

Challenges in Bayesian inference via Markov chain Monte Carlo for neural networks

Theodore Papamarkou, Jacob Hinkle, Michael Todd
Young and David Womble

Abstract. Markov chain Monte Carlo (MCMC) methods and neural networks are instrumental in tackling inferential and prediction problems. However, Bayesian inference based on joint use of MCMC methods and of neural networks is limited. This paper reviews the main challenges posed by neural networks to MCMC developments, including lack of parameter identifiability due to weight symmetries, prior specification effects, and consequently high computational cost and convergence failure. Population and manifold MCMC algorithms are combined to demonstrate these challenges via multilayer perceptron (MLP) examples and to develop case studies for assessing the capacity of approximate inference methods to uncover the posterior covariance of neural network parameters. Some of these challenges, such as high computational cost arising from the application of neural networks to big data and parameter identifiability arising from weight symmetries, stimulate research towards more scalable approximate MCMC methods or towards MCMC methods in reduced parameter spaces.

Key words and phrases: Bayesian inference, Bayesian neural networks, convergence diagnostics, manifold Langevin Monte Carlo, Markov chain Monte Carlo, power posteriors, prior specification, weight symmetries.

1. MOTIVATION

The *universal approximation theorem* (Cybenko, 1989) and its subsequent extensions (Hornik, 1991; Lu et al., 2017) state that feedforward neural networks with exponentially large width and width-bounded deep neural networks can approximate any continuous function arbitrarily well. This universal approximation capacity of neural networks along with available computing power explain the widespread use of deep learning nowadays.

Bayesian inference for neural networks is mainly performed via stochastic Bayesian optimization or via stochastic variational inference (Polson and Sokolov, 2017). MCMC methods have been explored in the context of neural networks, but have not evolved yet to the point of being broadly included in the Bayesian deep learning toolbox.

The slower evolution of MCMC methods for neural networks is partly attributed to the lack of scalability of existing MCMC algorithms for big data

*Computer Science and Engineering Division, Oak Ridge National Laboratory,
Oak Ridge, TN, USA*

and for high-dimensional parameter spaces. Furthermore, additional contributing factors hinder the adaptation of existing MCMC algorithms in deep learning, including the hierarchical structure of neural networks and the associated covariance between parameters, lack of identifiability arising from weight symmetries, lack of a priori knowledge about the parameter space and the associated impact of non-objective priors on the parameter posterior, and ultimately lack of convergence.

The purpose of this paper is twofold. Initially, a literature review is conducted to identify inferential challenges arising from attempts to develop MCMC methods for neural networks over the last three decades. Secondly, contemporary manifold and population MCMC algorithms are applied to small MLPs for the first time.

Manifold and population MCMC provide incremental improvements, are impeded by the same limitations other MCMC algorithms face, and therefore do not offer a solution for scalable Bayesian deep learning. Nevertheless, manifold and population MCMC help showcase many of the challenges reported in the MCMC literature for neural networks and offer a testbed for benchmarking scalable approximate inference methods on small neural networks. More specifically, manifold and population MCMC seem to capture the posterior covariance structure between parameters of small MLPs. Such ground truth is unknown, yet simulations with different manifold and population MCMC algorithms provide unanimous empirical evidence on what the posterior covariance structure might be.

An outline of the paper layout follows. Section 2 reviews the inferential challenges arising from the application of MCMC to neural networks. Section 3 provides an overview of the employed manifold and population MCMC algorithms, including simplified manifold Langevin Monte Carlo and power posteriors, and of the employed MCMC diagnostics, namely of the potential scale reduction factor (PSRF) and of the effective sample size (ESS). Section 4 introduces the truncated Metropolis-adjusted Langevin algorithm and the categorical distribution for swapping states between power posteriors that are used in the examples. An overview of the MLP model and of its likelihood for binary and multiclass classification are provided in section 5. MCMC simulations for two MLPs, one fitted to exclusive-or (XOR) and one to the Iris data, are summarized and interpreted in section 6. Directions for future research in section 7 conclude the paper.

2. INFERENTIAL CHALLENGES

A literature review of inferential challenges in the application of MCMC methods to neural networks is conducted in this section thematically, with each subsection being focused on a different challenge.

2.1 Computational cost

Existing MCMC algorithms do not scale with increasing number of parameters or of samples. For this reason, approximate inference methods, including variational inference (VI), are preferred in high-dimensional parameter spaces or in big data problems from a time complexity standpoint (Blei, Kucukelbir and McAuliffe, 2017). On the other hand, MCMC methods are known to be better than VI in terms of approximating the log-likelihood (Dupuy and Bach, 2017).

Literature on MCMC algorithms for neural networks is limited due to associated computational complexity implications. Sequential Monte Carlo and reversible jump MCMC have been applied on two types of neural network architectures, namely MLPs and radial basis function networks (RBFs), see for instance [Andrieu, de Freitas and Doucet \(1999\)](#); [de Freitas \(1999\)](#); [Andrieu, de Freitas and Doucet \(2000\)](#); [de Freitas et al. \(2001\)](#). For a review of MCMC approaches to neural networks, see [Titterington \(2004\)](#).

A resurgence of interest in scaling MCMC methods to big data has been reflected in literature over the last five years. The main focus has been on designing Metropolis-Hastings or Gibbs sampling variants that evaluate a costly log-likelihood on a subset (*minibatch*) of the data rather than on the entire data set ([Welling and Teh, 2011](#); [Chen, Fox and Guestrin, 2014](#); [Ma, Foti and Fox, 2017](#); [De Sa, Chen and Wong, 2018](#); [Nemeth and Sherlock, 2018](#); [Robert et al., 2018](#); [Seita et al., 2018](#); [Quiroz et al., 2019](#)).

Among recent attempts to scale MCMC algorithms to big data applications, there exists a smaller subset of studies applying such algorithms to neural networks ([Chen, Fox and Guestrin, 2014](#); [Gu, Ghahramani and Turner, 2015](#); [Gong, Li and Hernández-Lobato, 2019](#)). Minibatch MCMC approaches to neural networks pave the way towards *data-parallel deep learning*. On the other hand, to the best of the authors' knowledge, there is no published research on MCMC algorithms that evaluate the log-likelihood on a subset of neural network parameters rather than on the whole set of parameters, and therefore no reported research on *model-parallel deep learning* via MCMC.

2.2 Model structure

A neural network with ρ layers can be viewed as a hierarchical model with ρ levels, each network layer representing a level ([Williams, 2000](#)). Due to its non-linear activations, a neural network is specifically a *non-linear hierarchical model*.

MCMC methods for non-linear hierarchical models have been developed, see for example [Bennett, Racine-Poon and Wakefield \(1996\)](#); [Gilks and Roberts \(1996\)](#); [Daniels and Kass \(1998\)](#); [Sargent, Hodges and Carlin \(2000\)](#). However, existing MCMC algorithms for non-linear hierarchical models have not been harnessed by neural networks due to time complexity and convergence implications.

Although not designed to mirror the hierarchical structure of a neural network, recent hierarchical VI ([Ranganath, Tran and Blei, 2016](#); [Esmaeili et al., 2019](#); [Huang et al., 2019](#); [Titsias and Ruiz, 2019](#)) provides more expressive variational approximations of the parameter posterior of the neural network than mean-field VI. Introducing a hierarchical structure in the variational distribution induces correlation among parameters, in contrast to the mean-field variational distribution that assumes independent parameters. So, one of the goals of Bayesian inference for neural networks is to approximate the covariance structure among network parameters. In fact, there are published comparisons between MCMC and VI in terms of speed and accuracy of convergence to the posterior covariance, both in linear or mixture model examples ([Giordano, Broderick and Jordan, 2015](#); [Mandt, Hoffman and Blei, 2017](#); [Ong, Nott and Smith, 2018](#)) and in neural network examples ([Zhang et al., 2018a](#)).

Manifold Monte Carlo methods introduce a proposal mechanism that approx-

imates the covariance of the target posterior locally (Girolami and Calderhead, 2011). Thereby, manifold Monte Carlo can be used in toy neural networks to benchmark the capacity of large scale approximate Bayesian inference methods to capture the posterior covariance among neural network parameters.

2.3 Weight symmetries

The output of feedforward neural networks given some fixed input remains unchanged under a set of transformations determined by the choice of activations and by the network architecture more generally. For instance, certain weight permutations and sign flips in MLPs with hyperbolic tangent activations leave the output unchanged (Chen, Lu and Hecht-Nielsen, 1993).

If a parameter transformation leaves the output of a neural network unchanged given some fixed input, then the parameter posterior is invariant under the transformation. In other words, transformations, such as weight permutations and sign-flips, render neural networks *non-identifiable* (Pourzanjani, Jiang and Petzold, 2017).

It is known that the set of linear invertible parameter transformations that leaves the output unchanged is a subgroup T of the group of invertible linear mappings from the parameter space \mathbb{R}^n to itself (Hecht-Nielsen, 1990). T is a transformation group acting on the parameter space \mathbb{R}^n . It can be shown that for each permutable feedforward neural network, there exists a cone $H \subset \mathbb{R}^n$ dependent only on the network architecture such that for any parameter $\theta \in \mathbb{R}^n$ there exist $\eta \in H$ and $\tau \in T$ such that $\tau\eta = \theta$. This relation means that every network parameter is equivalent to a parameter in the proper subset H of \mathbb{R}^n (Hecht-Nielsen, 1990). Neural networks with convolutions, max-pooling and batch-normalization contain more types of weight symmetries than MLPs (Badrinarayanan, Mishra and Cipolla, 2015).

In practice, the parameter space of a neural network is set to be the whole of \mathbb{R}^n rather than a cone H of \mathbb{R}^n . Since a neural network posterior with support in the non-reduced parameter space of \mathbb{R}^n is invariant under weight permutations, sign-flips or other transformations, the posterior landscape includes multiple equally likely modes. This implies low acceptance rate, entrapment in local modes and convergence challenges for MCMC. Additionally, computational time is wasted during MCMC, since posterior modes represent equivalent solutions (Nalisnick, 2018). Such challenges manifest themselves in the MLP examples of section 6. For neural networks with higher number n of parameters in \mathbb{R}^n , the topology of the likelihood function is characterized by local optima embedded in high-dimensional flat plateaus (Brea et al., 2019). Thereby, larger neural networks lead to a multimodal target density with *sparse* modes for MCMC.

Seeking parameter symmetries in neural networks can lead to a variety of NP-hard problems (Ensign et al., 2017). Moreover, symmetries in neural networks pose identifiability and associated inferential challenges in Bayesian inference, but they also provide opportunities to develop inferential methods with reduced computational cost (Hu, Zagoruyko and Komodakis, 2019) or with improved predictive performance (Moore, 2016). Empirical evidence from stochastic optimization simulations suggests that removing weight symmetries has a negative effect on prediction accuracy in smaller and shallower convolutional neural networks (CNNs), but has no effect in prediction accuracy in larger and deeper

CNNs (Maddison et al., 2015). So, elimination or exploitation of weight symmetries provides scope for scalable Bayesian inference in deep learning. At the same time, finding weight symmetries is not a trivial problem.

2.4 Prior specification

Parameter priors have been used for generating Bayesian smoothing or regularization effects. For instance, de Freitas (1999) develops sequential Monte Carlo methods with smoothing priors for MLPs and Williams (1995) introduces Bayesian regularization and pruning to neural networks via a Laplace prior.

When parameter prior specification for a neural network is not driven by smoothing or regularization, the question becomes how to choose the prior. The choice of parameter prior for a neural network is crucial in that it affects the parameter posterior (Lee, 2004), and consequently the predictive posterior (Lee, 2005). The impact of prior choice on the parameter posterior is demonstrated in the MCMC simulations of section 6.1 by fitting an MLP model to XOR.

Neural networks are commonly applied to big data. For large amounts of data, practitioners usually do not have intuition about the relationship between input and output variables. Furthermore, it is an open research question for scientists to interpret neural network weights and biases. As a priori knowledge about big data sets and about neural network parameters is typically not available, *prior elicitation* from experts is not applicable to neural networks in practice.

It seems logical to choose a prior that reflects a priori ignorance about the parameters. A constant-valued prior is a possible candidate, with the caveat of being improper for unbounded parameter spaces, such as \mathbb{R}^n . However, for neural networks, an *improper prior* can result in an improper parameter posterior (Lee, 2005).

Typically, a *truncated flat prior* for neural networks is sufficient for ensuring a valid parameter posterior (Lee, 2005). At the same time, the choice of truncation bounds depends on weight symmetry and consequently on the allocation of equivalent points in the parameter space. Lee (2003) proposes a *restricted flat prior* for feedforward neural networks by bounding some of the parameters and by imposing constraints that guarantee layer-wise linear independence between activations, while Lee (2000) shows that this prior is asymptotically consistent for the posterior. Moreover, Lee (2003) demonstrates that such a restricted flat prior enables more effective MCMC sampling in comparison to alternative prior choices.

Objective prior specification is an area of statistics that has not infiltrated substantially Bayesian inference for neural networks. Alternative ideas for constructing objective priors with minimal effect on posterior inference exist in the statistics literature. For example, *Jeffreys priors* are invariant to differentiable one-to-one transformations of the parameters (Jeffreys, 1962), *maximum entropy priors* maximize the Shannon entropy and therefore provide the least possible information (Jaynes, 1968), *reference priors* maximize the expected Kullback-Leibler divergence from the associated posteriors and in that sense are the least informative priors (Bernardo, 1979), and *penalised complexity priors* penalise the complexity induced by deviating from a simpler base model (Simpson et al., 2017).

To the best of the authors' knowledge, there are only two published lines

of research on objective priors for neural networks; a theoretical derivation of Jeffreys and reference priors for feedforward neural networks by Lee (2007), and an approximation of reference priors via Monte Carlo sampling of a differentiable non-centered parameterization of MLPs and CNNs by Nalisnick (2018).

More broadly, research on prior specification for Bayesian neural networks (BNNs) has been published recently (Pearce et al., 2019; Vladimirova et al., 2019). For a more thorough review of prior specification for BNNs, see Lee (2005).

2.5 Convergence

MCMC convergence depends on the shape of the target density, namely on multi-modality, model sparsity and lack of smoothness. A small MLP with as few as twenty seven parameters makes convergence in fixed sampling time challenging for contemporary manifold and population MCMC algorithms (see example of section 6.2).

Attaining MCMC convergence is not the only challenge. Assessing whether a finite sample from an MCMC algorithm represents an underlying target density can not be done with certainty (Cowles and Carlin, 1996). MCMC diagnostics can fail to detect the type of convergence failure they were designed to identify. Combinations of diagnostics are thus used in practice to evaluate MCMC convergence with reduced risk of false diagnosis. In this paper, the potential scale reduction factor (PSRF) and the effective sample size (ESS) are used jointly to assess MCMC convergence (see section 3.2).

MCMC diagnostics have been designed with asymptotically exact MCMC algorithms in mind. Recently, research activity on approximate MCMC methods has emerged (Welling and Teh, 2011; Chen, Fox and Guestrin, 2014; Mandt, Hoffman and Blei, 2017; Rudolf and Schweizer, 2018; Chen et al., 2019) along with new diagnostics that quantify convergence of approximate MCMC methods (Chwialkowski, Strathmann and Gretton, 2016).

Quantization and *discrepancy* are two notions pertinent to approximate MCMC algorithms. The quantization of a target density p by an empirical measure \hat{p} provides an approximation to the target (Graf and Luschgy, 2007), while the notion of discrepancy quantifies how well the empirical measure \hat{p} approximates the target density p (Chen et al., 2019). The *kernel Stein discrepancy* (KSD) and the *maximum mean discrepancy* (MMD) constitute two instances of discrepancy; for more details, see Chen et al. (2019) and Gretton et al. (2012), respectively. Rudolf and Schweizer (2018) provides an alternative way of assessing the quality of approximation of a target density p by \hat{p} in the context of approximate MCMC using the notion of *Wasserstein distance* between \hat{p} and p .

A remark follows, which is beyond the scope of the present paper and towards future approximate MCMC developments. For parametric models, such as neural networks, there is no prior knowledge about the true parameter posterior. For this reason, it seems useful to attempt constructing approximate MCMC algorithms by measuring the discrepancy or Wasserstein distance between predictions and output data (Rudolf and Schweizer, 2018) instead of measuring discrepancy in the parameter space.

3. OVERVIEW OF MCMC ALGORITHMS AND DIAGNOSTICS

Sections 3.1 and 3.2 describe the respective MCMC algorithms and MCMC diagnostics used in the examples of this paper.

3.1 MCMC algorithms

Interest is in sampling from a possibly unnormalized *target density* $p : E \rightarrow [0, \infty)$ on a parameter space E . For a neural network, the parameter space E consists of the weights and biases of the network.

