Monthly Notices
of the
ROYAL ASTRONOMICAL SOCIETY

MNRAS **496,** 328–338 (2020)
Advance Access publication 2020 May 28

doi:10.1093/mnras/staa1469

Accelerated Bayesian inference using deep learning

Adam Moss★

School of Physics and Astronomy, University of Nottingham, Nottingham NG7 2RD, England

Accepted 2020 May 19. Received 2020 May 15; in original form 2019 October 18

ABSTRACT

We present a novel Bayesian inference tool that uses a neural network (NN) to parametrize efficient Markov Chain Monte Carlo (MCMC) proposals. The target distribution is first transformed into a diagonal, unit variance Gaussian by a series of non-linear, invertible, and non-volume preserving flows. NNs are extremely expressive, and can transform complex targets to a simple latent representation. Efficient proposals can then be made in this space, and we demonstrate a high degree of mixing on several challenging distributions. Parameter space can naturally be split into a block diagonal speed hierarchy, allowing for fast exploration of subspaces where it is inexpensive to evaluate the likelihood. Using this method, we develop a nested MCMC sampler to perform Bayesian inference and model comparison, finding excellent performance on highly curved and multimodal analytic likelihoods. We also test it on *Planck* 2015 data, showing accurate parameter constraints, and calculate the evidence for simple one-parameter extensions to the standard cosmological model in ∼20D parameter space. Our method has wide applicability to a range of problems in astronomy and cosmology and is available for download from https://github.com/adammoss/nnest.

Key words: methods: data analysis – methods: statistical.

1 INTRODUCTION

In the last few years, we have witnessed a revolution in machine learning. The use of deep neural networks (NNs) has become widespread due to increased computational power, the availability of large data sets, and their ability to solve problems previously deemed intractable (see Lecun, Bengio & Hinton 2015 for an overview). Deep learning is particularly suited to the era of data-driven astronomy and cosmology, but so far applications have mainly focused on supervised learning tasks such as classification and regression.

Bayesian inference is now a standard technique, with codes such as COSMOMC (Lewis & Bridle 2002), COSMOSIS (Zuntz et al. 2015), EMCEE (Foreman-Mackey et al. 2013), MONTEPYTHON (Audren et al. 2013), MULTINEST (Feroz & Hobson 2008; Feroz, Hobson & Bridges 2009; Feroz et al. 2013), and POLYCHORD (Handley, Hobson & Lasenby 2015; Higson et al. 2017) used for parameter estimation and model selection. Many of these use Metropolis–Hastings Markov Chain Monte Carlo (MCMC) to draw samples from the posterior distribution. The dimensionality of the problem can be high, with $\gtrsim 20$ parameters for the *Planck* likelihood (including nuisance parameters), and potentially more for future large-scale surveys. For some problems the likelihood can be non-Gaussian (e.g. with curving degeneracies) and/or multimodal, making conventional MCMC techniques inefficient and liable to miss regions of parameter space. Likelihoods can also be expensive to calculate, so minimizing the total number of evaluations is advantageous. This was the motivation behind BAMBI (Graff et al. 2012; Handley, Scott & White 2019), which used an NN to approximate the likelihood function where possible. Efficient exploration of the posterior is crucial for Bayesian inference and model selection.

A good proposal function is vital to fully explore the distribution and ensure convergence of the chain. Deep learning has recently been shown to accelerate MCMC sampling by using NNs to parametrize efficient proposals that maintain detailed balance (see e.g. Levy, Hoffman & Sohl-Dickstein 2017; Song, Zhao & Ermon 2017). The proposal function can be trained, for example, to minimize the autocorrelation length of the chain. Some of these methods (e.g. generalizations of Hamiltonian Monte Carlo) exploit the gradient of the target, and analytic gradients are often not available for astronomical or cosmological models. The training time can also be prohibitive, offsetting the gain in efficiency when sampling. In this work, we use an NN to transform the likelihood to a simpler representation, which requires no gradient information and is very fast to train. This approach is inspired by *representation learning*, which hypotheses that deep NNs have the potential to yield representation spaces in which Markov chains mix faster (Bengio, Courville & Vincent 2012a; Bengio et al. 2012b).

The idea of optimizing the proposal function originated in Gelman, Roberts & Gilks (1996) and Haario et al. (2001), which suggested using a normal distribution $\mathcal{N}(\boldsymbol{x}, 2.38^2\,\boldsymbol{\Sigma}/d)$, where $\boldsymbol{\Sigma}$ is the covariance matrix, $\boldsymbol{x}$ the current state, and d the dimension. The factor of $2.38^2/d$ was shown to minimize the chain auto-

★ E-mail: adam.moss@nottingham.ac.uk

correlation length. This is equivalent to transforming variables by $\mathbf{L}^{-1}\boldsymbol{x}$, where $\mathbf{L}$ is a lower triangular matrix defined by the Cholesky decomposition of the covariance matrix, $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$. Proposals can then be made using the diagonal normal distribution $\mathcal{N}(\mathbf{x}, 2.38^2\,\mathbf{I}/d)$, where $\mathbf{I}$ is the identity matrix. Standard practice in cosmological parameter estimation is to use an initial set of samples to estimate the covariance matrix, and make proposals using the Cholesky decomposition. Linear transformations are also used in affine invariant MCMC methods (Goodman & Weare 2010; Foreman-Mackey et al. 2013).

This transformation works well when samples are well described by their covariance matrix, but can become inefficient for more complex distributions. In this paper, we use an NN to parametrize more expressive transformations that are suitable for curved and multimodal targets. The NN learns a non-linear, invertible, and non-volume preserving (NVP) mapping between data and a simpler latent space via maximum likelihood, and proposals are then made in this space. We apply our method to nested sampling, a commonly used tool to perform Bayesian inference and model comparison, although it could easily be incorporated into other MCMC frameworks. A major challenge in nested sampling is drawing new samples from a constrained target distribution, and we show that NNs can lead to improved performance over existing rejection and MCMC-based approaches.

The structure of the paper is as follows. In Section 2, we provide a short overview of nested sampling. In Section 3, we show how NNs can be trained to transform complex target data to a simpler latent space. We give further details of our algorithm in Section 4, and demonstrate results on both analytic likelihoods and *Planck* data in Section 5. We conclude and provide an outlook for future work in Section 6.

2 NESTED SAMPLING

The nested sampling algorithm (Skilling 2004; Skilling et al. 2006) was developed to accurately calculate the Bayesian evidence (or marginal likelihood). From Bayes' theorem, the posterior distribution of a set of parameters $\boldsymbol{x}$, given data $\boldsymbol{d}$, and model M is

$$p(\boldsymbol{x}|\boldsymbol{d}, M) = \frac{p(\boldsymbol{d}|\boldsymbol{x}, M)p(\boldsymbol{x}|M)}{p(\boldsymbol{d}|M)}, \tag{1}$$

where $p(\boldsymbol{d}|\boldsymbol{x}, M) = \mathcal{L}(\boldsymbol{x})$ is the likelihood, $p(\boldsymbol{x}|M) = \pi(\boldsymbol{x})$ is the prior and the normalizing constant $p(\boldsymbol{d}|M)$ is the evidence. This can be expressed as

$$Z = p(\boldsymbol{d}|M) = \int \mathcal{L}(\boldsymbol{x})\pi(\boldsymbol{x})\mathrm{d}\boldsymbol{x}. \tag{2}$$

Two competing models M_1 and M_2 can be compared by calculating the Bayes factor

$$B = \frac{p(M_1|\boldsymbol{d})}{p(M_2|\boldsymbol{d})} = \frac{\pi(M_1)}{\pi(M_2)}\frac{p(\boldsymbol{d}|M_1)}{p(\boldsymbol{d}|M_2)}, \tag{3}$$

which simplifies to the evidence ratio if the models have equal prior probability.

This integral (2) is typically hard to evaluate, but can be turned into a simpler 1D integral by a change of variables (Skilling 2004),

$$X(\lambda) = \int_{\mathcal{L}(\boldsymbol{x})>\lambda} \pi(\boldsymbol{x})\mathrm{d}\boldsymbol{x}, \quad Z = \int_0^1 \mathcal{L}(X)\mathrm{d}X, \tag{4}$$

such that $X(\lambda)$ is the prior volume associated with a likelihood constant $\mathcal{L}(\boldsymbol{x}) > \lambda$. Parameters are scaled to the unit hypercube, so that the prior is normalized, and X takes values between 0 and 1.

