TOWARDS EFFICIENT POSTERIOR SAMPLING IN DEEP NEURAL NETWORKS VIA SYMMETRY REMOVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian inference in deep neural networks is challenging due to the high-dimensional, strongly multi-modal posterior landscape. Markov Chain Monte Carlo approaches asymptotically recover the true, intractable posterior but are prohibitively expensive for large modern architectures. Local posterior approximations, while often yielding satisfactory results in practice, crudely disregard the posterior geometry, i.e., other functionally relevant modes. We propose to exploit well-known parameter symmetries induced by neuron interchangeability and output activation to retrieve a drastically reduced – yet exact – posterior over uniquely identified parametrizations. To this end, we provide an algorithm for explicit symmetry removal and develop an upper bound on the number of Monte Carlo chains required to capture the reduced posterior. Our experiments suggest that efficient sampling from the functionally relevant part of the posterior is indeed possible, opening up a promising path to faithful uncertainty quantification in deep learning.

1 INTRODUCTION

Uncertainty quantification (UQ) is crucial to typical deep learning applications in which model parametrizations are inferred from limited data. By equipping predictions with meaningful confidence estimates, UQ enables higher predictive accuracy, provides credible regions, and allows for the detection of out-of-distribution (OOD) data Ovadia et al. (2019). A key component of UQ in neural networks is the distribution over feasible network parametrizations, represented by the posterior distribution in Bayesian statistics (Hüllermeier & Waegeman, 2021). However, an analytical form of the posterior distribution is generally not tractable for Bayesian neural networks (BNNs). Experimental inspections of the posterior distribution have revealed complex geometries (Izmailov et al., 2021). The highly non-monotonic hypersurface in the weight space proves a challenging approximation target. Furthermore, the classical approach of Markov Chain Monte Carlo (MCMC) approximation is considered impractical due to the typically large number of posterior modes, preventing a reasonable mixing of chains (Izmailov et al., 2021). Hence, instead of capturing the entire distribution, recent approaches have tried to infer uncertainty from smaller regions of the parameter space. The Laplace approximation (LA) captures a single mode of the posterior distribution (MacKay, 1992; Daxberger et al., 2021), while Izmailov et al. (2020) propose to use a parameter subspace for quantifying uncertainty. Even an independence assumption for all posterior dimensions in mean-field variational inference approaches can work well in practice (Farquhar et al., 2020). Similarly, deep ensembles (DE; Lakshminarayanan et al., 2017) rely on a handful of trained networks to characterize the uncertainty induced by the entire posterior landscape. However, in contrast to MCMC, these methods *a priori* omit regions of the parameter space that might be decisive for a faithful quantification of uncertainty (as also shown in Section 6.4).

A rarely considered property of neural networks in the context of UQ is their unidentifiability, i.e., the existence of two or more equivalent parametrizations that describe the same input-output mapping. Sources in multi-layer perceptrons (MLPs) for equioutput parametrizations are activation functions, such as tanh, sigmoid or ReLU (Kůrková & Kainen, 1994; Chen et al., 1993; Petzka et al., 2020), and the free permutability of neuron parameters in hidden layers (Hecht-Nielsen, 1990). The functional redundancy arising from this phenomenon grows rapidly with the depth and width of a network (Chen et al., 1993), and thus also contributes to the high geometric complexity and symmetries of BNN posterior distributions.

Our contributions In this work, we analyze the role of posterior space redundancies in quantifying BNN uncertainty, making the following contributions: 1) We show that the full BNN's posterior predictive distribution can be obtained from a small fraction of its parameter space containing uniquely identified parametrizations in function space. 2) We propose an estimation procedure for the number of Monte Carlo chains – independent of network size – required to discover all functionally diverse modes, i.e., posterior modes that remain after complete symmetry removal. 3) We relate our findings to the state-of-the-art DE and LA methods and provide an explanation for their high-quality uncertainty but also point out their limitations. 4) We present a novel algorithm to remove equioutput-transformation-related symmetries from the posterior distribution of tanh-activated BNNs, demonstrating the superior interpretability and better approximation quality of our general proposal.

2 BACKGROUND AND NOTATION

In this work, we consider neural networks of the following form. Let $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}; \boldsymbol{x} \mapsto f_{\boldsymbol{\theta}}(\boldsymbol{x}) =: \hat{\boldsymbol{y}}$ be an MLP with K layers, where layer $l \in \{1, \dots, K\}$ consists of M_l neurons, mapping a feature vector $\boldsymbol{x} = (x_1, \dots, x_n)^\top \in \mathcal{X} \subseteq \mathbb{R}^n$ to an outcome (vector) $\hat{\boldsymbol{y}} = (\hat{y}_1, \dots, \hat{y}_m)^\top \in \mathcal{Y} \subseteq \mathbb{R}^m$, $m, n \in \mathbb{N}$, to estimate $\boldsymbol{y} = (y_1, \dots, y_m)^\top \in \mathcal{Y}$. For each hidden layer $l \in \{2, \dots, K-1\}$, the inputs are linearly transformed and then activated by a function a. Throughout this paper, we will use the tanh function $a(u) = \tanh(u)$ as an example for hidden layers and an identity activation $a(u) = u$ in the last (K-th) layer. We further define the activated outputs of the i-th neuron in a hidden layer as $z_{li} = a(o_{li})$ with neuron output $o_{li} = \sum_{j=1}^{M_{l-1}} w_{lij} z_{(l-1)j} + b_{li}$, layer weights w_{lij}, and bias term b_{li}. For the input layer, we have $z_{1i} = x_i$ for $i = 1, \dots, n$, and for the output layer $z_{Ki} = \hat{y}_i$ for $i = 1, \dots, M_K$. We summarize all weights of the MLP in the vector

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

Bayesian neural networks In the Bayesian paradigm, a prior distribution $p(\boldsymbol{\theta})$ is imposed on the parameters, typically as part of a Bayesian model of the data. Using Bayes' rule, the posterior distribution of parameters $p(\boldsymbol{\theta}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})/p(\mathcal{D})$ updates this prior information based on the information given by the data $\mathcal{D}$ and encoded in the likelihood $p(\mathcal{D}|\boldsymbol{\theta})$.

Equioutput transformations All MLPs with more than one neuron in at least one hidden layer potentially exhibit equioutput parametrizations (Hecht-Nielsen, 1990; Kůrková & Kainen, 1994) that can be converted into each other using suitable transformations. This means that 1) the $M_l > 1$ neurons of a hidden layer l can be freely interchanged by permuting their parameters, and 2) the signs of parameters can be flipped for odd activation functions (satisfying $a(-u) = -a(u)$) without changing the input-output mapping. In the second case, each neuron i has two parametrizations that are related by a reflection transformation flipping the signs of corresponding parameters. A common example for this is the tanh function. An instance of functional equality related to the permutation of two neurons and sign flipping is shown in Appendix A. Sussmann (1992) mentions another class of equioutput cases in MLPs arising from specific parameter values, such as zero-valued multiplicative weights allowing for arbitrary parametrizations in the other corresponding factor. As these are typically degenerated cases, we will not consider them further.

