Embarrassingly parallel MCMC using deep invertible transformations

Diego Mesquita **Paul Blomstedt** **Samuel Kaski**
Helsinki Institute for Information Technology HIIT, Department of Computer Science, Aalto University
{diego.mesquita, paul.blomstedt, samuel.kaski}@aalto.fi

Abstract

While MCMC methods have become a main work-horse for Bayesian inference, scaling them to large distributed datasets is still a challenge. Embarrassingly parallel MCMC strategies take a divide-and-conquer stance to achieve this by writing the target posterior as a product of subposteriors, running MCMC for each of them in parallel and subsequently combining the results. The challenge then lies in devising efficient aggregation strategies. Current strategies trade-off between approximation quality, and costs of communication and computation. In this work, we introduce a novel method that addresses these issues simultaneously. Our key insight is to introduce a deep invertible transformation to approximate each of the subposteriors. These approximations can be made accurate even for complex distributions and serve as intermediate representations, keeping the total communication cost limited. Moreover, they enable us to sample from the product of the subposteriors using an efficient and stable importance sampling scheme. We demonstrate that the approach outperforms available state-of-the-art methods in a range of challenging scenarios, including high-dimensional and heterogeneous subposteriors.

1 INTRODUCTION

Markov Chain Monte Carlo (MCMC) algorithms have cemented themselves as a cornerstone of practical Bayesian analysis. Nonetheless, accommodating large distributed datasets is still a challenge. For this purpose, methods have been proposed to speed up inference either using mini-batches (e.g. Ma et al., 2015; Quiroz et al., 2018) or exploiting parallel computing (e.g. Ahn et al., 2014; Johnson et al., 2013), or combinations thereof. For a comprehensive review about scaling up Bayesian inference, we refer to Angelino et al. (2016) and Robert et al. (2018).

A particularly efficient class of parallel algorithms are embarrassingly parallel MCMC methods, which employ a divide-and-conquer strategy to obtain samples from the posterior

$$p(\theta|\mathcal{D}) \propto p(\theta)p(\mathcal{D}|\theta),$$

where $p(\theta)$ is a prior, $p(\mathcal{D}|\theta)$ is a likelihood function and the data $\mathcal{D}$ are partitioned into K disjoint subsets $\mathcal{D}_1, \ldots, \mathcal{D}_K$. The general idea is to break the global inference into smaller tasks and combine their results, requiring coordination only in the final aggregation stage. More specifically, the target posterior is factorized as

$$p(\theta|\mathcal{D}) \propto \prod_{k=1}^{K} p(\theta)^{1/K} p(\mathcal{D}_k|\theta), \tag{1}$$

and the right-hand-side factors, referred to as *subposteriors*, are independently sampled from—in parallel—using an MCMC algorithm of choice. The results are then centralized in a coordinating server and aggregated. The core challenge lies in devising strategies which are both accurate and computationally convenient to combine subposterior samples.

The seminal work of Scott et al. (2016) approximates posterior samples as weighted averages of subposterior samples. Neiswanger et al. (2014) proposed parametric, semi-parametric and non-parametric strategies, the two former being based on fitting kernel density estimators to the subposterior samples. Wang et al. (2015) used random partition trees to learn a discrete approximation to the posterior. Nemeth and Sherlock (2018) fitted Gaussian process approximations to the log-subposteriors and took the product of their expected values. Except for the parametric method, which imposes overly simplistic local approximations that generally result in poor approximations of the target posterior, all of the aforementioned approaches require the subposterior samples to be centralized, incurring extensive communication costs. In fact, communication costs have been altogether ignored in the literature so far. Furthermore, sampling from the approximate posterior can become difficult, requiring expensive additional MCMC steps to obtain samples from the combined posterior.

In this work, we propose a novel embarrassingly parallel MCMC strategy termed *non-volume-preserving aggregation product* (NAP), which addresses the aforementioned issues while providing accurate posterior samples. Our work builds on the insight that subposteriors of arbitrary complexity can be mapped to densities of tractable form, making use of *real non-volume preserving trasformations* (real NVP), a recently developed class of neural-network based invertible transformations (Dinh et al., 2017). This enables us to accurately evaluate the subposterior densities and sample from the combined posterior using importance sampling. We prove that, under mild assumptions, our importance sampling scheme is stable, i.e., estimates for a test function h have finite variance.

Experimental results show that NAP outperforms state-of-the art methods in several situations, including heterogeneous subposteriors and intricate-shaped, multi-modal or high-dimensional posteriors. Finally, the proposed strategy results in communication costs which are constant in the number of subposterior samples, which is an appealing feature when communication between machines holding data shards and the server is expensive or limited.

The remainder of this work proceeds as follows. Section 2 introduces our method, covering the required background on real NVP transformations. Section 3 presents experimental results. We conclude with a discussion on the results and possible unfoldings of this work in Section 4.

2 METHOD

In this work, we employ real NVP transformations to approximate subposteriors using samples obtained from independent MCMC runs. In the following subsections, we 1) review the basics of real NVP transformations; 2) discuss how to combine them using importance sampling and 3) how to obtain samples from the approximate posterior using sampling/importance resampling.

