Data Subsampling for Bayesian Neural Networks

Eiji Kawasaki, Markus Holzmann

Data Subsampling for Bayesian Neural Networks

Eiji Kawasaki
Université Paris-Saclay,
CEA, List,
F-91120, Palaiseau, France

Markus Holzmann
Univ. Grenoble Alpes,
CNRS, LPMMC,
38000 Grenoble, France

Abstract

Markov Chain Monte Carlo (MCMC) algorithms do not scale well for large datasets leading to difficulties in Neural Network posterior sampling. In this paper, we apply a generalization of the Metropolis Hastings algorithm that allows us to restrict the evaluation of the likelihood to small mini-batches in a Bayesian inference context. Since it requires the computation of a so-called "noise penalty" determined by the variance of the training loss function over the mini-batches, we refer to this data subsampling strategy as Penalty Bayesian Neural Networks – PBNNs. Its implementation on top of MCMC is straightforward, as the variance of the loss function merely reduces the acceptance probability. Comparing to other samplers, we empirically show that PBNN achieves good predictive performance for a given mini-batch size. Varying the size of the mini-batches enables a natural calibration of the predictive distribution and provides an inbuilt protection against overfitting. We expect PBNN to be particularly suited for cases when data sets are distributed across multiple decentralized devices as typical in federated learning.

1 INTRODUCTION

The development of an effective Uncertainty Quantification (UQ) method that computes the predictive distribution by marginalizing over Deep Neural Network (DNN) parameter sets remains an important, challenging task [1]. Bayesian methods provide the posterior distribution which can be obtained either from Variational inference or via Monte Carlo sampling techniques, Markov chain Monte Carlo (MCMC) generally considered as the gold standard of Bayesian inference [2]. However, sampling a DNN parameter space by a Markov chain does not scale well for large systems and data sets, as it requires the evaluation of the log-likelihood over the whole data set at each iteration step. This obstacles the use of Bayesian Neural Networks (BNNs) posterior sampling in practice. Up to now, MCMC based BNNs only manage to handle limited sizes of data sets. Therefore, uninformative priors are commonly used to prevent overfitting, which on the other hand strongly harm the predictive performance. A current field of research is dedicated to developing BNN specific priors as reviewed by Fortuin (2021) [3].

In this article, we develop a data sub-sampling strategy for BNN posterior sampling explicitly reducing predictive overconfidence. This leads us to a variant of MCMC, based on subsampled batch data, which we refer to as Penalty Bayesian Neural Network - **PBNN**. The so-called "penalty method" [4] was first developed in the context of statistical and computational physics e.g. in the work of Pierleoni et al (2004) [5] to efficiently sample distributions with loss functions affected by statistical noise.

Naive BNN posterior sampling based on sub sampled data introduces a strong bias. This issue has already been carefully studied in the context of Bayesian inference where several MCMC sub-sampling methodologies have been proposed for scaling up the Metropolis-Hastings algorithm [6, 7, 8]. We show, both theoretically and empirically, that PBNN enables an unbiased posterior sampling by explicitly computing the variance of the loss of a subsampled data batch.

In the following we first introduce some related works, especially the Stochastic Gradient Langevin Dynamic (SGLD) algorithm [9] that confronts the effect of a noise in the loss computation. We then introduce PBNN as an unbiased posterior sampling strategy and show its benefits focusing on its ability to mitigate the predictive overconfidence effects. We then continue by giving some practical details on how to evaluate both, the noisy loss, and its uncertainty. We show that PBNN is compatible with state of the art MCMC proposal distribution for the Markov chain such as Langevin dynamics and Hybrid Monte Carlo (HMC) [10]. A benchmark is provided on a data set designed to trigger overfitting. Based on this example, PBNN

obtains good predictive performance. It includes a natural calibration parameter in the form of the size of the mini-batch. Lastly we study the impact of this parameter on the acceptance and on the overall performance of the model.