A formal description of equioutput transformations as given in, e.g., Chen et al. (1993), is provided in the following for the example of a tanh-activated MLP. Let $\mathcal{F}_{\boldsymbol{T}} : \Theta \to \Theta, \boldsymbol{\theta} \mapsto \boldsymbol{T}\boldsymbol{\theta}$, $\boldsymbol{T} \in \{-1, 0, 1\}^{d \times d}$ be an output-preserving reflection transformation of a parameter vector that encodes activation-related equioutput parametrizations. Similarly, let $\mathcal{F}_{\boldsymbol{P}} : \Theta \to \Theta, \boldsymbol{\theta} \mapsto \boldsymbol{P}\boldsymbol{\theta}$, $\boldsymbol{P} \in \{0, 1\}^{d \times d}$ be a permutation of parameter vector components such that the output remains unchanged. For a neural network with a d-dimensional parameter vector, the cardinality of the set of reflection transformation matrices $\mathcal{T}_l = \{\boldsymbol{T}_1, \dots, \boldsymbol{T}_{2^{M_l}}\}$ in layer l is 2^{M_l}, and the cardinality of the permutation set $\mathcal{P}_l = \{\boldsymbol{P}_1, \dots, \boldsymbol{P}_{M_l!}\}$ is $M_l!$. Layerwise equioutput transformations $\mathcal{E}_l$ are arbitrary combinations of elements from $\mathcal{T}_l$ and $\mathcal{P}_l$, such that $\mathcal{E}_l = \{\boldsymbol{T}\boldsymbol{P} \mid \boldsymbol{T} \in \mathcal{T}_l, \boldsymbol{P} \in \mathcal{P}_l\}$ with corresponding functional $\mathcal{F}_{\mathcal{E}_l} = \mathcal{F}_{\boldsymbol{T}} \circ \mathcal{F}_{\boldsymbol{P}}$. A full equioutput transformation of the MLP can be obtained by combining layerwise transformations

$$\mathcal{E} = \left\{ \prod_{l=2}^{K-1} \boldsymbol{A}_l \mid \boldsymbol{A}_l \in \mathcal{E}_l, \text{ for } l = 1, \dots, K-1 \right\}, \tag{1}$$

with $f_{\boldsymbol{\theta}}(\cdot) = f_{\boldsymbol{E}\boldsymbol{\theta}}(\cdot)$ for $\boldsymbol{E} \in \mathcal{E}$. The cardinality of this set, i.e., the amount of equioutput parametrizations, can be computed as

$$|\mathcal{E}| = \prod_{l=2}^{K-1} M_l! \cdot 2^{M_l}. \tag{2}$$

It becomes immediately clear from Equation 2 that the amount of functional redundancy increases rapidly with network size (see also Figure 1). As a result of equioutput parametrizations, the BNN posterior distribution, too, exhibits functional redundancy in the form of symmetries.

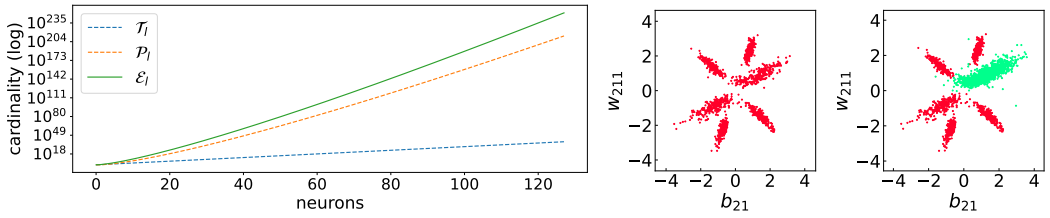

Figure 1: *Left*: cardinality of the equioutput transformation set (y-axis; on log-scale) for a single hidden layer with 1 to 128 neurons (the functional redundancy factor for a network with 128 neurons is at $1.31 \cdot 10^{254}$); *center*: induced symmetries in a ten-dimensional BNN posterior (bivariate marginal distribution); *right*: distribution of the functionally equivalent non-symmetric set of parametrizations (highlighted in green) obtained by our proposed method.

3 Related work

Equioutput parametrizations Non-unique network parametrizations have been considered in the literature before, first mentioned by Hecht-Nielsen (1990) with a focus on neuron interchangeability, and advanced by subsequent work on single-hidden-layer MLPs with tanh and more general self-affine activations (Sussmann, 1992; Kůrková & Kainen, 1994). An extension of MLPs with tanh activation to arbitrary depth was studied by Chen et al. (1993). More recently, Petzka et al. (2020) characterized equioutput parametrizations for MLPs with ReLU activation and a single hidden layer.

Functional redundancy Chen et al. (1993) provided a set of identifiability constraints alongside their characterization of equioutput transformations for tanh-activated MLPs that can be used to obtain an identifiable set of networks. However, their data-independent constraints do not respect the geometry of the posterior distribution and might intersect modes, such that removable symmetries between them remain. This phenomenon has been described in the context of finite mixture models, which possess comparable symmetries in their posterior distributions, under the term *label switching* (Frühwirth-Schnatter, 2001). Bardenet & Kégl (2012) introduced an adaptive Metropolis algorithm with online relabeling for mixture models, effectively removing permutation symmetries by optimizing over the discrete sets of permutation matrices. Unfortunately, such discrete optimization is not feasible in modern neural networks due to the vast amount of existing permutations.

Symmetries in neural networks A large body of work in deep learning attests to the various symmetries and modes in BNN posteriors. Draxler et al. (2018); Garipov et al. (2018) found modes in BNN posteriors to be linked by paths of near-constant loss (mode connectivity) and proposed methods to uncover these as more robust solutions. Recent work has also attempted to interpolate between pairs of close-by network parametrizations on lower-loss curves by removing symmetries in ReLU-activated networks: Tatro et al. (2020) permute parameters based on aligning layer-wise network embeddings, and Pittorino et al. (2022) maximize the cosine-similarity between normalized parameter vectors *post hoc*. Pourzanjani et al. (2017) propose an online-sampling algorithm for ReLU-activated MLPs by introducing constraints for symmetry removal, albeit ignoring posterior geometry and hence failing to remove all symmetries from the posterior distribution.

MCMC methods Posterior symmetries have been known to slow down convergence due to the possibility of visiting symmetric modes (Nalisnick, 2018; Papamarkou et al., 2022). Izmailov et al.

(2021) indeed found differences of MCMC chain-mixing between the parameter and function space in large-scale experiments, while imposing constraints to the sampling process improves mixing (Sen et al., 2020). We propose to address the problem by mapping every sample to a reference set of unique parameterizations. As a consequence of this approach, it is sufficient to only consider samples from this fraction of the posterior (see Section 4). These findings advocate focusing on functional instead of parameter space mixing during MCMC.

4 BAYESIAN INFERENCE USING A POSTERIOR REFERENCE SET

In supervised learning, a set of N feature vectors $\boldsymbol{x} \in \mathcal{X}$ and outcome vectors $\boldsymbol{y} \in \mathcal{Y}$ form the dataset $\mathcal{D} = \{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), \dots, (\boldsymbol{x}^{(N)}, \boldsymbol{y}^{(N)})\}$. The posterior predictive distribution $p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D})$ quantifies the predictive or functional uncertainty of a model for a new observation $(\boldsymbol{x}^*, \boldsymbol{y}^*) \in \mathcal{X} \times \mathcal{Y}$. Since $p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int_\Theta p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta}$, deriving this uncertainty requires access to the posterior $p(\boldsymbol{\theta}|\mathcal{D})$. As mentioned in Section 1, the posterior itself is generally not tractable for BNNs, and an MCMC approximation is considered to be practically infeasible for highly multi-modal posteriors. Motivated by the large number of symmetries from equioutput parametrizations, we revisit this infeasibility statement in the following. The presented results pave the way for a viable approach to sample from the true (non-approximate) posterior.

4.1 POSTERIOR REFERENCE SET

As stated by Chen et al. (1993), the neural network parameter space Θ can (with the exception of points at the boundaries) be dissected into $|\mathcal{E}|$ disjunctive sets of the same cardinality.