2.1 REAL NVP DENSITY ESTIMATION

Real NVP (Dinh et al., 2017) is a class of deep generative models in which a D-dimensional real-valued quantity of interest x is modeled as a composition of bijective transformations from a base latent variable z, with known density function p_Z, *i.e.*:

$$x = g_L \circ g_{L-1} \circ \ldots \circ g_1(z) = g(z),$$

such that $g_l : \mathbb{R}^D \to \mathbb{R}^D$ for all $1 \leq l \leq L$. The density $p_X(x)$ is then obtained using the change-of-variable formula

$$p_X(x) = p_Z\big(f(x)\big)\left|\det \frac{\partial f(x)}{\partial x^\top}\right|, \tag{2}$$

where

$$f = f_1 \circ f_2 \circ \ldots \circ f_L = g_1^{-1} \circ g_2^{-1} \circ \ldots \circ g_L^{-1} = g^{-1}.$$

To make (2) tractable, it is composed as follows. Let $\mathcal{I}_l \subset \{1, \ldots, D\}$ be a pre-defined proper subset of indices with cardinality $|\mathcal{I}_l|$, and denote its complement by $\overline{\mathcal{I}}_l$. Then, each transformation $v' = f_l(v)$ is computed as:

$$\begin{aligned} v'_{\mathcal{I}_l} &= v_{\mathcal{I}_l} \\ v'_{\overline{\mathcal{I}}_l} &= v_{\overline{\mathcal{I}}_l} \odot \exp\{s_l(v_{\mathcal{I}_l})\} + t_l(v_{\mathcal{I}_l}), \end{aligned} \tag{3}$$

where $\odot$ is an element-wise product. The functions $s_l, t_l : \mathbb{R}^{|\mathcal{I}_l|} \to \mathbb{R}^{|\overline{\mathcal{I}}_l|}$ are deep neural networks, which perform scale and translation, respectively. In particular, the Jacobian of f_l, has the form

$$\frac{\partial v'}{\partial v} = \begin{bmatrix} \mathbb{I}_{|\mathcal{I}_l|} & 0 \\ \frac{\partial v'_{\overline{\mathcal{I}}_l}}{\partial v_{\mathcal{I}_l}} & \operatorname{diag}\big(\exp\{s_l(v_{\mathcal{I}_l})\}\big) \end{bmatrix},$$

which avoids explicit computation of the Jacobian of the functions s_l and t_l. For observed data $(x_1, \ldots, x_N)$, the weights of the networks s_l and t_l that implicitly parameterize p_X are estimated via maximum likelihood.

Sampling from p_X is inexpensive and resumes to sampling $z \sim p_Z$, and computing $x = g(z)$. Here, each g_l is of the form

$$\begin{aligned} v_{\mathcal{I}_l} &= v'_{\mathcal{I}_l} \\ v_{\overline{\mathcal{I}}_l} &= (v'_{\overline{\mathcal{I}}_l} - t_l(v'_{\mathcal{I}_l})) \odot \exp\{-s_l(v'_{\mathcal{I}_l})\}, \end{aligned} \tag{4}$$

where (4) is obtained from (3) by straightforward inversion.

2.2 COMBINING LOCAL INFERENCES

Consider now a factorization of a target posterior density $p(\theta|\mathcal{D})$ into a product of K subposteriors according to Equation (1). In embarrassingly parallel MCMC, each worker runs MCMC independently on its respective subposterior,

$$p_k(\theta) := \frac{1}{Z_k} p(\theta)^{1/K} p(\mathcal{D}_k|\theta),$$

to obtain a set of draws $\{\theta_s^{(k)}\}_{s=1}^S$ from $p_k(\theta)$. The goal is then to produce draws from an approximate target posterior $\widehat{p}(\theta) \approx p(\theta|\mathcal{D})$, using the K sets of subposterior samples as input. This requires estimating the densities $p_k(\theta)$ from the subposterior samples, and sampling from the distribution induced by the product of approximations

$$\widehat{p}(\theta) \propto \widehat{p}_1(\theta)\widehat{p}_2(\theta) \ldots \widehat{p}_K(\theta), \tag{5}$$

typically resulting in a trade-off between accuracy and computational efficiency.

In this work, we make use of the fact that bijective transformations using real NVP offers both accurate density estimation and computationally efficient sampling for arbitrarily complex distributions. To this end, we first fit a separate real NVP network to estimate each p_k as $\widehat{p}_k$. The networks are then sent to a server that approximates the global posterior as in Equation (5).

In a typical scenario, one would ultimately be interested in using $\widehat{p}$ to compute the expectation of some function $h : \mathbb{R}^D \to \mathbb{R}$, such as a predictive density or a utility function. In our case, straightforward importance sampling can be used to weight samples drawn from any of the subposteriors. Thus, given a set of T samples drawn from any $\widehat{p}_k$, we obtain the estimate:

$$\overline{h}(\theta) = \sum_{t=1}^{T} w_t h(\theta_t),$$

where the importance weights $w_1, \ldots, w_T$ are normalized to sum to one, and given by

$$w_t \propto \frac{\prod_{k'=1}^{K} \widehat{p}_{k'}(\theta_t)}{\widehat{p}_k(\theta_t)}. \tag{6}$$

This strategy capitalizes on the key properties of real NVP transformations—ease of evaluation and sampling—and avoids the burden of running still more MCMC chains to sample from the aggregated posterior $\widehat{p}(\theta)$, which might be a complicated target due to the underlying neural networks.