The nested sampling algorithm first draws a set of $n_{\rm live}$ points from $\pi(\boldsymbol{x})$. The prior is typically uniform or Gaussian, so is easy to sample from. At each iteration, the point with the lowest likelihood is replaced by a new sample drawn from the prior, with the condition that the new point has a higher likelihood. The discarded samples are termed *dead* points. For a sequence of decreasing X values,

$$0 < X_M < \cdots < X_2 < X_1 < X_0 = 1, \tag{5}$$

the integral in equation (4) can then be estimated using trapezoidal integration

$$Z = \sum_{i=1}^{M} \mathcal{L}_i w_i, \tag{6}$$

where $\mathcal{L}_i = \mathcal{L}(X_i)$ and $w_i = \frac{1}{2}(X_{i-1} - X_{i+1})$. The error in $\log Z$ can be estimated by $\sqrt{H/n_{\rm live}}$, where H is the negative relative entropy (Feroz & Hobson 2008)

$$H = \sum_{i=1}^{M} \frac{\mathcal{L}_i w_i}{Z} \log \frac{\mathcal{L}_i}{Z}. \tag{7}$$

Nested sampling can also perform posterior inference by using the sequence of dead points (and the current set of live points), and assigning a weight p_i to the ith point

$$p_i = \frac{\mathcal{L}_i w_i}{Z}. \tag{8}$$

Since the posterior samples are independent but with different importance weights, the effective number of samples can be estimated by

$$N_{\rm eff} = \frac{\left(\sum_{i=1}^{M} w_i\right)^2}{\sum_{i=1}^{M} w_i^2}. \tag{9}$$

The main difficulty with nested sampling is drawing a new, independent sample subject to a hard likelihood constraint. This can be achieved by rejection sampling, using an envelope function that encloses the current set of live points. For Gaussian likelihoods, it is most efficient to use an ellipsoidal envelope (Mukherjee, Parkinson & Liddle 2006), and for multimodal distributions a set of (possibly) overlapping ellipsoids can be drawn around clusters of live points (Feroz & Hobson 2008; Feroz et al. 2009). The disadvantage of rejection sampling is that the envelope function requires scaling by an enlargement factor to ensure it contains the entire iso-likelihood contour. This can lead to poor scaling with dimension and introduces a choice of user specified hyper parameter.

An alternative to rejection sampling is MCMC. Starting from an existing live point, a new, independent sample is obtained after performing a 'sufficient' number of steps. MCMC scales better with dimension, but is not guaranteed to fully explore or mix the distribution, and is generally not well suited if the likelihood is highly curved and/or multimodal. Variants of MCMC developed to cope with challenging targets include Galilean dynamics (Feroz & Skilling 2013), diffusive sampling (Brewer, Pártay & Csányi 2009), and slice sampling (Handley et al. 2015; Higson et al. 2017). Some of these also use clustering algorithms to identify and sample from multimodal distributions. The key point is that they all try and choose better proposals – in our case we will use an NN to try and learn one.

3 NEURAL NETWORK SAMPLING

3.1 Non-volume preserving flows

Initially, we have a set of data $\{\boldsymbol{x}^i\}_{i=1}^N$, sampled from a target distribution $\boldsymbol{x} \sim p_X(\boldsymbol{x})$, where $\boldsymbol{x}$ has dimension $n_{\rm dim}$. The goal is to draw new samples from this distribution. Latent variables are drawn from a simpler prior distribution $\boldsymbol{z} \sim p_Z(\boldsymbol{z})$, with the same dimension. Given a bijection f: $X \rightarrow Z$, the change of variables formula gives the corresponding distribution on X

$$p_X(\boldsymbol{x}) = p_Z(f(\boldsymbol{x}))\left|\det\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|. \tag{10}$$

The inverse $f^{-1}(\boldsymbol{z})$ provides a mapping from latent to real space. The bijection can be parametrized by an NN, with trainable parameters θ. NNs, however, are not generally invertible, and the Jacobian determinant in equation (10) is not easily tractable.

NVP flows, introduced in Dinh, Sohl-Dickstein & Bengio (2016), can transform simple latent distributions into rich target distributions. They are invertible and allow for tractable Jacobians by using a specific architecture for the NN. They exploit the fact that the determinant of a triangular matrix is the product of its diagonal terms. The NVP transformation is (Dinh et al. 2016)

$$\boldsymbol{x}' = \boldsymbol{m}\odot\boldsymbol{x} + (1-\boldsymbol{m})\odot\left(\boldsymbol{x}\odot\exp\left(s_{\theta_s}(\boldsymbol{m}\odot\boldsymbol{x})\right) + t_{\theta_t}(\boldsymbol{m}\odot\boldsymbol{x})\right), \tag{11}$$

where $\boldsymbol{m}$ is a binary mask vector consisting of alternating 1's and 0's, s_{θ_s} and t_{θ_t} are separate (s)cale and (t)ranslation NN's with trainable parameters θ_s and θ_t, and $\odot$ is the element-wise product. The transformation for $n_{\rm dim} = 2$ with $\boldsymbol{m} = (1, 0)$, for example, is

$$\begin{aligned} x_1' &= x_1 \\ x_2' &= x_2\exp(s_{\theta_s}(x_1)) + t_{\theta_t}(x_1), \end{aligned} \tag{12}$$

with Jacobian determinant $\exp(s_{\theta_s}(x_1))$, and the inverse is

$$\begin{aligned} x_1 &= x_1' \\ x_2 &= \left(x_2' - t_{\theta_t}(x_1)\right)\exp(-s_{\theta_s}(x_1)), \end{aligned} \tag{13}$$

with Jacobian determinant $\exp(-s_{\theta_s}(x_1))$. Note that x_1 is unmodified in this transformation.

A series of transformations can be composed into a *flow* by permuting components of the inputs in successive transformations, such that those modified in one transformation are left unchanged in the next. This can be achieved by setting the mask in the next transformation to $1 - \boldsymbol{m}$, so that successive masks resemble a checkerboard pattern. The Jacobian determinant is still tractable, and is simply the product of each individual transformation. A flow transforms a target to latent space, and an inverse flow transforms latent space to the target. Each transformation step of the flow has separate s and t networks.

Given a set of samples, the s and t networks can be trained by minimizing the loss function

$$\begin{aligned} L &= -\sum_i^N \log p_X(\boldsymbol{x}^i) \\ &= -\sum_i^N \log p_Z(f(\boldsymbol{x}^i)) + \log\left|\det\frac{\partial f(\boldsymbol{x}^i)}{\partial \boldsymbol{x}^i}\right|. \end{aligned} \tag{14}$$

Parameters are updated by backpropagating the loss using gradient descent, and the minimum loss is equivalent to the maximum-likelihood estimate of θ_s and θ_t. Note that the entire flow, consisting of multiple NNs, is trained simultaneously.

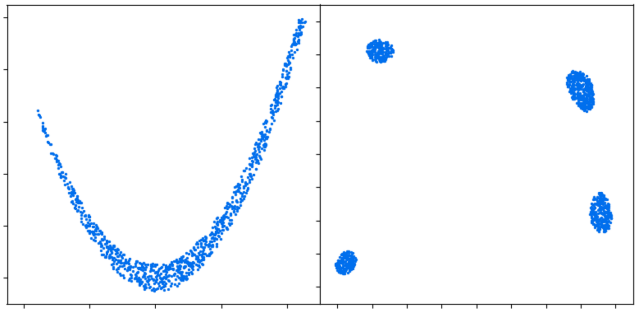

Figure 1. Example data $\boldsymbol{x} \sim p(\boldsymbol{x})$ of 1000 samples used to train the network for (left) the $n_{\rm dim} = 2$ Rosenbrock function, and (right) the Himmelblau function.

As an example, we fit flows to two data sets, shown in Fig 1. The initial data $\boldsymbol{x} \sim p(\boldsymbol{x})$ is obtained from the nested sampling of the 2D Rosenbrock and Himmelblau functions (defined in Section 5) for $n_{\rm live} = 1000$ points, evaluated when the prior volume $X = 0.02$. The target distribution is therefore uniform when $\mathcal{L}(\boldsymbol{x}) > \lambda_\star$, with $\lambda_\star$ defined by $X(\lambda_\star) = 0.02$, otherwise the probability density is zero. These functions are chosen as they are challenging examples of curved and multimodal distributions.