Proposition 1 (Parameter Space Dissection) *Similar to Chen et al. (1993), let us define $\mathcal{S}_1$ as the reference set of uniquely identified network parametrizations. Then, it holds that*

$$\Theta = \bigcup_{j=1}^{|\mathcal{E}|} \mathcal{S}_j, \text{ where } \mathcal{S}_j = \{\boldsymbol{\theta} \mid \boldsymbol{\theta} = \boldsymbol{E}_j \boldsymbol{\theta}' \quad \forall \boldsymbol{\theta}' \in \mathcal{S}_1, \boldsymbol{E}_j \in \mathcal{E}\}. \tag{3}$$

Since equioutput parametrizations have the same posterior probabilities $p(\boldsymbol{\theta}|\mathcal{D}) = p(\boldsymbol{E}_j\boldsymbol{\theta}|\mathcal{D})$ and, by definition, produce the same predictions $p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) = p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{E}_j\boldsymbol{\theta})$ for any $\boldsymbol{E}_j \in \mathcal{E}$ (see Appendix B, Equations 14-16), the following holds.

Corollary 1 (Reformulated Posterior Predictive Distribution) *Given a set of equioutput transformations $\mathcal{E}$ and a reference set $\mathcal{S}_1$, the posterior predictive distribution for a symmetric and unconstrained prior distribution can be reformulated as follows:*

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int_\Theta p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} = |\mathcal{E}| \cdot \int_{\mathcal{S}_1} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} \tag{4}$$

The proof of Corollary 1 is given in Appendix B and follows from Proposition 1 and the assumption of symmetric prior distributions, which is often satisfied in practice (e.g., for widely applied Gaussian priors). An alternative and more formal definition of this reference set can be found by using equivalence classes, where the elements of $\mathcal{S}_1$ are the representatives of each equivalence class. As a consequence of Corollary 1, the posterior predictive distribution can be obtained by only integrating over the set of uniquely identified parametrizations $\mathcal{S}_1$, up to 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 the set of uniquely identified parametrizations of the neural network and thus to obtain the full posterior predictive distribution. This simplifies inference drastically, as illustrated in Figure 1. For example, it allows the posterior space of a single-layer network with 128 neurons to be effectively reduced to a 10^{-254}-th of its original size.

How to sample from the reference set? In practice, when using Monte Carlo to approximate equation 4, it is not necessary to actually constrain the sampling procedure to $\mathcal{S}_1$, which might indeed not be straightforward. Since any equioutput transformation is known *a priori*, each sample can simply be mapped to its counterpart in the reference set after running the sampling procedure. In Section 5, we propose an algorithm to conduct this mapping (i.e., symmetry removal) in an automated manner.

4.2 AN UPPER BOUND FOR MARKOV CHAINS

The previous section showed how samples from the true posterior can be used efficiently and indicates that a substantially reduced amount of samples is required for the posterior approximation when accounting for all symmetries. The question remains how many samples are needed to approximate the set of uniquely identified parametrizations sufficiently well. Even after reducing the posterior space and removing redundant modes, BNN posteriors can exhibit multiple functionally diverse modes, representing structurally different hypotheses that cannot be distinguished in the light of limited data. The existence of such a set of plausible solutions depends on the network architecture and the underlying data-generating process. For example, in Section 6.4, we discuss the case of an underparametrized network that preserves three distinctive modes caused by its restricted capacity. In the following, we assume ν functionally diverse modes with the goal to visit every mode and its local proximity at least once when running MCMC. As the ability to switch from one mode to another within one 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 instead of 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 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 $\pi_1, \dots, \pi_\nu$ be the respective probabilities of the ν functionally diverse modes to be visited by an independently started Markov chain. Then, given ρ chains,*

$$\mathbb{P}(\mathcal{G} < \rho) \geq 1 - \frac{\sum_{q=0}^{\nu-1} (-1)^{\nu-1-q} \sum_{J:|J|=q} (1 - \Pi_J)^{-1}}{\rho} \quad \textit{with} \quad \Pi_J = \textstyle\sum_{j \in J} \pi_j. \tag{5}$$

The proof can be found in Appendix C. Note that this bound is independent of the neural network architecture and only depends on the assumptions about the number of functionally diverse modes ν, disregarding symmetric copies. In our experiments in Section 6, we find ν to be rather small (< 10).

Example Assume $\nu = 3$ functionally diverse modes in the set of uniquely identified parametrizations with probabilities $\pi_1 = 0.57, \pi_2 = 0.35, \pi_3 = 0.08$ (chosen to represent a rather diverse functional mode set). We then find ρ such that $\mathbb{P}(\mathcal{G} < \rho) \geq 0.99$. This results in an upper bound of $\rho = 1274$ chains to observe all functionally diverse modes, an amount typically feasible in practice.

5 POSTERIOR SYMMETRY REMOVAL

The previous section used the knowledge of equioutput parametrizations to derive an upper bound for the number of Markov chains to observe all functionally diverse modes of the network posterior distribution. However, the obtained samples will be scattered across the sets $\mathcal{S}_j$ (see Section 4 and Equation 3). This obscures any insights into the geometry of the network's posterior distribution and makes it infeasible to interpret anything but the obtained functional uncertainty. Ideally, all samples should therefore be mapped to the reference set. In the following, we derive a general methodology to achieve this symmetry removal in MLP posteriors, uncover their unique representative, and present an algorithm to implement this method in practice.

Reasoning Based on results of the previous section, assume G available posterior samples $\boldsymbol{\theta}^{(g)}, g \in \{1, \dots, G\}$. As equioutput transformations in MLPs factorize layerwise (see Section 2 and Equation 1), 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 is defined by $\boldsymbol{\phi}^{(g,l,i)} = (w_{li1}^{(g)}, \dots, w_{liM_{(l-1)}}^{(g)}, w_{(l+1)1i}^{(g)}, \dots, w_{(l+1)M_{(l+1)}i}^{(g)}, b_{li}^{(g)})^\top \in \mathbb{R}^{(M_{(l-1)}+M_{(l+1)}+1)}$, 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 2), the marginal posteriors of freely permutable neurons from the same hidden layer are identical. This implies that the marginal distribution of every $\boldsymbol{\phi}^{(g,l,i)}$ 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 parametrizations. 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).

While the previous considerations theoretically allow for the mapping of every sample to its reference set, the approach is practically infeasible due to the large number of transformation functions that would need to be defined. In the following, we present an exemplary algorithm for automatic symmetry removal in tanh-MLPs without the need to explicitly specify $\mathcal{E}$.

A symmetry removal algorithm for tanh-MLPs We follow the principle from Frühwirth-Schnatter (2001) to work in a selected parameter subspace of equioutput-transformation-related parameters, which is beneficial for finding identifiability constraints. Let $\mathcal{M}^{(g,l)} := \{\boldsymbol{\phi}^{(g,l,1)}, \dots, \boldsymbol{\phi}^{(g,l,M_l)}\}$ 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 distribution poses a data-dependent optimization problem of finding the right constraints. Here, we propose a zero-centered, linear hyperplane $\boldsymbol{\beta}_l \in \mathbb{R}^{M_{(l-1)}+M_{(l+1)}+1}$ that intersects as few states or clusters of neuron parameter vectors $\boldsymbol{\phi} \in \mathcal{M}^{(*,l)}$ as possible. We optimize this hyperplane using a variant of a support vector machine (SVM; Cortes & Vapnik, 1995) for unlabeled data, such that it has 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 $\boldsymbol{\beta}_l^\top \boldsymbol{\phi}$ for $\boldsymbol{\phi} \in \mathcal{M}^{(*,l)}$:

$$\mathcal{L}(\boldsymbol{\beta}_l) = \tfrac{1}{2}\boldsymbol{\beta}_l^\top \boldsymbol{\beta}_l + C \cdot \textstyle\sum_{\boldsymbol{\phi} \in \mathcal{M}^{(*,l)}} \max\left(0, 1 - \left|\boldsymbol{\beta}_l^\top \boldsymbol{\phi}\right|\right), \tag{6}$$

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 for flipping parameter vectors to one side of the hyperplane, i.e., applying the corresponding equioutput transformation. The full algorithm is described in Appendix D (Algorithm 1).