This section provides a description of the MCMC algorithms used in the examples. There are many available MCMC samplers, and no claim is made that the ones chosen in the paper are the most efficient among all. Langevin Monte Carlo has been chosen over Hamiltonian Monte Carlo on the basis of more feasible computational complexity of the respective manifold versions of these two families of samplers, since simplified manifold Langevin Monte Carlo can operate using only the gradient and Hessian of the target density, whereas manifold Hamiltonian Monte Carlo requires computing also the derivatives of the Hessian (Girolami and Calderhead, 2011).

3.1.1 Metropolis-Hastings algorithm Consider a *proposal density* $g_{\theta^{(k)}} : E \rightarrow [0, \infty)$ associated with state $\theta^{(k)} \in E$ at the k -th iteration of Metropolis-Hastings (MH). Let $\theta^* \in E$ be a state sampled from $g_{\theta^{(k)}}$. The MH *acceptance probability* is

$$(3.1) \quad r(\theta^{(k)}, \theta^*) := \min \left\{ \frac{p(\theta^*) g_{\theta^*}(\theta^{(k)})}{p(\theta^{(k)}) g_{\theta^{(k)}}(\theta^*)}, 1 \right\},$$

if $p(\theta^{(k)}) g_{\theta^{(k)}}(\theta^*) > 0$, and $r(\theta^{(k)}, \theta^*) := 1$ otherwise.

A normal proposal density $g_{\theta^{(k)}} = \mathcal{N}(\theta^{(k)}, \Sigma)$ with a constant covariance matrix Σ simplifies the acceptance probability to $\min \{p(\theta^*)/p(\theta^{(k)}), 1\}$, defining the *random walk Metropolis* (RWM) algorithm.

Roberts, Gelman and Gilks (1997) suggest an optimal acceptance rate of 23.4% for RWM under certain assumptions for the target. For a more recent account of optimal acceptance rates for RWM algorithms, see Bédard (2008).

3.1.2 Metropolis-adjusted Langevin algorithm The class of MH algorithms with normal proposal density

$$(3.2) \quad g_{\theta^{(k)}} = \mathcal{N}(\mu(\theta^{(k)}, M, \epsilon), \epsilon^2 M^{-1}),$$

$$(3.3) \quad \mu(\theta^{(k)}, M, \epsilon) = \theta^{(k)} + \frac{\epsilon^2}{2} M^{-1} \nabla \log p(\theta^{(k)}).$$

is known as the class of Metropolis-adjusted Langevin algorithms (MALA). M and ϵ are hyper-parameters known as the *preconditioning matrix* (Roberts and Stramer, 2002) and *integration stepsize*, respectively. MALA differs from RWM in that it uses the gradient $\nabla \log p(\theta^{(k)})$ of the log-target density to update the proposal density mean (3.3).

M is a positive-definite matrix of size $n \times n$, assuming an n -dimensional parameter space E . It is commonly set to the identity matrix $M = I$. In high-dimensional parameter spaces (as $n \rightarrow \infty$), the optimal integration stepsize $\epsilon \in \mathbb{R}_+$ is selected to attain a a limiting acceptance rate of 57.4% (Roberts and Rosenthal, 1998).

3.1.3 Simplified manifold Metropolis-adjusted Langevin algorithm Similarly to MALA, the simplified manifold Metropolis-adjusted Langevin algorithm (SMMALA) is also an MH algorithm. SMMALA is defined by the normal proposal density

$$(3.4) \quad g_{\theta^{(k)}} = \mathcal{N}(\mu(\theta^{(k)}), M(\theta^{(k)}), \epsilon, \epsilon^2 M^{-1}(\theta^{(k)})),$$

$$(3.5) \quad \mu(\theta^{(k)}, M(\theta^{(k)}), \epsilon) = \theta^{(k)} + \frac{\epsilon^2}{2} M^{-1}(\theta^{(k)}) \nabla \log p(\theta^{(k)}).$$

Proposal density (3.4) is a generalization over (3.2). The difference between SMMALA and MALA is that the former assumes a position-dependent preconditioning matrix $M(\theta^{(k)})$ at iteration k . Consequently, the covariance matrix of the SMMALA proposal density at iteration k depends on the current state $\theta^{(k)}$.

There is not any optimal scaling theory for tuning the stepsize ϵ of SMMALA towards an optimal acceptance rate. [Girolami and Calderhead \(2011\)](#) suggest empirically to tune ϵ to obtain an acceptance rate of around 70% for SMMALA.

3.1.4 Choice of metric tensor The SMMALA proposal density provides a sampling framework subject to the choice of position-dependent precondition matrix $M(\theta^{(k)})$. Differential geometry motivates the choice of $M(\theta^{(k)})$.

Consider a space of densities $\{p(y|\theta) : \theta\}$ parameterized by $\theta \in E = \mathbb{R}^n$. According to [Rao \(1945\)](#), the KL divergence between $p(y|\theta)$ and $p(y|\theta + \delta\theta)$ is given by

$$(3.6) \quad \text{KL}[p(y|\theta) \| p(y|\theta + \delta\theta)] \approx \frac{1}{2} \delta\theta^T M(\theta) \delta\theta,$$

where $M(\theta) = -\mathbf{E}_{y|\theta} \left[\frac{\partial^2}{\partial\theta^2} \log p(y|\theta) \right]$ is the expected Fisher information matrix of y conditional on θ . More generally, $M(\theta)$ is a metric tensor conveying a notion of distance in the space of densities $\{p(y|\theta) : \theta\}$ on the basis of (3.6), despite the fact that KL divergence is assymetric and therefore not a metric.

In a Bayesian setting, a posterior $p(\theta|y) \propto p(y|\theta)\pi(\theta)$ plays the role of target density, where $p(y|\theta)$ is the likelihood and $\pi(\theta)$ is the prior. One option is to then set $M(\theta^{(k)})$ at the k -iteration of SMMALA to be the *observed Fisher information matrix* of y given $\theta^{(k)}$, which is the negative Hessian of the log-likelihood $\log p(y|\theta^{(k)})$, plus the negative Hessian of the log-prior $\log \pi(\theta^{(k)})$:

$$(3.7) \quad M(\theta^{(k)}) = -\frac{\partial^2}{\partial\theta^2} \log p(y|\theta) \Big|_{\theta=\theta^{(k)}} - \frac{\partial^2}{\partial\theta^2} \log \pi(\theta) \Big|_{\theta=\theta^{(k)}}.$$

For more information about possible choices of metric tensor $M(\theta^{(k)})$ and for a more extensive treatment of manifold Langevin Monte Carlo methods, see [Girolami and Calderhead \(2011\)](#).

Sampling and evaluating the SMMALA proposal density require the Cholesky decomposition of $M(\theta^{(k)})$ and the inverse $M^{-1}(\theta^{(k)})$, respectively. Due to the observed Fisher information matrix in (3.7) not being always positive-definite, such linear algebra calculations can result in singularities. The SoftAbs metric by [Betancourt \(2013\)](#) is used for approximating metric tensors $M(\theta^{(k)})$ while avoiding such singularities. In the examples of the present paper, the SoftAbs metric is employed for appoximating metric (3.7).

Algorithm 1 Power posterior sampling

```

1: for  $k = 1, 2, \dots, v$  do
2:   for  $i = 0, 1, \dots, m$  do ▷ Within-chain moves
3:     Update state  $\theta^{(k,t_i)}$  via an MCMC step with target  $p^{t_i}(\cdot|y)$ 
4:   end for
5:
6:   for  $i = 0, 1, \dots, m$  do ▷ Between-chain moves
7:     Sample  $j \in \{0, 1, \dots, m\} \setminus \{i\}$  from  $p_i$ 
8:     Swap  $\theta^{(k,t_i)}$  with  $\theta^{(k,t_j)}$  with probability  $\min\left\{\frac{p^{t_i}(\theta^{(k,t_j)}|y)p^{t_j}(\theta^{(k,t_i)}|y)p_j(i)}{p^{t_i}(\theta^{(k,t_i)}|y)p^{t_j}(\theta^{(k,t_j)}|y)p_i(j)}, 1\right\}$ 
9:   end for
10: end for

```

3.1.5 Power posteriors Power posterior sampling by [Friel and Pettitt \(2008\)](#) is a population Monte Carlo algorithm. It involves $m + 1$ chains drawn from tempered versions $p^{t_i}(\theta|y)$ of a target posterior $p(\theta|y)$ for a *temperature schedule* $t_i \in [0, 1]$, $i \in \{0, 1, \dots, m\}$, where $t_m = 1$. At each iteration, the state of each chain is updated using an MCMC sampler associated with that chain and subsequently states between pairs of chains are swapped according to an MH algorithm. For the i -th chain, a sample j is drawn from a probability mass function p_i with probability $p_i(j)$, in order to determine the pair (i, j) for a possible swap. Algorithm 1 describes power posterior sampling in more detail; $\theta^{(k,t_i)} \in E$ denotes the parameter state at the k -th MCMC iteration of the chain associated with *power posterior* $p^{t_i}(\cdot|y)$.

Bayesian model selection, and more specifically log-marginal likelihood and Bayes factor computations, drive the development of power posteriors in [Friel and Pettitt \(2008\)](#). More generally, power posteriors are a useful population MCMC method for sampling from multimodal target densities. Power posteriors $p^{t_i}(\theta|y)$, $t_i < t_m$, are smooth approximations of the target density $p^{t_m}(\theta|y) = p(\theta|y)$, facilitating exploration of the parameter space via state transitions between chains of $p^{t_i}(\theta|y)$ and of $p(\theta|y)$.

3.2 Numerical MCMC diagnostics

This section outlines PSRF and ESS, which are the two numerical MCMC diagnostics used in the examples.

3.2.1 Potential scale reduction factor Convergence of a Markov chain to its stationary distribution can fail in various ways. For example, an MCMC algorithm may explore only one area in the support of the target density or may require more iterations to converge or may be sensitive to the choice of initial parameter value.

PSRF, commonly denoted by \hat{R} , is an MCMC diagnostic of convergence conceived by [Gelman and Rubin \(1992\)](#) and extended to its multivariate version by [Brooks and Gelman \(1998\)](#). To compute PSRF, several independent Markov chains are simulated. The idea behind PSRF is that the variance of all the chains will be higher than the variance of individual chains, if convergence has not been reached. [Gelman et al. \(2013\)](#) propose *split- \hat{R}* , a modification of \hat{R} that aims at comparing the distribution of the first half of each chain after warm up to the distribution of the second half of the chain.

Empirical evidence suggests that both \hat{R} and split- \hat{R} succeed in detecting lack

of convergence of the first moment as long as the Monte Carlo variance is not very high. [Vehtari et al. \(2019\)](#) suggest two transformations to make split- \hat{R} operational under high Monte Carlo variance; normalizing the chain around the median (referred to as *folding the chain*) provides a statistic sensitive to simulated chains with same location and different scales, while *rank-normalizing the folded chain* reduces the effect of heavy tails. The *folded-split- \hat{R}* is defined to be the value of split- \hat{R} computed on rank-normalized values of the folded chain, see [Vehtari et al. \(2019\)](#) for more details. In this paper, both split- \hat{R} and folded-split- \hat{R} are reported.

[Gelman et al. \(2004\)](#) recommend terminating MCMC sampling as soon as $\hat{R} < 1.1$. More recently, [Vats and Knudson \(2018\)](#) make an argument based on ESS that a cut-off of 1.1 for \hat{R} is too high to estimate a Monte Carlo mean with reasonable uncertainty. [Stan Development Team \(2019\)](#) recommend simulating at least four chains to compute \hat{R} and using a threshold of $\hat{R} < 1.05$.

3.2.2 Effective sample size \hat{R} and its variants can fail to diagnose poor mixing of a Markov chain, whereas low values of ESS are an indicator of poor mixing. It is thus recommended to check both \hat{R} and ESS ([Vehtari et al., 2019](#)). For a theoretical treatment of the relation between \hat{R} and ESS, see [Vats and Knudson \(2018\)](#).

The ESS of a Monte Carlo estimate obtained from a Markov chain is defined to be the number of independent samples that provide an estimate with variance equal to the variance of the Monte Carlo estimate. For a more extensive treatment entailing alternative definitions and estimators of ESS, see for example [Vats and Flegal \(2018\)](#); [Gong and Flegal \(2016\)](#); [Kass et al. \(1998\)](#).

[Vehtari et al. \(2019\)](#) proposes the commonly used effective sample size estimator $\hat{S} := (m + 1)v/\hat{\tau}$ across $m + 1$ independent chains of length v each, where $\hat{\tau}$ is an estimator of the *integrated autocorrelation time*. An average of within-chain autocorrelation across the v chains and an estimate of between-chain variance are combined to provide a between-chain autocorrelation estimate $\hat{\rho}$. The resulting $\hat{\rho}$ is then truncated to a maximum lag to reduce noise from autocorrelation estimation, and $\hat{\tau}$ is computed using $\hat{\rho}$ on the basis of the *initial monotone sequence estimator* by [Geyer \(1992\)](#).

The estimation of ESS in this paper bears a similarity to ([Vehtari et al., 2019](#)) in that both approaches involve computing the initial monotone sequence estimator of [Geyer \(1992\)](#) across independent chains. Despite this similarity, the ESS estimator in this paper differs from ([Vehtari et al., 2019](#)). For a Markov chain $\{\theta^{(k)} : k = 1, 2, \dots, v\}$ consisting of states $\theta^{(k)} \in E \subseteq \mathbb{R}^n$, the ESS estimator of the j -the coordinate is defined thereafter to be

$$(3.8) \quad \hat{S}_j := v \frac{\widehat{\text{Var}}^{(\text{IID})}(\theta_j^{(1:v)})}{\widehat{\text{Var}}^{(\text{MC})}(\theta_j^{(1:v)})},$$

where $\theta_j^{(1:v)} := \{\theta_j^{(k)} : k = 1, 2, \dots, v\}$, and $\theta_j^{(k)}$ is the j -coordinate of $\theta^{(k)}$. The term $\widehat{\text{Var}}^{(\text{IID})}(\theta_j^{(1:v)}) := \frac{1}{v-1} \sum_{k=1}^v (\theta_j^{(k)} - \bar{\theta}_j)^2$ refers to the variance of coordinate j under the assumption of independent identically distributed (IID) samples $\theta_j^{(1:v)}$, while the term $\widehat{\text{Var}}^{(\text{MC})}(\theta_j^{(1:v)})$ denotes an estimator of the Monte Carlo variance

of coordinate j . The initial monotone sequence estimator is used as $\widehat{\text{Var}}^{(\text{MC})}(\theta_j^{(1:v)})$ (Geyer, 1992; Papamarkou, Mira and Girolami, 2014).

To understand the intuition behind (3.8), start by observing that sample variance increases with smaller sample size, so sample variance is inversely proportional to sample size. Thus, assume the relation $y = a/x$ between the number of IID samples y and the sample variance x for these IID samples. $(x, y) = (v, \widehat{\text{Var}}^{(\text{IID})}(\theta_j^{(1:v)}))$ satisfies the relation $y = a/x$, so $a = v\widehat{\text{Var}}^{(\text{IID})}(\theta_j^{(1:v)})$, whence $y = v\widehat{\text{Var}}^{(\text{IID})}(\theta_j^{(1:v)})/x$. It now becomes obvious that the number of IID samples y with variance $x = \widehat{\text{Var}}^{(\text{MC})}(\theta_j^{(1:v)})$ is given by (3.8).

For a set of $m + 1$ independent chains $\theta^{(1:v,i)} := \{\theta^{(k,i)} : k = 1, 2, \dots, v\}$, $i \in \{0, 1, \dots, m\}$, each having ESS $\hat{S}_j^{(i)} = v\widehat{\text{Var}}^{(\text{IID})}(\theta_j^{(1:v,i)})/\widehat{\text{Var}}^{(\text{MC})}(\theta_j^{(1:v,i)})$ as defined by (3.8), the sample mean $\hat{S}_j := \frac{1}{m+1} \sum_{i=0}^m \hat{S}_j^{(i)}$ is computed in the examples of section 6.

The sample mean of ESS (3.8) across independent chains has been reported in the examples due to making more conservative claims in comparison to the ESS of Vehtari et al. (2019). A minimum value of 100 per chain for the ESS of Vehtari et al. (2019) is empirically suggested by Stan Development Team (2019), and such a threshold is employed in the present paper on the basis of ESS (3.8).

4. MCMC MODIFICATIONS

Two sampling tweaks are made in the examples. Firstly, some simulations assume neural network weights and biases with support in a bounded and closed interval $[l, u] \subset \mathbb{R}^n$ for some $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$, relying on a version of MALA with a truncated proposal density. Secondly, a categorical probability mass function is used in power posterior sampling for determining candidate pairs of chains for state swaps.