2 PRELIMINARIES

In the following we consider a vector θ that describes the parameters (weights and biases) of a Neural Network. We define $p(\theta)$ as a prior distribution over this set of parameters. Commonly used priors are Gaussian prior and Laplace prior that correspond respectively to an L2 and a L1 regularization of the vector θ. We refer as $p(y|x,\theta)$ the probability of a data item y given a data item x and parameter θ. As an example, we aim at sampling the posterior of a Neural Network designed for a supervised task. The posterior distribution over the parameters given a set of data can be written as $p(\theta|\mathcal{D}) \propto p(\theta)\prod_{i=1}^{N} p(y_i|x_i,\theta)$ where $\mathcal{D} = \{(y_i,x_i)\}_{i=1}^{N}$. Up to a constant, the log of the posterior can be written as a loss

$$\mathcal{L}_{\mathcal{D}}(\theta) = -\log p(\theta) - \sum_{i=1}^{N} \log p(y_i|x_i,\theta) \tag{1}$$

where the last term corresponds to the Negative Log-Likelihood (NLL). This is an illustrative choice that does not reduce the generality of PBNN as we could have also considered an unsupervised setup where $\mathcal{D} = \{(x_i)\}_{i=1}^{N}$ and $\mathcal{L}_{\mathcal{D}}(\theta) = -\log p(\theta) - \sum_{i=1}^{N} \log p(x_i|\theta)$.

Note that in the following $\mathcal{D}$ indicates precisely the ensemble of datum (y_i,x_i) used for the loss computation $\mathcal{L}_{\mathcal{D}}(\theta)$ in the equation 1. In particular, this data set can be a sub sample (mini-batch) of the larger data set containing all known data points of the training set.

3 RELATED WORK

We introduce in this section some relevant literature that studied how to take into account a noisy gradient estimate of $\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta)$ computed from a subset of the data. The link between noisy gradient and BNN posterior sampling is detailed in section 4.2.

Stochastic Gradient Langevin Dynamics Max Welling and Yee Whye Teh (2011) [9] showed that the iterates θ_t will converge to samples from the true posterior distribution as they anneal the stepsize by adding the right amount of noise to a standard stochastic gradient optimization algorithm. This is known as the Stochastic Gradient Langevin Dynamics (SGLD) where the parameter update is given by

$$\begin{aligned} \theta_{t+1} &= \theta_t - \eta_t \nabla_\theta \widetilde{\mathcal{L}_{\mathcal{D}}}(\theta_t) + \sqrt{2\eta_t}\epsilon_t \\ \widetilde{\mathcal{L}_{\mathcal{D}}}(\theta_t) &= -\log p(\theta) - \frac{N}{n}\sum_{i=1}^{n} \log p(y_i|x_i,\theta) \end{aligned} \tag{2}$$

where $\eta_t \in \mathbb{R}^+$ is a learning rate and ϵ_t is a centered normally distributed random vector. No rejection step is required for a vanishing step size. The positive whole number n corresponds to the size of the subsampled mini-batch. Chen (2014) [11] later extended this idea to HMC sampler.

Noisy Posterior Sampling Bias Due to a potentially high variance of the stochastic gradients $\nabla_\theta \widetilde{\mathcal{L}_{\mathcal{D}}}(\theta)$, Brosse (2018) [12] showed that the SGLD algorithm has an invariant probability measure which in general significantly departs from the target posterior for any non vanishing stepsize η. Furthermore, a recent work from Garriga-Alonso (2020) [13] suggests that recent versions of SGLD implementing an additional Metropolis-Hastings rejection step do not improve this issue, because the resulting acceptance probability is likely to vanish too.

Failures of Data Set Splitting Inference Other works have exploited parallel computing to scale Bayesian inference to large datasets by using a two-step approach. First, a MCMC computation is run in parallel on K (sub)posteriors defined on data partitions following $p(\theta|\mathcal{D}) \propto \prod_{i=1}^{K} p(\theta)^{1/K} p(\mathcal{D}_i|\theta)$. Then, a server combines local results. While efficient, this framework is very sensitive to the quality of subposterior sampling as showned by de Souza (2022) [14].