Following the removal of tanh-related symmetries using the SVM approach, the remaining parameter subspace in the marginal posterior distribution of layer l must be divided among the hidden-layer neurons in order to remove the permutation-related symmetries. For this, we consider one sample $\boldsymbol{\theta}^{(g)}$ with elements $\boldsymbol{\phi}^{(g,l,i)} \in \mathcal{M}^{(g,l)}$ 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 (Appendix D, Algorithm 2) on the elements of each set $\mathcal{M}^{(g,l)}$ to assign each neuron parameter vector $\boldsymbol{\phi}^{(g,l,i)}$ to its most likely neuron position in the hidden layer, followed by a permutation of the neuron parameter vectors of the layer according to the classification results (i.e., neurons are re-indexed according to their class assignment; Appendix D, Algorithm 3). 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 $(\boldsymbol{\phi}^{(g,l,i)}, \boldsymbol{\phi}^{(g,l,j)})$ 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. The complete algorithm is given in Algorithm 4 in Appendix D, where we also describe the choice of all hyperparameters.

6 EXPERIMENTS

We now investigate our theoretical findings and compare the resulting approach to other popular uncertainty methods. In all experiments, we employ a Bayesian regression model with Gaussian likelihood assumption, standard Gaussian prior for parameters $\boldsymbol{\theta}$ and a standard half-normal prior for the variance of the Gaussian likelihood, which we treat as a nuisance parameter. Depending on the task, we either use a No-U-Turn sampler (Hoffman & Gelman, 2014) 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 Appendix E.2, E.3).

6.1 PERFORMANCE COMPARISON

Table 1: Mean log pointwise predictive density (LPPD) values on test sets (larger is better; one standard deviation across samples in parentheses) for the small network (left) and large network (right). The best method per dataset and network is highlighted in bold.

	Small network f_1			**Large network** f_2		
	MCMC (Ours)	LA	DE	MCMC (Ours)	LA	DE
Sinusoidal ($\mathcal{D}_S$)	**-0.53** ($\pm$ 0.50)	-0.57 ($\pm$ 0.63)	-0.58 ($\pm$ 0.63)	**-0.59** ($\pm$ 0.63)	-2.42 ($\pm$ 0.04)	-2.13 ($\pm$ 0.19)
Izmailov ($\mathcal{D}_I$)	**0.79** ($\pm$ 0.50)	0.53 ($\pm$ 0.67)	0.56 ($\pm$ 0.57)	**0.91** ($\pm$ 0.78)	-1.81 ($\pm$ 0.09)	-2.02 ($\pm$ 0.16)
Regression2d ($\mathcal{D}_R$)	**0.64** ($\pm$ 0.72)	-27.4 ($\pm$ 26.3)	-1.46 ($\pm$ 0.47)	**0.95** ($\pm$ 0.57)	-2.33 ($\pm$ 0.03)	-2.20 ($\pm$ 0.17)
Airfoil	**-0.74** ($\pm$ 0.71)	-1.78 ($\pm$ 2.25)	-1.62 ($\pm$ 0.44)	**0.92** ($\pm$ 0.92)	-3.57 ($\pm$ 3.16)	-2.17 ($\pm$ 0.26)
Concrete	**-0.41** ($\pm$ 0.77)	-14.5 ($\pm$ 14.6)	-1.59 ($\pm$ 0.49)	**0.26** ($\pm$ 0.95)	-4.36 ($\pm$ 6.48)	-2.03 ($\pm$ 0.21)
Diabetes	**-1.20** ($\pm$ 0.69)	-1.46 ($\pm$ 0.84)	-1.47 ($\pm$ 0.63)	**-1.18** ($\pm$ 0.78)	-2.61 ($\pm$ 0.02)	-2.09 ($\pm$ 0.40)
Energy	**0.92** ($\pm$ 0.44)	-31.7 ($\pm$ 23.2)	-1.76 ($\pm$ 0.29)	**2.07** ($\pm$ 5.67)	-1.39 ($\pm$ 0.80)	-1.99 ($\pm$ 0.20)
Forest Fire	**-1.37** ($\pm$ 0.69)	-2.39 ($\pm$ 1.67)	-1.60 ($\pm$ 0.59)	**-1.43** ($\pm$ 4.54)	-2.80 ($\pm$ 0.05)	-2.20 ($\pm$ 0.23)
Wine	**9.67** ($\pm$ 1.34)	-24.5 ($\pm$ 25.2)	-1.15 ($\pm$ 0.62)	**8.62** ($\pm$ 0.64)	-2.81 ($\pm$ 0.02)	-2.08 ($\pm$ 0.26)
Yacht	**1.90** ($\pm$ 1.26)	-5.60 ($\pm$ 10.4)	-1.14 ($\pm$ 1.08)	**3.31** ($\pm$ 1.67)	-2.69 ($\pm$ 0.02)	-2.18 ($\pm$ 0.26)

In our first experiment, we demonstrate the performance of an MCMC-trained BNN that makes use of the derived upper bound for MCMC chains. We then compare the posterior approximation quality to the one of LA and DE, using ten ensemble members for the latter. We choose tanh-activated MLPs in this section to be consistent with our proposed symmetry removal algorithm but note that our bound is independent of the activation function. We use a small neural network f_1 with a single hidden layer containing three neurons and a larger network f_2 with three hidden layers having 16 neurons each. As in Section 4.2, we assume $\nu = 3$ and mode probabilities as in the given example. While this might seem restrictive, results confirm that this assumption yields enough functional flexibility. Table 1 shows the performance of our MCMC-based posterior approximation in comparison to LA and DE. To measure the goodness-of-fit of the resulting predictive posterior, we use the log point-wise predictive density (LPPD; Gelman et al., 2014)

$$\text{LPPD} = \log \int_{\Theta} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \approx \log \tfrac{1}{G} \textstyle\sum_{g=1}^{G} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}^{(g)})$$

averaged over N^* independent test data points $(\boldsymbol{x}^{*(1)}, \boldsymbol{y}^{*(1)}), \ldots, (\boldsymbol{x}^{*(N^*)}, \boldsymbol{y}^{*(N^*)})$. Using the LPPD, we can assess how well the predictive distribution defined by the entirety of estimated parameters is able to fit the test data. Our results clearly indicate that using only a reasonable amount of Markov chains yields better approximations of the posterior compared to LA and DE.

6.2 PRACTICAL EVALUATION OF COROLLARY 1

Next, we investigate the property derived in Corollary 1. We therefore analyze the posterior predictive distribution for the dataset $\mathcal{D}_I$ using the network f_2. For every newly collected sample in the MCMC run, the updated posterior predictive distribution is approximately computed on a two-dimensional (input/output) grid. Then the Kullback-Leibler (KL) divergence between consecutive distributions is calculated and averaged over the grid of input values of f_2 (details in Appendix F.1). As shown in Figure 2, despite the size of the network f_2 and the high amount of

equioutput parametrizations $|\mathcal{E}| = \left(16! \cdot 2^{16}\right)^3 \approx 2.58 \cdot 10^{54}$, the posterior predictive distribution converges within notably fewer samples compared to $|\mathcal{E}|$, and plots of the function space indicate the saturation of functional diversity already after 1274 samples from as many chains.

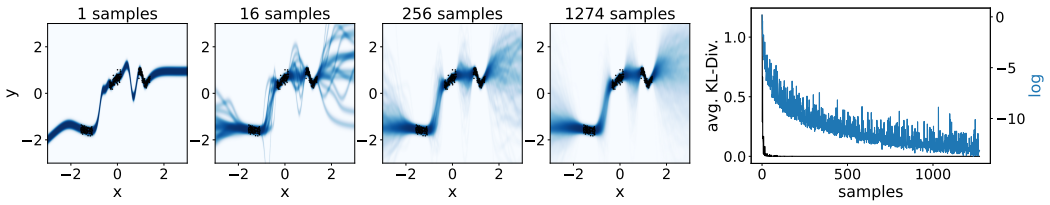

Figure 2: First four plots: approximated posterior predictive distribution of the network after 1, 16, 256, and 1274 samples; darker colors correspond to higher probabilities. Right plot: KL-divergence (black) and its value on the log-scale (blue) between consecutive distributions after adding another sample to the pool of samples.