We choose the prior distribution p_Z to be a diagonal Gaussian with unit variance, that is $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$. We use four successive transformations in the flow, each parametrized by a fully connected s and t network with an input layer of dimension 2, two hidden layers of dimension 128, and an output layer of dimension 2. We use rectified non-linear activation (ReLU) functions after the input and hidden layers in each network.

The resulting inverse flows are shown in Fig. 2, after training for 50 epochs (each epoch is a complete pass over training data). Each transformation step begins with an (x_1 dependent) scaling and translation of x_2 (the vertical axis in the plot), with x_1 unchanged. These are then permuted, and scaling and transformation operations are applied to (the now) x_1. The initial Gaussian can be flowed into the Rosenbrock function continuously, but narrow connecting ridges appear for the Himmelblau function. This is because it cannot continuously be deformed into the target distribution. Nevertheless, the volume of these ridges is quite small compared to the region where the target probability density is non-zero.

If the network could fit the target distribution perfectly, it would be trivial to generate new, independent samples. One could simply sample from latent space $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$ and perform the inverse flow to obtain $\boldsymbol{x}$. In reality, the fit is not perfect, but we can use the learnt mapping to improve the efficiency of MCMC by making proposals in the simpler latent space.

3.2 MCMC sampling

The acceptance probability required to maintain detailed balance in a Metropolis–Hastings update is

$$\alpha = \min\left(1, \frac{p_X(\boldsymbol{x}')q(\boldsymbol{x}|\boldsymbol{x}')}{p_X(\boldsymbol{x})q(\boldsymbol{x}'|\boldsymbol{x})}\right), \tag{15}$$

where the proposal function $q(\boldsymbol{x}'|\boldsymbol{x})$ is the conditional probability of state $\boldsymbol{x}'$ given $\boldsymbol{x}$. If the proposal function is symmetric (e.g. a Gaussian with the same covariance matrix for each state) then $q(\boldsymbol{x}'|\boldsymbol{x}) = q(\boldsymbol{x}|\boldsymbol{x}')$.

For proposals made in latent space $\mathbf{z}$, the acceptance probability must be modified by the Jacobian determinant to satisfy detailed

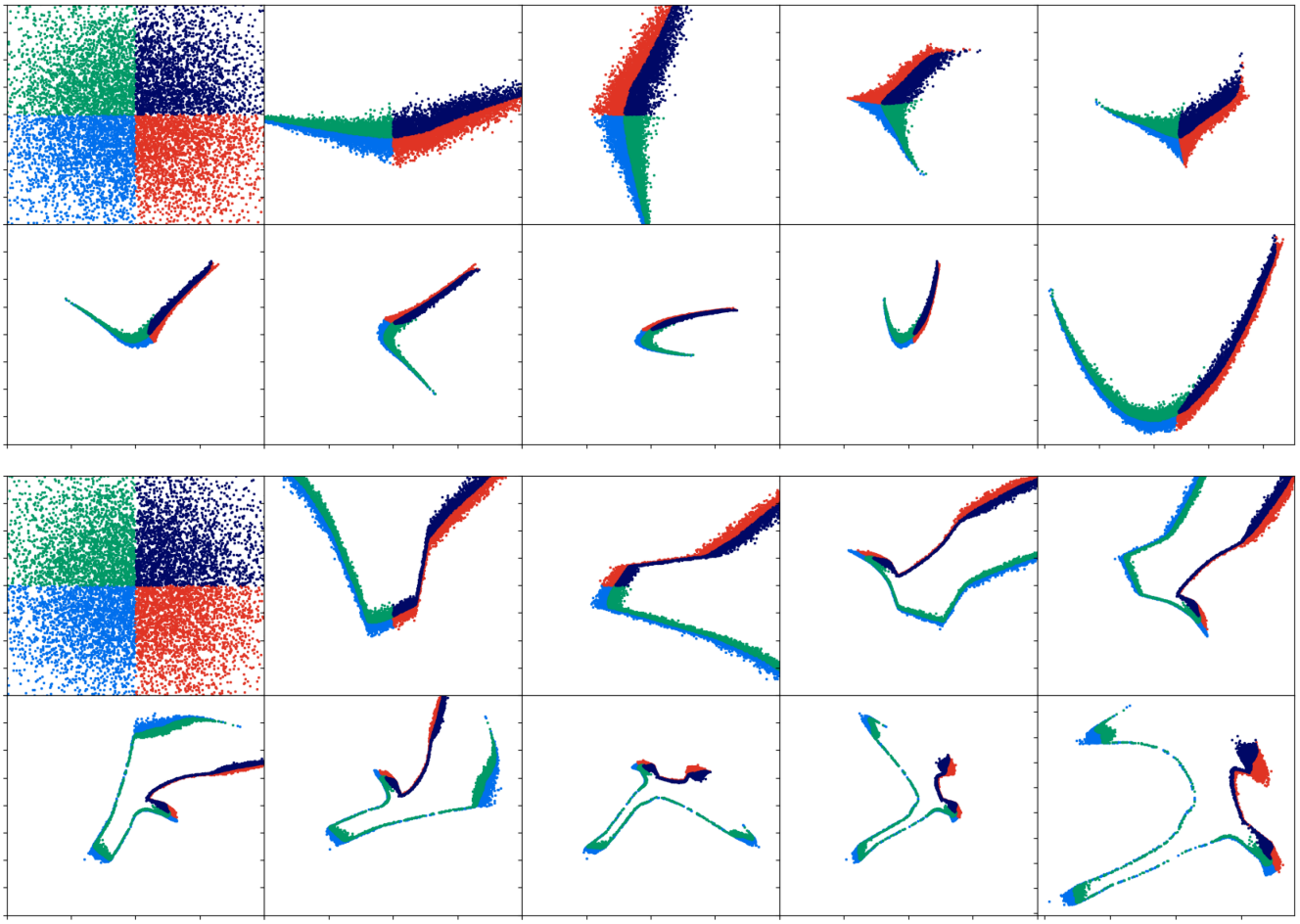

Figure 2. Inverse flow for (top) the $n_{\rm dim} = 2$ Rosenbrock function and (bottom) the Himmelblau function, trained on the initial data shown in Fig. 1. The inverse flow starts from latent space, $z \sim \mathcal{N}(0, \boldsymbol{I})$, and each step consists of a transformation followed by a permutation. The final image in each case ends at the target distribution and is zoomed-in for clarity. The colours are to help the reader track the flow of the space from one panel to the next, and do not have any specific meaning for the algorithm.

balance

$$\alpha = \min\left(1, \frac{p_X(f^{-1}(z'))q(z|z')\left|\det\frac{\partial f^{-1}(z')}{\partial z'}\right|}{p_X(f^{-1}(z))q(z'|z)\left|\det\frac{\partial f^{-1}(z)}{\partial z}\right|}\right). \tag{16}$$

Given the prior distribution of latent space is a diagonal, unit variance Gaussian, we use a symmetric proposal function

$$q(z'|z) = \mathcal{N}(z, \sigma^2\,\boldsymbol{I}), \tag{17}$$

where σ is a scaling parameter. Based on estimates of optimal proposals for Gaussian distributions (Gelman et al. 1996; Haario et al. 2001), we tune σ to give an acceptance rate of 50 per cent using the method in Feroz & Hobson (2008),

$$\sigma \rightarrow \begin{cases} \sigma e^{1/N_{\rm a}} & \text{if } N_{\rm a} > N_{\rm r} \\ \sigma e^{-1/N_{\rm r}} & \text{if } N_{\rm a} \leq N_{\rm r} \end{cases}, \tag{18}$$

where $N_{\rm a}$ and $N_{\rm r}$ are the number of accepted and rejected samples in the current MCMC chain.

