laplace Redux – Effortless Bayesian Deep Learning. In: 35th Conference on Neural Information Processing Systems (NeurIPS 2021) (2021)
10. Draxler, F., Veschgini, K., Salmhofer, M., Hamprecht, F.: Essentially No Barriers in Neural Network Energy Landscape. In: Proceedings of the 35th International Conference on Machine Learning. pp. 1309–1318. PMLR (2018)
11. Dua, D., Graff, C.: UCI machine learning repository (2017)
12. Ensign, D., Neville, S., Paul, A., Venkatasubramanian, S.: The complexity of explaining neural networks through (group) invariants. In: Proceedings of Machine Learning Research. vol. 76 (2017)
13. Eschenhagen, R., Daxberger, E., Hennig, P., Kristiadi, A.: Mixtures of Laplace Approximations for Improved Post-Hoc Uncertainty in Deep Learning. In: Bayesian Deep Learning Workshop, NeurIPS 2021 (2021)
14. Falcon, W.: PyTorch lightning. <https://github.com/PyTorchLightning> (2019)
15. Flajolet, P., Gardy, D., Thimonier, L.: Birthday paradox, coupon collectors, caching algorithms and self-organizing search. Discrete Applied Mathematics **39**(3), 207–229 (1992)
16. Frühwirth-Schnatter, S.: Markov chain Monte Carlo Estimation of Classical and Dynamic Switching and Mixture Models. Journal of the American Statistical Association **96**(453), 194–209 (2001)
17. Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D.P., Wilson, A.G.: Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. In: Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018) (2018)
18. Gelman, A., Hwang, J., Vehtari, A.: Understanding predictive information criteria for Bayesian models. Statistics and Computing **24**(6), 997–1016 (2014)
19. Graf, S., Luschgy, H.: Foundations of quantization for probability distributions. Springer (2007)
20. Hecht-Nielsen, R.: On the algebraic structure of feedforward network weight spaces. In: Advanced Neural Computers, pp. 129–135. Elsevier (1990)

21. Hinton, G., Srivastava, N., Swersky, K.: RMSProp: Divide the Gradient by a Running Average of Its Recent Magnitude (Lecture on Neural Networks for Machine Learning) (2012)
22. Hoffman, M.D., Gelman, A.: The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research* **15**(47), 1593–1623 (2014)
23. Hüllermeier, E., Waegeman, W.: Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *Machine Learning* (2021)
24. Izmailov, P., Maddox, W.J., Kirichenko, P., Garipov, T., Vetrov, D., Wilson, A.G.: Subspace Inference for Bayesian Deep Learning. In: Proceedings of The 35th Uncertainty in Artificial Intelligence Conference. pp. 1169–1179. PMLR (2020)
25. Izmailov, P., Vikram, S., Hoffman, M.D., Wilson, A.G.: What Are Bayesian Neural Network Posteriors Really Like? In: Proceedings of the 38th International Conference on Machine Learning, PMLR 139 (2021)
26. Kipf, T.N., Welling, M.: Semi-Supervised Classification with Graph Convolutional Networks. In: ICLR 2017 (2017)
27. Kůrková, V., Kainen, P.C.: Functionally Equivalent Feedforward Neural Networks. *Neural Computation* **6**(3), 543–558 (1994)
28. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In: Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017) (2017)
29. MacKay, D.J.C.: Bayesian Interpolation. *Neural Computation* **4**, 415–447 (1992)
30. Margossian, C.C., Hoffman, M.D., Sountsov, P., Riou-Durand, L., Vehtari, A., Gelman, A.: Nested \$\\hat{R}\$: Assessing the convergence of Markov chain Monte Carlo when running many short chains (2022)
31. Nalisnick, E.T.: On Priors for Bayesian Neural Networks. Ph.D. thesis, University of California, Irvine (2018)
32. Niepert, M.: Markov chains on orbits of permutation groups. In: Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence. p. 624–633. UAI'12, AUAI Press, Arlington, Virginia, USA (2012)
33. Niepert, M.: Symmetry-aware marginal density estimation. In: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence. p. 725–731. AAAI'13, AAAI Press (2013)
34. Papamarkou, T., Hinkle, J., Young, M.T., Womble, D.: Challenges in Markov Chain Monte Carlo for Bayesian Neural Networks. *Statistical Science* **37**(3) (2022)
35. Paszke, A., et al.: Pytorch: An imperative style, high-performance deep learning library. In: 33rd Conference on Neural Information Processing Systems (2019)
36. Pearce, T., Leibfried, F., Brinstrup, A.: Uncertainty in neural networks: Approximately Bayesian ensembling. In: Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 108, pp. 234–244. PMLR (26–28 Aug 2020)
37. Petzka, H., Trimmel, M., Sminchisescu, C.: Notes on the Symmetries of 2-Layer ReLU-Networks. Northern Lights Deep Learning Workshop **1** (2020)
38. Pittorino, F., Ferraro, A., Perugini, G., Feinauer, C., Baldassi, C., Zecchina, R.: Deep networks on toroids: Removing symmetries reveals the structure of flat regions in the landscape geometry. In: Proceedings of the 39th International Conference on Machine Learning, PMLR 162 (2022)
39. Pourzanjani, A.A., Jiang, R.M., Petzold, L.R.: Improving the Identifiability of Neural Networks for Bayesian Inference. In: Second Workshop on Bayesian Deep Learning (NIPS) (2017)

