Supplementary Material

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.

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

A.2 Equioutput Equivalence Relation

Let $f : \mathcal{X} \to \mathcal{Y}$ represent an MLP mapping a feature vector $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n, n \in \mathbb{N}$, to an outcome vector $f(\boldsymbol{x}) \in \mathcal{Y} \subseteq \mathbb{R}^m$, $m \in \mathbb{N}$. 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:

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

$\sim$ 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 $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}(\boldsymbol{x}) \,\forall\, \boldsymbol{x} \in \mathcal{X}$.
E2 (Symmetry) For $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \Theta$,

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

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

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

and

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

we have, by the transitivity of equality,

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

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 $\boldsymbol{E}_j \in \mathcal{E}$, the prior probability for $\boldsymbol{\theta}$ and $\boldsymbol{E}_j\boldsymbol{\theta}$ is identical if we assume the prior to be invariant under transformations $\mathcal{F}_{\boldsymbol{E}_j} : \Theta \to \Theta, \boldsymbol{\theta} \mapsto \boldsymbol{E}_j\boldsymbol{\theta}, \boldsymbol{E}_j \in \mathcal{E}$, i.e., $p(\boldsymbol{\theta}) = p(\boldsymbol{E}_j\boldsymbol{\theta})$. 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.

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

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

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

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

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

$\boldsymbol{T}$ is a diagonal matrix that is also orthogonal because the transformation it represents are designed precisely such that $\boldsymbol{T}^\top \boldsymbol{T}$ is the identity matrix (e.g., with diagonal values equal to -1 for tanh). The orthogonality of $\boldsymbol{E}_j$ 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 $\boldsymbol{E}_j\boldsymbol{\theta}$ are equioutput parameter states:

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

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

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

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

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \underset{(14)}{\overset{(13)}{=}} \frac{p(\mathcal{D}|\boldsymbol{E_j\theta})p(\boldsymbol{E}_j\boldsymbol{\theta})}{p(\mathcal{D})} = p(\boldsymbol{E}_j\boldsymbol{\theta}|\mathcal{D}) \tag{14}$$

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 $\mathcal{F}_{\boldsymbol{E}_j}$ induced by $\boldsymbol{E}_j \in \mathcal{E}$, s.t.

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

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

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

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

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

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 $\exists j \in \{1, \dots, |\mathcal{E}|\} : \boldsymbol{\theta} = \boldsymbol{E}_j \boldsymbol{\theta}' \ \Rightarrow \boldsymbol{\theta} \in \mathcal{S}_j$, 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:

$$p(\boldsymbol{y}^*|\boldsymbol{x}^*, \mathcal{D}) = \int_{\Theta} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} \tag{15}$$
$$= \sum_{i=1}^{|\mathcal{E}|} \int_{\mathcal{S}_i} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} \tag{16}$$
$$+ \underbrace{\int_{\mathcal{S}^0} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta}}_{c}$$
$$= \sum_{i=1}^{|\mathcal{E}|} \int_{\mathcal{S}_i} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} + c \tag{17}$$
$$= \sum_{i=1}^{|\mathcal{E}|} \int_{\mathcal{S}_1} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{E}_i\boldsymbol{\theta}) p(\boldsymbol{E}_i\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} + c \tag{18}$$
$$= \int_{\mathcal{S}_1} \sum_{i=1}^{|\mathcal{E}|} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{E}_i\boldsymbol{\theta}) p(\boldsymbol{E}_i\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} + c \tag{19}$$
$$= \int_{\mathcal{S}_1} |\mathcal{E}| \cdot p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} + c \tag{20}$$
$$= |\mathcal{E}| \cdot \int_{\mathcal{S}_1} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} + c \tag{21}$$

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

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

We can rewrite this inequality as

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

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:

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

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 $\boldsymbol{\theta}^{(g)}, g \in \{1, \ldots, G\}$. 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 $i \in \{1, \ldots, M_l\}$ 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 $\boldsymbol{\phi}^{(g,l,i)} \in \mathbb{R}^{(M_{(l-1)}+M_{(l+1)}+1)}$ is defined by