To demonstrate this gives the correct distribution on p_X, we generate new samples for the flows fitted to the 2D Rosenbrock and Himmelblau functions. Starting from an existing sample $\boldsymbol{x}_{\rm init}$, we run a chain of length $N_{\rm c} = 1000$ and repeat this for 20 000 chains, each time choosing the *same* initial $\boldsymbol{x}_{\rm init}$. In Fig. 3, we plot a histogram of the resulting samples after 5 and 20 MCMC iterations, also showing the first 20 proposed moves of an example chain (note that not all of these proposals are accepted). After only five iterations, the resulting distribution is non-uniform, with a higher probability near the initial point, but after 20 iterations the samples appear to have lost all memory of where they began. By sampling in latent space, the chain is able to take large steps in data space, even jumping directly between modes, with an overall acceptance rate in each case of around 40 per cent.

To quantify how many iterations are required to generate a new, independent sample, we calculate the effective sample size (ESS). Given a chain of $N_{\rm c}$ correlated samples $\{\boldsymbol{x}^i\}_{i=1}^{N_{\rm c}}$, the ESS is

$$\text{ESS} = \frac{N_{\rm c}}{1 + 2\sum_{s=1}^{N_{\rm c}-1}(1 - s/N_{\rm c})\boldsymbol{\rho}_s}, \tag{19}$$

where $\boldsymbol{\rho}_s$ is the autocorrelation of $\boldsymbol{x}$ at lag s. Since there is an autocorrelation and ESS for each parameter, we use the (worst-case) minimum ESS to set the chain length requirement. We use the following estimate for $\boldsymbol{\rho}_s$,

$$\hat{\boldsymbol{\rho}}_s = \frac{1}{\hat{\boldsymbol{\sigma}}^2(N_{\rm c} - s)}\sum_{n=s+1}^{N_{\rm c}}(\boldsymbol{x}^n - \hat{\boldsymbol{\mu}})(\boldsymbol{x}^{n-s} - \hat{\boldsymbol{\mu}}), \tag{20}$$

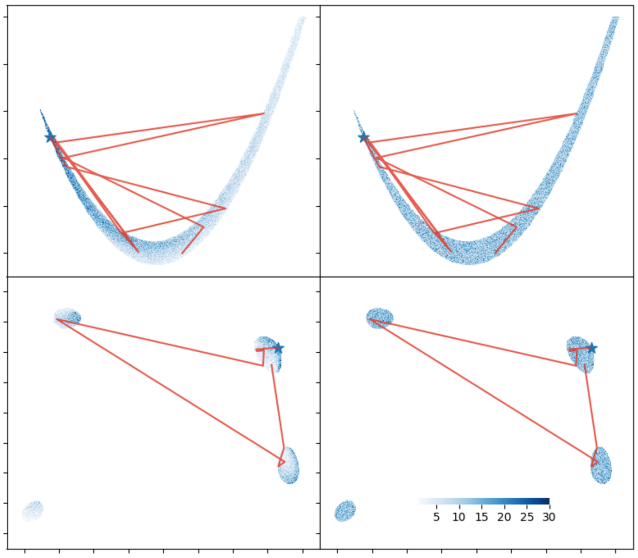

Figure 3. MCMC chains for (top) the $n_{\rm dim} = 2$ Rosenbrock function and (bottom) the Himmelblau function. On the left, we show the histogram of samples after 5 MCMC iterations, and on the right after 20 MCMC iterations. The initial position of all chains is indicated by a star. After 20 iterations, the samples appear to have lost all memory of where they began. We also show the first 20 proposed moves of an example chain.

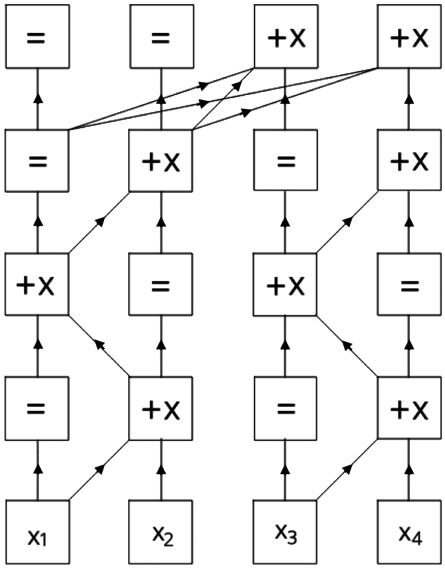

Figure 4. Illustration of transformations for a single fast–slow hierarchy, where x_1 and x_2 are slow parameters and x_3 and x_4 are fast parameters. The $+$ and $\times$ represent translation and scale operations of the NNs respectively, and $=$ indicates the input is unmodified. A sequence of three transformations is applied to the slow and fast subspaces, then they are decorrelated by applying a further transformation. Changes to only fast parameters do not change slow parameters.

where $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}^2$ are the mean and variance of the initial data. We truncate the sum over $\boldsymbol{\rho}_s$ when $\hat{\boldsymbol{\rho}}_s < 0.05$, as the estimate can become dominated by noise for large lags (Carlin & Louis 2008).

For the 2D Rosenbrock and Himmelblau functions, we obtain an average minimum ESS of $\sim$100 for $N_{\rm c} = 1000$. This suggests that, on average, it takes around 10 iterations to generate a new, independent sample. Empirically, we find this scales as $\sim n_{\rm dim}$ for higher dimensions.

3.3 Fast–slow decorrelation

In practice, the likelihood function can be computationally more expensive to evaluate for some parameters ('slow' parameters) than others ('fast' parameters). In astronomy/cosmology applications, for example, nuisance parameters are often much faster to evaluate than physical parameters of the model, when keeping physical parameters fixed. It is therefore desirable to split parameter space into a speed hierarchy, allowing for fast exploration of subspaces where it is inexpensive to evaluate the likelihood (Lewis 2013).

Fast–slow decorrelation can naturally incorporated into our method by fitting flows to each subspace and performing a further transformation to decorrelate them. In the case of a single hierarchy, for example, we fit separate flows to the slow ($\boldsymbol{x}_s$) and fast ($\boldsymbol{x}_f$) subspaces, concatenate the output into the vector ($\boldsymbol{y}_s, \boldsymbol{y}_f$), and then apply a transformation with mask ($\mathbf{1}, \mathbf{0}$). This means that slow parameters are unchanged by updating only the fast block, and a slow update changes both fast and slow parameters. This is illustrated in Fig. 4.

Given a speed hierarchy, we choose the sampling rate to be proportional to the number of parameters in each block. In the case of a single hierarchy, with $n_{\rm dim} = n_{\rm slow} + n_{\rm fast}$, where $n_{\rm slow}$ is the number of slow parameters and $n_{\rm fast}$ the number of fast parameters, at each MCMC iteration we perform a fast update with probability $n_{\rm fast}/(n_{\rm slow} + n_{\rm fast})$, otherwise performing a slow update. In our experiments, we find this leads to a minimum ESS similar to the full update of all parameters.

4 NEURAL NEST ALGORITHM

In this section, we give further details of our algorithm applied to nested sampling, although it could easily be incorporated into other MCMC frameworks. As outlined in Section 2, the algorithm begins by drawing $n_{\rm live}$ points from the prior distribution $\pi(\boldsymbol{x})$. At each iteration, the point with the lowest likelihood (denoted by $\lambda_\star$) is replaced by a new sample drawn from the prior, subject to the condition that $\mathcal{L}(\boldsymbol{x}) > \lambda_\star$.

To obtain a new, independent sample, we first train a flow on the current set of live points. Starting from an existing live point (chosen at random), we transform to latent space, perform sampling, do the reverse transformation and finally accept the new point after $n_{\rm MCMC}$ steps, with the requirement that it must have made at least one move (in practice it will make many moves). Further pertinent details of our implementation are:

Number of MCMC iterations: based on estimates of the ESS, we set $n_{\rm MCMC} = 5n_{\rm dim}$. Empirically, we find this works well for a range of target distributions. We monitor the ESS to ensure it does not significantly drop below 1 as the algorithm progresses, and that the chain performs a large number of updates by adjusting the proposal width as in equation (18).

Fast–slow hierarchy: we will consider models with either no hierarchy ($n_{\rm dim} = n_{\rm slow}$) or a single fast–slow hierarchy ($n_{\rm dim} = n_{\rm slow} + n_{\rm fast}$). In the latter case, we perform fast updates at each iteration with probability $n_{\rm fast}/(n_{\rm slow} + n_{\rm fast})$, otherwise performing a slow update that changes all parameters. On average, there will be approximately $5n_{\rm slow}$ slow likelihood evaluations per chain.