40. Rosenthal, J.S.: Parallel computing and Monte Carlo algorithms. *Far East Journal of Theoretical Statistics* **4**, 207–236 (2000)
41. Sen, D., Papamarkou, T., Dunson, D.: Bayesian neural networks and dimensionality reduction (2020), arXiv: 2008.08044
42. Sussmann, H.J.: Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural Networks* **5**(4), 589–593 (1992)
43. Tattro, N.J., Chen, P.Y., Das, P., Melnyk, I., Sattigeri, P., Lai, R.: Optimizing Mode Connectivity via Neuron Alignment. In: Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020) (2020)
44. Tsanas, A., Xifara, A.: Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and buildings* **49**, 560–567 (2012)
45. Vlačić, V., Bölcseki, H.: Affine symmetries and neural network identifiability. *Advances in Mathematics* **376**, 107485 (2021). <https://doi.org/https://doi.org/10.1016/j.aim.2020.107485>, <https://www.sciencedirect.com/science/article/pii/S0001870820305132>
46. Wilson, A.G., Izmailov, P.: Bayesian deep learning and a probabilistic perspective of generalization. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS’20, Curran Associates Inc., Red Hook, NY, USA (2020)
47. Yeh, I.C.: Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research* **28**(12), 1797–1808 (1998)

A Derivations

A.1 Functional Equality of Equioutput Parameter States

We consider permutation and activation function-related equioutput transformation of the parameter states. The following shows the functional equality for the interchange of neurons and two activation functions, tanh and ReLU.

Permutation-Related Equality. Due to the commutative property of addition, the permutation of two neurons v, w in the $(l-1)$ -th layer does not change the pre-activation o_{li} of the i -th neuron in the l -th layer.

$$\begin{aligned} o_{li} &= (w_{li1}z_{(l-1)1} + b_{l1}) + \cdots + (w_{liv}z_{(l-1)v} + b_{lv}) + (w_{liw}z_{(l-1)w} + b_{lw}) \\ &\quad + \cdots + (w_{liM_{l-1}}z_{(l-1)M_{l-1}} + b_{lM_{l-1}}) \\ &= (w_{li1}z_{(l-1)1} + b_{l1}) + \cdots + (w_{liw}z_{(l-1)w} + b_{lw}) + (w_{liv}z_{(l-1)v} + b_{lv}) \\ &\quad + \cdots + (w_{liM_{l-1}}z_{(l-1)M_{l-1}} + b_{lM_{l-1}}) \end{aligned}$$

Tanh-Related Equality. Since tanh is an odd function, flipping the signs of all parameters tethered to the outputs of incoming neurons (from layer $l-1$) and outgoing neurons (to layer $l+1$) as well as the associated bias parameter of layer l leaves the output unchanged.

$$\begin{aligned} w_{(l+1)ki} \tanh(o_{li}) &= w_{(l+1)ki} \tanh \left(\sum_{j=1}^{M_{l-1}} w_{lij} z_{(l-1)j} + b_{li} \right) \\ &= -w_{(l+1)ki} \tanh \left(\sum_{j=1}^{M_{l-1}} -w_{lij} z_{(l-1)j} - b_{li} \right) \\ &= -w_{(l+1)ki} \tanh(-o_{li}) \end{aligned}$$

ReLU-Related Equality. Similar to tanh, in the case of ReLU, rescaling the parameters of incoming (from layer $l-1$) and outgoing neuron outputs (to layers $l+1$) and the bias parameter of layer l generates equioutput parameter states.

$$\begin{aligned} w_{(l+1)ki} \text{ReLU}(o_{li}) &= w_{(l+1)ki} \text{ReLU} \left(\sum_{j=1}^{M_{l-1}} w_{lij} z_{(l-1)j} + b_{li} \right) \\ &= c \cdot w_{(l+1)ki} \text{ReLU} \left(\sum_{j=1}^{M_{l-1}} c^{-1} \cdot w_{lij} z_{(l-1)j} + c^{-1} \cdot b_{li} \right) \\ &= c \cdot w_{(l+1)ki} \text{ReLU}(c^{-1} \cdot o_{li}) \end{aligned}$$

A.2 Equioutput Equivalence Relation

Let $f : \mathcal{X} \rightarrow \mathcal{Y}$ represent an MLP mapping a feature vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n, n \in \mathbb{N}$, to an outcome vector $f(\mathbf{x}) \in \mathcal{Y} \subseteq \mathbb{R}^m, m \in \mathbb{N}$. Two parameter states $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta$ are considered *equioutput* if the maps $f_{\boldsymbol{\theta}}, f_{\boldsymbol{\theta}'}$ yield the same outputs for all possible inputs. We denote this relation by \sim and write:

$$\boldsymbol{\theta} \sim \boldsymbol{\theta}' \Leftrightarrow f_{\boldsymbol{\theta}}(\mathbf{x}) = f_{\boldsymbol{\theta}'}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{X}, \quad \boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta.$$

\sim constitutes an equivalence relation, which follows immediately from equality being an equivalence relation:

E1 (Reflexivity) For $\boldsymbol{\theta} \in \Theta$, $\boldsymbol{\theta} \sim \boldsymbol{\theta}$ since $f_{\boldsymbol{\theta}}(\mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{X}$.

E2 (Symmetry) For $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta$,

$$\boldsymbol{\theta} \sim \boldsymbol{\theta}' \Leftrightarrow f_{\boldsymbol{\theta}}(\mathbf{x}) = f_{\boldsymbol{\theta}'}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{X} \Rightarrow f_{\boldsymbol{\theta}'}(\mathbf{x}) = f_{\boldsymbol{\theta}}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{X} \Leftrightarrow \boldsymbol{\theta}' \sim \boldsymbol{\theta}$$