6.3 Posterior symmetry removal

In order to test our symmetry removal approach, we apply the proposed Algorithm 4 to tanh-activated neural networks. We demonstrate the efficacy of the approach for two architectures of different sizes, f_1 having a single layer with 3 neurons and f_3 having a single layer with 16 neurons, yielding 48 and $1.37 \cdot 10^{18}$ equioutput parametrizations, respectively. We use two exemplary synthetic datasets $\mathcal{D}_S$ and $\mathcal{D}_I$, described in the Appendix E.1, and fit these with f_1 and f_3, respectively. The resulting neuron parameter subspace of our approach is visualized in Figure 3 for both experiments, including the steps of the symmetry removal algorithm. Different colors encode the current neuron index in the hidden layer of the respective neuron parameter vector. In experiment A (Figure 3, top) initially, the neuron parameter vectors are distributed identically and symmetries of the posterior distributions are clearly noticeable (Figure 3a). Upon the first step of the algorithm (Figure 3b), the parameter space is effectively halved, and three clusters remain. The algorithm's second (clustering) step completely removes all permutation symmetries from the full posterior distribution 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 Figure 3d and e, the univariate marginal distribution of each neuron's inner weight w_{2i1} reveals their reassignment to the parameter space. After running our algorithm, each neuron is now assigned to a distinct unimodal distribution, resulting in a unimodal distribution in the full parameter space (visualization in Appendix G.1).

In the case of experiment B (bottom row of Figure 3), the initial parameter subspace of hidden neurons reveals less distinct states, instead neuron parametrizations coincide at the center. However, the symmetry removal algorithm manages to assign a coherent region of the parameter space to each neuron, such that they are unimodal. Due to the discovered unimodal distribution of the uniquely identified set of parametrizations in these two examples, an LA could result in reasonable UQ here as well, even though the unreduced BNN posterior contains many more (symmetric) modes. Similarly, a deep ensemble consisting of only one model would be sufficient for a comparable predictive performance in this case.

6.4 Interpretability

Lastly, we demonstrate the benefit of removing symmetries from the BNN posterior distribution for improved interpretation and approximation purposes. Here we impose the same assumption on the functionally diverse modes of the posterior of f_1 as before and apply the proposed symmetry removal algorithm after running MCMC. After convergence of our algorithm, three modes remain in the BNN posterior and the transformed samples are clustered using a spectral clustering approach (details in Appendix F.2). Figure 4 visualizes network parametrizations in the function space, showing three functionally diverse hypotheses the network can potentially learn from the given dataset. Having removed all functionally redundant modes, our approach allows for a better-suited approximation of the uniquely identified reference set by, e.g., using a mixture of Laplace approximations

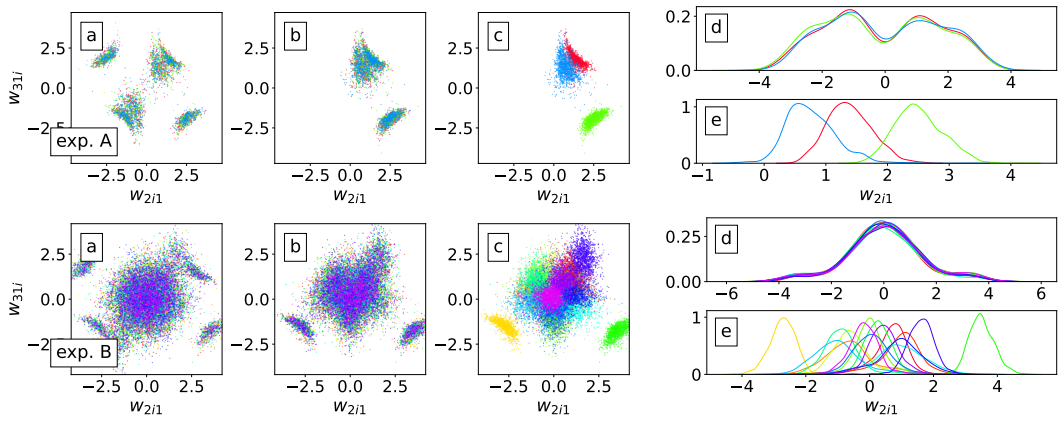

Figure 3: Visualization of the initial parameter subspaces of hidden neurons (a) and their transformation and reassignment during the steps of the algorithm (b, c) for experiment A using the network f_1 on $\mathcal{D}_S$ (top row) and experiment B using f_3 on $\mathcal{D}_I$ (bottom row). Plots on the right: Univariate marginal distributions of the neurons before (d) and after (e) the application of the algorithm.

(MoLA; Eschenhagen et al., 2021), as provided in Appendix G.2. With a functional redundancy of $|\mathcal{E}| = (3! \cdot 2^3) = 48$, we have a total of 144 modes in the unreduced posterior and can discard all but three modes for the approximation to observe the full function space. Note that standard LA would have captured only a third of the functional diversity, which highlights its limitation in providing high-quality uncertainty in cases where multiple modes exist in the set of uniquely identified parametrizations. In contrast, an ensemble approach such as DE will likely recover the different modes and thus be more suitable for the given application.

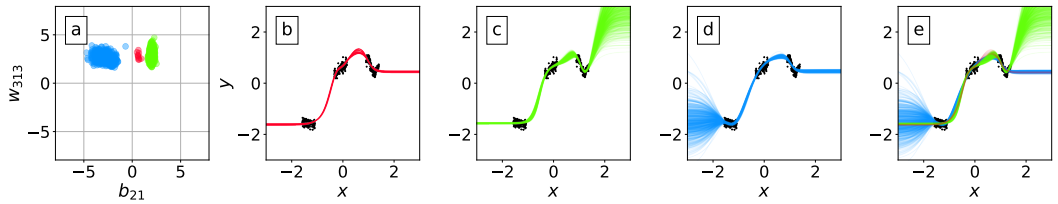

Figure 4: From left to right: The three remaining modes in the BNN's posterior after clustering, visualized in the bivariate marginal space of two weights (a); resulting functionally diverse network parametrizations based on the three parameter clusters (b - d); the observed function space as a composition of the three parameter clusters (e) by combining b - d.

7 Discussion

We proposed a principled approach toward Bayesian inference that makes explicit use of symmetries in the posterior distribution of neural networks. Specifically, we argue that removing such functional redundancies enables exact uncertainty quantification in a substantially reduced posterior space and thus improves over local approximations. Our considerations are based on the idea of an upper bound for the number of Monte Carlo chains required to sample all functionally diverse posterior modes that remain after the elimination of symmetries. We further developed an algorithm for symmetry removal in tanh-activated MLPs. With this work, we hope to help pave the way to exact inference (up to a Monte Carlo error) in deep learning. Ultimately, a better understanding of the posterior geometry is paramount to the reliability and interpretability of neural networks. In the future, we plan to address the current limitations of our symmetry removal algorithm, such as scalability to larger multi-layer architectures, and explore other architectures and potential sources of symmetry.

REFERENCES

Rémi Bardenet and Balázs Kégl. An adaptive Monte-Carlo Markov chain algorithm for inference from mixture signals. *Journal of Physics: Conference Series*, 368:012044, June 2012. Publisher: IOP Publishing.