4 PENALTY BAYESIAN NEURAL NETWORK

4.1 Unbiased Posterior Sampling

We suppose that the data set points (y_i,x_i) are sampled from an unknown distribution $p(y,x)$ that can provide an infinite number of independent and identically distributed (i.i.d) random data points. This allows us to properly define a true unbiased loss $\mathcal{L}(\theta)$ as the mean over all possible data sets $\mathcal{L}(\theta) \simeq \mathcal{L}_{\mathcal{D}}(\theta)$ with

$$\mathcal{L}(\theta) = -\log p(\theta) - N\mathbb{E}_{(y_i,x_i)\sim p(y,x)}[\log p(y_i|x_i,\theta)] \tag{3}$$

given a fixed size N of a random data set $\mathcal{D}$. In order to sample from this log-posterior, we note that detailed balance is a sufficient but not necessary condition to ensure that a Markov process possesses a stationary distribution proportional to $e^{-\mathcal{L}(\theta)}$. Concretely, the detailed balance can be written as

$$A(\theta,\theta')q(\theta|\theta')e^{-\Delta(\theta',\theta)} = A(\theta',\theta)q(\theta'|\theta) \tag{4}$$

where $A(\theta',\theta)$ corresponds to the acceptance of the move from θ to θ' and $q(\theta'|\theta)$ is a proposal distribution. The loss difference writes

$$\Delta(\theta',\theta) = \mathcal{L}(\theta') - \mathcal{L}(\theta) \tag{5}$$

In the following, we assume that the true loss difference $\Delta(\theta', \theta)$ is unknown, and loss differences can only be estimated based on random data sets $\mathcal{D}$. Then we can introduce a random variable $\delta(\theta', \theta)$ providing an unbiased estimator of $\Delta(\theta', \theta)$ which we assume as normally distributed

$$\delta(\theta', \theta) \sim \mathcal{N}(\Delta(\theta', \theta), \sigma^2(\theta', \theta)) \tag{6}$$

The variance $\sigma^2(\theta', \theta)$ typically decreases with the size N of the random data sets $\mathcal{D}$.

This noisy loss $\delta(\theta', \theta)$ introduces a bias in the posterior sampling if not correctly taken into account. In the context of statistical physics and computational chemistry, Ceperley and Dewing (1999) [4] have generalized the Metropolis-Hastings random walk algorithm to the situation where the loss is noisy and can only be estimated. They showed that it is possible to still sample the exact distribution even with very strong noise by modifying the acceptance probability and applying a noise penalty $-\sigma^2(\theta', \theta)/2$ to the loss difference in the acceptance ratio A such that

$$A(\delta, \theta', \theta) = \min\left(1, e^{-\delta(\theta', \theta) - \sigma^2(\theta', \theta)/2}\right) \tag{7}$$

One can then show that detailed balance is satisfied on average

$$\begin{aligned} &\int d\delta A(\delta, \theta, \theta') q(\theta|\theta') \mathcal{N}(\delta; \Delta(\theta', \theta), \sigma^2(\theta', \theta)) e^{-\delta} \\ &= \int d\delta A(\delta, \theta', \theta) q(\theta'|\theta) \mathcal{N}(\delta; \Delta(\theta, \theta'), \sigma^2(\theta, \theta')) \end{aligned} \tag{8}$$

which is sufficient condition for the Markov chain to sample the unbiased distribution in the stationary regime. The penalty method can further be extended to a non symmetric proposal distribution $q(\theta'|\theta)$ used in algorithm 1.

Algorithm 1 PBNN Metropolis Adjusted Algorithm

```
θ_t ← θ_0
for t ← 0 to T do
    θ' ~ q(θ'|θ_t)
    A(δ, θ', θ_t) ← min(1, [q(θ_t|θ')/q(θ'|θ_t)] e^{-δ(θ', θ_t) - σ²(θ', θ_t)/2})
    u ~ U(0, 1)
    if u ≤ A(δ, θ', θ_t) then
        θ_{t+1} ← θ'
    else
        θ_{t+1} ← θ_t
    end if
    t ← t + 1
end for
```

From equation 7 one can immediately recognize the drawback of PBNN leading to an exponential suppression of the acceptance since the variance $\sigma^2(\theta', \theta)$ is always non negative. Note further, that in the case of BNN posterior sampling, $\sigma^2(\theta', \theta)$ is in general not known either, and can only be estimated, too. Whereas it is possible to extend the scheme to account for noisy variances [4], we will not pursue this here.