Initial rejection sampling: in the initial stages of selecting a new point, the prior volume $X \sim 1$. In this case, it is more efficient to use

rejection sampling from the prior hypercube. We switch to MCMC when the rejection efficiency is equal to the MCMC efficiency, that is after the prior volume has decreased by a factor of $1/(5n_{\rm slow})$.

Training updates: the set of live points changes relatively slowly, so we only retrain the flow every $n_{\rm live}$ iterations. We train each update for 50 epochs. Training is fast, taking <60 s on a CPU. The NN is trained by backpropagating the loss in equation (14) using the Adam optimizer (Kingma & Ba 2014).

Adding jitter: we add 'jitter' (random perturbations) to the set of live points during training to reduce overfitting. Jitter is chosen to be Gaussian with zero mean and a standard deviation of 0.2 times the average nearest neighbour separation between live points. The level of jitter therefore reduces as the algorithm progresses.

Validation data: during training, we use 90 per cent of the current live points to train the flow. The remaining 10 per cent are used as validation data to ensure the loss does not increase due to overfitting.

Termination: the algorithm is terminated on determining the fractional remaining Z to 0.5 in log-evidence (see Feroz & Hobson 2008 for details).

Parallelization: we parallelize our code using Message Passing Interface (MPI), communicating the set of live points between processes. Each process trains a separate flow to the (same) set of points, providing a type of ensembling across NNs. A new live point is generated by each process, and these are communicated back to the master process, which is responsible for updating the set of live points. Although this means that each process uses a slightly different proposal, this is valid as each new sample is independent and is uniformly drawn from the prior distribution, as shown in Fig. 3.

The NN was coded using the PYTORCH library[1] and the nested sampling code is available for download from https://github.com/adammoss/nnest.

5 RESULTS

5.1 Analytic likelihoods

We first test our method on several challenging analytic likelihoods:

Mixture of four Gaussians: this is the same multimodal distribution given in Higson et al. (2017), with a likelihood function

$$\mathcal{L}(\boldsymbol{x}) = \sum_{m=1}^{M} W^{(m)} \left(2\pi {\boldsymbol{\sigma}^{(m)}}^2\right)^{-d/2} \exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{\mu}^{(m)}|^2}{2{\boldsymbol{\sigma}^{(m)}}^2}\right). \quad (21)$$

We also consider $M = 4$, with weights $W^{(1)} = 0.4$, $W^{(2)} = 0.3$, $W^{(3)} = 0.2$, and $W^{(4)} = 0.1$. The only non-zero components of $\boldsymbol{\mu}$ are $\mu_2^{(1)} = -\mu_2^{(2)} = \mu_1^{(3)} = -\mu_1^{(4)} = 4$. The standard deviation is $\boldsymbol{\sigma}^{(m)} = 1$ for all m. We choose a uniform prior of $\mathcal{U}(-10, 10)$ on the parameters $\boldsymbol{x}$. The analytic expression for the evidence is $\log Z = -n_{\rm dim}\log 20$.

Rosenbrock function: the is the archetypal example of a banana-shaped degeneracy, with a log-likelihood

$$\log \mathcal{L}(\boldsymbol{x}) = -\sum_{i=1}^{n_{\rm dim}-1} \left[(1-x_i)^2 + 100\left(x_{i+1}-x_i^2\right)^2\right]. \quad (22)$$

We choose uniform priors $\mathcal{U}(-5, 5)$ on the parameters $\boldsymbol{x}$. The analytic evidence for $n_{\rm dim} = 2$ is $\log Z = -5.80$ (Graff et al. 2012).

[1] https://pytorch.org/

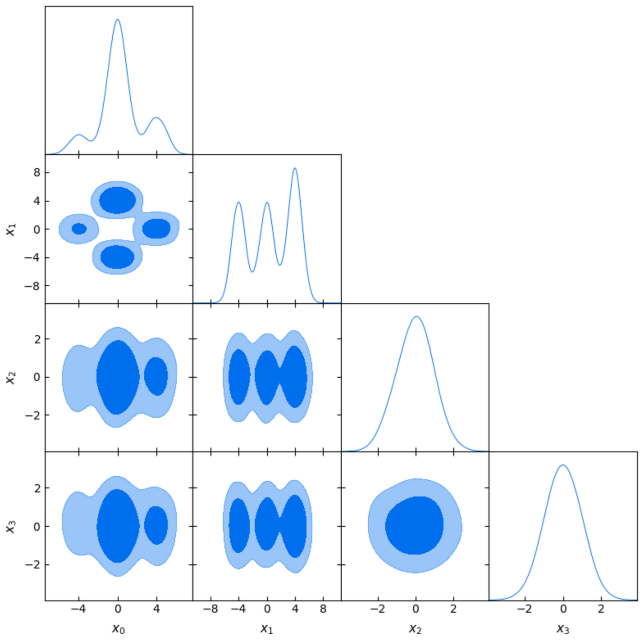

Figure 5. Marginalized 1D and 2D posterior distributions for the Gaussian mixture model with $n_{\rm dim} = 4$.

There is no analytic expression for $n_{\rm dim} > 2$, so for $n_{\rm dim} = 3$ we perform numerical integration to obtain the ground truth value. For higher dimensions, we found this too expensive to compute numerically.

Himmelblau function: this is an example of a multimodal distribution, with a log-likelihood

$$\log \mathcal{L}(\boldsymbol{x}) = -\left(x_1^2 + x_2 - 11\right)^2 - \left(x_1 + x_2^2 - 7\right)^2. \quad (23)$$

We also choose uniform priors $\mathcal{U}(-5, 5)$ on the parameters $\boldsymbol{x}$. The Himmelblau function has four identical local minima at (3, 2), (−2.81, 3.13), (−3.78, −3.28), and (3.58, −1.85). There is also no analytic expression for the evidence, so we perform numerical integration to obtain the ground truth value for Z.

In Fig. 5, we show the marginalized 1D and 2D posterior distributions for the Gaussian mixture model with $n_{\rm dim} = 4$. These marginalized values agree well with the expected values.

In Table 1, we show $\log Z$ and the number of likelihood evaluations for each of the three analytic likelihoods. We compare our results to the nested sampling codes MULTINEST (Feroz et al. 2009) and POLYCHORD (Handley et al. 2015). MULTINEST uses multimodal ellipsoidal rejection sampling, and POLYCHORD uses MCMC slice sampling with multimodal clustering. Each is run with their default settings,[2] and are set to stop on determining the fractional remaining log-evidence to 0.5.

In our code, we use five transformations in the NVP flow, each parametrized by a fully connected s and t network with an input layer of dimension 2, two hidden layers of dimension 128, and an output layer of dimension 2. We use ReLU functions after the input and hidden layers in each network. In all three codes, we set $n_{\rm live} = 1000$ and perform five separate runs to obtain summary statistics of $\log Z$ and the number of likelihood evaluations. Each code also produces an estimate of the evidence error for each run.

[2] An efficiency factor of 0.3 is used for MULTINEST and $n_{\rm repeats} = 5n_{\rm dim}$ for POLYCHORD.

Table 1. Average $\log Z$ (top line of each entry), number of slow likelihood evaluations (second line of each entry) and effective number of posterior samples (third line of each entry) for the analytic likelihoods. Values and errors are averaged over five runs. Ground truth values denoted by a * were obtained by numerical integration. POLYCHORD is also capable of a fast–slow hierarchy.