E3 (Transitivity) Let $\boldsymbol{\theta}_1 \sim \boldsymbol{\theta}_2$ and $\boldsymbol{\theta}_2 \sim \boldsymbol{\theta}_3$ for $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3 \in \Theta$. Since

$$\boldsymbol{\theta}_1 \sim \boldsymbol{\theta}_2 \Leftrightarrow f_{\boldsymbol{\theta}_1}(\mathbf{x}) = f_{\boldsymbol{\theta}_2}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{X}$$

and

$$\boldsymbol{\theta}_2 \sim \boldsymbol{\theta}_3 \Leftrightarrow f_{\boldsymbol{\theta}_2}(\mathbf{x}) = f_{\boldsymbol{\theta}_3}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{X},$$

we have, by the transitivity of equality,

$$f_{\boldsymbol{\theta}_1}(\mathbf{x}) = f_{\boldsymbol{\theta}_3}(\mathbf{x}) \forall \mathbf{x} \in \mathcal{X} \Leftrightarrow \boldsymbol{\theta}_1 \sim \boldsymbol{\theta}_3.$$

A.3 Posterior Density Reference Set

Prior Invariance For a neural network parameter state $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$ and any equioutput parameter state $\mathbf{E}_j \in \mathcal{E}$, the prior probability for $\boldsymbol{\theta}$ and $\mathbf{E}_j \boldsymbol{\theta}$ is identical if we assume the prior to be invariant under transformations $\mathcal{F}_{\mathbf{E}_j} : \Theta \rightarrow \Theta, \boldsymbol{\theta} \mapsto \mathbf{E}_j \boldsymbol{\theta}, \mathbf{E}_j \in \mathcal{E}$, i.e., $p(\boldsymbol{\theta}) = p(\mathbf{E}_j \boldsymbol{\theta})$. Since equioutput parameter states produce, by definition, the same functional mapping, it is only reasonable to assign them identical prior probability. A notable example is the universally-used isotropic Gaussian prior. The following shows the equivalence of prior probabilities of the original and transformed weight vector for this case.

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} | \mathbf{0}, \mathbf{I}) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{\theta}\right) \quad (7)$$

$$= \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}\boldsymbol{\theta}^\top \mathbf{E}_j^\top \mathbf{E}_j \boldsymbol{\theta}\right) \quad (8)$$

$$= \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2}(\mathbf{E}_j \boldsymbol{\theta})^\top (\mathbf{E}_j \boldsymbol{\theta})\right) \quad (9)$$

$$= \mathcal{N}(\mathbf{E}_j \boldsymbol{\theta} | \mathbf{0}, \mathbf{I}) = p(\mathbf{E}_j \boldsymbol{\theta}) \quad (10)$$

The orthogonality of \mathbf{E}_j follows from the fact that $\mathbf{E}_j = \mathbf{T} \mathbf{P}$ is the product of orthogonal matrices. \mathbf{P} is a permutation matrix and, therefore, orthogonal.

\mathbf{T} is a diagonal matrix that is also orthogonal because the transformation it represents are designed precisely such that $\mathbf{T}^\top \mathbf{T}$ is the identity matrix (e.g., with diagonal values equal to -1 for \tanh). The orthogonality of \mathbf{E}_j is used to pass from Equation (7) to Equation (8). Note, however, that scaling transformations such as those induced by ReLU can generally not be represented by orthogonal matrices. Due to the infinite amount of possible scaling transformations, these pose a less tangible problem.

Likelihood Invariance Similarly, we show that the likelihood is invariant to equioutput parameter states, using the fact that $\boldsymbol{\theta}$ and $\mathbf{E}_j \boldsymbol{\theta}$ are equioutput parameter states:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \boldsymbol{\theta}) = \prod_{i=1}^N p(\mathbf{y}^{(i)}|f_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) \quad (11)$$

$$= \prod_{i=1}^N p(\mathbf{y}^{(i)}|f_{\mathbf{E}_j \boldsymbol{\theta}}(\mathbf{x}^{(i)})) \quad (12)$$

$$= \prod_{i=1}^N p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \mathbf{E}_j \boldsymbol{\theta}) = p(\mathcal{D}|\mathbf{E}_j \boldsymbol{\theta}) \quad (13)$$

Posterior Invariance Following from Equations 7 to 13, the posterior probabilities for two equioutput parameter states are identical:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \stackrel{(13)}{=} \frac{p(\mathcal{D}|\mathbf{E}_j \boldsymbol{\theta})p(\mathbf{E}_j \boldsymbol{\theta})}{p(\mathcal{D})} = p(\mathbf{E}_j \boldsymbol{\theta}|\mathcal{D}) \quad (14)$$

A.4 Proof Sketch for Proposition 1

Here, we provide a proof sketch for Proposition 1, which fundamentally builds on [6] and contains some considerations on handling edge and boundary cases. Let $\mathcal{S}_1 \subset \Theta$ be a complete set of representatives from the equivalence classes induced by the equioutput relation \sim , s.t. $\forall \boldsymbol{\theta} \in \Theta \exists_1 \boldsymbol{\theta}' \in \mathcal{S}_1 : \boldsymbol{\theta} \sim \boldsymbol{\theta}'$. We call \mathcal{S}_1 the *reference set* of uniquely identified network parameter states (cf. open minimal sufficient search sets in [6]). Let \mathcal{S}_j be the image of \mathcal{S}_1 under the transformation $\mathcal{F}_{\mathbf{E}_j}$ induced by $\mathbf{E}_j \in \mathcal{E}$, s.t.

$$\mathcal{S}_j = \mathcal{F}_{\mathbf{E}_j}(\mathcal{S}_1) = \{\mathbf{E}_j \boldsymbol{\theta} : \boldsymbol{\theta} \in \mathcal{S}_1, \mathbf{E}_j \in \mathcal{E}\}, \quad j = 1, \dots, |\mathcal{E}|.$$

Now, we collect all edge and boundary cases in the residual set \mathcal{S}^0 by the following operations:

$$\mathcal{S}^0 := \Theta \setminus \bigcup_{j=1}^{|\mathcal{E}|} \mathcal{S}_j$$

$$\mathcal{S}^0 = \mathcal{S}^0 \cup \bigcup_{i \neq j} (\mathcal{S}_i \cap \mathcal{S}_j)$$

$$\begin{aligned} \forall j = 1, \dots, |\mathcal{E}| : \mathcal{S}_j &= \mathcal{S}_j \setminus \mathcal{S}^0 \\ \Rightarrow \forall i, j \in \{1, \dots, |\mathcal{E}|\}, i \neq j : \mathcal{S}_i \cap \mathcal{S}_j &= \emptyset, \quad \mathcal{S}_j \cap \mathcal{S}^0 = \emptyset \end{aligned}$$