4.1 Truncated Metropolis-adjusted Langevin algorithm

Consider a target density $p : E \rightarrow [0, \infty)$ with support in $E = [l, u] \subset \mathbb{R}^n$, $l \in \mathbb{R}^n$, $u \in \mathbb{R}^n$. To sample from p via MALA, the truncated normal proposal density

$$(4.1) \quad g_{\theta^{(k)}} = \prod_{i=1}^n \mathcal{T}\mathcal{N}(\mu_i(\theta^{(k)}, I, \epsilon), \epsilon^2, l_i, u_i)$$

can be employed, which corresponds to a normal density $\mathcal{N}(\mu(\theta^{(k)}, I, \epsilon), \epsilon^2 I)$ bounded in $[l, u]$. The terms $\mu_i(\theta^{(k)}, I, \epsilon)$, l_i and u_i refer to the i -th coordinates of $\mu(\theta_k, I, \epsilon)$, l and u , respectively.

The acceptance probability of MALA with truncated normal proposal density (4.1) is derived from (3.1) as

$$r(\theta^{(k)}, \theta^*) = \min \left\{ \frac{p(\theta^*) \prod_{i=1}^n \phi\left(\frac{\theta^{(k)} - \mu_i(\theta^*)}{\epsilon}\right) \Phi\left(\frac{u_i - \mu_i(\theta^{(k)})}{\epsilon}\right) \Phi\left(\frac{l_i - \mu_i(\theta^{(k)})}{\epsilon}\right)}{p(\theta^{(k)}) \prod_{i=1}^n \phi\left(\frac{\theta^* - \mu_i(\theta^{(k)})}{\epsilon}\right) \Phi\left(\frac{u_i - \mu_i(\theta^*)}{\epsilon}\right) \Phi\left(\frac{l_i - \mu_i(\theta^*)}{\epsilon}\right)}, 1 \right\},$$

if $p(\theta^{(k)}) \prod_{i=1}^n \phi\left(\frac{\theta^* - \mu_i(\theta^{(k)})}{\epsilon}\right) \Phi\left(\frac{u_i - \mu_i(\theta^*)}{\epsilon}\right) \Phi\left(\frac{l_i - \mu_i(\theta^*)}{\epsilon}\right) > 0$, and $r(\theta^{(k)}, \theta^*) = 1$ otherwise. ϕ and Φ denote the respective density and cumulative distribution

function of the standard normal distribution, while $\mu_i(\theta^{(k)})$ is a shorthand for $\mu_i(\theta^{(k)}, I, \epsilon)$.

4.2 Categorical distribution for power posterior state swaps

In Friel and Pettitt (2008), a discrete Laplacian probability mass function $p_i(j)$ is suggested for proposing a neighbouring chain j of i . In the current paper, a categorical probability mass function $p_i(j)$ is chosen, in an attempt to disseminate conceptual details in the implementation of power posterior sampling.

Assuming $m + 1$ power posteriors, a neighbouring chain j of i is sampled from the categorical probability mass distribution

$$(4.2) \quad p_i := \mathcal{C}(\alpha_i(0), \alpha_i(1), \dots, \alpha_i(i-1), \alpha_i(i+1), \dots, \alpha_i(m)),$$

$$(4.3) \quad \alpha_i(j) := \frac{\exp(-\beta|j-i|)}{\gamma_i},$$

$$(4.4) \quad \gamma_i := \frac{\exp(-\beta)(2 - \exp(-\beta i) - \exp(-\beta(m-i)))}{1 - \exp(-\beta)},$$

where $i \in \{0, 1, \dots, m\}$ and $j \in \{0, 1, \dots, m\} \setminus \{i\}$. For a derivation of the normalizing constant γ_i appearing in event probability $\alpha_i(j)$, see Appendix A. β is a hyper-parameter typically set to $\beta = 0.5$, a value which makes a jump to $i \pm 1$ roughly three times more likely than a jump to $i \pm 3$ (Friel and Pettitt, 2008).

5. OVERVIEW OF THE MULTILAYER PERCEPTRON MODEL

Both examples in this paper rely on an MLP model, one consisting of nine and one consisting of twenty seven parameters. MCMC inference has been performed on MLPs before, see for example de Freitas (1999); Vehtari, Sarkka and Lampinen (2000). Manifold Langevin Monte Carlo methods and power posteriors have not been used in the context of MLPs. The use of such contemporary geometric and population MCMC methods in neural networks is not an end in itself, it is a means for acquiring an understanding of the challenges arising from the application of MCMC methods on neural networks and for developing benchmark tools for more scalable MCMC algorithms.

MLPs have been chosen as a more tractable class of neural networks. CNNs are the most widely used deep learning models. However, even small CNNs, such as AlexNet (Krizhevsky, Sutskever and Hinton, 2012), SqueezeNet (Iandola et al., 2016), Xception (Chollet, 2017), MobileNet (Howard et al., 2017), ShuffleNet (Zhang et al., 2018b), EffNet (Freeman, Roese-Koerner and Kummert, 2018) or DCTI (Truong, Nguyen and Tran, 2018), have at least two orders of magnitude higher number of parameters, thus amplifying issues of computational complexity, model structure, weight symmetry, prior specification, posterior shape, MCMC convergence and sampling effectiveness.

5.1 The multilayer perceptron

An MLP is a feedforward neural network consisting of an input layer, one or more hidden layers and an output layer (Rosenblatt, 1958; Minsky and Papert, 1988; Hastie, Tibshirani and Friedman, 2016). Let $j \in \{0, 1, \dots, \rho\}$ be an index indicating the layer for some natural number $\rho \geq 2$, where $j = 0$ refers to the input layer, $j = 1, 2, \dots, \rho - 1$ to one of the $\rho - 1$ hidden layers and $j = \rho$ to the output layer. Let κ_j be the number of neurons in layer j and use $\kappa_{(0:\rho)} := \{\kappa_0, \kappa_1, \dots, \kappa_\rho\}$

as a shorthand for the sequence of neuron counts per layer. Under such notation, $\text{MLP}(\kappa_{(0:\rho)})$ refers to an MLP with $\rho - 1$ hidden layers and κ_j neurons at layer j .

An $\text{MLP}(\kappa_{(0:\rho)})$ with $\rho - 1 \geq 1$ hidden layers and κ_j neurons at layer $j \in \{0, 1, \dots, \rho\}$ is defined recursively as

$$(5.1) \quad g^{(j)}(x^{(i)}, \theta_{(1:j)}) := W^{(j)} h^{(j-1)}(x^{(i)}, \theta_{(1:j-1)}) + b^{(j)},$$

$$(5.2) \quad h^{(j)}(x^{(i)}, \theta_{(1:j)}) := \phi^{(j)}(g^{(j)}(x^{(i)}, \theta_{(1:j)})),$$

for $j = 1, 2, \dots, \rho$. A data point $x^{(i)} \in \mathbb{R}^{\kappa_0}$ is used as input $h^{(0)}(x^{(i)}) := x^{(i)}$ to the input layer, yielding the sequence $g^{(1)}(x^{(i)}, \theta_{(1)}) = W^{(1)}x^{(i)} + b^{(1)}$ in the first hidden layer. $W^{(j)}$ and $b^{(j)}$ are the respective weights and biases at layer $j = 1, 2, \dots, \rho$, which together constitute the parameters $\theta_{(j)} := (W^{(j)}, b^{(j)})$ at layer j . $\theta_{(1:j)} := \{\theta_{(1)}, \theta_{(2)}, \dots, \theta_{(j)}\}$ is a shorthand for all weights and biases up to layer j . Functions $\phi^{(j)}(g^{(j)})$, known as *activations*, are applied elementwise to their input $g^{(j)}$.

The default recommendation of activation in neural networks is a rectified linear unit (ReLU), see for instance [Jarrett et al. \(2009\)](#); [Nair and Hinton \(2009\)](#); [Goodfellow, Bengio and Courville \(2016\)](#). If an activation is not present at layer j , then $\phi^{(j)}$ in (5.2) corresponds to the identity function $\phi^{(j)}(g^{(j)}) = g^{(j)}$.

The weight matrix $W^{(j)}$ in (5.1) has κ_j rows and κ_{j-1} columns, while the vector $b^{(j)}$ of biases has length $\kappa^{(j)}$. Concatenating all $\theta_{(j)}$ across hidden and output layers gives a parameter vector $\theta := \theta_{(1:\rho)} \in \mathbb{R}^n$ of length $n := \sum_{j=1}^{\rho} \kappa_j(\kappa_{j-1} + 1)$. To define θ uniquely, the convention to traverse weight matrix elements row-wise is made. Apparently, each of $g^{(j)}$ in (5.1) and $h^{(j)}$ in (5.2) has length κ_j .

The notation $W_{k,l}^{(j)}$ is introduced to point to the (k,l) -the element of weight matrix $W^{(j)}$ at layer j . Analogously, $b_k^{(j)}$ points to the k -th coordinate of bias vector $b^{(j)}$ at layer j .

5.2 Likelihood for binary classification

Consider s samples $(x^{(i)}, y^{(i)})$, $i = 1, 2, \dots, s$, consisting of some input $x^{(i)} \in \mathbb{R}^{\kappa_0}$ and of a binary output $y^{(i)} \in \{0, 1\}$. An $\text{MLP}(\kappa_0, \kappa_1, \dots, \kappa_\rho = 1)$ with a single neuron in its output layer can be used for setting the likelihood function $L(\{y^{(i)}\} | \{x^{(i)}\}, \theta)$ of labels $\{y^{(i)}\} := \{y^{(i)} : i\}$ given the input $\{x^{(i)}\} := \{x^{(i)} : i\}$ and MLP parameters θ .

Firstly, the *sigmoid activation function* $\phi^{(\rho)}(g^{(\rho)}) = 1/(1 + \exp(-g^{(\rho)}))$ is applied at the output layer of the MLP. So, the *event probabilities* $\Pr(y^{(i)} = 1 | x^{(i)}, \theta)$ are set to

(5.3)

$$\Pr(y^{(i)} = 1 | x^{(i)}, \theta) = h^{(\rho)}(x^{(i)}, \theta) = \phi^{(\rho)}(g^{(\rho)}(x^{(i)}, \theta)) = \frac{1}{1 + \exp(-g^{(\rho)}(x^{(i)}, \theta))}.$$

Assuming that the labels are outcomes of s independent draws from Bernoulli probability mass functions with event probabilities given by (5.3), the likelihood becomes

$$(5.4) \quad L(\{y^{(i)}\} | \{x^{(i)}\}, \theta) = \prod_{i=1}^s (h^{(\rho)}(x^{(i)}, \theta))^{y^{(i)}} (1 - h^{(\rho)}(x^{(i)}, \theta))^{1-y^{(i)}}.$$

The log-likelihood $\ell(\{y^{(i)}\}|\{x^{(i)}\}, \theta) := \log(L(\{y^{(i)}\}|\{x^{(i)}\}, \theta))$ follows as (5.5)

$$\ell(\{y^{(i)}\}|\{x^{(i)}\}, \theta) = \sum_{i=1}^s y^{(i)} \log(h^{(\rho)}(x^{(i)}, \theta)) + (1 - y^{(i)}) \log(1 - h^{(\rho)}(x^{(i)}, \theta)).$$

The negative value of log-likelihood (5.5) is known as the *binary cross entropy* (BCE). To infer the parameters θ of $\text{MLP}(\kappa_0, \kappa_1, \dots, \kappa_\rho = 1)$, the binary cross entropy or a different loss function is minimized using stochastic optimization methods, such as stochastic gradient descent.

5.3 Likelihood for multiclass classification

Consider some output variable $y_i \in \{1, 2, \dots, \kappa_\rho\}$, which can take $\kappa_\rho \geq 2$ values. An $\text{MLP}(\kappa_0, \kappa_1, \dots, \kappa_\rho)$ with κ_ρ neurons in its output layers can be used for setting the likelihood function $L(\{y^{(i)}\}|\{x^{(i)}\}, \theta)$.

A *softmax activation function* $\phi^{(\rho)}(g^{(\rho)}) = \exp(g^{(\rho)}) / \sum_{k=1}^{\kappa_\rho} \exp(g_k^{(\rho)})$ is applied at the output layer of the MLP, where $g_k^{(\rho)}$ denotes the k -th coordinate of the κ_ρ -length vector $g^{(\rho)}$. Thus, the event probabilities $\Pr(y^{(i)} = k|x^{(i)}, \theta)$ are (5.6)

$$\Pr(y^{(i)} = k|x^{(i)}, \theta) = h_k^{(\rho)}(x^{(i)}, \theta) = \phi^{(\rho)}(g_k^{(\rho)}(x^{(i)}, \theta)) = \frac{\exp(g_k^{(\rho)}(x^{(i)}, \theta))}{\sum_{k=1}^{\kappa_\rho} \exp(g_k^{(\rho)}(x^{(i)}, \theta))}.$$

It is assumed that the labels are outcomes of s independent draws from categorical probability mass functions with event probabilities given by (5.6), so the likelihood is

$$(5.7) \quad L(\{y^{(i)}\}|\{x^{(i)}\}, \theta) = \prod_{i=1}^s \prod_{k=1}^{\kappa_\rho} (h_k^{(\rho)}(x^{(i)}, \theta))^{\mathbb{1}(y^{(i)}=k)}.$$

$\mathbb{1}$ denotes the indicator function, that is $\mathbb{1}(y^{(i)} = k) = 1$ if $y^{(i)} = k$, and $\mathbb{1}(y^{(i)} = k) = 0$ otherwise. The log-likelihood follows as

$$(5.8) \quad \ell(\{y^{(i)}\}|\{x^{(i)}\}, \theta) = \sum_{i=1}^s \sum_{k=1}^{\kappa_\rho} \mathbb{1}(y^{(i)} = k) \log(h_k^{(\rho)}(x^{(i)}, \theta)).$$

The negative value of log-likelihood (5.8) is also known as *cross entropy*, and it is used as loss function for stochastic optimization in multiclass classification MLPs.

It is noted that for $\kappa_\rho = 2$, an $\text{MLP}(\kappa_0, \kappa_1, \dots, \kappa_\rho = 2)$ with two neurons at the output layer, event probabilities given by softmax activation (5.6) and log-likelihood (5.8) can be used for binary classification. Such a formulation provides an alternative to an $\text{MLP}(\kappa_0, \kappa_1, \dots, \kappa_\rho = 1)$ with one neuron at the output layer, event probabilities given by sigmoid activation (5.3) and log-likelihood (5.5). The difference between the two MLP models is the parameterization of event probabilities, since a categorical distribution with $\kappa_\rho = 2$ levels otherwise coincides with a Bernoulli distribution.

6. EXAMPLES

Two examples of MLPs are used for showcasing challenges in MCMC inference for neural networks. An $\text{MLP}(2, 2, 1)$ and an $\text{MLP}(4, 3, 3)$ are used in the context

of a binary and of a multiclass classification example, respectively. The focus of this paper is on inferring the parameter posterior $p(\theta|\{x^{(i)}\}, \{y^{(i)}\})$ of an MLP using MCMC, rather than inferring the predictive posterior.

The unnormalized parameter posterior of an MLP with binary or categorical output is $p(\theta|\{x^{(i)}\}, \{y^{(i)}\}) \propto L(\{y^{(i)}\}|\{x^{(i)}\}, \theta)\pi(\theta)$, where the likelihood function $L(\{y^{(i)}\}|\{x^{(i)}\}, \theta)$ corresponds to (5.4) or (5.7). $\pi(\theta)$ is the prior of MLP parameters. In the examples, the unnormalized log-target density sampled via MCMC is $\ell(\{y^{(i)}\}|\{x^{(i)}\}, \theta) + \log \pi(\theta)$, where the log-likelihood $\ell(\{y^{(i)}\}|\{x^{(i)}\}, \theta)$ for binary or multiclass classification corresponds to (5.5) or (5.8).

Ten and four Markov chains are simulated per MCMC sampler for the respective MLP(2, 2, 1) and MLP(4, 3, 3) examples to compute PSRF and ESS. The increased computational cost incurred by power posterior simulations, which are run only for the MLP(4, 3, 3) example, is the reason for simulating fewer chains per sampler for MLP(4, 3, 3) in comparison to MLP(2, 2, 1). 110000 iterations are run per chain, 10000 of which are discarded as burn-in. Samplers are tuned empirically to attain acceptance rates recommended in the literature (see section 3.1). The SofAbs metric (Betancourt, 2013) is used for approximating the Hessian of the target density in all simulations involving SMMALA and power posteriors with SMMALA chains.

Maximum PSRF values across all n parameters are reported for each MLP. For ease of exposition, the suppressed notation split- \hat{R} and folded-split- \hat{R} is henceforth preferred over $\max_{j=1,\dots,n} \{\text{split-}\hat{R}_j\}$ and $\max_{j=1,\dots,n} \{\text{folded-split-}\hat{R}_j\}$. In a similar fashion, the mean ESS of each parameter is computed across simulated chains, and subsequently the minimum, median and maximum ESS is taken across parameters. The shorter notation \hat{S}_{\min} , \hat{S}_{median} and \hat{S}_{\max} is preferred over $\min_{j=1,\dots,n} \{\hat{S}_j\}$, $\text{median}_{j=1,\dots,n} \{\hat{S}_j\}$ and $\max_{j=1,\dots,n} \{\hat{S}_j\}$.