While importance sampling estimates can be unreliable if their variance is very high or infinite, we can provide guarantees that $\overline{h}(\theta)$ has finite variance. Geweke (1989) showed that, for a broad class of test functions, it suffices to prove that $\prod_{k'=1}^{K} \widehat{p}_{k'}(\theta)/\widehat{p}_k(\theta) \leq M \; \forall\theta$, i.e., the importance weights are bounded. We first note that the denominator of the weight in Equation (6) is included as a factor in the numerator, so that $\widehat{p}_k(\theta) = 0 \Rightarrow \prod_{k'=1}^{K} \widehat{p}_{k'}(\theta) = 0$. The remaining thing to check is that $\widehat{p}_k(\theta)$ is bounded for all k and all θ.

We begin by making the following assumption on the structure of the neural networks which define the real NVP transformations.

Assumption 1. *The neural networks $s_1^{(k)}$, …, $s_L^{(k)}$ associated with the real NVP estimate $\widehat{p}_k$ are equipped with bounded activation functions in their individual output layers.*

Remark 1. *Note that Assumption 1 is satisfied, for example, when the activation functions in the last layer of the scale networks are the hyperbolic tangent or the logistic function.*

We place no further assumption on the structure of the remaining layers of $s_1^{(k)}, \ldots, s_L^{(k)}$ or in the overall structure of the translation networks $t_1^{(k)}, \ldots, t_L^{(k)}$.

With the additional condition that we choose an appropriate density for the base variable of the NVP network, we can prove that $\widehat{p}_k$ itself is bounded.

Lemma 2.1. *Given a bounded base density p_Z, the distribution resulting from L transformations is bounded.*

Proof. As p_Z is bounded, there exists some constant $M > 0$ such that

$$p_Z(z) \leq M \quad \forall z \in \mathbb{R}^D.$$

Let $v_1 = g_1(z)$. Applying the change-of-variable formula we get

$$\begin{aligned}\log p_{v_1}(v_1) &= \log p_Z(z) + \log \left| \det \frac{\partial z}{\partial v_1^{\top}} \right| \\ &= \log p_Z(z) + \left| \mathrm{Tr}\ \mathrm{diag}\left(s_l(v_{\mathcal{I}_l})\right) \right|\end{aligned}$$

Let $\mathcal{B}_z > 0$ be the constant bounding p_Z. Using Assumption 1, since all of the outputs of the neural networks s_l are bounded, their sum is bounded by some $\mathcal{B}_s > 0$. Then, it follows:

$$\log p_{v_1}(v_1) \leq \mathcal{B}_z + \mathcal{B}_s,$$

i.e., p_{v_1} is bounded. Repeating the argument we get a proof by induction on the number of transformations L.

□

As a direct application of Lemma 1, we get the desired bound the importance weights.

Theorem 2.2. *For any $1 \leq k \leq K$, there exists $M > 0$ such that for all $\theta \in \mathbb{R}^D$, $\prod_{k'} \widehat{p}_{k'}(\theta)/\widehat{p}_k(\theta) \leq M$.*

Proof. Using Lemma 2.1, let $\mathcal{U}_{k'}$ be the upper bound for $\widehat{p}_{k'}$ and let $M = \prod_{k' \neq k} \mathcal{U}_{k'}$, from which the statement follows. □

This provides the sufficient conditions underlined by Geweke (1989), so that we achieve the following result regarding the overall stability of the importance sampling estimates.

Corollary 2.2.1. *Suppose $\theta_1, \ldots, \theta_T$ are samples from $\widehat{p}_k$ for some $1 \leq k \leq K$. Let $w_t \propto \prod_{k'} \widehat{p}_{k'}(\theta_t)/\widehat{p}_k(\theta_t)$, $\sum_t w_t = 1$ and $\mathrm{Var}_{\widehat{p}}[h] < \infty$. Then, the importance sampling estimate $\overline{h}(\theta) = \sum_{t=1}^{T} w_t h(\theta_t)$ has finite variance.*

2.3 SAMPLING FROM THE APPROXIMATE POSTERIOR

We can also use the samples $\theta_1, \ldots, \theta_T$ from $\widehat{p}_k$ and their associated importance weights $w_1, \ldots, w_T$ to obtain approximate samples $\theta_1^\star, \ldots, \theta_R^\star$ from $\widehat{p}$ using sampling/importance resampling (SIR). With this, $\mathbb{E}_{\widehat{p}}[h(\theta)]$ can be directly estimated as a Monte Carlo integral over the new samples. This procedure easily is done by choosing $\theta_r^\star = \theta_t$ with probability proportional to w_t. The required steps are detailed in Algorithm 1.