We thus have sets $\mathcal{S}_i, \mathcal{S}_j$ that are pairwise disjoint and whose intersection with the residual set \mathcal{S}^0 is empty. It remains to be shown that the union over all these sets is equal to Θ . For this, consider $\boldsymbol{\theta} \in \Theta$ with $\boldsymbol{\theta} \notin \left(\mathcal{S}^0 \cup \bigcup_{j=1}^{|\mathcal{E}|} \mathcal{S}_j\right)$. Since the reference set contains, by definition, a representative of each equivalence class, there exists $\boldsymbol{\theta}' \in \mathcal{S}_1 : \boldsymbol{\theta} \sim \boldsymbol{\theta}'$. From this, however, it follows that $\exists j \in \{1, \dots, |\mathcal{E}|\} : \boldsymbol{\theta} = \mathbf{E}_j \boldsymbol{\theta}' \Rightarrow \boldsymbol{\theta} \in \mathcal{S}_j$, and in particular, $\boldsymbol{\theta} \in \left(\mathcal{S}^0 \cup \bigcup_{j=1}^{|\mathcal{E}|} \mathcal{S}_j\right)$, such that we reach a contradiction.

A.5 Proof for Corollary 1

Combining the results from A.3 and A.4, it follows that the posterior predictive density expresses as:

$$p(\mathbf{y}^* | \mathbf{x}^*, \mathcal{D}) = \int_{\Theta} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} \quad (15)$$

$$= \sum_{i=1}^{|\mathcal{E}|} \int_{\mathcal{S}_i} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} \quad (16)$$

$$+ \underbrace{\int_{\mathcal{S}^0} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta}}_c \quad (17)$$

$$= \sum_{i=1}^{|\mathcal{E}|} \int_{\mathcal{S}_i} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} + c \quad (17)$$

$$= \sum_{i=1}^{|\mathcal{E}|} \int_{\mathcal{S}_1} p(\mathbf{y}^* | \mathbf{x}^*, \mathbf{E}_i \boldsymbol{\theta}) p(\mathbf{E}_i \boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} + c \quad (18)$$

$$= \int_{\mathcal{S}_1} \sum_{i=1}^{|\mathcal{E}|} p(\mathbf{y}^* | \mathbf{x}^*, \mathbf{E}_i \boldsymbol{\theta}) p(\mathbf{E}_i \boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} + c \quad (19)$$

$$= \int_{\mathcal{S}_1} |\mathcal{E}| \cdot p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} + c \quad (20)$$

$$= |\mathcal{E}| \cdot \int_{\mathcal{S}_1} p(\mathbf{y}^* | \mathbf{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta} | \mathcal{D}) d\boldsymbol{\theta} + c \quad (21)$$

Equation (1) enables the transition from Equation (15) to 16. The likelihood and posterior invariance, as stated in Equations (13) and (14), allow to rewrite Equation (17) as (18).

A.6 Upper Bound of Markov Chains

We now prove the upper bound derived in Equation (5). We first note that Markov's inequality for the number of chains G yields

$$P(G \geq \rho) \leq \mathbb{E}(G) \cdot \rho^{-1}. \quad (22)$$

We can rewrite this inequality as

$$P(G < \rho) \geq 1 - \mathbb{E}[G] \cdot \rho^{-1}. \quad (23)$$

The number of independent chains that are required to visit every mode at least once can be framed as a generalized coupon collector's problem [15], where the number of draws (independent chains) needed to collect (visit) all ν coupons (modes) is estimated. This gives us an expression for the expected number of chains:

$$\mathbb{E}(G) = \sum_{q=0}^{\nu-1} (-1)^{\nu-1-q} \sum_{J:|J|=q} (1 - \Pi_J)^{-1}, \quad \text{with } \Pi_J = \sum_{j \in J} \pi_j. \quad (24)$$

Putting together Equation (23) and Equation (24), we get Equation (5).

B Posterior Symmetry Removal

Section 4.2 uses the knowledge of equioutput parameter states to derive an upper bound on the number of Markov chains required to observe all functionally diverse modes of the MLP parameter posterior density. However, the obtained samples are scattered across the sets \mathcal{S}_j ; see Section 4 and Equation (2). This obscures any insights into the geometry of the network's parameter posterior density and makes it infeasible to interpret anything but the obtained functional uncertainty. Ideally, all samples should therefore be mapped to a representative set. In the following, we derive a symmetry removal algorithm for tanh-activated MLPs.

Reasoning. Assume G available posterior samples $\boldsymbol{\theta}^{(g)}, g \in \{1, \dots, G\}$. As equioutput transformations in MLPs factorize layerwise (see Section 3), symmetries can be removed by iterating through the $K - 2$ hidden network layers. We propose to operate on the layers in reverse order, motivated by the idea that the output of the previous layer $l - 1$ can be interpreted as an input to a single-layer MLP with M_l neurons. Therefore, processing an MLP in this way is comparable to removing symmetries from $K - 2$ independent single-layer MLPs.