The mean acceptance rate across simulated chains is reported for each sampler. In the case of power posteriors, if the state of a chain corresponding to the target density $p^{t_m}(\theta|y) = p(\theta|y)$ changes after taking the within and between-chain sub-steps of an MCMC step, then the state counts as accepted, otherwise it counts as rejected.

Gaussian and Beta kernels are adopted for evaluating kernel density estimators (KDEs) of densities with unbounded and bounded support, respectively. KDEs relying on Beta kernels are based on the work of (Chen, 1999).

6.1 MLP for exclusive-or data

The XOR function $f : \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$ returns 1 if exactly one of its binary input values is equal to 1, otherwise it returns 0. The four $s = 4$ data points defining XOR are $(x^{(1)}, y^{(1)}) = ((0, 0), 0)$, $(x^{(2)}, y^{(2)}) = ((0, 1), 1)$, $(x^{(3)}, y^{(3)}) = ((1, 0), 1)$ and $(x^{(4)}, y^{(4)}) = ((1, 1), 0)$.

A perceptron without a hidden layer can not learn the XOR function (Minsky and Papert, 1988). On the other hand, an MLP(2, 2, 1) with a single hidden layer of two neurons can learn the XOR function (Goodfellow, Bengio and Courville, 2016).

An MLP(2, 2, 1) has a parameter vector θ of length $n = 9$, since $W^{(1)}, b^{(1)}, W^{(2)}$ and $b^{(2)}$ have respective dimensions $2 \cdot 2, 2 \cdot 1, 2 \cdot 1$ and $1 \cdot 1$. MCMC is run to learn the posterior of θ given the four XOR data points $\{(x^{(i)}, y^{(i)}) : i = 1, 2, 3, 4\}$. The sigmoid function is used as activation $\phi^{(1)}$ on the hidden layer, since it achieves

Prior	\hat{R}		\hat{S}			Acceptance
	Split- \hat{R}	Folded-split- \hat{R}	Min	Median	Max	
$\mathcal{N}(10, 3I)$	1.000082	1.000047	28982	29521	29818	57.67
$\mathcal{N}(0, 3I)$	1.000334	1.000151	9878	10313	14036	57.38
$\mathcal{N}(0, 10I)$	1.000787	1.000369	2751	3094	4031	58.12
$\mathcal{N}(0, 100I)$	1.014097	1.004643	199	264	323	57.08
$\mathcal{U}(-20, 20)$	1.023772	1.015454	92	121	134	57.75

TABLE 1

MCMC diagnostics of MALA simulations for an MLP(2, 2, 1) model fitted to XOR. Every row corresponds to MALA simulations with a different prior. Ten chains are simulated for each prior. PSRF, ESS and acceptance rates averaged across the ten chains are reported. See section 3.2 and beginning of section 6 for details on how these diagnostics are computed.

higher acceptance rate in the XOR example than a ReLU, according to MCMC pilot runs.

6.1.1 Passing convergence diagnostics Firstly, ten chains are simulated via MH with prior $\pi(\theta) \sim \mathcal{N}(0, 3I)$. The values of split- $\hat{R} = 1.001071$ and folded-split- $\hat{R} = 1.000509$ are below the upper PSRF threshold of 1.05. Moreover, the ESS summaries across the ten chains are $\hat{S}_{\min} = 2600$, $\hat{S}_{\text{mean}} = 2689$ and $\hat{S}_{\max} = 3616$. The minimum ESS value of $\hat{S}_{\min} = 2600$ per chain is above the lower ESS threshold of 100. Thus, the PSRF and ESS diagnostics do not detect lack of convergence. Moreover, a mean acceptance rate of 23.78% across the ten chains and the trace plot of parameter $\theta_4 = W_{2,2}^{(1)}$ shown in Figure 6a of Appendix C do not show signs of slow mixing.

Next, ten chains are simulated via MALA with the same prior $\pi(\theta) \sim \mathcal{N}(0, 3I)$. As seen in Table 1, MALA with $\mathcal{N}(0, 3I)$ prior does not fail the PSRF and ESS diagnostics either. Moreover, MALA has a minimum ESS value of $\hat{S}_{\min} = 9878$, higher than the minimum ESS of 2600 for MH. This higher sampling effectiveness of MALA is possibly due to the use of gradient information in MALA.

6.1.2 Convergence with a weakly informative prior MCMC diagnostics for the chains generated via MH and MALA with $\mathcal{N}(0, 3I)$ prior give no reason to doubt that the chains are representative of the parameter posterior of the MLP(2, 2, 1) under consideration.

Nevertheless, an attempt follows to evaluate prior effects. To this end, MALA simulations are run with more weakly informative priors, such as $\mathcal{N}(0, 10I)$ and $\mathcal{N}(0, 100I)$. The term ‘weakly informative’ is not used rigorously here from an information-theoretic perspective of objective priors, but rather refers to more flat priors.

According to Table 1, split- \hat{R} and folded-split- \hat{R} for $\mathcal{N}(0, 10I)$ and $\mathcal{N}(0, 100I)$ priors are below the upper threshold of 1.05, although PSRF increases with increasing prior variance. $\hat{S}_{\min} = 2751$ with $\mathcal{N}(0, 10I)$ prior and $\hat{S}_{\min} = 199$ with $\mathcal{N}(0, 100I)$ prior are higher than the lower ESS threshold of 100. Sampling effectiveness drops with increasing prior variance. MALA does not fail the PSRF and ESS diagnostics with any of $\mathcal{N}(0, 3I)$, $\mathcal{N}(0, 10I)$ and $\mathcal{N}(0, 100I)$ priors.

Figure 6h shows that the running mean of a single chain realization via MALA for weight $\theta_4 = W_{2,2}^{(1)}$ stabilizes with $\mathcal{N}(0, 3I)$ or $\mathcal{N}(0, 10I)$ prior, but not with $\mathcal{N}(0, 100I)$ prior. The loss of sampling effectiveness of MALA with increasing normal prior variance is visible in the autocorrelations of Figure 6g (the plotted

autocorrelation lines correspond to the chains visualized in Figures 6b-6f).

The trace plots of Figures 6c, 6d and 6e provide an explanation for the reduced sampling effectiveness of MALA with increasing normal prior variance (black straight horizontal lines represent Monte Carlo means for the respective traces). As the prior variance increases, MALA explores wider regions in the support of θ_4 , and mixing becomes slower. These three trace plots demonstrate the waste of computational time during MCMC due to weight symmetries (Nalisnick, 2018). Equivalent marginal posterior modes of θ_4 are scattered over the real line, so MCMC mixing worsens and additional computational effort is required to explore the marginal posterior landscape.

As a note of caution, consider the effects of using a prior with arbitrary mean and relatively small variance. For instance, MALA simulations with $\mathcal{N}(10, 3I)$ prior do not fail MCMC diagnostics (see relevant PSRF and ESS values in Table 1) and attain a minimum ESS of 28982 per chain. Such a high ESS is spurious and does not represent high sampling effectiveness. Figure 6c shows the trace plots of two chains for parameter θ_4 realized via MALA with priors $\mathcal{N}(10, 3I)$ and $\mathcal{N}(0, 3I)$. Both chains are entrapped in local modes and therefore neither of the corresponding priors help explore the space of parameter θ_4 effectively. Priors $\mathcal{N}(10, 3I)$ and $\mathcal{N}(0, 3I)$ yield $\hat{S}_{\min} = 28982$ and $\hat{S}_{\min} = 9878$, so it seems that prior $\mathcal{N}(10, 3I)$ leads to more effective sampling. However, Figure 3 indicates otherwise, since prior $\mathcal{N}(10, 3I)$ seems to prohibit MALA from capturing the posterior covariance among parameters, in contrast to prior $\mathcal{N}(0, 3I)$.

Different priors for the parameters of the MLP(2, 2, 1) under consideration yield chains that get entrapped in different local modes or explore parameter regions of varying magnitude (Figures 6c-6e), and yet convergence diagnostics do not fail in any of the examined scenarios (Table 1). Thus, prior specification can impact the neural network parameter posterior inferred via MCMC (Lee, 2004) and MCMC diagnostics can fail to detect this issue.

6.1.3 Parameter symmetries explored via optimization and MCMC To explore parameter symmetries and their role in MCMC, gradient descent (GD) is run until 10000 optimization solutions are obtained for the MLP(2, 2, 1) fitted to XOR. The MLP(2, 2, 1) used for optimization applies a sigmoid activation $\phi^{(1)}$ to the hidden layer and minimizes the BCE loss, that is the negative log-likelihood (5.5).

Since only four data points make up XOR, the data are not split into training and test set. For each optimization run, GD learns the nine MLP(2, 2, 1) parameters given the four XOR data points, and the learnt parameters are then used for predicting the XOR output given the XOR input for the same four XOR data points.

The event probability (5.3) for each of the four XOR data points is computed by a sigmoid activation $\phi^{(2)}$ at the output layer of MLP(2, 2, 1). The classification threshold is set to be $\Pr(y^{(i)} = 1|x^{(i)}, \theta) > 0.9$. An optimization solution is accepted if it achieves 100% prediction accuracy, that is if it predicts correctly the output of all four XOR data points.

Initial parameter values for optimization are sampled from a prior. Two priors $\pi(\theta)$ are employed, namely $\pi(\theta) \sim \mathcal{N}(0, 10I)$ and $\pi(\theta) \sim \mathcal{U}(-20, 20)$. The notation $\pi(\theta) \sim \mathcal{U}(-20, 20)$ refers to a prior of IID parameter coordinates, with each coordinate admitting uniform $\mathcal{U}(-20, 20)$ probability density function. 10000 optimization solutions are acquired for each of the two priors. 2000 epochs are run

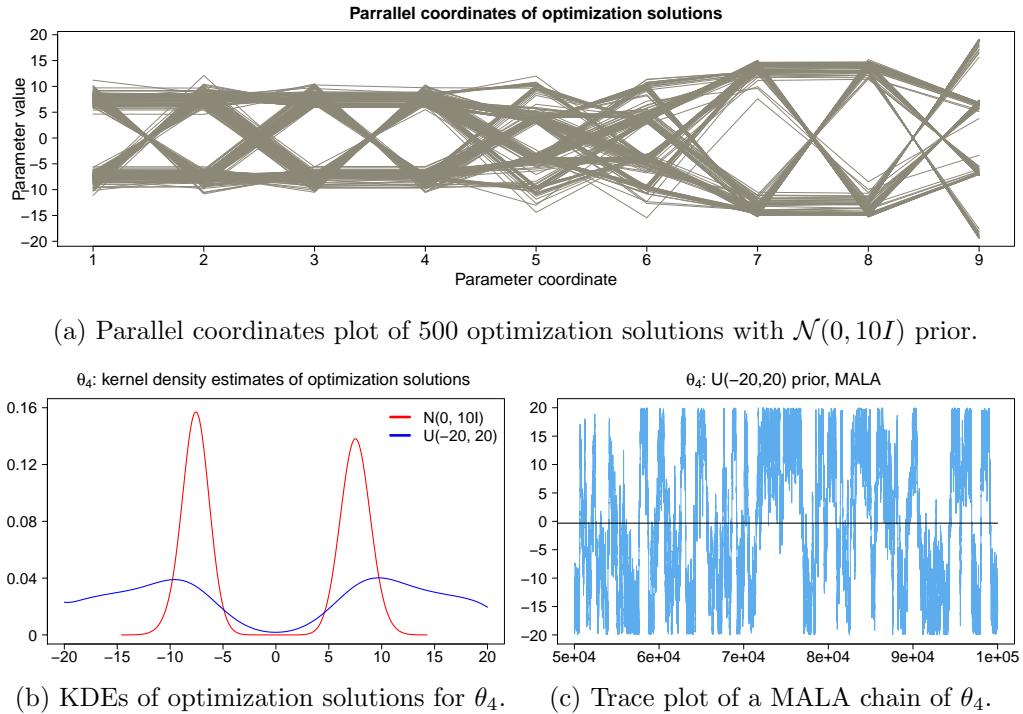


Fig 1: Exploration of parameter symmetries for an $\text{MLP}(2, 2, 1)$ model fitted to XOR. Gradient descent is run with initial values drawn from $\mathcal{N}(0, 10I)$ and $\mathcal{U}(-20, 20)$ priors. 10000 optimization solutions per prior are obtained. Parameter symmetries are visualized via a parallel coordinates plot of 500 solutions under the $\mathcal{N}(0, 10I)$ prior. KDEs of optimization solutions for parameter $\theta_4 = W_{2,2}^{(1)}$ under these two priors, and a trace plot of a Markov chain of parameter θ_4 generated by MALA with a $\mathcal{U}(-20, 20)$ prior are displayed. The horizontal black line in the trace plot represents the mean of the chain.

for each of the 10000 optimizations, with learning rate set to one.

Figure 1a displays a parallel coordinates plot (Inselberg and Dimsdale, 1990) of 500 out of the 10000 optimization solutions with $\mathcal{N}(0, 10I)$ prior. Every line plotted along the horizontal axis connects the nine parameter coordinates of a single optimization solution. The parallel coordinates plot exhibits symmetries along the horizontal axis $\theta = 0$. Folding Figure 1a along the horizontal axis $\theta = 0$, visualizes permutations that leave the output of the $\text{MLP}(2, 2, 1)$ unchanged.

Figure 1b shows the KDEs of GD solutions attained by sampling the initial parameter values for GD from $\mathcal{N}(0, 10I)$ or from $\mathcal{U}(-20, 20)$. Each KDE in Figure 1b can be interpreted as a posterior of ‘good’ optimization solutions. In other words, each KDE approximates a posterior of parameters that lead to 100% prediction accuracy on the four XOR data points. Those GD and MCMC simulations that use the same prior do not provide estimates of the same posterior for the $\text{MLP}(2, 2, 1)$ parameters, since the likelihood functions for GD and MCMC differ. The BCE loss minimized by GD coincides with the negative log-likelihood (5.5) used in MCMC. However, a GD solution obtained by minimizing the BCE loss is accepted only if it meets the requirement of 100% prediction accuracy on the four XOR data points. Adding this extra requirement of predictive quality on

top of the BCE loss leads to a different loss function, and therefore to a different underlying likelihood.

As seen in Figure 1a, most of the lines pass through the regions of values $[-10, -5]$ and $[5, 10]$ of parameter θ_4 . Figure 1b shows that the posterior KDE associated with $\mathcal{N}(0, 10I)$ prior is bimodal with modes in $[-10, -5]$ and $[5, 10]$.

Both KDEs in Figure 1b have very low probability mass in the region $[-5, 5]$ in the vicinity of $\theta = 0$, are bimodal and have one modal peak in $[-10, -5]$ and one in $[5, 10]$. On the other hand, the KDE associated with $\mathcal{U}(-20, 20)$ prior distributes probability mass widely in $[-20, 5]$ and $[5, 20]$, whereas the more highly peaked KDE associated with $\mathcal{N}(0, 10I)$ prior concentrates most of its probability mass in $[-10, -5]$ and in $[5, 10]$.

Figure 1c shows a Markov chain generated by MALA with $\mathcal{U}(-20, 20)$ prior. The chain switches between two regions, located towards the bounds of the support $[-20, 20]$, and explores much less the region $[-5, 5]$ in the vicinity of $\theta = 0$. Thus, the chain simulated via MALA with $\mathcal{U}(-20, 20)$ prior explores the two regions away from the center of the support, which correspond to the two modal peaks $[-20, 5]$ and $[5, 20]$ of the posterior KDE of GD solutions with $\mathcal{U}(-20, 20)$ prior (see Figures 1c and 1b in relation to one another).

Figure 2e shows the posterior KDE of a chain simulated using MALA with $\mathcal{U}(-20, 20)$ prior, where the KDE employs a Beta kernel. The posterior KDE of the MALA chain with flat $\mathcal{U}(-20, 20)$ prior (Figure 2e) is bimodal and distributes probability mass towards the two modes of the posterior KDE of GD simulations with $\mathcal{U}(-20, 20)$ prior (Figure 1b). On the contrary, the posterior Gaussian kernel KDEs of chains simulated using MALA with priors $\mathcal{N}(0, 3I)$, $\mathcal{N}(0, 10I)$ and $\mathcal{N}(0, 100I)$ (Figures 2b, 2c and 2d) have their mass concentrated in $\theta = 0$. In summary, MCMC and optimization-based posteriors for the MLP(2, 2, 1) parameters are in closer agreement under a flat prior. Another interpretation of the simulations is that parameter posteriors arising from MCMC simulations with a flat prior shift towards posteriors of optimization solutions of high predictive accuracy. As an empirical conclusion, prior specification not only affects the MCMC-based parameter posterior of the MLP(2, 2, 1), but it also has an effect on whether parameter regions of high predictive accuracy dominate the MCMC-based parameter posterior.