Algorithm 1 NAP-SIR

```
Input: Subposterior approximations $\widehat{p}_1, \ldots, \widehat{p}_K$, number
    of candidate samples $T$, final number of samples $R$,
    chosen subposterior index $k \in \{1, \ldots, K\}$.
Output: $R$ samples $\theta_1^\star, \ldots, \theta_R^\star$ from $\widehat{p}$.

1: for $t = 1, \ldots, T$ do
2:     Sample $\theta_t \sim \widehat{p}_k$
3:     $w_t \leftarrow \prod_{k'} \widehat{p}_{k'}(\theta_t)/\widehat{p}_k(\theta_t)$
4: $c \leftarrow \sum_t w_t$
5: $\boldsymbol{w} \leftarrow (w_1/c, \ldots, w_T/c)$
6: for $r = 1, \ldots, R$ do
7:     Draw $t$ from $\mathrm{Categorical}(\boldsymbol{w})$
8:     $\theta_r^\star \leftarrow \theta_t$
```

Note that Algorithm 1 provides, for any single k, a valid sampler for the approximate posterior $\widehat{p}$. However, in practice it is beneficial to apply the algorithm for many or all k to provide better exploration of the parameter space.

2.4 TIME COMPLEXITY

We now analyze the time complexity of the proposed method with respect to the number of subposteriors K, the number of samples S drawn from each of the subposteriors $p_1, \ldots, p_K$, and the number of samples R which we wish to obtain from the aggregated posterior.

Obtaining R samples from the approximate posterior using NAP consists of a single pass of the two following steps:

Step 1. In parallel, for $k = 1, \ldots, K$, fit a real NVP transformation to the samples drawn from the kth subposterior at worker k.

Step 2. Gather the subposterior approximations. Choose a $k \in \{1, \ldots, K\}$, choose $T \geq R$ and use Algorithm 1 to draw R samples from $\widehat{p}$.

Step 1 involves the usual costs of learning real NVP networks, which can be done using gradient-based methods, such as ADAM (Kingma and Ba, 2014). Assuming the number of layers and weights per layer in each network is fixed, evaluating $\widehat{p}(\theta) = \prod_k \widehat{p}_k(\theta)$ takes linear time in DK. Further taking $T = \lceil cR \rceil$ for some constant $c \geq 1$, we conclude Step 2 can be executed in $\mathcal{O}(RDK + R^2)$.

2.5 COMMUNICATION COSTS

It is important to note that typically $S \gg |\mathcal{D}_k|$, i.e., a worker ouputs a much larger number of subposterior samples than the size of the data subset $\mathcal{D}_k$ it processes. Even if the dataset is split among workers to improve computational efficiency through parallel inference, sending subposterior samples back to the server for aggregation can amount to considerable communication costs. Therefore, we also examine communication cost of NAP, and contrast it to currently available methods.

The communication cost of the proposed NAP amounts to $\mathcal{O}(KD)$, corresponding to the cost of communicating the NVP networks to the server, which does not depend on S. On the other hand, current methods have their accuracy intrinsically tied to the number of subposterior samples communicated to the server, resulting in $\mathcal{O}(SKD)$ communication costs. For example, the partition tree based method of Wang et al. (2015) requires recursive pair-wise aggregation of subposteriors, which calls for centralization of subposterior samples. The non-parametric and semi-parametric methods proposed by Neiswanger et al. (2014) require computing kernel-density estimates defined on each subposterior individually and centralizing them to subsequently execute an MCMC step to sample from their product. Similar costs are implied by the strategy of Nemeth and Sherlock (2018), which fits Gaussian process approximations and centralizes them to use MCMC to sample from the product of the expected values of their exponentiated predictives.

In other words, with NAP, subposteriors can be made arbitrarily accurate by drawing more subposterior samples (as long as local resources allow) with no additional effect on the cost of communicating the networks to the server.

3 EXPERIMENTAL RESULTS

We evaluated the performance of the proposed method in four different experiments, comparing it against several aggregation methods[1]:

- **Parametric (PARAM):** approximates the posterior as a product of multivariate normal densities fitted to each subposterior (Neiswanger et al., 2014).
- **Non-parametric (NP)**: uses kernel density estimates to approximate the subposteriors, takes their product and samples from it using Gibbs sampling (Neiswanger et al., 2014).
- **Semi-parametric (SP)**: a hybrid between the two former approaches (Neiswanger et al., 2014).
- **Consensus (CON)**: takes weighted averages of subposterior samples to obtain approximate samples from the target posterior (Scott et al., 2016).
- **Parallel aggregation using random partition trees (PART)**: uses partition trees to fit hyper-histograms to

[1]We have used the implementations available at https://github.com/richardkwo/random-tree-parallel-MCMC

the target posterior using subposterior samples (Wang et al., 2015).

In the first experiment, we target a uni-modal distribution of an intricate shape. In the second, we approximate a bivariate multi-modal distribution. In the third, we evaluate the performance of our method when approximating logistic regression posteriors in high dimensions. Finally, in the last one we analyze the performance of our method when there is a clear discrepancy among the subposteriors being merged.

All MCMC simulations were carried out using the python interface of the Stan probabilistic programming language(Carpenter et al., 2017), which implements the no-U-turn sampler. For each subposterior, we draw 4000 samples using 16 chains with an equal number of samples as warm-up. The same holds for the target (ground truth) posterior, computed on centralized data. The real NVP networks were implemented with PyTorch[2] using three transformations ($L = 3$) and Gaussian spherical base densities. The scale and translation networks were all implemented as multi-layer perceptrons with two hidden-layers comprising 256 nodes each. The layers of these networks were all equipped with rectified linear units except for the the last layer of each scale network, which was equipped with the hyperbolic tangent activation function. The network parameters were optimized using ADAM (Kingma and Ba, 2014) over 1000 iterations with learning rate 10^{-4}.