Likelihood	$n_{\rm slow}$	$n_{\rm fast}$	Ground truth	MULTINEST (Feroz et al. 2009)	POLYCHORD (Handley et al. 2015)	Ours
Gaussian mix.	5	0	−14.98	−14.94 ± 0.04 34 459 5389	−14.91 ± 0.06 949 706 3791	−14.91 ± 0.05 139 755 5141
Gaussian mix.	10	0	−29.96	−29.90 ± 0.06 73 041 7710	−29.97 ± 0.10 38 11 072 3588	−29.96 ± 0.09 582 780 7431
Gaussian mix.	20	0	−59.91	−59.21 ± 0.12 249 826 11 092	−59.91 ± 0.22 151 46 455 1728	−59.95 ± 0.16 25 77 875 10 747
Gaussian mix.	30	0	−89.87	−88.42 ± 0.09 697 089 13 633	−89.85 ± 0.33 336 50 878 1516	−89.79 ± 0.10 63 40 351 12 871
Gaussian mix.	2	3	−14.98	N/A	−14.98 ± 0.12 381 181 3812	−15.01 ± 0.04 58 460 5040
Gaussian mix.	2	8	−29.96	N/A	−30.01 ± 0.18 767 216 3530	−29.93 ± 0.08 121 668 7324
Gaussian mix.	2	18	−59.91	N/A	−59.72 ± 0.08 15 04 787 2185	−59.80 ± 0.17 261 827 10 693
Gaussian mix.	2	28	−89.87	N/A	−89.75 ± 0.27 22 41 159 1 218	−89.72 ± 0.06 425 564 12 732
Himmelblau	2	0	−5.54 ± 0.02*	−5.51 ± 0.06 21 110 3360	−5.44 ± 0.04 (259 506) 3,216	−5.48 ± 0.08 (47 880) 3162
Rosenbrock	2	0	−5.80	−5.83 ± 0.03 21 612 3224	−5.82 ± 0.12 3 40 022 2894	−5.77 ± 0.08 42 173 3026
Rosenbrock	3	0	−10.46 ± 0.03*	−10.46 ± 0.09 37 658 4089	−10.41 ± 0.11 7 75 309 3539	−10.44 ± 0.13 97 648 3820
Rosenbrock	4	0	N/A	−14.94 ± 0.11 58 606 5017	−15.13 ± 0.11 14 43 530 3668	−15.14 ± 0.09 1 95 704 4738
Rosenbrock	5	0	N/A	−19.63 ± 0.08 78 346 5895	−19.82 ± 0.08 23 08 802 3843	−19.67 ± 0.04 3 19 325 5557
Rosenbrock	10	0	N/A	−42.13 ± 0.10 3 85 446 9854	−42.81 ± 0.45 97 42 368 3003	−43.04 ± 0.18 14 68 468 8965
Rosenbrock	20	0	N/A	−87.67 ± 0.31 64 59 763 16 061	−91.63 ± 0.95 380 47 353 1455	−91.83 ± 0.29 68 03 067 16 524
Rosenbrock	30	0	N/A	−134.05 676 12 863 10 607	−138.79 ± 1.74 877 61 897 787	−141.81 ± 0.28 163 05 276 21 011

For the Gaussian mixture model, we obtain results consistent with the ground truth values. The number of required likelihood evaluations is around a factor of 5–10 higher (i.e. less efficient) than MULTINEST, primarily due to the $n_{\rm MCMC} = 5n_{\rm dim}$ iterations we perform to obtain a new, independent sample. In contrast, the number of evaluations required per sample for MULTINEST is only ∼3. Using default settings, however, MULTINEST tends to overestimate $\log Z$ for $n_{\rm dim} \geq 10$ – this can be alleviated by decreasing the efficiency parameter, but at the cost of more likelihood evaluations. Decreasing the efficiency to 0.05, for example, gives $\log Z = -59.92 \pm 0.07$ for $n_{\rm dim} = 20$, with an average 703 182 likelihood evaluations, and $\log Z = -89.95 \pm 0.15$ for $n_{\rm dim} = 30$, with 962 111 likelihood evaluations. This means that MULTINEST is still a factor ∼5 times more efficient. Our error estimate in $\log Z$ using equation (7) is consistent with our summary statistics over five runs, being 0.08, 0.12, 0.17, and 0.21 for $n_{\rm dim} = 5$, 10, 20, and 30, respectively.

We also consider the same Gaussian mixture model but with a fast/slow hierarchy. In this case, we choose x_1 and x_2 to be slow parameters, with the remainder fast parameters. We take the number of likelihood evaluations to be the number of *slow evaluations*. This number is now significantly reduced and is comparable to MULTINEST at low dimensions, which does not implement any speed hierarchy, and is more efficient at high dimensions. One advantage of POLYCHORD is that it is capable of a hierarchy with multiple 'speeds'. For comparison, we use a single fast/slow hierarchy, with the number of repeats $n_{\rm repeats} = 5n_{\rm slow}$ and a grade fraction of $n_{\rm slow}/(n_{\rm slow} + n_{\rm fast})$ for the two slow parameters. The grade fraction for the fast parameters is set to $n_{\rm fast}/(n_{\rm slow} + n_{\rm fast})$, meaning there will be approximately $n_{\rm fast}/n_{\rm slow}$ more fast repeats. This makes the fast/slow hierarchy comparable to ours. Results are given in Table 1 and show a similar improvement in the number of slow likelihood evaluations. The evidence values and posterior distributions are unaffected in both codes by using a fast/slow hierarchy.

For the Himmelblau function, we obtain results consistent with the ground truth value. The number of likelihood evaluations is now only around a factor of 2 higher than MULTINEST. This is an improvement over the mixture model at the same dimension, as ellipsoidal rejection sampling is less efficient for non-Gaussian distributions.

For the Rosenbrock function, we also obtain results consistent with the available ground truth values. For $n_{\rm dim} = 2$, the efficiency is within a factor of 2 of MULTINEST, and for $n_{\rm dim} \geq 20$, the poor scaling of rejection sampling becomes apparent. There is a difference in $\log Z$ compared to MULTINEST for $n_{\rm dim} \geq 20$, and although the results of POLYCHORD are consistent with ours, the lack of a ground truth value makes it difficult to draw conclusions. Our error estimate in $\log Z$ using equation (7) is again consistent with our summary statistics, being 0.07, 0.09, 0.11, 0.13, 0.19, 0.28, and 0.34 for $n_{\rm dim} =$ 2, 3, 4, 5, 10, 20, and 30 respectively.

Compared to POLYCHORD, also an MCMC sampler, our method requires a lower number of likelihood evaluations, by a factor of ∼5. POLYCHORD also performs $n_{\rm MCMC} = 5n_{\rm dim}$ repeats to obtain a new sample, but requires additional evaluations to determine the width of the slice. Recently, DYPOLYCHORD (Higson et al. 2017) has been developed, which dynamically allocates live points during sampling. We have not performed direct comparisons with DYPOLYCHORD in Table 1, as we wish to compare the efficiency of each algorithm with a constant number of live points. From our experiments, however, DYPOLYCHORD has similar performance to POLYCHORD at low dimensions, but significantly improves the scaling at higher dimensions. For the Rosenbrock function, for example, an average of 66 60 409 likelihood evaluations are required for $n_{\rm dim} = 20$ and only 91 82 957 for $n_{\rm dim} = 30$.

Other MCMC-based results in the literature are limited, but in Feroz & Skilling (2013), it was shown that Galilean dynamics required around 120 000 and 220 000 likelihood evaluations for the Himmelblau and $n_{\rm dim} = 2$ Rosenbrock functions, respectively.

Finally, it is also instructive to evaluate the performance at estimating the posterior distribution. In nested sampling, the effective number of posterior samples, $N_{\rm eff}$, can be estimated using equation (9). The larger the effective number of samples, the smaller the MCMC error. In Table 1, we give $N_{\rm eff}$ for each of the three methods on the different analytic likelihoods. The sampling efficiency, given by the ratio of $N_{\rm eff}$ to the number of likelihood evaluations, can become small for high dimensions, but this is a generic issue for nested sampling, as the posterior weight is small for much of the prior volume. We obtain similar $N_{\rm eff}$ compared to MULTINEST, and this can be also verified qualitatively by comparing the posterior distributions.

5.2 *Planck*

Our PYTHON implementation can easily be integrated with codes such as MONTEPYTHON (Audren et al. 2013; Brinckmann & Lesgourgues 2019) and COBAYA[3] to perform cosmological parameter estimation and model selection. The *Planck* data sets used in our analysis come from the 2015 mission (Aghanim et al. 2016; Ade et al. 2016a). In particular, we use the TT+lowP+lensing combination, which contains the 100, 143, and 217-GHz binned half-mission TT cross-spectra for $\ell = 30 - 2508$ with cosmic microwave background-cleaned 353-GHz map, CO emission maps, and *Planck* catalogues for the masks and 545-GHz maps for the dust residual contamination template. It also uses the joint temperature and the E and B cross-spectra for $\ell = 2 - 29$ with E and B maps from the 70-GHz low-frequency instrument (LFI) full-mission data and foreground contamination determined by 30-GHz LFI and 353-GHz high-frequency instrument maps. The *Planck* lensing likelihood (Ade et al. 2016b) uses both temperature and polarization data in the multipole range $\ell = 100 - 2048$ to estimate the lensing power spectrum.