6.1.4 Lack of convergence with a flat prior The ten chains simulated via MALA with $\mathcal{U}(-20, 20)$ prior do not fail PSRF convergence diagnostics, ($\text{split-}\hat{R} = 1.023772 < 1.05$ and $\text{folded-split-}\hat{R} = 1.015454 < 1.05$, see Table 1), but they fail ESS diagnostics ($\hat{S}_{\min} = 92 < 100$, see Table 1). So, it is likely that MALA with a truncated flat prior converges, but samples ineffectively. Figure 6g corroborates the lack of sampling effectiveness of MALA with $\mathcal{U}(-20, 20)$ prior.

PSRF diagnostics and a comparison between GD and MALA simulations indicate that a truncated flat prior is likely to be helping MCMC uncover clues about the shape of the MLP(2, 2, 1) parameter posterior despite the lack of effective sampling. This is in line with the remark of (Lee, 2005) about the help that truncated flat priors can provide towards learning the parameter posterior of a neural network in some cases. Nonetheless, PSRF and ESS diagnostics are employed jointly to reduce the risk of false diagnosis about convergence (Cowles and Carlin, 1996; Vehtari et al., 2019). Due to the minimum ESS value of $\hat{S}_{\min} = 92 < 100$, it is thus concluded that MALA simulations with $\mathcal{U}(-20, 20)$ prior have not converged.

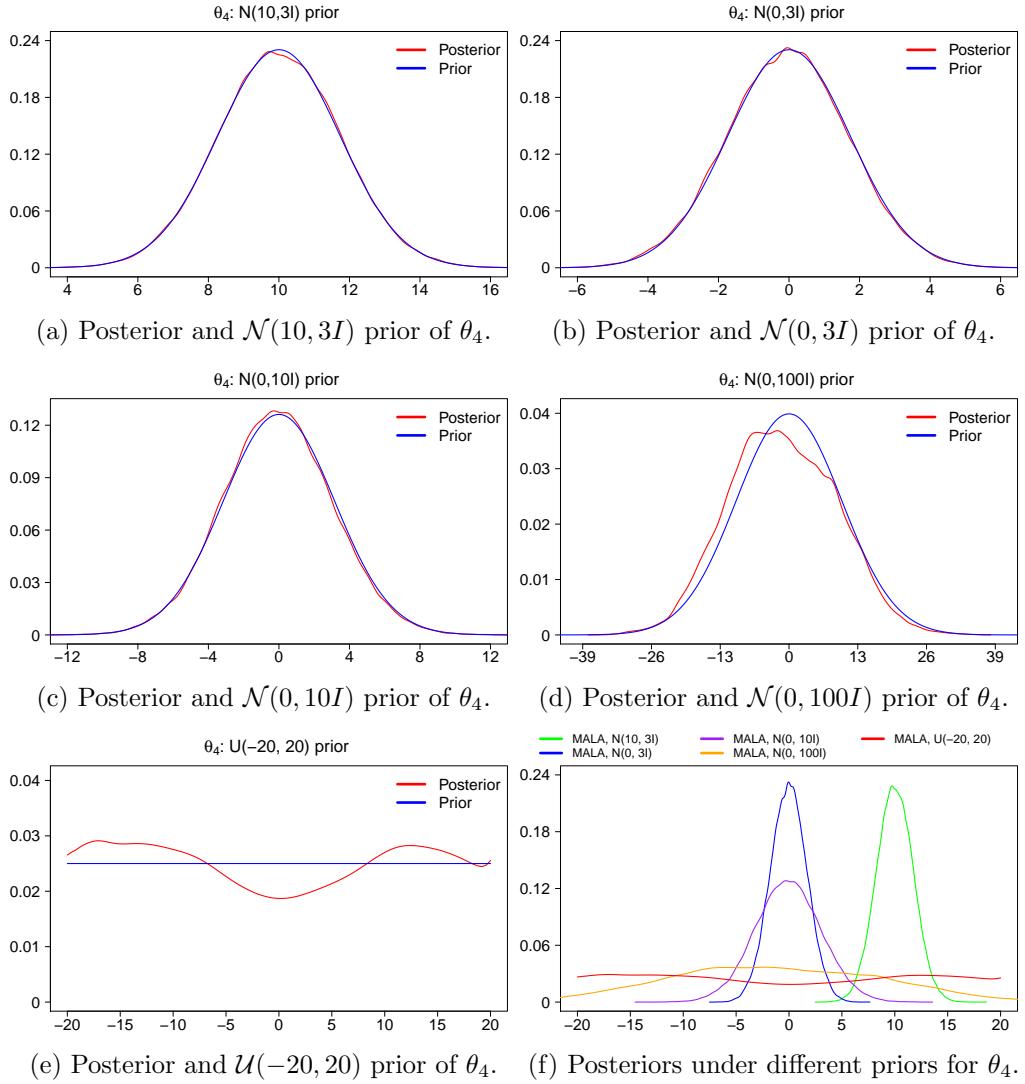


Fig 2: Posterior KDEs based on MALA chains of parameter $\theta_4 = W_{2,2}^{(1)}$ of an $\text{MLP}(2, 2, 1)$ model fitted to XOR. MALA chains have been simulated for the five different priors mentioned in Table 1. Each posterior KDE is plotted against its corresponding prior. Moreover, all posterior KDEs are overlaid in a single plot.

MCMC simulations with a truncated flat prior, which are likely to provide some information about the parameter posterior of the $\text{MLP}(2, 2, 1)$, fail convergence diagnostics, whereas MCMC simulations with a normal prior, which are dominated by the prior without learning the parameter posterior of the $\text{MLP}(2, 2, 1)$, do not fail convergence diagnostics. In summary, weight symmetries give rise to multimodal parameter posteriors and thus raise convergence challenges, prior misspecification can steer away from the valid parameter posterior of a neural network, and MCMC convergence diagnostics can give the green light without detecting prior misspecification.

6.1.5 Comparison of marginal parameter posteriors for different priors Figure 2 displays marginal posterior KDEs and associated priors for single-chain realizations of parameter $\theta_4 = W_{2,2}^{(1)}$ generated by MALA. Each prior nearly coin-

cides with the resulting marginal posterior KDE of θ_4 , apart from the $\mathcal{U}(-20, 20)$ flat prior, which results in a non-flat posterior KDE with two very mild modal peaks. So, posteriors do not learn from the underlying $\text{MLP}(2, 2, 1)$ likelihood, and instead priors dominate parameter inference. Figure 2f shows that the overlaid marginal posterior KDEs of θ_4 arising from different priors differ from one another. This is a sign of spurious parameter inference, since the goal is to ensure a valid parameter posterior not arbitrarily influenced by the choice of prior.

Figure 7 of Appendix D portrays a similar picture for weight $\theta_8 = W_{1,2}^{(2)}$. All priors apart from $\mathcal{U}(-20, 20)$ and $\mathcal{N}(10, 3I)$ tend to coincide with the resulting marginal posterior KDEs of θ_8 . Thus, priors outweigh $\text{MLP}(2, 2, 1)$ likelihood effects, leading to spurious parameter inference. As seen in Figure 7f, the marginal posterior KDEs of θ_8 arising from different priors differ from one another.

Table 3 shows the Monte Carlo means of all nine $\text{MLP}(2, 2, 1)$ parameters for MALA simulations under different priors. Ten chains are simulated via MALA for each prior, the Monte Carlo mean of each chain is computed, and the mean of the ten Monte Carlo means is tabulated per parameter. Under the zero-centered priors $\mathcal{N}(0, 3I)$, $\mathcal{N}(0, 10I)$ and $\mathcal{N}(0, 100I)$, all Monte Carlo means are in the vicinity of zero. Such numerics agree with Figures 2 and 7 in that the marginal posteriors of the parameters are dominated by the choice of prior.

Under the $\mathcal{U}(-20, 20)$ truncated flat prior, most of the nine Monte Carlo means are also in the vicinity of the prior mean of zero (see Table 3). However, these Monte Carlo means do not correspond to unimodal marginal posteriors with mode close to zero. For example Figure 2e, shows that the marginal posterior KDE of parameter θ_4 has two modal peaks away from zero and roughly symmetric around zero, thus cancelling each other out and resulting in a marginal posterior mean close to zero (-0.2238 as shown in Table 3).

6.1.6 Comparison of parameter posterior covariance for different priors Figure 3 shows the empirical posterior covariance matrix among the nine $\text{MLP}(2, 2, 1)$ parameters for six different MCMC settings. The six settings comprise MH with $\mathcal{N}(0, 3I)$ prior, and MALA with $\mathcal{N}(0, 3I)$, $\mathcal{N}(0, 10I)$, $\mathcal{N}(0, 100I)$, $\mathcal{U}(-20, 20)$ and $\mathcal{N}(10, 3I)$ priors. Ten chains are simulated per setting, and the ten chains per setting are combined to derive the empirical posterior covariance matrix associated with the setting (Plummer et al., 2019).

Figure 3 suggests that five out of the six MCMC simulations uncover similar posterior covariance structure. MALA with $\mathcal{N}(10, 3I)$ prior disagrees with the other five simulations and shows pairwise independence across the nine parameters. It is highly unlikely that the independence suggested by the MALA simulation with $\mathcal{N}(10, 3I)$ prior is representative of the posterior covariance structure, not only because the majority vote is by large in favour of the other five MCMC simulations, but also because the parameters of a hierarchical model, such as the $\text{MLP}(2, 2, 1)$, are likely to exhibit some correlation.

Since the MALA simulation with $\mathcal{N}(10, 3I)$ prior seems to fail to capture the posterior covariance structure, the shifting away of the marginal posterior KDE of parameter θ_8 from the corresponding $\mathcal{N}(10, 3I)$ prior in Figure 7a may not be representative of the marginal posterior of θ_8 .

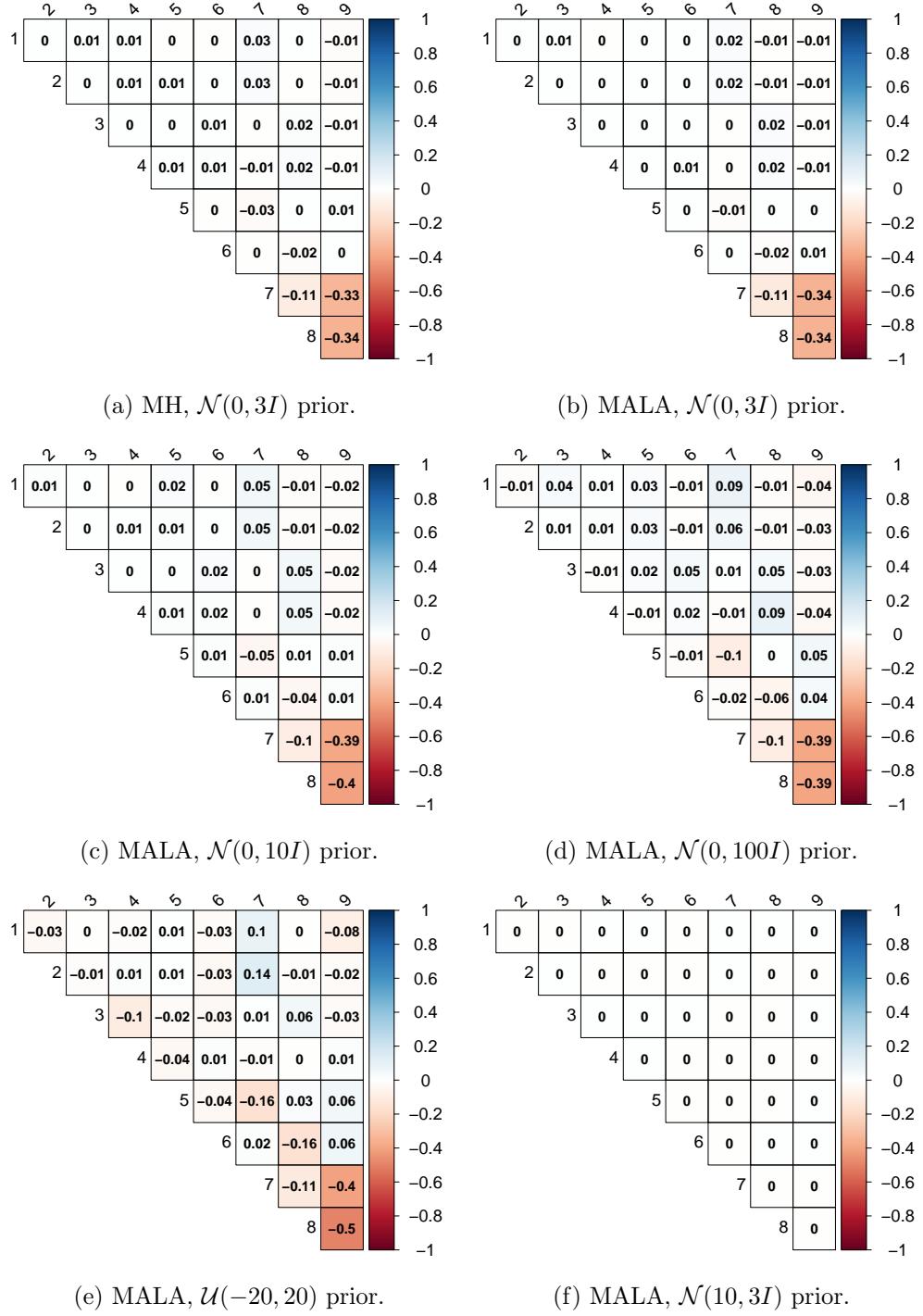


Fig 3: Empirical posterior covariances of chains generated by MH and MALA for different priors. Covariances are rounded off to two decimal digits.

Moreover, the $\mathcal{N}(10, 3I)$ prior leads to the highest ESS among the five priors used with MALA for sampling the MLP(2, 2, 1) parameters, according to Table 1. In the MLP(2, 2, 1) example, MALA with $\mathcal{N}(10, 3I)$ prior seems to have the highest sampling effectiveness (Table 1), but it seems the worst sampling scheme

in terms of capturing the posterior covariance (Figure 3f). This can be explained by the fact that ESS depends on the Monte Carlo variance of a Markov chain by definition, but not on the covariance between coordinates of the chain. Thus, it is possible for ESS to have high value indicating small Monte Carlo variance, while failing to detect that the underlying Markov chain has not captured the covariance of the target density. So, combining MCMC diagnostics reduces the risk of failing to detect lack of convergence (Cowles and Carlin, 1996). More specifically, using both PSRF and ESS is a practical guideline reported in the literature (Vehtari et al., 2019). The current MLP(2, 2, 1) example points out the need to develop a diagnostic for assessing the extent to which a Markov chain approximates the covariance of the target density. PSRF, ESS and such a diagnostic for assessing sampling effectiveness with respect to Monte Carlo covariance could then be used as a triplet of MCMC diagnostics.

Coming back to Figure 3 and setting aside the $\mathcal{N}(10, 3I)$ prior, all five other priors uncover similar empirical posterior covariance via MH and MALA simulations. More specifically, pairwise negative correlation is detected between weights $\theta_7 = W_{1,1}^{(2)}$, $\theta_8 = W_{1,2}^{(2)}$ and bias $\theta_9 = b_1^{(2)}$ at the second layer. Furthermore, both $\mathcal{N}(0, 100I)$ and $\mathcal{U}(-20, 20)$ priors with MALA detect weak positive correlation between θ_1 and θ_7 , θ_2 and θ_7 , and weak negative correlation between θ_5 and θ_7 . This suggests that using weakly informative or non-informative priors in neural networks can help capture the covariance structure of the parameter posterior.

Despite not having approximated the marginal posteriors of the MLP(2, 2, 1) parameters (Figures 2 and 7), the parameter posterior covariance structure is likely to have been uncovered by the MCMC simulations (Figure 3). Thus, existing MCMC algorithms used in small dimensional data and parameter spaces can help build benchmarks based on small scale MLPs to assess the capacity of scalable inference methods to represent the parameter covariance structure of neural networks.

6.2 MLP for Iris data

The *Iris flower data set* consists of fifty samples from each of three species of Iris flowers, namely Iris setosa, Iris virginica and Iris versicolor (Anderson, 1936; Fisher, 1936). The species $y^{(i)} \in \{0, 1\}^3$, expressed in one hot encoding form, and four features $x^{(i)} \in (0, \infty)^4$, that is the length and width of the sepals and petals, are available for each sample $i = 1, 2, \dots, 150$.

In this example, an MLP(4, 3, 3) is fitted to the Iris data. The parameter vector θ of MLP(4, 3, 3) has length $n = 27$, since $W^{(1)}, b^{(1)}, W^{(2)}$ and $b^{(2)}$ have respective dimensions $3 \cdot 4, 3 \cdot 1, 3 \cdot 3$ and $3 \cdot 1$. This example focuses on assessing the capacity of MCMC to learn the posterior of θ given the Iris data points $\{(x^{(i)}, y^{(i)}) : i = 1, 2, \dots, 150\}$, and not on solving the species prediction task. The sigmoid function is preferred over ReLU as activation $\phi^{(1)}$ on the hidden layer, since MCMC pilot runs indicate that ReLU drops the MCMC acceptance rate to zero when fitting an MLP(4, 3, 3) to the Iris data.