In each step, it is sufficient to consider the parameters of neurons from the current hidden layer l . For a hidden neuron $i \in \{1, \dots, M_l\}$ in the l -th layer, the corresponding parameters are the weights and biases from the neuron output o_{li}

and weights connecting the neuron to the following layer. The neuron parameter vector $\phi^{(g,l,i)} \in \mathbb{R}^{(M_{(l-1)}+M_{(l+1)})+1}$ is defined by

$$\phi^{(g,l,i)} = (w_{li1}^{(g)}, \dots, w_{l(M_{l-1})}^{(g)}, w_{(l+1)1i}^{(g)}, \dots, w_{(l+1)M_{(l+1)}i}^{(g)}, b_{li}^{(g)})^\top,$$

with $w_{lij}^{(g)}, b_{li}^{(g)}$ for $2 \leq l \leq K-1, 1 \leq i \leq M_l, 1 \leq j \leq M_{(l-1)}$. This vector is an element of a subspace of Θ whose dimensionality depends on the width of the previous and subsequent layer. As neurons can be freely interchanged (see Section 3), the marginal posteriors of freely permutable neurons from the same hidden layer are identical. This implies that the marginal density of every $\phi^{(g,l,i)}$ contains at least M_l different regions to which a neuron might be assigned in its mapping to the reference set, allowing for $M_l!$ different permutation configurations. The permutation-related symmetries can be removed by finding a valid reference assignment of states to hidden neurons, which effectively dissects the parameter subspace into M_l regions. For tanh-activated MLPs, as considered in the subsequent paragraph, the number of states is further doubled due to sign-flip equioutput parameter states. Therefore, prior to the permutation symmetry removal, we cut the parameter subspace in half in a geometry-respecting manner for tanh-related symmetry removal (details below). In the following, we present an exemplary algorithm for automatic symmetry removal in tanh-activated MLPs without the need to explicitly specify \mathcal{E} .

Detailed Algorithm. We need to find identifiability constraints, i.e., a selection of hyperplanes in parameter space that distribute the available space uniquely to the neurons of a layer. As a result, neurons cannot possess symmetric configurations, which effectively removes symmetries. For this, we follow the principle from [16] and work layer-wise in the subspace of the neuron parameters that are involved in the corresponding equioutput transformations of that layer. Let $\mathcal{M}^{(g,l)} := \{\phi^{(g,l,1)}, \dots, \phi^{(g,l,M_l)}\}$ be the collection of all parameters of the l -th layer for a sample g , and let $\mathcal{M}^{(*,l)} = \bigcup_{g=1}^G \mathcal{M}^{(g,l)}$. In order to remove the tanh-related symmetries from a layer l , the parameter subspace of the respective neurons must be reduced by half. Halving the space while respecting the geometry of the posterior density poses a data-dependent optimization problem of finding the right constraints. Here, we propose a zero-centered linear hyperplane $\beta_l \in \mathbb{R}^{M_{(l-1)}+M_{(l+1)}+1}$ that intersects as few states or clusters of neuron parameter vectors $\phi \in \mathcal{M}^{(*,l)}$ as possible. We optimize this hyperplane using a variant of a support vector machine (SVM; [7]) for unlabeled data, such that it has a maximum distance to all neuron parameter vectors. The loss function follows the hinge-loss formulation of SVMs but substitutes the labels for absolute values of $\beta_l^\top \phi$ for $\phi \in \mathcal{M}^{(*,l)}$, taking the form

$$\mathcal{L}(\beta_l) = \frac{1}{2} \beta_l^\top \beta_l + C \cdot \sum_{\phi \in \mathcal{M}^{(*,l)}} \max(0, 1 - |\beta_l^\top \phi|),$$

where $C > 0$ is a cost-related hyperparameter and $|\cdot|$ a user-defined norm (we use the L_1 norm in analogy to the hinge loss). The loss-minimal hyperplane is chosen as a geometry-respecting constraint and is used for flipping parameter

vectors to one side of the hyperplane, i.e., applying the a tanh-related equioutput transformation and flipping the signs of the corresponding parameter vector entries (Algorithm 1). In practice we run $K_{\beta} > 1$ optimizations with randomly initialized hyperplanes since the global optimum is not guaranteed from this SVM variation.

Algorithm 1 Geometry-respecting tanh removal algorithm.

```

procedure TANHREMOVAL( $\mathcal{M}^{(*,l)}$ ,  $K_{\beta}$ )
     $\mathcal{R} \leftarrow \emptyset$ 
    for each  $k$  in range  $K_{\beta}$  do
         $\beta_k \leftarrow \arg \min_{\beta} \mathcal{L}(\beta)$ 
         $\mathcal{R} \leftarrow \mathcal{R} \cup \{\beta_k\}$ 
    end for
     $\beta^* \leftarrow \beta \in \mathcal{R}$  with minimal loss  $\mathcal{L}(\beta)$ 
    for each  $\phi \in \mathcal{M}^{(*,l)}$  do
        if  $\beta^{*\top} \phi < 0$  then
             $\phi \leftarrow -\phi$ 
        end if
    end for
end procedure

```

Following the removal of tanh-related symmetries using the SVM approach, the remaining parameter subspace in the marginal posterior density of the parameters in layer l must be divided among the hidden-layer neurons in order to remove the permutation-related symmetries. For this, we consider one sample $\theta^{(g)}$ with elements $\phi^{(g,l,i)} \in \mathcal{M}^{(g,l)}$ at a time and assign classification labels $c^{(g,l,i)}, i \in \{1, \dots, M_l\}$. Initially, each element is labeled with its neuron index i . We then perform a greedy constrained k -NN classification (Algorithm 2) on the elements of each set $\mathcal{M}^{(g,l)}$ based on their pairwise spatial distances using the Gaussian similarity function $s(a, b) = \exp(-\|a - b\|^2 \cdot (2\sigma^2)^{-1})$ with $\sigma = 1$ to assign each neuron parameter vector $\phi^{(g,l,i)}$ to its most likely neuron position in the hidden layer. The hyperparameter k should be carefully selected. We generally recommend high values, such that the classification is based on as many neighbors as possible. We set $k = 1024$ in all experiments.