Let us stress that the penalty term serves to exactly account for the uncertainty in calculating the loss $\mathcal{L}(\theta)$ of equation 3 for a finite number of random data. However, it does not address the actual uncertainty introduced by setting $\mathcal{L}(\theta) \approx \mathcal{L}_{\mathcal{D}}(\theta)$, e.g. equation 3 and equation 1.

4.2 Langevin Dynamic Penalty

Choosing a non symmetric proposal distribution $q(\theta'|\theta)$ can speed up the mixing of the Markov Chain and help PBNN scale to larger systems by maximizing the acceptance $A(\theta', \theta)$. We first consider the situation where the proposal distribution depends only on the two states θ and θ' and not on any given mini-batch $\mathcal{D}$. In the absence of noise, the Metropolis-Hastings acceptance writes

$$\begin{aligned} A(\theta', \theta) &= \min\left(1, \frac{q(\theta|\theta')}{q(\theta'|\theta)} e^{-\Delta(\theta', \theta)}\right) \\ &\approx \min\left(1, \frac{q(\theta|\theta')}{q(\theta'|\theta)} e^{-(\theta - \theta') \cdot (\nabla_\theta \mathcal{L}(\theta') + \nabla_\theta \mathcal{L}(\theta))/2}\right) \end{aligned} \tag{9}$$

where we have Taylor expanded the loss $\mathcal{L}(\theta)$ around θ' assuming a sufficiently small step from θ to θ'. The maximization of $A(\theta', \theta)$ leads to a Langevin equation where the gradient of the loss introduces a drift in the Gaussian proposal distribution

$$q(\theta'|\theta) = \mathcal{N}(\theta'; \theta - \eta \nabla_\theta \mathcal{L}(\theta), 2\eta) \tag{10}$$

Sampling a new state θ' from the proposal distribution $q(\theta'|\theta)$ corresponds exactly to drawing a centered reduced normal random variable ϵ, and computing

$$\theta' = \theta - \eta \nabla_\theta \mathcal{L}(\theta) + \sqrt{2\eta}\epsilon \tag{11}$$

The non-trivial term in the Metropolis-Hastings acceptance then writes

$$\begin{aligned} &\log\left(\frac{q(\theta|\theta')}{q(\theta'|\theta)} e^{-\Delta(\theta', \theta)}\right) \\ &\quad = \frac{-1}{4\eta} \left\| \eta \left(\nabla_\theta \mathcal{L}(\theta') + \nabla_\theta \mathcal{L}(\theta)\right) - \sqrt{2\eta}\epsilon \right\|^2 \\ &\quad + \frac{1}{4\eta} \left\| \sqrt{2\eta}\epsilon \right\|^2 - \mathcal{L}(\theta') + \mathcal{L}(\theta) \end{aligned} \tag{12}$$

In order to use a noisy gradient $\nabla_\theta \mathcal{L}(\theta) \simeq \nabla_\theta \widetilde{\mathcal{L}_{\mathcal{D}}}(\theta)$ as an approximation of the Gaussian mean's drift, SGLD [9] requires a vanishing learning rate to dominate the noise and maximize the acceptance. On the other hand, one could design an optimized proposal distribution $q(\theta|\theta')$ and set a non zero step size while computing the full Metropolis-Hastings acceptance

$$A(\theta', \theta) = \min\left(1, \frac{q(\theta_t|\theta')}{q(\theta'|\theta_t)} e^{-\delta(\theta', \theta_t) - \sigma^2(\theta', \theta_t)/2}\right) \tag{13}$$

as shown in algorithm 1. The PBNN's noise penalty explicitly targets the bias introduced by a noisy loss. In fact, in the equation 13, the corresponding Monte Carlo loss minimization (i.e. introducing a zero temperature limit) corresponds to finding the set of parameters θ that minimizes both the noisy regularized loss and its associated uncertainty.