An Mei Chen, Haw-minn Lu, and Robert Hecht-Nielsen. On the Geometry of Feedforward Neural Network Error Surfaces. *Neural Computation*, 5(6):910–927, November 1993.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

Paulo Cortez and Aníbal de Jesus Raimundo Morais. A data mining approach to predict forest fires using meteorological data, 2007.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace Redux – Effortless Bayesian Deep Learning, 2021. arXiv: 2106.14806.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially No Barriers in Neural Network Energy Landscape. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1309–1318. PMLR, 2018.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

Runa Eschenhagen, Erik Daxberger, Philipp Hennig, and Agustinus Kristiadi. Mixtures of Laplace Approximations for Improved Post-Hoc Uncertainty in Deep Learning, 2021. arXiv: 2111.03577.

William Falcon. PyTorch lightning. https://github.com/PyTorchLightning/pytorch-lightning, 2019.

Sebastian Farquhar, Lewis Smith, and Yarin Gal. Liberty or Depth: Deep Bayesian Neural Nets Do Not Need Complex Weight Posterior Approximations. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Philippe Flajolet, Danièle Gardy, and Loÿs Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 39(3):207–229, 1992.

Sylvia Frühwirth-Schnatter. Markov chain Monte Carlo Estimation of Classical and Dynamic Switching and Mixture Models. *Journal of the American Statistical Association*, 96(453):194–209, 2001.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

Andrew Gelman, Jessica Hwang, and Aki Vehtari. Understanding predictive information criteria for Bayesian models. *Statistics and Computing*, 24(6):997–1016, 2014.

Robert Hecht-Nielsen. On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers*, pp. 129–135. Elsevier, 1990.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. RMSProp: Divide the Gradient by a Running Average of Its Recent Magnitude (Lecture on Neural Networks for Machine Learning), 2012.

Matthew D. Hoffman and Andrew Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(47):1593–1623, 2014.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and Epistemic Uncertainty in Machine Learning: An Introduction to Concepts and Methods. *Machine Learning*, 2021.

Pavel Izmailov, Wesley J. Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace Inference for Bayesian Deep Learning. In *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, pp. 1169–1179. PMLR, 2020. ISSN: 2640-3498.

Pavel Izmailov, Sharad Vikram, Matthew D. Hoffman, and Andrew Gordon Wilson. What Are Bayesian Neural Network Posteriors Really Like? In *Proceedings of the 38th International Conference on Machine Learning, PMLR 139,*, 2021.

Věra Kůrková and Paul C. Kainen. Functionally Equivalent Feedforward Neural Networks. *Neural Computation*, 6(3):543–558, 1994.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

David J. C. MacKay. Bayesian Interpolation. *Neural Computation*, 4:415–447, 1992.

Eric Thomas Nalisnick. *On Priors for Bayesian Neural Networks*. PhD thesis, University of California, Irvine, 2018.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua V. Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019.

Theodore Papamarkou, Jacob Hinkle, M. Todd Young, and David Womble. Challenges in Markov Chain Monte Carlo for Bayesian Neural Networks. *Statistical Science*, 37(3), 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019),*, December 2019.

Henning Petzka, Martin Trimmel, and Cristian Sminchisescu. Notes on the Symmetries of 2-Layer ReLU-Networks. *Proceedings of the Northern Lights Deep Learning Workshop*, 1, 2020.

Fabrizio Pittorino, Antonio Ferraro, Gabriele Perugini, Christoph Feinauer, Carlo Baldassi, and Riccardo Zecchina. Deep Networks on Toroids: Removing Symmetries Reveals the Structure of Flat Regions in the Landscape Geometry, 2022.

Arya A Pourzanjani, Richard M Jiang, and Linda R Petzold. Improving the Identifiability of Neural Networks for Bayesian Inference. In *Second Workshop on Bayesian Deep Learning (NIPS 2017*, 2017.

Deborshee Sen, Theodore Papamarkou, and David Dunson. Bayesian neural networks and dimensionality reduction, 2020. arXiv: 2008.08044.

Héctor J. Sussmann. Uniqueness of the Weights for Minimal Feedforward Nets with a given input-output Map, 1992.

N. Joseph Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. Optimizing Mode Connectivity via Neuron Alignment. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Athanasios Tsanas and Angeliki Xifara. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and buildings*, 49:560–567, 2012.

I-C Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998.

A FUNCTIONAL EQUALITY OF EQUIOUTPUT PARAMETRIZATIONS

Permutation-related equality The commutable property of addition directly implies that the permutation of two neurons v, w in the $(l-1)$-th layer does not change the output o_{li} of the i-th neuron in the l-th layer.

$$o_{li} = (w_{li1} z_{(l-1)1} + b_{l1}) + \cdots + (w_{liv} z_{(l-1)v} + b_{lv}) + (w_{liw} z_{(l-1)w} + b_{lw}) \tag{7}$$
$$+ \cdots + (w_{liM_{l-1}} z_{(l-1)M_{l-1}} + b_{lM_{l-1}}) \tag{8}$$
$$= (w_{li1} z_{(l-1)1} + b_{l1}) + \cdots + (w_{liw} z_{(l-1)w} + b_{lw}) + (w_{liv} z_{(l-1)v} + b_{lv}) \tag{9}$$
$$+ \cdots + (w_{liM_{l-1}} z_{(l-1)M_{l-1}} + b_{lM_{l-1}}) \tag{10}$$

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.

$$w_{(l+1)ki} \tanh(o_{li}) = w_{(l+1)ki} \tanh\left(\sum_{j=1}^{M_{l-1}} w_{lij} z_{(l-1)j} + b_{li} \right) \tag{11}$$
$$= -w_{(l+1)ki} \tanh\left(\sum_{j=1}^{M_{l-1}} -w_{lij} z_{(l-1)j} - b_{li} \right) = -w_{(l+1)ki} \tanh(-o_{li}) \tag{12}$$

B DERIVATIONS FOR POSTERIOR DISTRIBUTION REFERENCE SET

For a neural network parametrization $\boldsymbol{\theta} \in \mathbb{R}^d$ and any equioutput parametrization $\boldsymbol{E}_j \in \mathcal{E}$, the prior probability for $\boldsymbol{\theta}$ and $\boldsymbol{E}_j\boldsymbol{\theta}$ is identical if we assume a symmetric prior (which is, by definition of functional symmetry, invariant to permutations in its input arguments). Here, we provide exemplary proof for the popular Gaussian prior (note that non-unitary precision would lead to the same result).

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \boldsymbol{I}) = \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} \right) \overset{\boldsymbol{E}_j \text{ orthogonal}}{=} \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2} \boldsymbol{\theta}^\top \boldsymbol{E}_j^\top \boldsymbol{E}_j \boldsymbol{\theta} \right) \tag{13}$$
$$= \frac{1}{(2\pi)^{d/2}} \exp\left(-\frac{1}{2} (\boldsymbol{E}_j\boldsymbol{\theta})^\top (\boldsymbol{E}_j\boldsymbol{\theta}) \right) = \mathcal{N}(\boldsymbol{E}_j\boldsymbol{\theta}|\mathbf{0}, \boldsymbol{I}) = p(\boldsymbol{E}_j\boldsymbol{\theta}) \tag{14}$$

Similarly, we can show that the likelihood is invariant to equioutput parametrizations.

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\boldsymbol{y}^{(i)}|f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) \overset{\text{equioutput}}{=} \prod_{i=1}^{N} p(\boldsymbol{y}^{(i)}|f_{\boldsymbol{E}_j\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) \tag{15}$$
$$= \prod_{i=1}^{N} p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}, \boldsymbol{E}_j\boldsymbol{\theta}) = p(\mathcal{D}|\boldsymbol{E}_j\boldsymbol{\theta}) \tag{16}$$

Following from the above, the posterior probabilities for two equioutput parametrizations are identical.