6.2.1 Comparison of effectiveness of MCMC samplers MALA and SMMALA sampling, as well as power posterior sampling with MALA chains (P-MALA) and power posterior sampling with SMMALA chains (P-SMMALA) are compared under the same $\mathcal{N}(0, 3I)$ prior in terms of their convergence and their effectiveness in sampling the MLP(4, 3, 3) parameters. Comparisons are made on the basis of

a single prior due to the high computational cost of power posterior sampling. It can take nearly up to a day to complete a P-SMMALA simulation when fitting an MLP(4, 3, 3) model to the Iris data (see section 6.4 for more details on hardware specifications and runtimes), and multiple P-SMMALA simulations are required for PSRF and ESS computations.

Each of P-MALA and P-SMMALA sampling schemes use ten power posteriors. The temperature is set to $t_i = 1$ for each $i \in \{0, 1, \dots, 9\}$. Thereby, each power posterior $p^{t_i}(\theta | \{x^{(k)}\}, \{y^{(k)}\})$ corresponds to the parameter posterior $p(\theta | \{x^{(k)}\}, \{y^{(k)}\})$. Such a constant temperature schedule has been used before in the literature of population MCMC algorithms, see for instance [Braak \(2006\)](#). Empirical tuning of P-MALA and P-SMMALA for this example has shown better mixing with the constant temperature schedule. It is speculated that better mixing with non-tempered power posteriors ($t_i = 1$) in this example is due to the high number of local proximal modes in the parameter posterior of the MLP(4, 3, 3) fitted to the Iris data.

Four chains are simulated per sampler to compute PSRF and ESS (Table 2). None of the four samplers have PSRF lower than the upper threshold of 1.05, therefore all of them fail to converge. However, it is noted that P-SMMALA has split- $\hat{R} = 1.050341$ and folded-split- $\hat{R} = 1.066138$, both of which are below the cut-off of 1.1 suggested by [Gelman et al. \(2004\)](#). In other words, P-SMMALA reduces PSRF in comparison to the other three samplers, and deciding on its convergence depends on the choice of PSRF threshold. Letting P-SMMALA run for more iterations to try to further reduce PSRF is an option. However, such an option incurs high computational cost and poses numerical challenges associated with the evaluation of the Hessian of the target density. The SoftAbs metric used for approximating the Hessian is more likely to fail for higher number of iterations due to numerical runtime errors related to Cholesky decomposition. Such numerical issues are not insurmountable, since there are potential engineering workarounds, such as storing P-SMMALA iterations on file and restoring them to resume simulation upon numerical errors.

As seen in Table 2, the median ESS \hat{S}_{median} for MALA, SMMALA, P-MALA and P-SMMALA corresponds to 21, 51, 429 and 464. Some observations are made in relation to these \hat{S}_{median} values. Manifold MCMC increases sampling effectiveness (SMMALA samples more effectively than MALA). The power posterior samplers P-MALA and P-SMMALA improve sampling effectiveness over their MALA and SMMALA counterparts. Combining manifold and population MCMC under the P-SMMALA scheme gives the highest sampling effectiveness among all four samplers.

Table 2 also shows the minimum ESS \hat{S}_{\min} for MALA, SMMALA, P-MALA and P-SMMALA to be 9, 7, 160 and 129, respectively. Both MALA and SMMALA fail to attain \hat{S}_{\min} higher than the lower threshold of 100. On the other hand, P-MALA and P-SMMALA yield \hat{S}_{\min} higher than 100.

It takes a combination of population and manifold MCMC via P-SMMALA to increase sampling effectiveness and to substantially reduce PSRF. At the same time, the target density of the MLP(4, 3, 3) with support in the small dimensional parameter space of \mathbb{R}^{27} poses convergence challenges to contemporary MCMC sampling strategies, such as P-SMMALA.

The trace plots of chains of parameter $\theta_{23} = W_{3,2}^{(2)}$, which are shown in Figure

Sampler	\hat{R}		\hat{S}			Acceptance
	Split- \hat{R}	Folded-split- \hat{R}	Min	Median	Max	
MALA	2.112773	1.930060	9	21	44	58.62
SMMALA	1.918026	1.764620	7	51	92	28.56
P-MALA	1.339584	1.385501	160	429	14547	84.45
P-SMMALA	1.050341	1.066138	129	464	1521	88.36

TABLE 2

MCMC diagnostics of MALA, SMMALA, P-MALA and P-SMMALA simulations for an MLP(4, 3, 3) model fitted to the Iris data. All four samplers employ the same $\mathcal{N}(0, 3I)$ prior. Four chains are simulated per sampler. PSRF, ESS and acceptance rates averaged across the four chains are reported. See section 3.2 and beginning of section 6 for details on how these diagnostics are computed.

4, suggest that SMMALA (Figure 4b) has faster mixing than MALA (Figure 4a) and that P-SMMALA (Figure 4d) has the fastest mixing. Moreover, the autocorrelation plot (Figure 4e) for the chains visualized in Figures 4a-4d shows that SMMALA slightly reduces autocorrelation in comparison to MALA, while power posteriors reduce autocorrelation to a great extent. The P-SMMALA chain gives the least autocorrelated samples. Thus, the visual summaries of Figure 4 are in agreement with the numerical summaries of Table 2.

6.2.2 Comparison of parameter posterior covariance for different MCMC samplers According to Figure 5, the three geometric and population MCMC samplers (SMMALA, P-MALA and P-SMMALA) uncover visually similar posterior covariance structure among the MLP(4, 3, 3) parameters. MALA appears to be in partial agreement, but deviates locally from the empirical covariance structure that the other three samplers share.

Although convergence has not been achieved by SMMALA and P-MALA and it may have or may have not been achieved by P-SMMALA subject to the choice of PSRF cut-off threshold, all three samplers seem to share the same picture about the underlying posterior covariance. Of these three samplers, P-SMMALA is likely to represent the posterior covariance structure more accurately due to its more satisfactory PSRF and ESS (Figure 5d).

One piece of circumstantial evidence in Figures 5b-5d that makes an encouraging argument for having approximated the unknown posterior covariance among MLP(4, 3, 3) parameters is the displayed block covariance structure. Hierarchical models have block-diagonal covariance (Goldstein, 1986), and neural networks are non-linear hierarchical models. Three $2 \cdot 2$ block matrices are observed at the top left of Figures 5b-5d. Each of these $2 \cdot 2$ block matrices corresponds to the weights that connect a neuron in the hidden layer to the four neurons of the input layer.

Additional covariance structure is observed in Figures 5b-5d. For instance, there seems to be positive correlation between the weights connecting the first and second input variables to the neurons in the hidden layer and the weight connecting the third neuron in the hidden layer to the third neuron in the output layer. At the same time, there seems to be negative correlation between the weights connecting the third and fourth input variables to the neurons in the hidden layer and the weight connecting the third neuron in the hidden layer to the third neuron in the output layer.

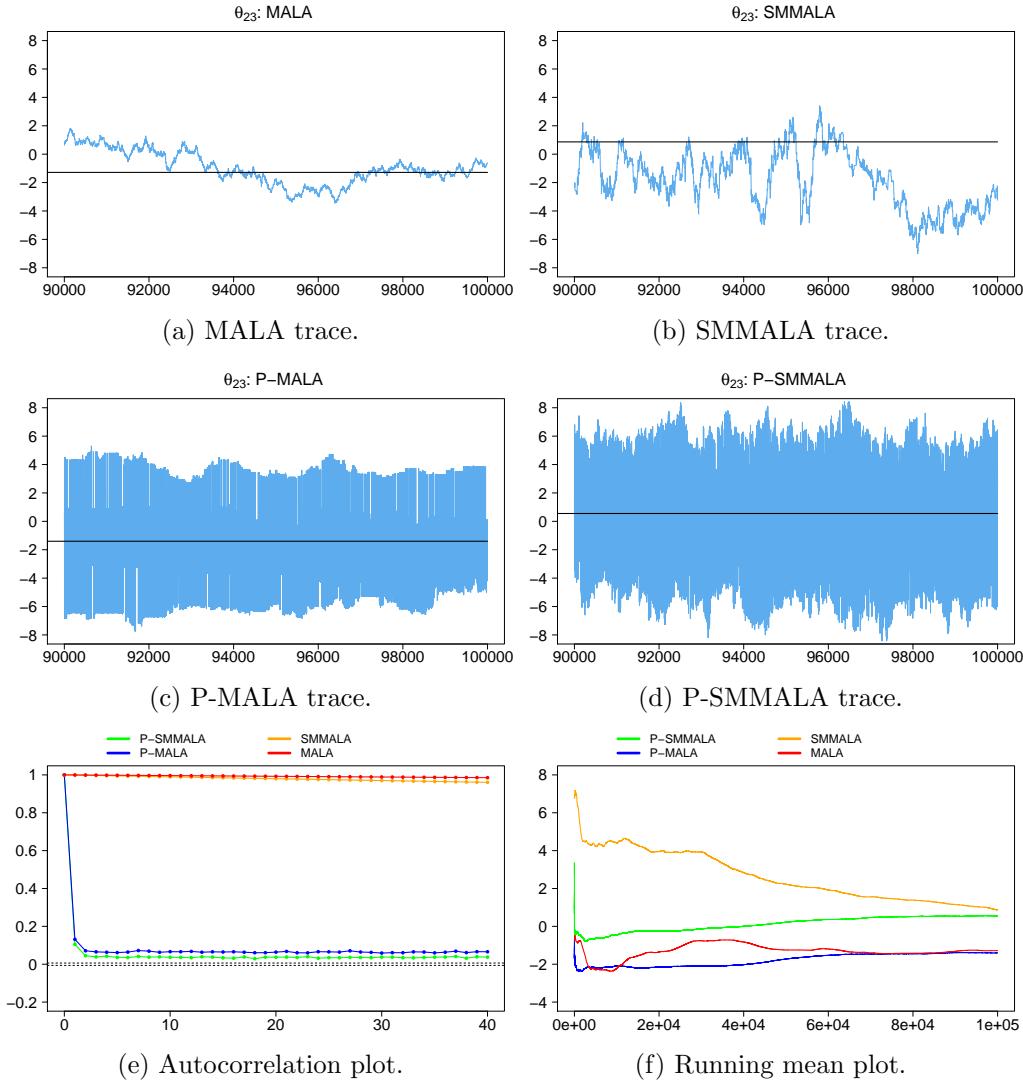


Fig 4: Trace plots, autocorrelation plot and running mean plot of chains of parameter $\theta_{23} = W_{3,2}^{(2)}$ generated by MALA, SMMALA, P-MALA and P-SMMALA with $\mathcal{N}(0, 3I)$ prior.

Uncovering the covariance structure among the parameters of small neural networks, such as MLPs and RBFs, with the help of power posteriors and manifold MCMC provides two benefits. Firstly, P-SMMALA simulations on small neural networks can be utilized for constructing benchmarks to assess the capacity of scalable inference methods to learn the parameter covariance structure of neural networks. Secondly, P-SMMALA simulations can be used for studying parameter relations and consequently neural network architecture characteristics at small scale.

6.3 Implementation

PyTorch (Steiner et al., 2019) is used for implementing the MLP model, as defined by (5.1)-(5.2). An MLP class is set to be a subclass of `torch.nn.Module`, with log-likelihood (5.5) for binary classification equal to the negative value of `torch.nn.BCELoss` and with log-likelihood (5.8) for multiclass classification equal

to the negative value of `torch.nn.CrossEntropyLoss`.

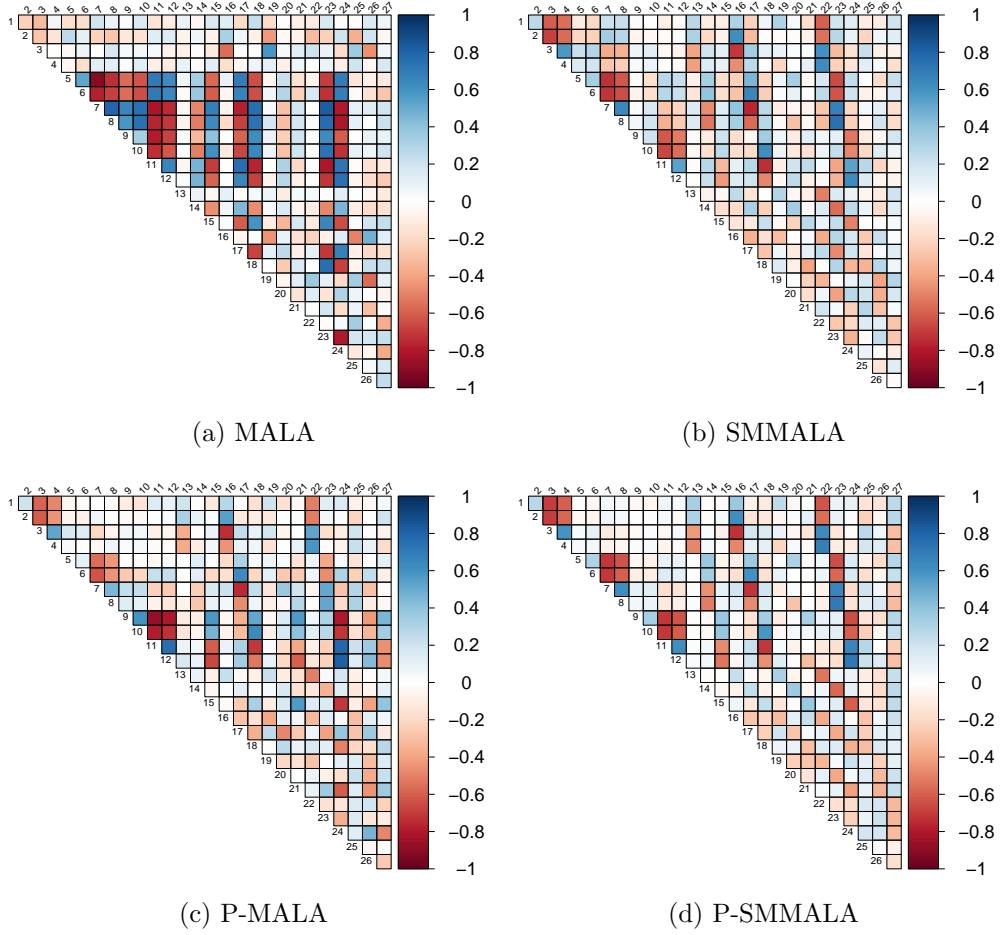


Fig 5: Empirical posterior covariances of chains generated by MALA, SMMALA, P-MALA and P-SMMALA with $\mathcal{N}(0, 3I)$ prior. Covariances are rounded off to two decimal digits.

A Python package called `eeyore` has been developed to implement the MCMC algorithms of the paper on neural networks. `eeyore` is built on top of PyTorch and uses PyTorch tensors for computations. Each MCMC algorithm takes an instance of `torch.nn.Module` as input, with the logarithm of the target density being a `log_target` method of the instance.

Log-target density gradients for MALA are computed via the automatic differentiation functionality of the `torch.autograd` package of PyTorch. SMMALA requires additionally the Hessian of the log-target density. There is no built-in support for Hessian computations in `torch.autograd`. For this reason, automatic differentiation functionality for Hessian computations has been developed in `eeyore`. To compute second order derivatives for the Hessian via automatic differentiation, the `grad` method of `torch.autograd` is called in two nested levels. Calling the `backward` method of `torch.autograd` and subsequently calling `grad` is not a viable alternative, as it leads to memory leak.

Optimization via GD for $\text{MLP}(2, 2, 1)$ in section 6.1 is run using PyTorch. The loss function for optimization is computed via `torch.nn.BCELoss`. This

loss function corresponds to the negative log-likelihood function (5.5) involved in MCMC, thus linking the optimization and MCMC simulations. GD is coded manually instead of calling an optimization algorithm of the `torch.optim` package of PyTorch. Gradients for optimization are computed calling the `backward` method.

R packages are used for computing MCMC diagnostics, KDEs and empirical covariance matrices. PSRFs, namely split- \hat{R} and folded-split- \hat{R} , are computed via the `rstan` interface of Stan (Stan Development Team, 2019). The initial monotone sequence estimate of Monte Carlo variance $\widehat{\text{Var}}^{(\text{MC})}(\theta_j^{(1:v)})$, utilized in the calculations of ESS (3.8), is generated by the `mcmc` package (Geyer and Johnson, 2019). The KDEs for densities of bounded support based on Beta kernels (Chen, 1999) are produced by the `bde` package (Santafe et al., 2015). Empirical covariance matrices are computed using the `coda` package (Plummer et al., 2019).

The `eyore` package is available at <https://github.com/scidom/eyore>, and the `eyore`-based code for running the MCMC examples of section 6 can be found at https://code.ornl.gov/9tp/mcmc_challenges_for_bnns.