3.1 WARPED GAUSSIAN

We first consider inference in a warped Gaussian model which exhibits a banana-shaped posterior and is described by the generative model:

$$y \sim \mathcal{N}(\mu_1 + \mu_2^2, \sigma^2),$$

where the true values of the parameters μ_1 and μ_2 are 0.5 and 0, respectively. The variance σ^2 is set to 2 and treated as a known constant. We draw 10000 observations from the model and distribute them in $K = 10$ disjoint sets. Gaussian priors with zero mean and variance 25 were placed both on μ_1 and μ_0.

For NAP, we used Algorithm 1 to draw 4000 samples from the approximate posterior, using 16000 samples drawn from the individual subposterior approximation. To avoid possible underflow from normalizing a large number of importance weights, we do this in K installments, in each of which a suposterior approximation is used as a proposal. The same number of samples was drawn using each of the competing methods.

Figure 1 shows[3] the samples from the approximate posterior obtained with different aggregation methods, plotted against the posterior obtained using the entire sample set. Of all the methods, only NAP and PART were flexible enough to mimic the banana shape of the posterior. PART, however, is overly concentrated when compared to the ground truth, while NAP more faithfully spreads the mass of the distribution.

3.2 MIXTURE OF GAMMAS

We now consider performing inference in the shape parameters α_1 and α_2 of the following two-component mixture of Gammas:

$$p(y|\alpha_1, \alpha_2) = \frac{1}{2}\text{Gamma}(y|\alpha_1, \beta_1) + \frac{1}{2}\text{Gamma}(y|\alpha_2, \beta_2),$$

where the true values of the parameters of interest are $\alpha_1 = 0.5$ and $\alpha_2 = 1.0$. Furthermore, $\beta_1 = \beta_2 = 1$ are known constants, which makes the model clearly bi-modal. Independent Gamma priors with shape 0.5 and scale 1.0 were placed on α_1 and α_2.

As before, we drew 10000 observations from the model and distributed them in $K = 10$ disjoint sets for parallel inference. Samples from the approximate posterior for each aggregation algorithm were drawn in the same fashion as in the previous experiment.

Figure 2 shows[4] the samples from the approximate posteriors obtained with each aggregation method plotted against the target posterior, obtained using the entire sample set. The proposed method and PART clearly are the only ones that capture the multi-modality of the posterior. NAP, however, presented a better fit to the true posterior while PART placed more mass in low-density regions.

3.3 BAYESIAN LOGISTIC REGRESSION

We now explore how our method behaves in higher dimensions in comparison to its alternatives. For this purpose we consider inference on the simple logistic regression model with likelihood

$$y_i \sim \text{Bernoulli}\big(\sigma(\theta_{1:p} \cdot x_i + \theta_0)\big) \quad \forall 1 \leq i \leq N,$$

where $\cdot$ denotes the dot product, $\sigma(t) = (1 + e^{-t})^{-1}$ is the logistic function, and $\theta_0, \ldots, \theta_p$ receive independent $\mathcal{N}(0, \sqrt{5})$ priors. The true value θ_0' of θ_0 is held at -3 and the remaining $\theta_1', \ldots, \theta_p'$ are independently drawn from a normal distribution with zero-mean and variance 0.25.

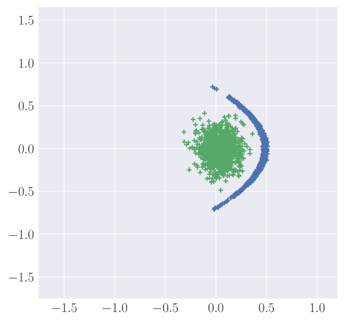
To generate a sample pair (x_i, y_i), we first draw x_i from $\mathcal{N}(\mathbf{0}, \Sigma)$, where the covariance matrix Σ is such that

$$\Sigma_{i,j} = 0.9^{|i-j|} \quad \forall 1 \leq i, j \leq p.$$

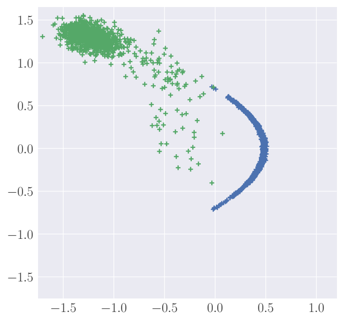
[2]https://pytorch.org

[3]The experiment was repeated with multiple random seeds, yielding similar results.

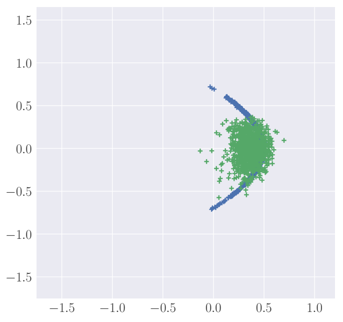
[4]The experiment was repeated with multiple random seeds, yielding similar results.