For large size models, biased samplers like the Unrestricted Langevin Algorithm (ULA) are known to be very effective as they skip the rejection step i.e set $A(\theta', \theta) = 1$ for a sufficiently small step size η resulting in an unrestricted Langevin sampling as

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t) + \sqrt{2\eta}\epsilon_t \tag{14}$$

In order to model a noisy estimate of the loss, it is tempting to replace the drift $\eta \nabla_\theta \mathcal{L}(\theta_t)$ with an unbiased estimator such that

$$\eta \nabla_\theta \mathcal{L}(\theta) = \eta \nabla_\theta \widetilde{\mathcal{L}_{\mathcal{D}}}(\theta) + \eta\sigma(\theta) \tag{15}$$

where $\widetilde{\mathcal{L}_{\mathcal{D}}}(\theta)$ is defined in the equation 2. For a vanishing step size $\eta \to 0$, one may then expect that the additional noise term $\eta\sigma(\theta)$ gets negligible compared to the random noise of order $\eta^{1/2}$ in the equation 14. However, the uncertainty of the loss gradient $\sigma(\theta)$ does in general not result in white noise, but is correlated between different parameters θ. For non-vanishing η the noisy loss gradient can thus trigger a significant departure from the target posterior, see also [12].

As we will see in the numerical experiments section, PBNN's ability to evaluate the likelihood over small mini-batches even in the presence of a strong noise allows us to calibrate the Bayesian predictive distribution. This is especially convenient in comparison with usual BNNs as the regularization is handled solely by the prior distribution $p(\theta)$. Commonly used uninformative priors can lead to poor performances as they do not target explicitly overfitting but rather the complexity of the model i.e. L2 and L1 penalties. As a reminder, other conventional methods such as early stopping are not compatible with the Bayesian approach developed for BNN.

4.3 Noise Penalty Estimation

As showed in equation 7 in the case of a symmetric proposal distribution $q(\theta'|\theta)$, the energy difference $\delta(\theta', \theta_t)$ has to dominate the always positive variance $\sigma^2(\theta', \theta_t)/2$ in order to obtain a reasonable acceptance $A(\delta, \theta', \theta_t)$. However, in practice a noise penalty usually strongly dominates any gain in energy from θ_t to θ' if the energy difference is computed only on a single small mini-batch $\mathcal{D}$. This leads to an exponentially suppressed acceptance and long correlation times of the MCMC.

To prevent this situation, we define $\delta(\theta', \theta)$ as an empirical average over the loss difference

$$\delta(\theta', \theta) = \frac{1}{M} \sum_{j=1}^{M} \left(\mathcal{L}_{\mathcal{D}_j}(\theta') - \mathcal{L}_{\mathcal{D}_j}(\theta) \right) \tag{16}$$

where $\mathcal{D}_j$ corresponds to randomly chosen mini-batches. By definition the average is an unbiased estimator such that $\mathbb{E}[\delta(\theta', \theta)] = \Delta(\theta', \theta)$. We notice that the central limit theorem ensures that $\delta(\theta', \theta)$ is normally distributed in the limit of large M as required by equation 6.

The variance of the random variable $\delta(\theta', \theta)$ strictly decreases with the number of mini-batches M since $\sigma^2(\theta', \theta) = \sigma^2_{\mathcal{D}}(\theta', \theta)/M$ where $\sigma^2_{\mathcal{D}}$ corresponds to the expected variance of a single loss difference computed over a mini-batch $\mathcal{D}$. Both $\sigma^2_{\mathcal{D}}$ and σ^2 are unknown, but we can compute an estimate of $\sigma^2(\theta', \theta) \simeq \chi^2(\theta', \theta)$ using an unbiased chi-squared estimator

$$\chi^2(\theta', \theta) = \frac{1}{M(M-1)} \sum_{j=1}^{M} \left(\mathcal{L}_{\mathcal{D}_j}(\theta') - \mathcal{L}_{\mathcal{D}_j}(\theta) - \delta(\theta', \theta) \right)^2 \tag{17}$$

In the following, we do not take into account the error over the estimation of the variance $\sigma^2(\theta', \theta)$ which corresponds to the hypothesis that variations of χ^2 as a function of θ' and θ largely dominate over the noise. Leading order corrections in this noise are discussed in [4].