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta})}{p(\mathcal{D})} \overset{14,16}{=} \frac{p(\mathcal{D}|\boldsymbol{E}_j\boldsymbol{\theta}) p(\boldsymbol{E}_j\boldsymbol{\theta})}{p(\mathcal{D})} = p(\boldsymbol{E}_j\boldsymbol{\theta}|\mathcal{D}) \tag{17}$$

We now derive the full implication of symmetries for the posterior predictive distribution.

$$
\begin{aligned}
p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = & \int_{\Theta} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} && (18)\\
\overset{1}{=} & \int_{\mathcal{S}_1} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} + \cdots + \int_{\mathcal{S}_{|\mathcal{E}|}} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} && (19)\\
\overset{14,16}{=} & \int_{\mathcal{S}_1} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} + \cdots + \int_{\mathcal{S}_1} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{E}_{|\mathcal{E}|}\boldsymbol{\theta}) p(\boldsymbol{E}_{|\mathcal{E}|}\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} && (20)\\
= & \int_{\mathcal{S}_1} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) + \cdots + p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{E}_{|\mathcal{E}|}\boldsymbol{\theta}) p(\boldsymbol{E}_{|\mathcal{E}|}\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} && (21)\\
= & \int_{\mathcal{S}_1} |\mathcal{E}| \cdot p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} && (22)\\
= & \, |\mathcal{E}| \cdot \int_{\mathcal{S}_1} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} && (23)
\end{aligned}
$$

C Derivations for 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

$$P(G \geq \rho) \leq \mathbb{E}(G) \cdot \rho^{-1}. \tag{24}$$

We can rewrite this inequality as

$$P(G < \rho) \geq 1 - \mathbb{E}[G] \cdot \rho^{-1}. \tag{25}$$

As 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 (CCP; Flajolet et al., 1992), 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:

$$\mathbb{E}(G) = \sum_{q=0}^{\nu-1} (-1)^{\nu-1-q} \sum_{J:|J|=q} (1 - \Pi_J)^{-1}, \quad \text{with } \Pi_J = \sum_{j \in J} \pi_j. \tag{26}$$

Putting together Equation 25 and Equation 26, we get Equation 5.

D Symmetry removal algorithm

In Algorithm 1, we describe how to remove tanh sign-flip symmetries by projecting parameter vectors to one side of a hyperplane with maximum distance to all parameters.

Algorithm 1 Geometry-respecting tanh removal algorithm.

procedure TanhRemoval($\mathcal{M}^{(*,l)}$, $K_{\boldsymbol{\beta}}$)
  $\mathcal{R} \leftarrow \emptyset$
  for each k in range $K_{\boldsymbol{\beta}}$ **do**
    $\boldsymbol{\beta}_k \leftarrow \arg\min_{\boldsymbol{\beta}} \mathcal{L}(\boldsymbol{\beta})$
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{\boldsymbol{\beta}_k\}$
  end for
  $\boldsymbol{\beta}^* \leftarrow \boldsymbol{\beta} \in \mathcal{R}$ with minimal loss $\mathcal{L}(\boldsymbol{\beta})$
  FlipParametersByHyperplane($\mathcal{M}^{(*,l)}$, $\boldsymbol{\beta}^*$)
end procedure

We choose $K_{\boldsymbol{\beta}} > 1$ since in practice we observe that hyperplanes sometimes get stuck in local optima. Note that any hyperplane is valid for the removal of tanh-related equioutput parametrizations. However, in posterior symmetry removal, we strive for a solution that respects the geometry for

optimal mode reduction rate. FLIPPARAMETERSBYHYPERPLANE is a generic method that tests for a neuron parameter vector $\boldsymbol{\phi}$ whether $\boldsymbol{\beta}^\top \boldsymbol{\phi} < 0$ and if yes, projects it to the other side of the hyperplane.

Algorithm 2 demonstrates how to assign each neuron parameter vector its most likely neuron position in hidden layer l. Here, KNNCLASSIFICATION is just a standard k-NN classification algorithm that returns the class probabilities for a vector $\boldsymbol{\phi}$ (here: M_l classes) and that is based on the Gaussian similarity function $s(a, b) = \exp\left(-||a - b||^2 \cdot (2\sigma^2)^{-1}\right)$, with $\sigma = 1.0$. The hyperparameter k should be carefully selected, however, we generally recommend high values, such that the classification is based on as many neighbors as possible. In all the provided experiments we used $k = 1024$.

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

Employing Algorithm 2, we show in Algorithm 3 how permutation symmetries can be removed from a hidden layer l. The method PERMUTE permutes the neuron index of neuron parameter vectors according to the classification results. Since the classification by the greedy constrained k-NN classification algorithm is based on local relationships between neuron parameter vectors, the procedure has to be repeated multiple times I in order to obtain a globally coherent clustering of neuron parameter vectors. We use $I = 256$ in all our experiments.

Algorithm 3 Permutation symmetry removal for a hidden layer l.

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

Lastly, Algorithm 4 summarizes the above steps to remove all considered symmetries while respecting the posterior geometry.

Algorithm 4 Complete geometry-respecting symmetry removal algorithm.

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

E EXPERIMENTAL SETUP

E.1 DATASETS

The sinusoidal dataset $\mathcal{D}_S$ is adopted from an example provided in Daxberger et al. (2021) and the izmailov dataset $\mathcal{D}_B$ is a synthetic dataset from Izmailov et al. (2020). The Regression2d dataset $\mathcal{D}_R$ is another synthetic dataset that we generate as follows (U denoting a uniform distribution).

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

All synthetic datasets are standardized and visualized in Figure 5. We split the data in 80% training and 20% test observations.

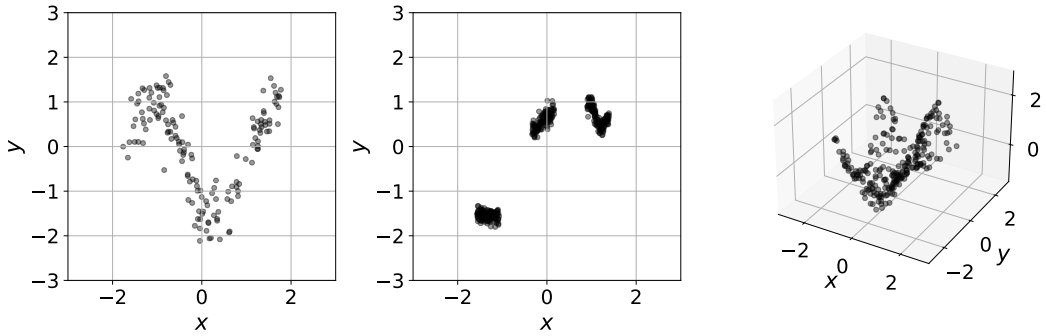

Figure 5: Left, center, right: Visualization of datasets $\mathcal{D}_S$, $\mathcal{D}_I$ and $\mathcal{D}_R$

Together with the synthetic datasets we further provide descriptives and references also for the UCI datasets used in our benchmarks in Table 2.

Table 2: Dataset characteristics and references.

Dataset	# Observations	# Features	Reference
Sinusoidal ($\mathcal{D}_S$)	150	1	adapted from Daxberger et al. (2021)
Izmailov ($\mathcal{D}_i$)	400	1	adapted from Izmailov et al. (2020)
Regression2d ($\mathcal{D}_R$)	256	2	–
Airfoil	1503	5	Dua & Graff (2017)
Concrete	1030	8	Yeh (1998)
Diabetes	442	10	Dua & Graff (2017)
Energy	768	8	Tsanas & Xifara (2012)
Forest Fire	517	12	Cortez & Morais (2007)
Wine	178	13	Dua & Graff (2017)
Yacht	308	6	Dua & Graff (2017)

E.2 Markov chain Monte Carlo

MCMC sampling from the Bayesian models was performed using the `NumPyro` implementation of the No U-Turn Sampler with default settings. Specifically, we only provide our model to the method and 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.