Algorithm 2 Algorithm to perform greedy constrained k -NN classification.

```

procedure GREEDYCONSTRAINEDKNNCLASSIFICATION( $\mathcal{M}^{(g,l)}$ ,  $\mathcal{M}^{(*,l)}$ ,  $k$ )
     $\mathcal{R} \leftarrow \emptyset$ 
    for each  $i$  in range  $M_l$  do
         $\phi^{(g,l,i)} \in \mathcal{M}^{(g,l)}$ 
         $(p_i(c=1), \dots, p_i(c=M_l)) \leftarrow \text{KNNCLASSIFICATION}(\phi^{(g,l,i)}, \mathcal{M}^{(*,l)}, k)$ 
         $\mathcal{R} \leftarrow \mathcal{R} \cup \{(p_i(c=1), \dots, p_i(c=M_l))\}$ 
    end for
     $i \leftarrow 1$ 
    repeat
        Find the prob. vector  $p_j$  from  $\mathcal{R}$  with highest probability for a class  $c^*$ .
         $c^{(g,l,j)} \leftarrow \arg \max_c p_j(c=c^*)$ 
         $i \leftarrow i + 1$ 
    until  $i = M_l$ 
    Set  $p_j(c=c^*) = 0$  for all  $j$ .
    return  $\{c^{(g,l,1)}, \dots, c^{(g,l,M_l)}\}$ 
end procedure

```

This is followed by a permutation of the neuron parameter vectors of the layer according to the classification results, i.e., neurons are re-indexed to their class assignment (Algorithm 3) in the PERMUTE operation. This effectively clusters the parameter subspace into M_l regions that implicitly define constraints and remove permutation-related symmetries. The k -NN classification is constrained in such a way that no pair of neuron parameter vectors ($\phi^{(g,l,i)}, \phi^{(g,l,j)}$) in the set $\mathcal{M}^{(g,l)}$ is allowed to have the same class. This is done greedily by assigning the element with the highest probability for any class first, followed by the remaining elements in decreasing order of probabilities, until all vectors are assigned. The process is repeated for I iterations over all $K-1$ hidden layers and G Monte Carlo samples to obtain a globally coherent clustering of neuron parameter vectors, since the classification by the greedy constrained k -NN classification algorithm is based on local relationships. We set $I = 256$ in all experiments.

Algorithm 3 Permutation symmetry removal for a hidden layer l .

```

procedure PERMUTATIONREMOVAL( $\mathcal{M}^{(1,l)}, \dots, \mathcal{M}^{(G,l)}, \mathcal{M}^{(*,l)}$ ,  $k, I$ )
    for  $I$  times do
        for each  $g$  in range  $G$  do
             $\{c^{(g,l,1)}, \dots, c^{(g,l,M_l)}\} \leftarrow$ 
            GREEDYCONSTRAINEDKNNCLASSIFICATION( $\mathcal{M}^{(g,l)}$ ,  $\mathcal{M}^{(*,l)}$ ,  $k$ )
        end for
        for each  $s$  in range  $S$  do
            PERMUTE( $\mathcal{M}^{(g,l)}, \{c^{(g,l,1)}, \dots, c^{(g,l,M_l)}\}$ )
        end for
    end for
end procedure

```

Algorithm 4 outlines the entire algorithm for removing symmetries from a Markov chain sampled from an MLP parameter posterior density. Algorithm 4 iteratively invokes Algorithms 1 and 3.

Algorithm 4 Complete geometry-respecting symmetry removal algorithm.

```

procedure GEOMETRYREMOVAL( $\{\theta^1, \dots, \theta^G\}$ ,  $I$ ,  $k$ ,  $K_\beta$ )
    for each layer  $l$  (reverse) do
        Construct neuron parameter sets  $M^{(1,l)}, \dots, M^{(G,l)}$  from MCMC samples
         $\{\theta^1, \dots, \theta^G\}$ 
         $\mathcal{M}^{(*,l)} \leftarrow \mathcal{M}^{(1,l)} \cup \dots \cup \mathcal{M}^{(G,l)}$ 
        TANHREMOVAL( $\mathcal{M}^{(*,l)}$ ,  $K_\beta$ ); see Algorithm 1
        PERMUTATIONREMOVAL( $\mathcal{M}^{(1,l)}, \dots, \mathcal{M}^{(G,l)}, \mathcal{M}^{(*,l)}$ ,  $k$ ,  $I$ ); see Algorithm 3
    end for
end procedure

```

C Experimental Setup

C.1 Datasets

The sinusoidal dataset \mathcal{D}_S is adopted from an example provided in [9], and the dataset \mathcal{D}_I is a synthetic dataset from [24]. The Regression2d dataset \mathcal{D}_R is another synthetic dataset that we generate as follows:

$$\begin{aligned}
p(x_1) &= \mathcal{U}(a = -2.0, b = 2.0) \\
p(x_2) &= \mathcal{U}(a = -2.0, b = 2.0) \\
f(x_1, x_2) &= x_1 \cdot \sin(x_1) + \cos(x_2) \\
p(y|x_1, x_2) &= \mathcal{N}(\mu = f(x_1, x_2), \sigma = 0.1) \\
p(\mathcal{D}) &= \prod_{i=1}^{256} p(y^{(i)}|x_1^{(i)}, x_2^{(i)})p(x_1^{(i)})p(x_2^{(i)})
\end{aligned}$$

The three synthetic datasets \mathcal{D}_S , \mathcal{D}_I and \mathcal{D}_R were standardized for all experiments and are visualized in Figure 4. We split the data in 80% training and 20% test observations. Together with the synthetic datasets we further provide descriptions and references also for the UCI datasets used in our benchmarks in Table 2.