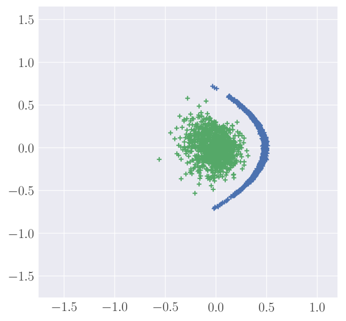
(a) Parametric



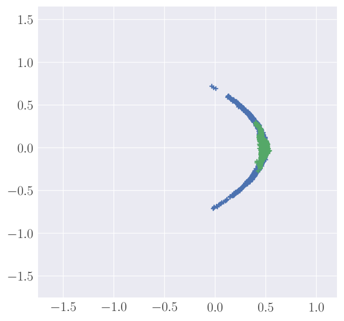
(b) Semi-parametric



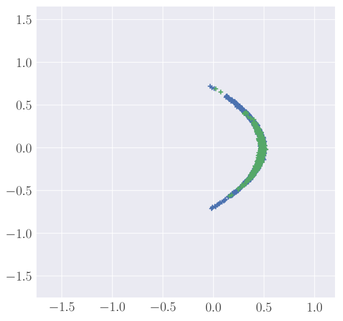
(c) Non-parametric



(d) Consensus



(e) PART



(f) NAP

Figure 1: MCMC samples for the warped Gaussian model obtained on the centralized dataset (ground truth), in blue, against samples from posterior approximations using different embarassingly parallel MCMC methods, in green.

Then, y_i is computed by rounding $\sigma(\theta'_{1:p} \cdot x_i + \theta'_0)$ to one if it is at least 0.5, and to zero otherwise.

For each value of $p \in \{25, 50, 100\}$, we draw $N = 10000$ sample pairs using the scheme described above and distribute them in $K = 50$ disjoint sets for parallel inference. As in previous experiments, we use NAP and its counterparts to merge the subposteriors and draw 4000 samples from the approximate posterior.

Table 1 presents the results for each of the aggregation methods in terms of the following performance measures:

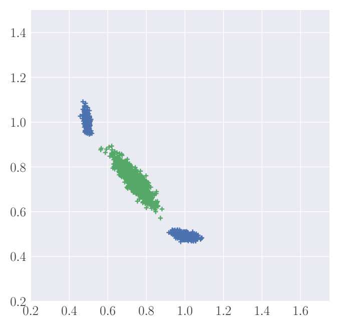
- **Root mean squared error** (RMSE) between the mean $\overline{\theta}$ of the approximate posterior samples $\{\theta^\star_r\}_{r=1}^R$ and the mean $\overline{\theta}'$ of samples $\{\theta'_r\}_{r=1}^R$ from the ground truth posterior;
- **Posterior concentration ratio** ($\mathcal{R}$), computed as:
$$\sqrt{\sum_r \|\theta_r - \overline{\theta}'\|_2^2 / \sum_r \|\theta'_r - \overline{\theta}'\|_2^2},$$
comparing the concentration of the two posteriors around the ground truth mean (values close to one are desirable);
- **KL divergence**; ($\mathrm{D_{KL}}$) between a multivariate normal approximation of the aggregated posterior and a multivariante normal approximation of the true one, both computed from samples.

Experiments were repeated ten times for each value of p, in each of which a new θ' was drawn. Additionally, average computing times for each aggregation method are shown in Table 2.

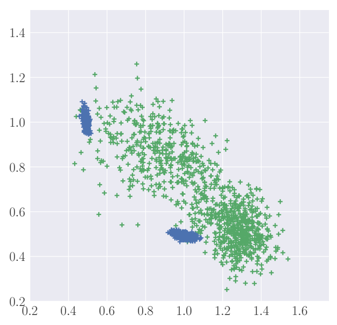
When compared to the other methods, for all values of p, NAP presents a mean closer to the one obtained using centralized inference (smaller RMSE) and has a more accurate spread around it ($\mathcal{R}$ closer to one). In terms of KL divergence, only at $p = 25$, PARAM outperforms NAP by a relatively small margin. Besides this case, NAP performs orders of magnitude better than the other methods, with increasing disparity as p grows.

3.4 RARE CATEGORICAL EVENTS

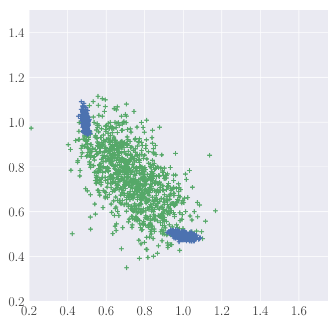
In the scenarios explored in the previous experiments, there is no specific reason to believe that the subposteriors differ drastically from each other. We now consider parallel inference on the parameters $(\lambda_1, \lambda_2, \lambda_3)$ of a categorical model,



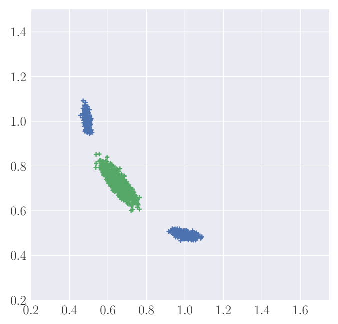
(a) Parametric



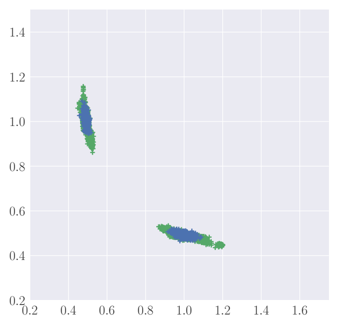
(b) Semi-parametric



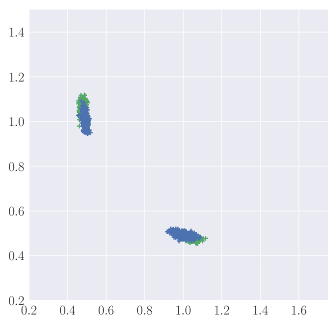
(c) Non-parametric



(d) Consensus



(e) PART



(f) NAP

Figure 2: MCMC samples for the mixture of gammas model obtained on the centralized dataset (ground truth), in blue, against samples from posterior approximations using different embarassingly parallel MCMC methods, in green.