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

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 $\boldsymbol{\phi}^{(g,l,i)}$ 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 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 $\mathcal{M}^{(g,l)} := \{\boldsymbol{\phi}^{(g,l,1)}, \ldots, \boldsymbol{\phi}^{(g,l,M_l)}\}$ 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 posterior density 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; [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 $\boldsymbol{\beta}_l^\top \boldsymbol{\phi}$ for $\boldsymbol{\phi} \in \mathcal{M}^{(*,l)}$, taking the form

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

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 $K_{\boldsymbol{\beta}} > 1$ 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($\mathcal{M}^{(*,l)}$, $K_{\boldsymbol{\beta}}$)
  $\mathcal{R} \leftarrow \emptyset$
  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 $\mathcal{L}(\boldsymbol{\beta})$
  for each $\boldsymbol{\phi} \in \mathcal{M}^{(*,l)}$ **do**
    if $\boldsymbol{\beta}^{*\top} \boldsymbol{\phi} < 0$ **then**
      $\boldsymbol{\phi} \leftarrow -\boldsymbol{\phi}$
    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 $\boldsymbol{\theta}^{(g)}$ with elements $\boldsymbol{\phi}^{(g,l,i)} \in \mathcal{M}^{(g,l)}$ at a time and assign classification labels $c^{(g,l,i)}, i \in \{1, \ldots, M_l\}$. 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 $s(a, b) = \exp\left(-||a - b||^2 \cdot (2\sigma^2)^{-1}\right)$ with $\sigma = 1$ to assign each neuron parameter vector $\boldsymbol{\phi}^{(g,l,i)}$ 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**
    $\phi^{(g,l,i)} \in \mathcal{M}^{(g,l)}$
    $(p_i(c=1), \ldots, p_i(c=M_l)) \leftarrow$ KNNCLASSIFICATION($\boldsymbol{\phi}^{(g,l,i)}$, $\mathcal{M}^{(*,l)}$, k)
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{(p_i(c=1), \ldots, p_i(c=M_l))\}$
  end for
  $i \leftarrow 1$
  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 $\{c^{(g,l,1)}, \ldots, c^{(g,l,M_l)}\}$
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 $(\boldsymbol{\phi}^{(g,l,i)}, \boldsymbol{\phi}^{(g,l,j)})$ 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 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($\mathcal{M}^{(1,l)}$, ..., $\mathcal{M}^{(G,l)}$, $\mathcal{M}^{(*,l)}$, 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

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($\{\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^G\}$, 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 $\{\boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^G\}$
    $\mathcal{M}^{(*,l)} \leftarrow \mathcal{M}^{(1,l)} \cup \cdots \cup \mathcal{M}^{(G,l)}$
    TANHREMOVAL($\mathcal{M}^{(*,l)}$, $K_{\boldsymbol{\beta}}$); see Algorithm 1
    PERMUTATIONREMOVAL($\mathcal{M}^{(1,l)}$, …, $\mathcal{M}^{(G,l)}$, $\mathcal{M}^{(*,l)}$, 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.

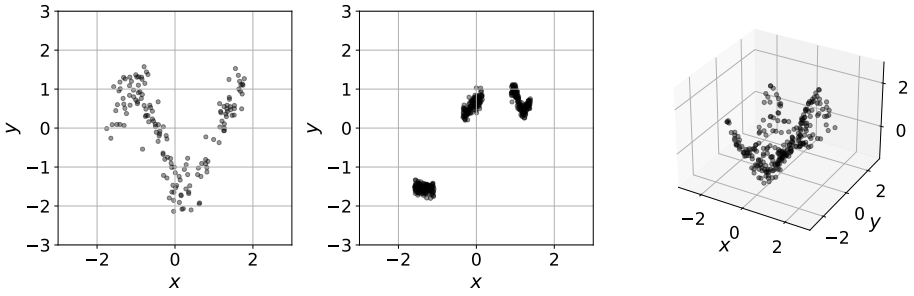

Fig. 4. Visualization of synthetic datasets $\mathcal{D}_S$ (left), $\mathcal{D}_I$ (center), and $\mathcal{D}_R$ (right)

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

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

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

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

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 $s(a, b) = \exp\left(-||a - b||^2 \cdot (2\sigma^2)^{-1}\right)$, with $\sigma = 1$ as a distance measure, encoded via the adjacency matrix $\boldsymbol{A}$. We then compute the normalized graph Laplacian $\boldsymbol{L}_{norm}$ from $\boldsymbol{A}$ as

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

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

where $\boldsymbol{D}_{ij}$ is the (i, j) element of the degree matrix $\boldsymbol{D}$ of the graph. Subsequently, the eigenvalue spectrum of $\boldsymbol{L}_{norm}$ 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.