A second approximation can be introduced in case of a limited access to the data: drawing M mini-batches with replacement artificially decreases the variance at the cost of introducing a bias (i.e. violates the i.i.d. hypothesis of the energy differences).

5 EXPERIMENTS

5.1 Data Set

We study the performance of PBNN on a synthetic data set that contains the positions of a double pendulum over a simulated time t. These positions are obtained by integrating Euler-Lagrange equations casted as ordinary differential equations. We turn a time series forecasting problem into a supervised regression task. We model the distribution $p(y|x)$ where $y \in \mathbb{R}^4$ corresponds to the four Cartesian coordinates of the two masses of the double pendulum and $x \in \mathbb{R}^{4*5}$ are 5 given y past positions of the masses. The data set $\mathcal{D} = \{(y_i, x_i)\}_{i=1}^{N}$ inputs x_i write

$$x_t \equiv (y_{t-20}, ..., y_{t-24}) \tag{18}$$

where t is the discrete simulation time. The system is strongly chaotic such that learning on a given limited data

set can lead to strong predictive overconfidence as suggested in figure 1. We thus expect the noise penalty to play an important role on the realization of a supervised task based on this data set.

The code that generates the double pendulum dataset is available following this link: `https://gitlab.com/eijikawasaki/double-pendulum`

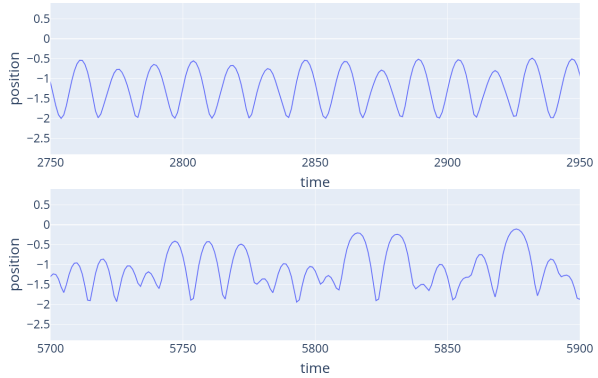

Figure 1: Pendulum data set extracts from a single simulation run. The blue curve corresponds to one of the Cartesian coordinates of one of the masses in function of time. As an example, the behavior on the bottom (part of the test data set) is hard to predict knowing only the data from the top (part of the training data).

5.2 Benchmark setup

The goal of this section is to compare the performances between PBNN and other BNN that do not include any noise penalty. We show empirically that the PBNN obtains good predictive performances even while evaluating the loss on small mini-batches and therefore introducing a strong noise. On the other hand, as expected PBNN has a much lower MCMC acceptance rate. We also compare the PBNN to SGLD which is designed to take into account a stochastic noise in the loss computation.

In the following, we sample the posterior based on MCMC random walkers where $q(\theta'|\theta_t)$ is a symmetric proposal density. We use a Gaussian distribution centered around θ_t and adjust its variance to ensure the ergodicity of the Markov process. We model the data distribution with a single multivariate Gaussian likelihood $p(y|x,\theta) = \mathcal{N}(y;\mu_\theta(x),\Sigma^2_\theta(x))$ where $\mu_\theta(x) \in \mathbb{R}^4$ and $\Sigma^2_\theta(x)$ is a positive diagonal covariance matrix parameterized by $\sigma^2_{\theta,d}(x)$ where $d \in \{1,2,3,4\}$. This model is thus heteroscedastic: we use a Mixture Density Network [15] such that both $\mu_\theta(x)$ and $\Sigma^2_\theta(x)$ are outputs of a neural network that takes x as an input. We observe empirically that a homoscedastic model based on a Mean Squared Error loss has a noise penalty that is several orders of magnitude smaller than its heteroscedastic counterpart.