	$p = 25$			$p = 50$			$p = 100$		
	RMSE	$\mathcal{R}$	D_{KL}	RMSE	$\mathcal{R}$	D_{KL}	RMSE	$\mathcal{R}$	D_{KL}
NAP	**1.95**	**13.95**	791.86	**1.07**	**13.08**	**1539.32**	**0.63**	**12.43**	**3493.35**
PART	3.29	24.38	4263.53	2.44	31.27	20159.64	1.51	31.80	75423.10
PARAM	2.56	18.34	**589.12**	1.99	24.37	2568.57	1.32	26.07	11245.58
SP	2.43	17.36	1586.80	2.02	24.62	7589.26	1.39	27.26	36994.45
NP	2.39	17.07	1343.88	2.01	24.54	7202.50	1.39	27.26	35313.77
CON	3.51	25.43	10654.78	3.08	37.92	56001.25	2.02	39.96	186275.86

Table 1: Comparison of different aggregation methods for embarassingly parallel MCMC inference on the logistic regression model with p covariates. The values presented are averages over ten repetitions of the experiments. The best results are in bold.

with respective outcomes A_1, A_2 and A_3, where the probability of observing one outcome is much higher than the others, i.e. $\lambda_3 \gg \lambda_1, \lambda_2$.

We simulate $N = 10000$ data points from $\text{Categorical}(\lambda_1', \lambda_2', \lambda_3')$ with $\lambda_1' = \lambda_2' = 2K/N$. We then partition the data into $K = 10$ disjoint subsets, run MCMC in parallel and apply different aggregation methods to obtain samples from the approximate posterior. The aggregated posteriors are then compared with the one obtained using the complete data.

Since the expected number of A_1 and A_2 per partition is 2, it often occurs than some partitions have only A_3. Figure 3 illustrates how disparate the subposteriors can be, depending on the specific partitioning of data.

To compensate for the variability in experimental results due to the random partitioning of the subsets, we repeated the experiments one hundred times with different random seeds, and report average results in Table 3. NAP clearly outperforms its competitors, with results that are orders of magnitude better.

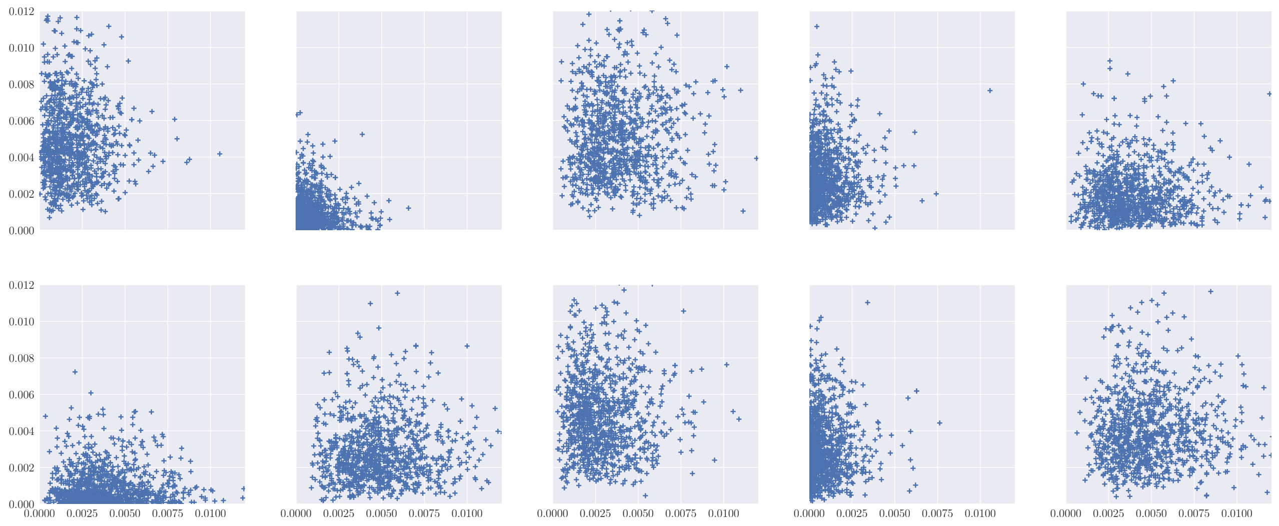


Figure 3: Scatter plots of the marginal for (r_1, r_2) for each of the $K = 10$ subposteriors within one of the experiment rounds.

	$p = 25$	$p = 50$	$p = 100$
NAP	337.28	363.88	426.95
PART	117.10	245.85	727.99
PARAM	33.36	62.45	115.40
SP	476.90	749.07	18378.862
NP	43.47	72.28	127.34
CON	32.59	62.01	125.49

Table 2: Average computing times for different aggregation methods for the logistic regression experiment.