E.3 Point estimates

For the LA and DE estimates, both the small architecture f_1 (one hidden layer of three neurons) and the larger architecture f_2 (three hidden layers of 16 neurons each) were trained with an RMSProp optimizer (Hinton et al., 2012) for 500 (f_1) and 1000 (f_2) epochs, using a constant learning rate of 10^{-4} and no weight decay. In the case of DE we initialized each weight vector with a different random seed and did not use data bootstrapping, following the approach described in the original work by Lakshminarayanan et al. (2017). LA samples were drawn directly from the posterior. Due to the small dataset sizes, we performed full-batch training. Our loss function is derived from the negative log posterior, which is to be minimized over the network parameters $\boldsymbol{\theta}$ and nuisance parameter σ, namely $\min_{\boldsymbol{\theta},\sigma} -\log p(\boldsymbol{\theta}|\mathcal{D})$, s.t.

$$\mathcal{L}(\boldsymbol{\theta}, \sigma) = -\log p(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{2\sigma^2} \sum_{i=1}^{N} (f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \boldsymbol{y}_i)^2 + N \cdot \log \sigma + \frac{1}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} \tag{27}$$

The code is mainly based on the `PyTorch` (Paszke et al., 2019) and `PyTorch Lightning` (Falcon, 2019) libraries, as well as the `Laplace` library by Daxberger et al. (2021).

F Methods and metrics

F.1 KL-Divergence for the posterior predictive

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

$$D_{KL}(p||q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x. \tag{28}$$

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

$$D_{KL}(p||q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}. \tag{29}$$

In Section 6.2 the KL-divergence of consecutive posterior predictive distributions 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 measure.

F.2 SPECTRAL CLUSTERING

The spectral clustering in Section 6.4 was performed by first constructing a 4-NN graph using the Gaussian similarity function $s(a, b) = \exp\left(-||a - b||^2 \cdot (2\sigma^2)^{-1}\right)$, with $\sigma = 1.0$ as a distance measure, defined as the adjacency matrix $\boldsymbol{A}$. We then compute the normalized graph Laplacian from $\boldsymbol{A}$ as follows.

$$\boldsymbol{D} := \text{degree matrix} \tag{30}$$

$$\boldsymbol{D}_{ij} = \begin{cases} \sum_{k=1}^{N} \boldsymbol{A}_{ik} & \text{if } i = j \\ 0 & \text{else} \end{cases} \tag{31}$$

$$\boldsymbol{L}_{norm} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{1/2} \tag{32}$$

Afterwards, the eigenvalue spectrum of the Laplacian was determined and KMeans clustering was performed for $K = 3$.

G ADDITIONAL RESULTS

G.1 SYMMETRY REMOVAL

Full posterior Figure 6 visualizes the full posterior distribution of the BNN of f_1 on dataset $\mathcal{D}_S$ as investigated in Section 6.3 before (red) and after (green) the application of the symmetry removal algorithm. The resulting posterior distribution is unimodal and much simpler compared to the original posterior distribution, yet functionally they are identical.

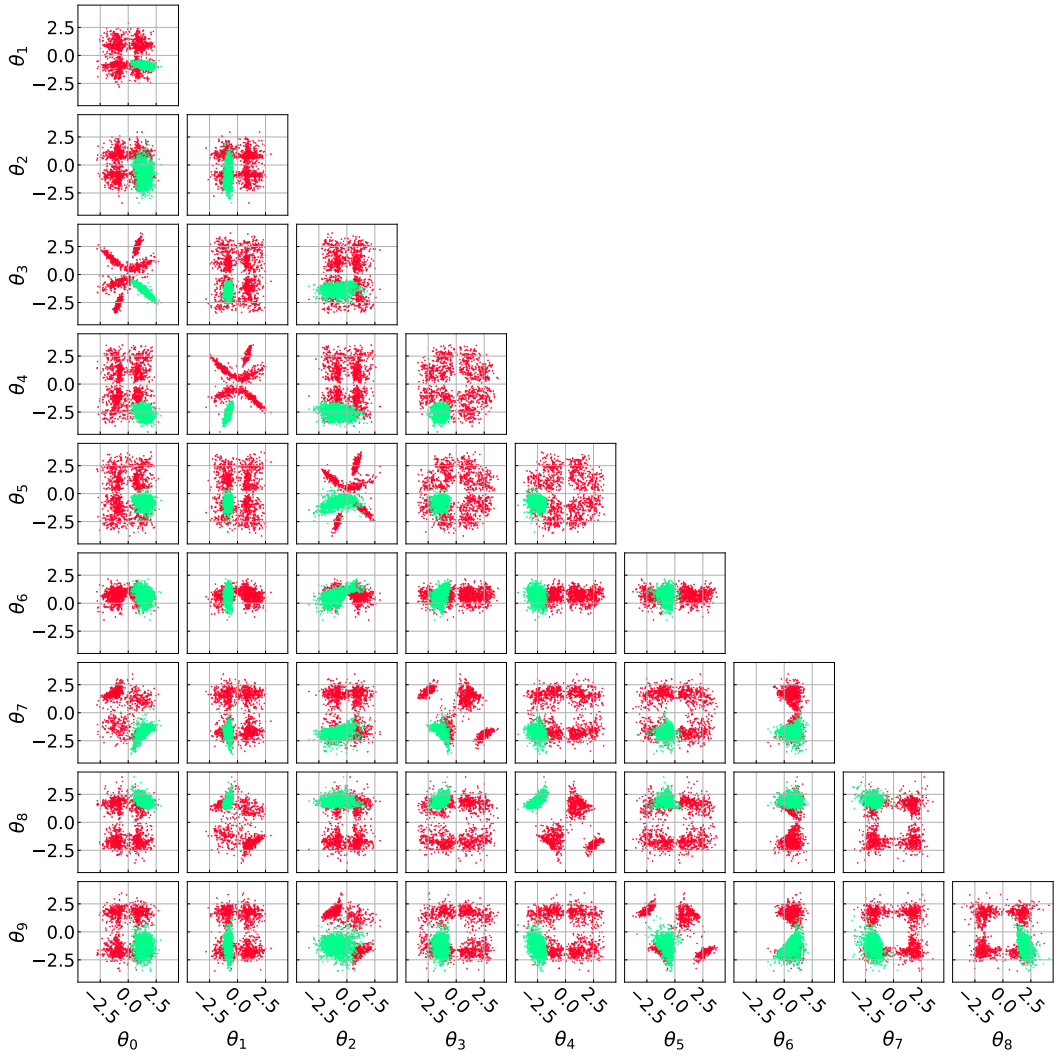

Figure 6: Visualization of the posterior of the BNN f_1 in a triangle plot of pairwise bivariate marginal distributions. The full posterior as collected from MCMC (red) shows a symmetric geometry, the transformed posterior distribution (green) is unimodal.

G.2 Interpretability example approximated

The three remaining modes in the BNN's posterior of the interpretability example can be analytically approximated using a mixture of LA upon clustering. Figure 7 depicts the approximated function space component-wise and as a mixture.

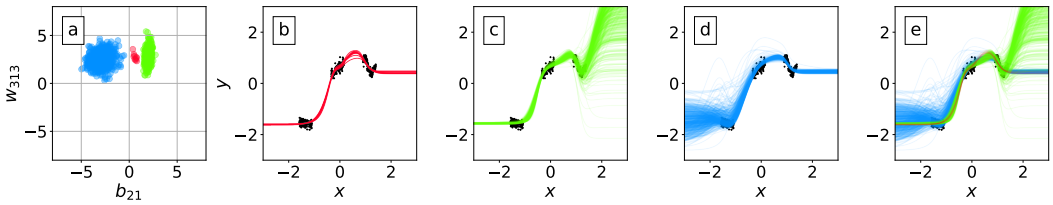

Figure 7: From left to right: The three remaining modes in the BNN's posterior after clustering approximated by MoLA, visualized in the bivariate marginal space of two weights (a); resulting functionally diverse network parametrizations based on the three Gaussian components (b - d); the resulting function space as a composition of the samples of the three Gaussian distributions (e) by combining b - d.