We use the full version of the *Planck* likelihood with MONTEPYTHON, fitting for a total of 6 base L-cold dark matter (LCDM) parameters and 15 nuisance parameters. We also fit for simple one-parameter extensions to LCDM with a variable effective number of neutrino species $N_{\rm eff}$ and curvature density $\Omega_{\rm K}$. We assume uniform priors on the cosmological parameters, with the upper and lower limits corresponding to approximate *Planck* $\pm 5\sigma$ values. To account for any Gaussian priors on nuisance parameters used in the *Planck* analysis, we use uniform priors with $\pm 5\sigma$ limits, and add an additional term to the likelihood function. The prior ranges, along with a description of each parameter, are shown in Table 2.

For *Planck*, the total computational time is dominated by the calculation of the cosmological observables, so we use a fast hierarchy for the nuisance parameters. We use $n_{\rm live} = 500$ points and the same network architecture as in the previous section. In Table 3, we give the marginalized cosmological parameters for the base LCDM, LCDM + $N_{\rm eff}$, and LCDM + $\Omega_{\rm K}$ models. These agree very well with the published *Planck* values, and in Fig. 6 we show marginalized 1D and 2D posterior distributions for the base LCDM model, compared to results from standard MCMC. These again

[3] https://github.com/CobayaSampler/cobaya

Table 2. Prior ranges for the base LCDM (top), one-parameter extensions (middle), and nuisance parameters (bottom), together with the resulting posterior values.

Lower	Parameter	Upper	Description
0.0211	$\leq\Omega_b h^2\leq$	0.0234	Physical baryon density
0.109	$\leq\Omega_c h^2\leq$	0.131	Physical CDM density
1.038	$\leq 100\Theta_s\leq$	1.044	Ratio of angular diameter distance to sound horizon
2.91	$\leq\ln(10^{10}A_s)\leq$	3.27	Scalar amplitude
0.93	$\leq n_s\leq$	1.0	Scalar spectral index
0.05	$\leq\tau\leq$	0.15	Optical depth to reionization
1.5	$\leq N_{\rm eff}\leq$	4.5	Effective number of neutrinos
−0.1	$\leq\Omega_K\leq$	0.05	Curvature density
0 μK^2	$\leq A_{217}^{\rm CIB}\leq$	200 μK^2	CIB amplitude at 217 GHz
0 μK^2	$\leq A^{\rm kSZ}\leq$	10 μK^2	kSZ amplitude at 143 GHz
0 μK^2	$\leq A_{143}^{\rm tSZ}\leq$	10 μK^2	tSZ amplitude at 143 GHz
0	$\leq\xi^{\rm tSZ\times CIB}\leq$	1	tSZ–CIB template amplitude
0 μK^2	$\leq A_{100}^{\rm PS}\leq$	400 μK^2	Point source amplitude at 100 GHz
0 μK^2	$\leq A_{143}^{\rm PS}\leq$	400 μK^2	Point source amplitude at 143 GHz
0 μK^2	$\leq A_{143\times217}^{\rm PS}\leq$	400 μK^2	Point source amplitude at 143x217 GHz
0 μK^2	$\leq A_{217}^{\rm PS}\leq$	400 μK^2	Point source amplitude at 217 GHz
0 μK^2	$\leq A_{100}^{{\rm dust}TT}\leq$	17 μK^2	Dust amplitude at 100 GHz
0 μK^2	$\leq A_{143}^{{\rm dust}TT}\leq$	19 μK^2	Dust amplitude at 143 GHz
0 μK^2	$\leq A_{143\times217}^{{\rm dust}TT}\leq$	63.5 μK^2	Dust amplitude at 143 × 217 GHz
0 μK^2	$\leq A_{217}^{{\rm dust}TT}\leq$	180 μK^2	Dust amplitude at 217 GHz
0.994	$\leq c_{100}\leq$	1.004	Calibration factor for 100/143 GHz
0.985	$\leq c_{217}\leq$	1.005	Calibration factor for 217/143 GHz
0.9875	$\leq y_{\rm cal}\leq$	1.0125	Total *Planck* calibration

Table 3. Marginalized values for the base LCDM, LCDM + $N_{\rm eff}$, and LCDM + Ω_K cosmological parameters.

Parameter	Base LCDM	$+N_{\rm eff}$	$+\Omega_K$
$\Omega_b h^2$	$0.02229^{+0.00023}_{-0.00024}$	$0.02237^{+0.00033}_{-0.00033}$	$0.02233^{+0.00024}_{-0.00027}$
$\Omega_c h^2$	$0.1182^{+0.0019}_{-0.0020}$	$0.1190^{+0.0042}_{-0.0036}$	$0.1176^{+0.0022}_{-0.0024}$
$100\Theta_s$	$1.0420^{+0.0004}_{-0.0004}$	$1.0420^{+0.0007}_{-0.0007}$	$1.0420^{+0.0005}_{-0.0005}$
$\ln(10^{10}A_s)$	$3.078^{+0.020}_{-0.032}$	$3.084^{+0.030}_{-0.041}$	$3.070^{+0.021}_{-0.036}$
n_s	$0.9690^{+0.0055}_{-0.0061}$	$0.9726^{+0.014}_{-0.013}$	$0.9708^{+0.0066}_{-0.0069}$
τ	$0.073^{+0.010}_{-0.018}$	$0.076^{+0.012}_{-0.020}$	$0.0703^{+0.006}_{-0.020}$
$N_{\rm eff}$	3.046	$3.11^{+0.30}_{-0.29}$	3.046
Ω_K	0.0	0.0	$-0.0042^{+0.0089}_{-0.0073}$

agree extremely well, showing that we obtain accurate parameter constraints using our nested sampler.

We have also calculated the evidence, finding the Bayes factor to be $\log B = -1.7 \pm 0.2$ and -2.1 ± 0.2 for LCDM + $N_{\rm eff}$ and LCDM + Ω_K, respectively. The error on the Bayes factor is obtained from adding the log-evidence errors in quadrature. In the revised Jeffreys scale (Kass & Raftery 1995), $|\log B| > 1$ is regarded as positive evidence, $|\log B| > 3$ as strong evidence, and $|\log B| > 5$ as very strong. These results therefore suggest that *Planck* disfavours both extensions to LCDM. Although the evidence is dependant on the choice of priors, our results are consistent with those in Heavens et al. (2017), who reuse MCMC chains produced for parameter inference to calculate the evidence.

6 CONCLUSIONS

In this paper, we have trained an NN to parametrize efficient MCMC proposals, by transforming the target distribution to a simpler latent representation. This approach is inspired by representative learning, which suggests that deep NNs yield latent spaces in which Markov chains can mix faster. Our method is a non-linear extension of the commonly used technique of transforming parameter space using the Cholesky decomposition of the covariance matrix.

We have applied this method to nested sampling, finding excellent performance on highly curved and multimodal targets. At low dimensions, the efficiency is within a factor of a few times that of state-of-the-art multimodal rejection sampling, but has better scaling in higher dimensions. Parameter space can also naturally be split into a speed hierarchy, making it suitable for models with a subset of parameters where it is inexpensive to evaluate the likelihood. We demonstrate this for *Planck* data in ∼20D parameter space, accurately recovering the expected posterior distributions. As an example, we calculate the Bayesian evidence for variable effective number of neutrino species $N_{\rm eff}$ and curvature density Ω_K, finding the data disfavours these extensions to LCDM.

There are several possibilities for future work. First, it would be interesting to see if the flow model can more naturally be extended to multimodal distributions. Currently, the latent representation forms narrow connecting ridges between modes, which reduces the efficiency on models with a very high number of modes. One could also potentially use more general types of NN, but these may not have the desirables properties of being invertible with tractable Jacobian determinants.

In terms of nested sampling, it has recently been shown that dynamically allocating the number of live points can significantly improve performance (Higson et al. 2017). It would be interesting to apply this technique to our method, potentially even using an NN to estimate the posterior mass $\mathcal{L}(X)X$ and control the allocation of points.