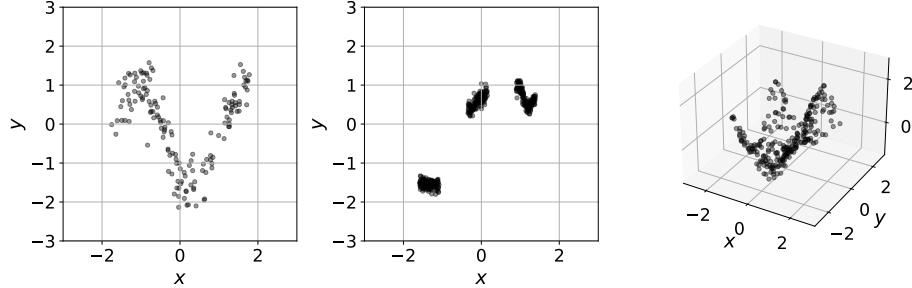


Fig. 4. Visualization of synthetic datasets \mathcal{D}_S (left), \mathcal{D}_I (center), and \mathcal{D}_R (right).

Table 2. Dataset characteristics and references.

Dataset	Total Observations	# Train	# Test	Features	Reference
Sinusoidal (\mathcal{D}_S)	150	120	30	1	Adapted from [9]
Izmailov (\mathcal{D}_i)	400	320	80	1	Adapted from [24]
Regression2d (\mathcal{D}_R)	256	304	52	2	–
Airfoil	1503	1202	301	5	[11]
Concrete	1030	824	206	8	[47]
Diabetes	442	353	89	10	[11]
Energy	768	614	154	8	[44]
Forest Fire	517	413	104	12	[8]
Yacht	308	246	62	6	[11]

C.2 Markov Chain Monte Carlo

MCMC sampling of the MLP parameters is performed using the NumPyro implementation of the No U-Turn Sampler (NUTS) with default settings. Specifically, we only provide our models and data to NUTS, set the step size to 1.0 and allow for adaptation during the warm-up phase. We use a diagonal inverse mass matrix which is initialized with the identity matrix and is allowed to adapt during the warm-up phase. The target acceptance probability is set to 0.8.

C.3 Point Estimates

For the LA and DE estimates, both the smaller architecture f_1 (one hidden layer of three neurons) and the larger architecture f_2 (three hidden layers of 16 neurons each) are trained with an RMSProp optimizer [21] for 500 and 1000 epochs, respectively, using a constant learning rate of 10^{-4} and no weight decay. In the case of DE, we initialize each weight vector with a different random

seed and do not use data bootstrapping, following the approach described in the original work of [28]. LA samples are drawn directly from the parameter posterior density. Due to the small dataset sizes, we perform full-batch training. Our loss function is set to be the negative log-posterior

$$\mathcal{L}(\boldsymbol{\theta}, \sigma) = -\log p(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{2\sigma^2} \sum_{i=1}^N (f_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{y}_i)^2 + N \cdot \log \sigma + \frac{1}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta}. \quad (25)$$

The loss function is to be minimized over the network parameters $\boldsymbol{\theta}$ and nuisance parameter σ . The code for LA and DE estimates is mainly based on the PyTorch [35] and PyTorch Lightning [14] libraries, as well as the Laplace library by [9].

D Methods

D.1 KL-Divergence for the Posterior Predictive Density

The KL-divergence for two densities p, q of continuous random variables is defined as

$$D_{KL}(p||q) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx. \quad (26)$$

The analogous formulation for random variables with discrete probability densities p, q is

$$D_{KL}(p||q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}. \quad (27)$$

In Section 5.2 the KL-divergence of consecutive posterior predictive densities is calculated on a regular grid of values $y \in [-3.0, 3.0]$ for a particular input x . Further, this KL-divergence is averaged over a regular grid of input values $x \in [-3.0, 3.0]$ to obtain the final value.

D.2 Spectral Clustering

The spectral clustering in Section 5.3 is performed by first constructing a 4-NN graph using the Gaussian similarity function $s(a, b) = \exp(-\|a - b\|^2 \cdot (2\sigma^2)^{-1})$, with $\sigma = 1$ as a distance measure, encoded via the adjacency matrix \mathbf{A} . We then compute the normalized graph Laplacian \mathbf{L}_{norm} from \mathbf{A} as

$$\mathbf{D}_{ij} = \begin{cases} \sum_{k=1}^N \mathbf{A}_{ik} & \text{if } i = j \\ 0 & \text{else} \end{cases}, \quad (28)$$

$$\mathbf{L}_{norm} = \mathbf{D}^{-1/2} \mathbf{L} \mathbf{D}^{1/2}, \quad (29)$$

where \mathbf{D}_{ij} is the (i, j) element of the degree matrix \mathbf{D} of the graph. Subsequently, the eigenvalue spectrum of \mathbf{L}_{norm} is determined and K-means clustering is performed for $K = 3$.

E Additional Results

E.1 Performance Comparison Between our Approach and LA

We compare our approach of calculating the PPD based on MCMC sampling to LA in terms of predictive performance. For our MCMC sampling, we generate 1274 chains, which is the derived upper bound for the number of chains. Table 3 summarizes this comparison.

Table 3. Mean log pointwise predictive density (LPPD) values on test sets (larger is better; one standard error in parentheses). The highest performance per dataset and network is highlighted in bold.