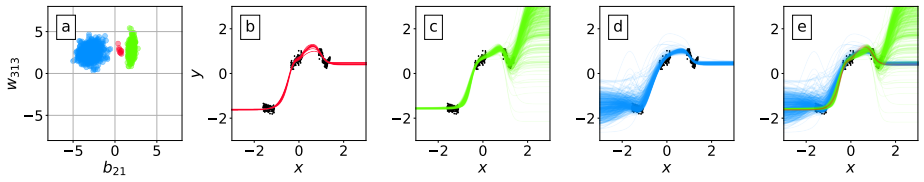

Fig. 5. From left to right: The three remaining modes in the BNN's posterior after clustering approximated by MoLA, visualized in the bivariate marginal space of two weights (a); resulting functionally diverse network parameter states based on the three Gaussian components (b-d); the resulting function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

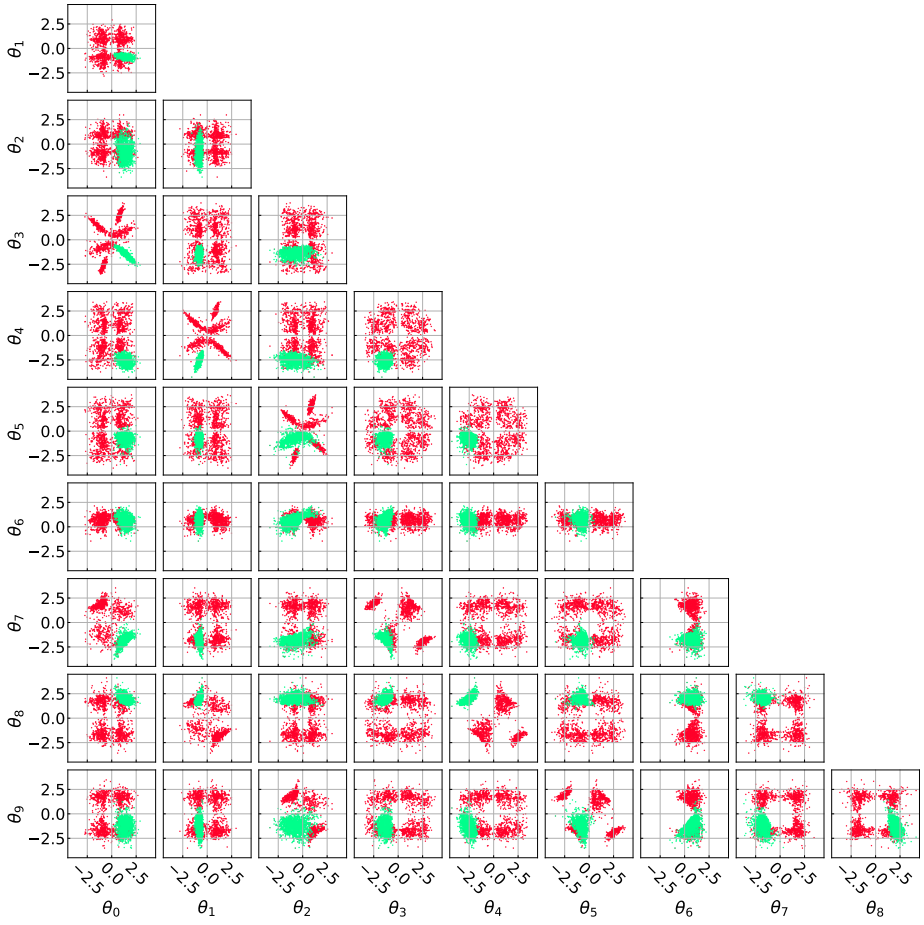

Fig. 6. Visualization of pairwise profiles of the parameter posterior density of BNN f_1. The parameter posterior density sampled via MCMC (red) exhibits symmetries, while the transformed approximate parameter posterior density (green) is unimodal.