In follow-up work, we will develop an NN sampler specifically designed for fast inference, that can easily be integrated into standard parameter estimation codes. This would improve on the

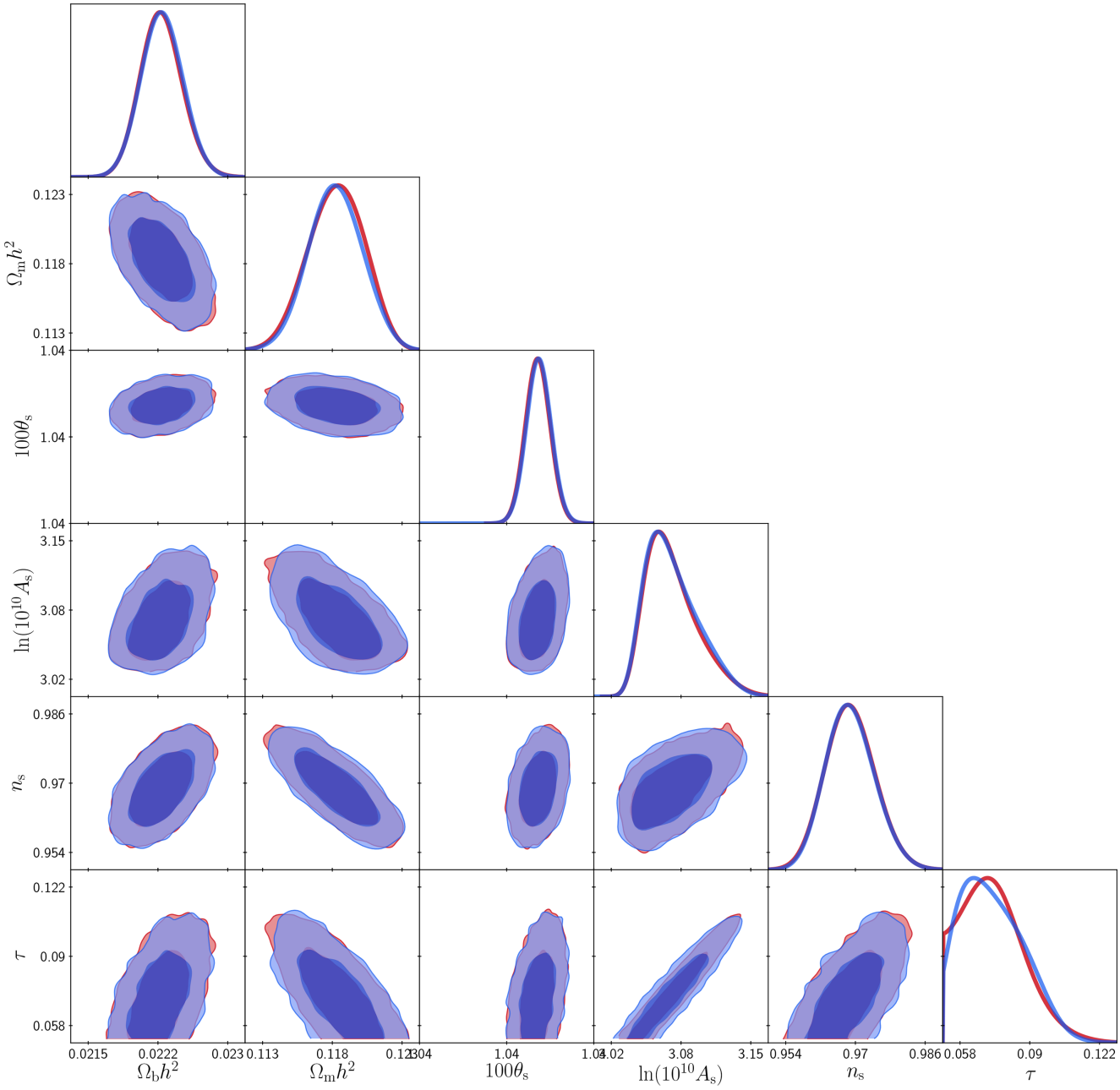

Figure 6. Marginalized 1D and 2D posterior distributions for the base LCDM cosmological parameters. In red, we show results from our nested sampling method, and in blue results from standard parameter estimation using Metropolis–Hastings MCMC.

standard technique of using the covariance matrix to parametrize the proposal function, working for both highly curved and multimodal likelihoods. With the ability of NNs to characterize complex data by simple representations, we expect they will become useful tools to improve the speed of inference on a variety of problems.

ACKNOWLEDGEMENTS

We appreciate helpful conversations with Steven Bamford, Simon Dye, Juan Garrahan and Dominic Rose, and Will Handley for very useful comments on the nested sampling algorithm. We also thank the referee (Joe Zuntz) for very helpful comments to improve the manuscript. AM is supported by a Royal Society University Research Fellowship.

REFERENCES

Ade P. A. R. et al., 2016a, A&A, 594, A13
Ade P. A. R. et al., 2016b, A&A, 594, A15
Aghanim N. et al., 2016, A&A, 594, A11
Audren B., Lesgourgues J., Benabed K., Prunet S., 2013, J. Cosmol. Astropart. Phys., 1302, 001
Bengio Y., Courville A., Vincent P., 2013, IEEE transactions on pattern analysis and machine intelligence, 35, 1798
Bengio Y., Mesnil G., Dauphin Y., Rifai S., 2013, International conference on machine learning, 552
Brewer B. J., , Pártay L. B., Csányi G., 2011, Statistics and Computing, 21, 649
Brinckmann T., Lesgourgues J., 2019, Phys. Dark Universe, 24, 100260

Carlin B. P., Louis T. A., 2008, Bayesian Methods for Data Analysis. CRC Press, Boca Raton, FL
Dinh L., Sohl-Dickstein J., Bengio S., 2016, preprint (arXiv:1605.08803)
Feroz F., Hobson M. P., 2008, MNRAS, 384, 449
Feroz F., Skilling J., 2013, in Toussaint U. V., ed., AIP Conf. Ser. Vol. 1553, Am. Inst. Phys., New York, p. 106
Feroz F., Hobson M. P., Bridges M., 2009, MNRAS, 398, 1601
Feroz F., Hobson M. P., Cameron E., Pettitt A. N., 2013, preprint (arXiv:1306.2144)
Foreman-Mackey D. et al., 2013, Astrophysics Source Code Library , record ascl:1303.002
Gelman A., Roberts G. O., Gilks W. R., 1996, in Oxford Sci. Publ., Bayesian Statistics, 5 (Alicante, 1994). Oxford Univ. Press, New York, p. 599
Goodman J., Weare J., 2010, Commun. Appl. Math. Comput. Sci., 5, 65
Graff P., Feroz F., Hobson M. P., Lasenby A., 2012, MNRAS, 421, 169
Haario H., Saksman E., Tamminen J., 2001, Bernoulli, 7, 223
Handley W. J., Hobson M. P., Lasenby A. N., 2015, MNRAS, 453, 4384
Handley W., Scott P., White M., 2019, DarkMachines/pyBAMBI: pyBAMBI beta
Heavens A., Fantaye Y., Sellentin E., Eggers H., Hosenie Z., Kroon S., Mootoovaloo A., 2017, Phys. Rev. Lett., 119, 101301
Higson E., Handley W., Hobson M., Lasenby A., 2019, Statistics and Computing, 29, 891
Kass R. E., Raftery A. E., 1995, J. Am. Stat. Assoc., 90, 773
Kingma D., Ba J., 2014, preprint (arXiv:1412.6980)
Lecun Y., Bengio Y., Hinton G., 2015, Nature, 521, 436
Levy D., Hoffman M. D., Sohl-Dickstein J., 2017, preprint (arXiv:1711.09268)
Lewis A., 2013, Phys. Rev. D, 87, 103529
Lewis A., Bridle S., 2002, Phys. Rev. D, 66, 103511
Mukherjee P., Parkinson D., Liddle A. R., 2006, ApJ, 638, L51
Skilling J. et al., 2006, Bayesian Anal., 1, 833
Skilling J., 2004, in Fischer R., Preuss R., Toussaint U. V., eds, AIP Conf. Ser. Vol. 735, Am. Inst. Phys., New York, p. 395
Song J., Zhao S., Ermon S., 2017, Advances in Neural Information Processing Systems 30, 5140
Zuntz J. et al., 2015, Astron. Comput., 12, 45

This paper has been typeset from a TEX/LATEX file prepared by the author.