	Smaller network f_1			Larger network f_2		
	MCMC (ours)	MCMC (s.c.)	LA	MCMC (Ours)	MCMC (s. c.)	LA
\mathcal{D}_S	-0.53 (± 0.09)	-0.56 (± 0.11)	-0.57 (± 0.10)	-0.59 (± 0.12)	-0.59 (± 0.12)	-2.42 (± 0.01)
\mathcal{D}_I	0.79 (± 0.06)	0.65 (± 0.07)	0.53 (± 0.07)	0.91 (± 0.09)	0.91 (± 0.09)	-1.81 (± 0.01)
\mathcal{D}_R	0.64 (± 0.10)	0.75 (± 0.11)	-27.39 (± 3.65)	0.95 (± 0.08)	0.95 (± 0.08)	-2.33 (± 0.00)
Airfoil	-0.74 (± 0.04)	-0.80 (± 0.05)	-1.78 (± 0.13)	0.92 (± 0.05)	0.72 (± 0.10)	-3.57 (± 0.18)
Concrete	-0.41 (± 0.05)	-0.44 (± 0.06)	-14.49 (± 1.02)	0.26 (± 0.07)	0.25 (± 0.07)	-4.36 (± 0.47)
Diabetes	-1.20 (± 0.07)	-1.20 (± 0.07)	-1.46 (± 0.09)	-1.18 (± 0.08)	-1.22 (± 0.09)	-2.61 (± 0.00)
Energy	0.92 (± 0.04)	0.69 (± 0.12)	-31.74 (± 1.88)	2.07 (± 0.46)	2.38 (± 0.11)	-1.39 (± 0.06)
ForestF	-1.37 (± 0.07)	-1.37 (± 0.07)	-2.39 (± 0.16)	-1.43 (± 0.45)	-1.69 (± 0.49)	-2.80 (± 0.00)
Yacht	1.90 (± 0.16)	1.29 (± 0.56)	-5.60 (± 1.51)	3.31 (± 0.21)	0.15 (± 0.09)	-2.69 (± 0.00)

E.2 Interpretability Example Approximated

The three remaining modes in the BNN’s posterior of the interpretability example in Section 5.3 can be analytically approximated using a mixture of LA (MoLA; [13]) upon clustering. Figure 5 depicts the approximated function space component-wise and as a mixture.

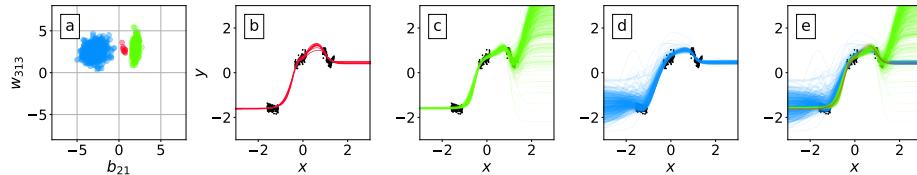


Fig. 5. From left to right: The three remaining modes in the BNN’s posterior after clustering approximated by MoLA, visualized in the bivariate marginal space of two weights (a); resulting functionally diverse network parameter states based on the three Gaussian components (b-d); the resulting function space as a composition of the samples of the three Gaussian distributions (e) by combining b-d.

E.3 Profile of Parameter Posterior Density

Figure 6 visualizes pairwise profiles of the parameter posterior density of BNN f_1 on dataset \mathcal{D}_S as investigated in Section 5.3 before (red) and after (green) the application of the symmetry removal algorithm. The resulting approximate parameter posterior density is unimodal and much simpler in comparison to the original parameter posterior density, yet, functionally, they are identical.

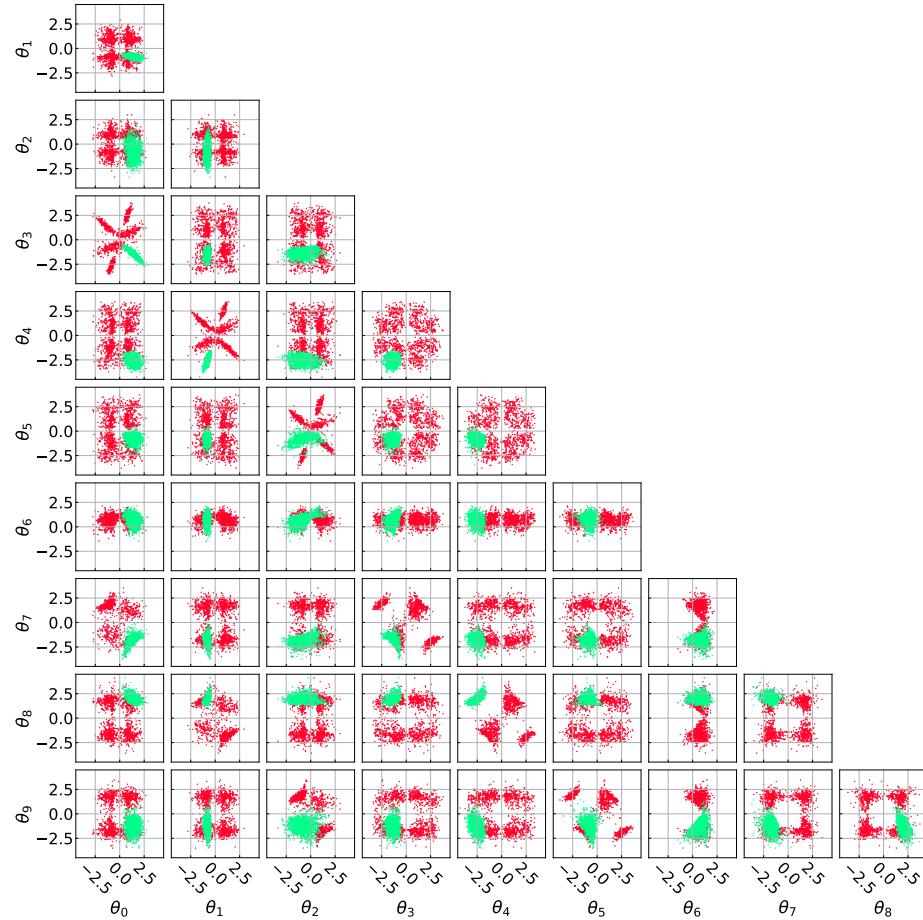


Fig. 6. Visualization of pairwise profiles of the parameter posterior density of BNN f_1 . The parameter posterior density sampled via MCMC (red) exhibits symmetries, while the transformed approximate parameter posterior density (green) is unimodal.