6.4 Hardware

The workload of MCMC simulations of section 6 has been split between two separate servers, an NVIDIA DGX-1 server and a virtual server instance on CADES Cloud at Oak Ridge National Laboratory. Pilot MCMC runs on these servers have indicated a three-fold increase in speed by using CPUs instead of GPUs. The GPU slowdown is explained by the overhead of copying PyTorch tensors between GPUs and CPUs for small neural networks, such as the ones used in the examples.

The NVIDIA-DGX1 server has forty Intel Xeon E5-2698 CPUs (v4, 2.20GHz), and the hypervisor of the Compute and Data Environment for Science (CADES) Cloud virtual server instance uses twenty Intel Xeon E5-2698 CPUs (v4, 2.20GHz). PyTorch multi-threading has utilized automatically twenty CPUs at MCMC runtime on each of the two servers, as observed via the `htop Unix` process viewer.

Setting aside heterogeneities in hardware configuration between the two servers and in order to provide an indication of relative computational cost across different MCMC samplers, MCMC simulation runtimes are provided for the example of applying an MLP(4, 3, 3) to the Iris data. The median runtimes across the four simulated chains per MALA, SMMALA, P-MALA and P-SMMALA are 00 : 04 : 18, 00 : 42 : 11, 02 : 50 : 43 and 21 : 49 : 30, respectively (runtimes are formatted as ‘hours : minutes : seconds’).

7. FUTURE WORK

Existing MCMC algorithms do not scale in the high-dimensional parameter space of a neural network fitted to big data for a number of reasons explained in the paper. Weight symmetries may be an ally in the research quest for scalable Bayesian inference for deep learning via MCMC methods. A reduced parameter space via exploitation of transformations that leave the neural network likelihood invariant may reduce the computational cost of MCMC and may simplify the associated problem of prior specification. Along these lines, future research may entail finding transformations that reduce the parameter space, identifying one of the equivalence classes that partition the parameter space, deducing whether

a given weight vector belongs to such an equivalence class, and designing MCMC algorithms on reduced parameter spaces. [Moore \(2016\)](#), [Pourzanjani, Jiang and Petzold \(2017\)](#) and [Hu, Zagoruyko and Komodakis \(2019\)](#) provide entry points for breaking symmetries or for imposing symmetry constraints in neural networks, and the effects of these approaches in MCMC sampling remain to be learnt.

Approximate MCMC is another future research direction towards scalable MCMC for deep learning. For instance, it is of interest to study stochastic gradient MCMC ([Welling and Teh, 2011](#); [Gong, Li and Hernández-Lobato, 2019](#)) for neural networks by replacing the underlying variational inference problem ([Mandt, Hoffman and Blei, 2017](#)) with a likelihood based on the kernel Stein discrepancy between predictions and observations. More specifically, the covariance structure and the predictive capacity of the approximate parameter posterior based on such a likelihood, and the scalability of the resulting MCMC scheme are the two intended lines of inquiry.

APPENDIX A: CATEGORICAL DISTRIBUTION IN POWER POSTERIORS

Starting from the fact that the event probabilities $\alpha_i(j)$ in (4.3) add up to one, γ_i in (4.4) is derived as follows:

$$\begin{aligned}
\gamma_i &= \sum_{\substack{j=0 \\ j \neq i}}^m \alpha_i(j) \\
&= \sum_{j=0}^{i-1} \exp(-\beta(i-j)) + \sum_{j=i+1}^m \exp(-\beta(j-i)) \\
&= \sum_{j=1}^i \exp(-\beta j) + \sum_{j=1}^{m-i} \exp(-\beta j) \\
&= \exp(-\beta) \left(\frac{1 - \exp(-\beta i)}{1 - \exp(-\beta)} \right) + \exp(-\beta) \left(\frac{1 - \exp(-\beta(m-i))}{1 - \exp(-\beta)} \right) \\
&= \frac{\exp(-\beta)(2 - \exp(-\beta i) - \exp(-\beta(m-i)))}{1 - \exp(-\beta)}.
\end{aligned}$$

APPENDIX B: MEANS OF PARAMETERS OF MLP(2, 2, 1)

Parameter	Prior				
	$\mathcal{N}(10, 3I)$	$\mathcal{N}(0, 3I)$	$\mathcal{N}(0, 10I)$	$\mathcal{N}(0, 100I)$	$\mathcal{U}(-20, 20)$
$\theta_1 = W_{1,1}^{(1)}$	10.0007	-0.0418	-0.1050	-0.2573	-1.1139
$\theta_2 = W_{1,2}^{(1)}$	9.9943	-0.0303	-0.1479	-0.3739	0.1023
$\theta_3 = W_{2,1}^{(1)}$	10.0034	-0.0481	-0.1180	-0.3951	-0.1049
$\theta_4 = W_{2,2}^{(1)}$	10.0022	-0.0394	-0.0881	-0.3542	-0.2238
$\theta_5 = b_1^{(1)}$	9.9976	-0.0911	-0.2755	-0.9010	-1.0763
$\theta_6 = b_2^{(1)}$	9.9986	-0.0917	-0.2250	-1.0149	-0.0799
$\theta_7 = W_{1,1}^{(2)}$	4.0058	0.0012	0.0038	0.1769	0.2347
$\theta_8 = W_{1,2}^{(2)}$	4.0045	-0.0138	0.0018	-0.0236	0.0952
$\theta_9 = b_1^{(2)}$	4.0032	0.0016	-0.0030	0.0314	0.6039

TABLE 3

Monte Carlo means of the nine parameters of an MLP(2, 2, 1) model fitted to XOR. Every row corresponds to one of the nine parameters. Every column corresponds to a MALA simulation with a different prior. Ten chains are simulated per prior, the Monte Carlo mean of each chain is computed, and the mean of the ten Monte Carlo means is tabulated for each parameter.

APPENDIX C: PLOTS FOR PARAMETER θ_4 OF MLP(2, 2, 1)

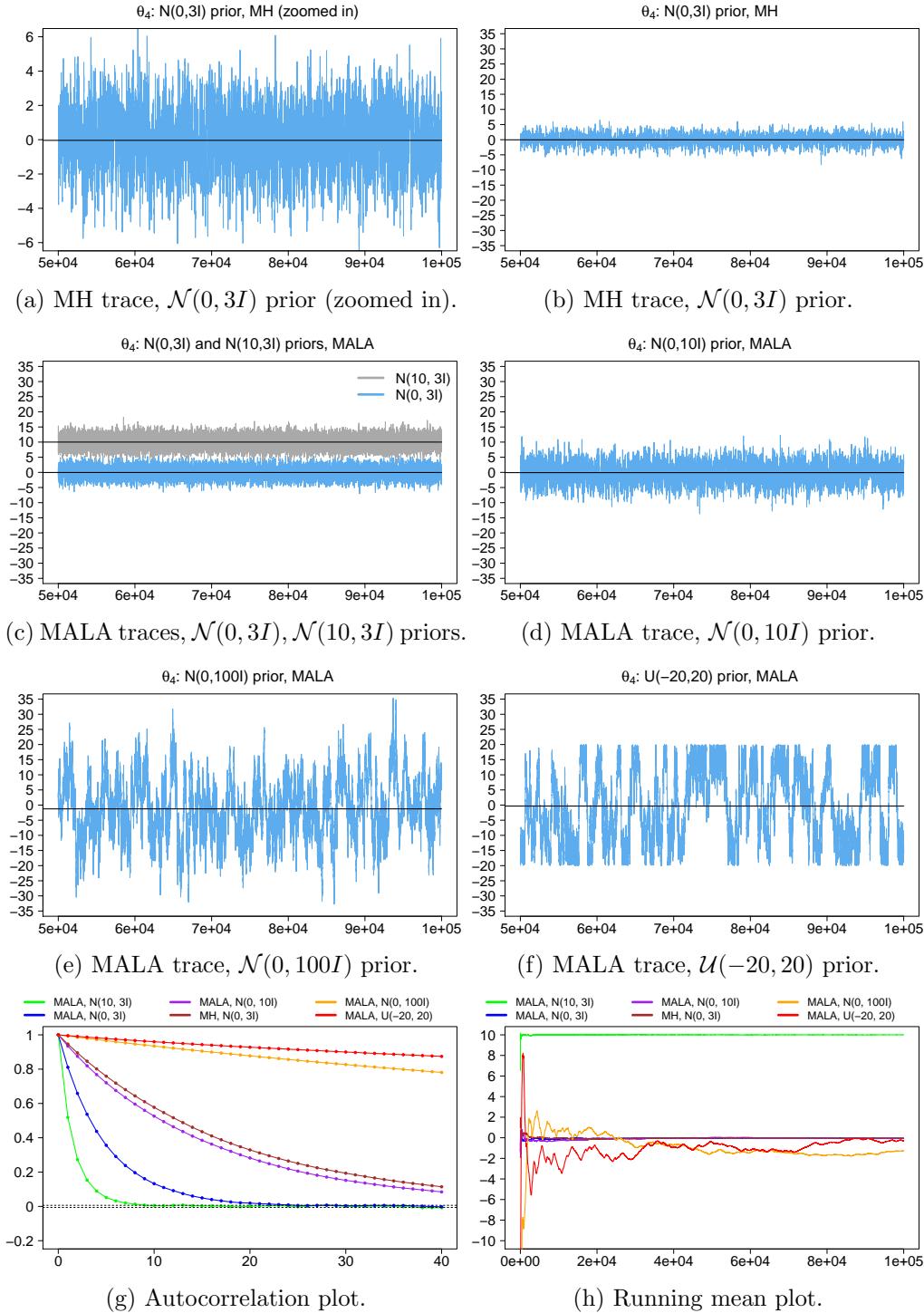


Fig 6: Trace plots, autocorrelation plot and running mean plot of chains of parameter $\theta_4 = W_{2,2}^{(1)}$ generated by MH and MALA for different priors.

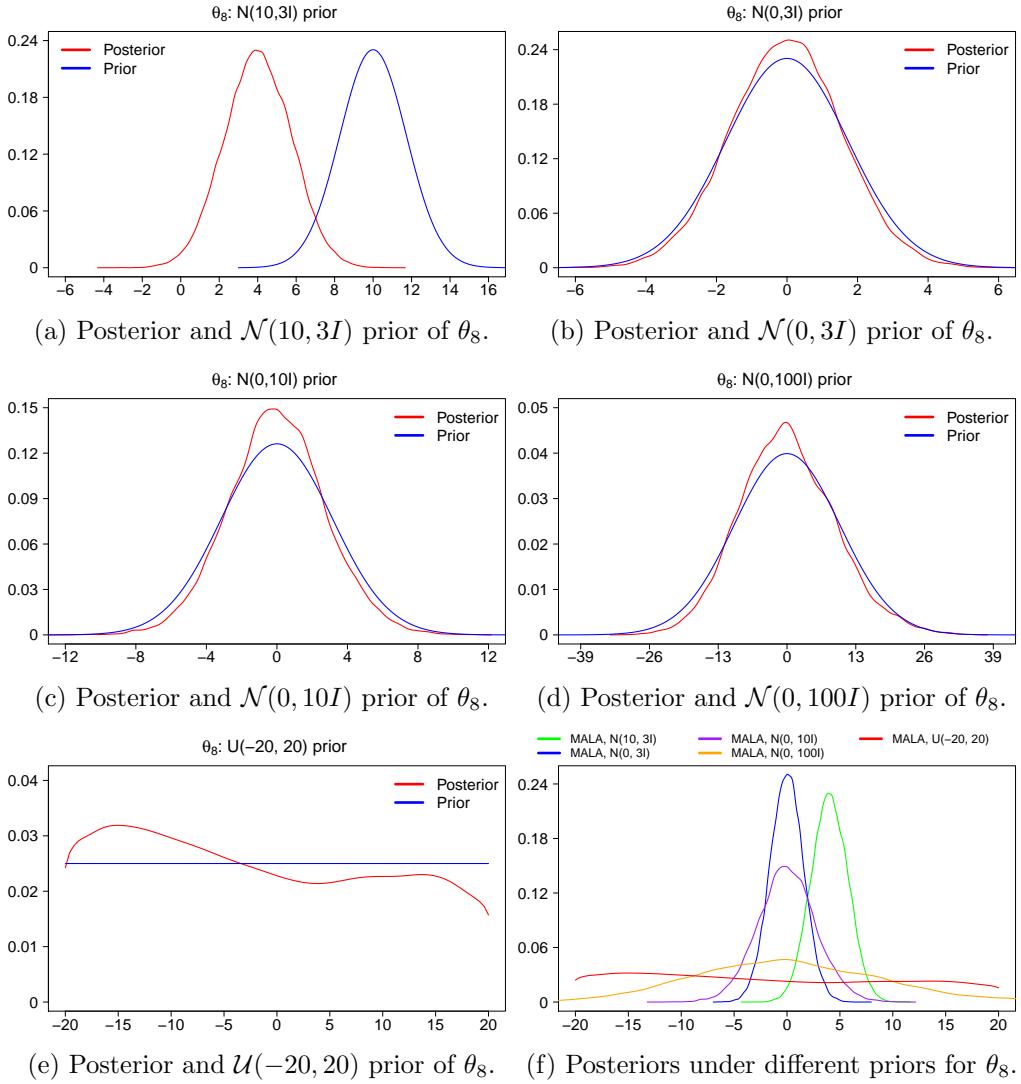
APPENDIX D: PLOTS FOR PARAMETER θ_8 OF MLP(2, 2, 1)


Fig 7: Posterior KDEs based on MALA chains of parameter $\theta_8 = W_{1,2}^{(2)}$ of an MLP(2, 2, 1) model fitted to XOR. MALA chains have been simulated for the five different priors mentioned in Table 1. Each posterior KDE is plotted against its corresponding prior. Moreover, all posterior KDEs are overlaid in a single plot.

ACKNOWLEDGEMENTS

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the US Department of Energy under contract DE-AC05-00OR22725.

This research used resources of the Compute and Data Environment for Science (CADES) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

REFERENCES

- ANDERSON, E. (1936). The Species Problem in Iris. *Annals of the Missouri Botanical Garden* **23** 457–509. [23](#)
- ANDRIEU, C., DE FREITAS, J. F. G. and DOUCET, A. (1999). Sequential Bayesian Estimation And Model Selection Applied To Neural Networks. [3](#)
- ANDRIEU, C., DE FREITAS, N. and DOUCET, A. (2000). Reversible Jump MCMC Simulated Annealing for Neural Networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence* 11–18. Morgan Kaufmann Publishers Inc. [3](#)
- BADRINARAYANAN, V., MISHRA, B. and CIOPPOLA, R. (2015). Symmetry-invariant optimization in deep networks. *arXiv*. [4](#)
- BÉDARD, M. (2008). Optimal acceptance rates for Metropolis algorithms: Moving beyond 0.234. *Stochastic Processes and their Applications* **118** 2198 - 2222. [7](#)
- BENNETT, J. E., RACINE-POON, A. and WAKEFIELD, J. C. (1996). MCMC for nonlinear hierarchical models. In *Markov chain Monte Carlo in practice* (W. R. Gilks, S. Richardson and D. Spiegelhalter, eds.) 339–358. Chapman and Hall/CRC. [3](#)
- BERNARDO, J. M. (1979). Reference Posterior Distributions for Bayesian Inference. *Journal of the Royal Statistical Society. Series B (Methodological)* **41** 113–147. [5](#)
- BETANCOURT, M. (2013). A General Metric for Riemannian Manifold Hamiltonian Monte Carlo. In *Geometric Science of Information* 327–334. Springer Berlin Heidelberg. [8](#), [15](#)
- BLEI, D. M., KUCUKELBIR, A. and McAULIFFE, J. D. (2017). Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association* **112** 859-877. [2](#)
- BRAAK, C. J. F. T. (2006). A Markov chain Monte Carlo version of the genetic algorithm differential evolution: easy Bayesian computing for real parameter spaces. *Statistics and Computing* **16** 239–249. [24](#)
- BREA, J., SIMSEK, B., ILLING, B. and GERSTNER, W. (2019). Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv*. [4](#)
- BROOKS, S. P. and GELMAN, A. (1998). General Methods for Monitoring Convergence of Iterative Simulations. *Journal of Computational and Graphical Statistics* **7** 434–455. [9](#)
- CHEN, S. X. (1999). Beta kernel estimators for density function. *Computational Statistics and Data Analysis* **31** 131–145. [15](#), [28](#)
- CHEN, T., FOX, E. and GUESTRIN, C. (2014). Stochastic Gradient Hamiltonian Monte Carlo. In *Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research* **32** 1683–1691. [3](#), [6](#)
- CHEN, A. M., LU, H. and HECHT-NIELSEN, R. (1993). On the Geometry of Feedforward Neural Network Error Surfaces. *Neural Computation* **5** 910–927. [4](#)
- CHEN, W. Y., BARP, A., BRIOL, F.-X., GORHAM, J., GIROLAMI, M., MACKEY, L. and OATES, C. (2019). Stein Point Markov Chain Monte Carlo. In *Proceedings of the 36th International Conference on Machine Learning* **97** 1011–1021. PMLR. [6](#)
- CHOLLET, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* 1251–1258. [12](#)
- CHWIAŁKOWSKI, K., STRATHMANN, H. and GRETTON, A. (2016). A Kernel Test of Goodness of Fit. In *Proceedings of The 33rd International Conference on Machine Learning* **48** 2606–2615. PMLR. [6](#)
- COWLES, M. K. and CARLIN, B. P. (1996). Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. *Journal of the American Statistical Association* **91** 883–904. [6](#), [19](#), [23](#)