	RMSE	$\mathcal{R}$	D_{KL}
NAP	$\mathbf{0.71 \times 10^{-3}}$	**2.61**	$\mathbf{106010.45 \times 10^{0}}$
PART	$0.27 \times 10-2$	179.39	623035.37×10^{1}
PARAM	0.11×10^{-1}	39.73	469483.18×10^{2}
SP	0.51×10^{-2}	267.30	945558.50×10^{4}
NP	0.19×10^{-1}	268.97	968806.96×10^{4}
CON	0.16×10^{0}	550.67	276976.28×10^{3}

Table 3: Comparison of different aggregation methods for embarassingly parallel MCMC inference on the rare categorical events model. The best results are in bold.

4 DISCUSSION

We proposed an embarrassingly parallel MCMC scheme in which each subposterior density is mapped to a tractable form using a deep invertible generative model. We capitalized on the ease of sampling from the mapped subposteriors and evaluating their log density values to build an efficient importance sampling scheme to merge the subposteriors. Imposing mild assumptions on the structure of the network, we proved that our importance sampling scheme is stable.

While in this work we gave special attention to the use of real NVP networks, our approach could potentially employ other invertible models, such as the Glow transform (Kingma and Dhariwal, 2018) or FFJORD (Grathwohl et al., 2019), without losing theoretical properties, as long as one can guarantee log densities remain bounded. If the bounds are difficult to verify, one could still resort to truncated forms of importance sampling (Ionides, 2008; Vehtari et al., 2015) to control the variance of importance sampling estimates.

Our experimental results demonstrated that NAP is capable of capturing intricate posteriors and coping with heteregenous subposteriors. In particular, we observed that it significantly outperformed current methods in high-dimensional settings. A possible explanation for this is that, unlike the density estimation techniques underlying the competing methods, the real NVP transformations used in our method, are specifically designed for high-dimensional data such as images.

Finally, the generative models we use serve as a intermediate representation to the subposterior, the size of which does not depend on the number of subposterior samples. Thus, workers can produce arbitrarily accurate subposterior estimates by drawing additional samples, without affecting the cost of communicating the subposteriors to the server, or the computational cost of aggregating them into a final posterior estimate.

Acknowledgements

DM, PB and SK were funded by the Academy of Finland, grants 319264 and 294238. The authors gratefully acknowledge the computational resources provided by the Aalto Science-IT project and support from the Finnish Center for Artificial Intelligence (FCAI).

References

Sungjin Ahn, Babak Shahbaba, and Max Welling. Distributed stochastic gradient MCMC. In *Proceedings of the 31st International Conference on International Conference on Machine Learning*, ICML'14, pages II–1044–II–1052. JMLR.org, 2014.

Elaine Angelino, Matthew James Johnson, and Ryan P. Adams. Patterns of scalable Bayesian inference. *Foundations and Trends in Machine Learning*, 9(2-3):119–247, 2016. doi: 10.1561/2200000052.

Bob Carpenter, Andrew Gelman, Matthew Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software*, 76(1), 2017.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

John Geweke. Bayesian inference in econometric models using Monte Carlo integration. *Econometrica*, 57(6): 1317–1339, 1989.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJxgknCcK7`.

Edward L Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17(2): 295–311, 2008. doi: 10.1198/106186008X320456.

Matthew Johnson, James Saunderson, and Alan Willsky. Analyzing hogwild parallel Gaussian Gibbs sampling. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2715–2723. Curran Associates, Inc., 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. *arXiv e-prints*, art. arXiv:1807.03039, Jul 2018.

Yi-An Ma, Tianqi Chen, and Emily B. Fox. A complete recipe for stochastic gradient MCMC. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 2917–2925, Cambridge, MA, USA, 2015. MIT Press.

Willie Neiswanger, Chong Wang, and Eric P. Xing. Asymptotically exact, embarrassingly parallel MCMC. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, pages 623–632, Arlington, Virginia, United States, 2014. AUAI Press.

Christopher Nemeth and Chris Sherlock. Merging MCMC subposteriors through Gaussian-process approximations. *Bayesian Analysis*, 13(2):507–530, 06 2018. doi: 10.1214/17-BA1063.

Matias Quiroz, Robert Kohn, Mattias Villani, and Minh-Ngoc Tran. Speeding up MCMC by efficient data subsampling. *Journal of the American Statistical Association*, 114:831–843, 2018. doi: 10.1080/01621459.2018.1448827.

Christian P. Robert, Vctor Elvira, Nick Tawn, and Changye Wu. Accelerating MCMC algorithms. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10(5):e1435, 2018. doi: 10.1002/wics.1435.

Steven L. Scott, Alexander W. Blocker, Fernando V. Bonassi, Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. Bayes and big data: The consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*, 11: 78–88, 2016.

Aki Vehtari, Andrew Gelman, and Jonah Gabry. Pareto Smoothed Importance Sampling. *arXiv e-prints*, art. arXiv:1507.02646, Jul 2015.

Xiangyu Wang, Fangjian Guo, Katherine A. Heller, and David B. Dunson. Parallelizing MCMC with random partition trees. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 451–459, Cambridge, MA, USA, 2015. MIT Press.