- CYBENKO, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* **2** 303–314. [1](#)
- DANIELS, M. J. and KASS, R. E. (1998). A Note on First-Stage Approximation in Two-Stage Hierarchical Models. *Sankhyā: The Indian Journal of Statistics, Series B (1960-2002)* **60** 19–30. [3](#)
- DE FREITAS, N. (1999). Bayesian Methods for Neural Networks, PhD thesis, Trinity College. University of Cambridge. [3](#), [5](#), [12](#)
- DE FREITAS, N., ANDRIEU, C., HØJEN-SØRENSEN, P., NIRANJAN, M. and GEE, A. (2001). *Sequential Monte Carlo Methods for Neural Networks In Sequential Monte Carlo Methods in Practice* 359–379. Springer New York. [3](#)
- DE SA, C., CHEN, V. and WONG, W. (2018). Minibatch Gibbs Sampling on Large Graphical Models. In *Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research* **80** 1165–1173. [3](#)
- DUPUY, C. and BACH, F. (2017). Online but Accurate Inference for Latent Variable Models with Local Gibbs Sampling. *Journal of Machine Learning Research* **18** 1–45. [2](#)
- ENSIGN, D., NEVILLE, S., PAUL, A. and VENKATASUBRAMANIAN, S. (2017). The Complexity of Explaining Neural Networks Through (group) Invariants. In *Proceedings of the 28th International Conference on Algorithmic Learning Theory* **76** 341–359. [4](#)
- ESMAEILI, B., WU, H., JAIN, S., BOZKURT, A., SIDDHARTH, N., PAIGE, B., BROOKS, D. H., DY, J. and VAN DE MEENT, J.-W. (2019). Structured Disentangled Representations. In *Proceedings of Machine Learning Research* **89** 2525–2534. [3](#)
- FISHER, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics* **7** 179–188. [23](#)
- FREEMAN, I., ROESE-KOERNER, L. and KUMMERT, A. (2018). Effnet: An efficient structure for convolutional neural networks. In *2018 25th IEEE International Conference on Image Processing (ICIP)* 6–10. [12](#)
- FRIEL, N. and PETTITT, A. N. (2008). Marginal likelihood estimation via power posteriors. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **70** 589–607. [9](#), [12](#)
- GELMAN, A. and RUBIN, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science* **7** 457–472. [9](#)
- GELMAN, A., CARLIN, J. B., STERN, H. S. and RUBIN, D. B. (2004). *Bayesian data analysis*, second ed. Chapman and Hall/CRC. [10](#), [24](#)
- GELMAN, A., CARLIN, J. B., STERN, H. S., DUNSON, D. B., VEHTARI, A. and RUBIN, D. B. (2013). *Bayesian data analysis*, third ed. Chapman and Hall/CRC. [9](#)
- GEYER, C. J. (1992). Practical Markov Chain Monte Carlo. *Statistical Science* **7** 473–483. [10](#), [11](#)
- GEYER, C. J. and JOHNSON, L. T. (2019). mcmc: Markov chain Monte Carlo. R package version 0.9-6. [28](#)
- GILKS, W. R., RICHARDSON, S. and SPIEGELHALTER, D., eds. (1996). *Markov chain Monte Carlo in practice*. Chapman and Hall/CRC.
- GILKS, W. R. and ROBERTS, G. O. (1996). Strategies for improving MCMC. In *Markov chain Monte Carlo in practice* (W. R. Gilks, S. Richardson and D. Spiegelhalter, eds.) 89–114. Chapman and Hall/CRC. [3](#)
- GIORDANO, R. J., BRODERICK, T. and JORDAN, M. I. (2015). Linear Response Methods for Accurate Covariance Estimates from Mean Field Variational Bayes. In *Advances in Neural Information Processing Systems 28* 1441–1449. Curran Associates, Inc. [3](#)
- GIROLAMI, M. and CALDERHEAD, B. (2011). Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **73** 123–214. [4](#), [7](#), [8](#)
- GOLDSTEIN, H. (1986). Multilevel mixed linear model analysis using iterative generalized least squares. *Biometrika* **73** 43–56. [25](#)
- GONG, L. and FLEGAL, J. M. (2016). A Practical Sequential Stopping Rule for High-Dimensional Markov Chain Monte Carlo. *Journal of Computational and Graphical Statistics* **25** 684–700. [10](#)
- GONG, W., LI, Y. and HERNÁNDEZ-LOBATO, J. M. (2019). Meta-Learning For Stochastic Gradient MCMC. In *International Conference on Learning Representations*. [3](#), [29](#)
- GOODFELLOW, I., BENGIO, Y. and COURVILLE, A. (2016). *Deep learning*. MIT press. [13](#), [15](#)
- GRAF, S. and LUSCHGY, H. (2007). *Foundations of quantization for probability distributions*. Springer. [6](#)

- GRETTON, A., BORGWARDT, K. M., RASCH, M. J., SCHÖLKOPF, B. and SMOLA, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research* **13** 723–773. [6](#)
- GU, S. S., GHAHRAMANI, Z. and TURNER, R. E. (2015). Neural Adaptive Sequential Monte Carlo. In *Advances in Neural Information Processing Systems 28* 2629–2637. Curran Associates, Inc. [3](#)
- HASTIE, T., TIBSHIRANI, R. and FRIEDMAN, J. (2016). *The elements of statistical learning: data mining, inference and prediction*, second ed. Springer. [12](#)
- HECHT-NIELSEN, R. (1990). On the algebraic structure of feedforward network weight spaces. In *Advanced Neural Computers* (R. Eckmiller, ed.) 129–135. North-Holland. [4](#)
- HORNIK, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks* **4** 251–257. [1](#)
- HOWARD, A. G., ZHU, M., CHEN, B., KALENICHENKO, D., WANG, W., WEYAND, T., ANDREETTO, M. and ADAM, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv*. [12](#)
- HU, S. X., ZAGORUYKO, S. and KOMODAKIS, N. (2019). Exploring weight symmetry in deep neural networks. *Computer Vision and Image Understanding* **187** 102786. [4](#), [29](#)
- HUANG, C.-W., SANKARAN, K., DHEKANE, E., LACOSTE, A. and COURVILLE, A. (2019). Hierarchical Importance Weighted Autoencoders. In *Proceedings of the 36th International Conference on Machine Learning* **97** 2869–2878. [3](#)
- IANDOLA, F. N., HAN, S., MOSKEWICZ, M. W., ASHRAF, K., DALLY, W. J. and KEUTZER, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and 0.5 MB model size. *arXiv*. [12](#)
- INSELBERG, A. and DIMSDALE, B. (1990). Parallel Coordinates: A Tool for Visualizing Multi-dimensional Geometry. In *Proceedings of the 1st Conference on Visualization* 361–378. IEEE Computer Society Press. [18](#)
- JARRETT, K., KAVUKCUOGLU, K., RANZATO, M. and LECUN, Y. (2009). What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision* 2146–2153. [13](#)
- JAYNES, E. T. (1968). Prior Probabilities. *IEEE Transactions on Systems Science and Cybernetics* **4** 227–241. [5](#)
- JEFFREYS, H. (1962). *The theory of probability*, 3rd ed. OUP Oxford. [5](#)
- KASS, R. E., CARLIN, B. P., GELMAN, A. and NEAL, R. M. (1998). Markov Chain Monte Carlo in Practice: A Roundtable Discussion. *The American Statistician* **52** 93–100. [10](#)
- KRIZHEVSKY, A., SUTSKEVER, I. and HINTON, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger, eds.) 1097–1105. Curran Associates, Inc. [12](#)
- LEE, H. K. H. (2000). Consistency of posterior distributions for neural networks. *Neural Networks* **13** 629–642. [5](#)
- LEE, H. K. H. (2003). A Noninformative Prior for Neural Networks. *Machine Learning* **50** 197–212. [5](#)
- LEE, H. K. (2004). Priors for neural networks. In *Classification, Clustering, and Data Mining Applications* (D. Banks, F. R. McMorris, P. Arabie and W. Gaul, eds.) 141–150. Springer. [5](#), [17](#)
- LEE, H. K. (2005). Neural networks and default priors. In *Proceedings of the American Statistical Association, Section on Bayesian Statistical Science*. [5](#), [6](#), [19](#)
- LEE, H. K. (2007). Default Priors for Neural Network Classification. *Journal of Classification* **24** 53–70. [6](#)
- LU, Z., PU, H., WANG, F., HU, Z. and WANG, L. (2017). The Expressive Power of Neural Networks: A View from the Width. In *Advances in Neural Information Processing Systems 30* 6231–6239. Curran Associates, Inc. [1](#)
- MA, Y.-A., FOTI, N. J. and FOX, E. B. (2017). Stochastic Gradient MCMC Methods for Hidden Markov Models. In *Proceedings of the 34th International Conference on Machine Learning* (D. PRECUP and Y. W. TEH, eds.). *Proceedings of Machine Learning Research* **70** 2265–2274. [3](#)
- MADDISON, C. J., HUANG, A., SUTSKEVER, I. and SILVER, D. (2015). Move evaluation in Go using deep convolutional neural networks. In *International Conference on Learning Representations*. [5](#)
- MANDT, S., HOFFMAN, M. D. and BLEI, D. M. (2017). Stochastic Gradient Descent as Ap-

- proximate Bayesian Inference. *Journal of Machine Learning Research* **18** 1–35. 3, 6, 29
- MINSKY, M. L. and PAPERT, S. A. (1988). *Perceptrons: expanded edition*. MIT press. 12, 15
- MOORE, D. A. (2016). Symmetrized variational inference. In *NIPS Workshop on Advances in Approximate Bayesian Inference*. 4, 29
- NAIR, V. and HINTON, G. E. (2009). 3D Object Recognition with Deep Belief Nets. In *Advances in Neural Information Processing Systems 22* 1339–1347. Curran Associates, Inc. 13
- NALISNICK, E. T. (2018). On Priors for Bayesian Neural Networks, PhD thesis, UC Irvine. 4, 6, 17
- NEMETH, C. and SHERLOCK, C. (2018). Merging MCMC Subposterior through Gaussian-Process Approximations. *Bayesian Analysis* **13** 507–530. 3
- ONG, V. M. H., NOTT, D. J. and SMITH, M. S. (2018). Gaussian Variational Approximation With a Factor Covariance Structure. *Journal of Computational and Graphical Statistics* **27** 465–478. 3
- PAPAMARKOU, T., MIRA, A. and GIROLAMI, M. (2014). Zero Variance Differential Geometric Markov Chain Monte Carlo Algorithms. *Bayesian Analysis* **9** 97–128. 11
- PEARCE, T., ZAKI, M., BRINTRUP, A. and NEELY, A. (2019). Expressive Priors in Bayesian Neural Networks: Kernel Combinations and Periodic Functions. In *Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence*. 6
- PLUMMER, M., BEST, N., COWLES, K., VINES, K., SARKAR, D., BATES, D., ALMOND, R. and MAGNUSSON, A. (2019). coda: Output Analysis and Diagnostics for MCMC. R package version 0.19-3. 21, 28
- POLSON, N. G. and SOKOLOV, V. (2017). Deep Learning: A Bayesian Perspective. *Bayesian Analysis* **12** 1275–1304. 1
- POURZANJANI, A. A., JIANG, R. M. and PETZOLD, L. R. (2017). Improving the Identifiability of Neural Networks for Bayesian Inference. In *NIPS Workshop on Bayesian Deep Learning*. 4, 29
- QUIROZ, M., KOHN, R., VILLANI, M. and TRAN, M.-N. (2019). Speeding Up MCMC by Efficient Data Subsampling. *Journal of the American Statistical Association* **114** 831–843. 3
- RANGANATH, R., TRAN, D. and BLEI, D. (2016). Hierarchical Variational Models. In *Proceedings of The 33rd International Conference on Machine Learning* **48** 324–333. 3
- RAO, R. (1945). Information and accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society* **37** 81–91. 8
- ROBERT, C. P., ELVIRA, V., TAWN, N. and WU, C. (2018). Accelerating MCMC algorithms. *Wiley Interdisciplinary Reviews: Computational Statistics* **10** e1435. 3
- ROBERTS, G. O., GELMAN, A. and GILKS, W. R. (1997). Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability* **7** 110–120. 7
- ROBERTS, G. O. and ROSENTHAL, J. S. (1998). Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **60** 255–268. 7
- ROBERTS, G. O. and STRAMER, O. (2002). Langevin Diffusions and Metropolis-Hastings Algorithms. *Methodology And Computing In Applied Probability* **4** 337–357. 7
- ROSENBLATT, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* **65** 386. 12
- RUDOLF, D. and SCHWEIZER, N. (2018). Perturbation theory for Markov chains via Wasserstein distance. *Bernoulli* **24** 2610–2639. 6
- SANTAFE, G., CALVO, B., PEREZ, A. and LOZANO, J. A. (2015). bde: Bounded Density Estimation. R package version 1.0.1. 28
- SARGENT, D. J., HODGES, J. S. and CARLIN, B. P. (2000). Structured Markov chain Monte Carlo. *Journal of Computational and Graphical Statistics* **9** 217–234. 3
- SEITA, D., PAN, X., CHEN, H. and CANNY, J. (2018). An Efficient Minibatch Acceptance Test for Metropolis-Hastings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18* 5359–5363. International Joint Conferences on Artificial Intelligence Organization. 3
- SIMPSON, D., RUE, H., RIEBLER, A., MARTINS, T. G. and SØRBYE, S. H. (2017). Penalising Model Component Complexity: A Principled, Practical Approach to Constructing Priors. *Statistical Science* **32** 1–28. 5
- STEINER, B., DEVITO, Z., CHINTALA, S., GROSS, S., PASZKE, A., MASSA, F., LERER, A., CHANAN, G., LIN, Z., YANG, E., DESMAISON, A., TEJANI, A., KOPF, A., BRADBURY, J., ANTIGA, L., RAISON, M., GIMELSHEIN, N., CHILAMKURTHY, S., KILLEEN, T., FANG, L. and

- BAI, J. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32* Curran Associates, Inc. 26
- STAN DEVELOPMENT TEAM (2019). RStan: the R interface to Stan. R package version 2.19.2. 10, 11, 28
- TITSIAS, M. K. and RUIZ, F. (2019). Unbiased Implicit Variational Inference. In *Proceedings of Machine Learning Research* 89 167–176. 3
- TITTERINGTON, D. M. (2004). Bayesian Methods for Neural Networks and Related Models. *Statistical Science* 19 128–139. 3
- TRUONG, T.-D., NGUYEN, V.-T. and TRAN, M.-T. (2018). Lightweight Deep Convolutional Network for Tiny Object Recognition. In *ICPRAM* 675–682. 12
- VATS, D. and FLEGAL, J. M. (2018). Lugsail lag windows and their application to MCMC. *arXiv*. 10
- VATS, D. and KNUDSON, C. (2018). Revisiting the Gelman-Rubin Diagnostic. *arXiv*. 10
- VEHTARI, A., SARKKA, S. and LAMPINEN, J. (2000). On MCMC sampling in Bayesian MLP neural networks. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium* 1 317–322. 12
- VEHTARI, A., GELMAN, A., SIMPSON, D., CARPENTER, B. and BURKNER, P.-C. (2019). Rank-normalization, folding, and localization: An improved R for assessing convergence of MCMC. *arXiv*. 10, 11, 19, 23
- VLADIMIROVA, M., VERBEEK, J., MESEJO, P. and ARBEL, J. (2019). Understanding Priors in Bayesian Neural Networks at the Unit Level. In *Proceedings of the 36th International Conference on Machine Learning* 97 6458–6467. PMLR. 6
- WELLING, M. and TEH, Y. W. (2011). Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning* 681–688. Omnipress. 3, 6, 29
- WILLIAMS, P. M. (1995). Bayesian Regularization and Pruning Using a Laplace Prior. *Neural Computation* 7 117–143. 5
- WILLIAMS, C. K. I. (2000). An MCMC Approach to Hierarchical Mixture Modelling. In *Advances in Neural Information Processing Systems 12* 680–686. MIT Press. 3
- ZHANG, G., SUN, S., DUVENAUD, D. and GROSSE, R. (2018a). Noisy Natural Gradient as Variational Inference. In *Proceedings of the 35th International Conference on Machine Learning* 80 5852–5861. 3
- ZHANG, X., ZHOU, X., LIN, M. and SUN, J. (2018b). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 6848–6856. 12