The loss of a NN is known to have numerous local minima and we don't use in the experiment any transition kernel for PBNN designed to jump between separate minima. We thus limit our study on the impact of the penalty method on a relatively small model that is easier to sample. The dimension of the vector parameter θ is equal to 419 as we consider a NN with 2 hidden layers each containing 10 neurons. The data consists in 9975 data points sequentially (i.e. not randomly) split into a 2992 points as a training data and 6983 points as a test data set. We use a Gaussian uninformative prior $p(\theta) \propto e^{-\lambda\|\theta\|_2^2}$ corresponding to a tiny L2 regularization of the NN parameters of magnitude $\lambda = 10^{-5}$. During the posterior sampling, only the training data is used in order to compute the loss $\mathcal{L}_\mathcal{D}(\theta)$ and thus both $\delta(\theta',\theta)$ and $\chi^2(\theta',\theta)$.

The performance of the prediction (based on the inferred parameters θ sampled from the posterior) is measure by the the average Negative-Log-Likelihood (NLL) that we define as

$$\overline{\text{NLL}_\mathcal{D}} = -\frac{1}{L}\sum_{i=1}^{L}\log\left(\frac{1}{J}\sum_{j=1}^{J}p(y_i|x_i,\theta^{(j)})\right) \tag{19}$$

where the data set $\mathcal{D}$ of size L corresponds either to the train or the test sets. $\theta^{(j)}$ are J i.i.d. samples of the MDN parameters obtained from the Markov chain. Note that this measure should not depend on θ since the MDN prediction is marginalized over the posterior parameters distribution.

The mini-batch size N in equation 1 determines the target posterior distribution that is sampled and therefore changes the value of $\overline{\text{NLL}_\mathcal{D}}$. For a constant uninformative prior, decreasing N corresponds to increasing the variance of the predictive function. This is a straightforward consequence of Bayes theorem using an uninformative Gaussian prior with a huge variance. In order to compare the performances of the prediction of a BNN and a reference standard "vanilla" BNN that does not use mini-batches, we compute one-sigma confidence intervals as drawn in the figure 2. As an example, we compare them to a Gaussian predictive distribution and target an expected coverage of approximately 68.2%. To check the accuracy of the UQ method, we compute the Average Coverage Error (ACE) defined in the equation 20.

$$\text{ACE} = \left|68.2\% - \frac{1}{4L}\sum_{i=1}^{L}\sum_{d=1}^{4}\rho_{i,d}\right| \qquad \text{with } \rho_{i,d} = \begin{cases} 1 & \text{if } y_{i,d} \in [\mu^*_d(x_i) - \sigma^*_d(x_i), \mu^*_d(x_i) + \sigma^*_d(x_i)] \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

where μ^*_d and σ^*_d are the mean and the standard deviation in one of the four dimension d of the empirical predictive distribution as defined in equation 21.

5.3 Numerical Results

From table 1 we empirically show that PBNN achieves a better overall performance than other biased sub sampled models for a small mini-batch size. As an illustration, figure 2 shows the error bars of the prediction over a random period of time for each model. The empirical predictive distribution is defined as

$$\mathbb{E}[p(y_t|x_t,\theta)] \simeq \frac{1}{J}\sum_{j=1}^{J}p(y_t|x_t,\theta^{(j)}) \tag{21}$$

where t is the simulated time and $\theta^{(j)}$ are J samples obtained by the MCMC computation. The expected value in the equation 21 is computed over the targeted posteriors which are different for every model studied in the benchmark.

It is important to note here that even for a same likelihood weight

Table 1: Performance benchmark. The top and bottom groups of models correspond to the predictive performance based on a likelihood weight corresponding respectively to $N = 2992$ and $N = 60$.

Model	Test $\overline{\text{NLL}_{\mathcal{D}}}$	Train $\overline{\text{NLL}_{\mathcal{D}}}$	Test ACE
Vanilla BNN	-4.11 ± 0.01	-5.36 ± 0.01	$7.1\% \pm 0.3\%$
Tempered BNN	-1.96 ± 0.08	-2.05 ± 0.10	$3.9\% \pm 1.1\%$
Batched BNN	-1.68 ± 0.17	-1.74 ± 0.19	$1.7\% \pm 1.6\%$
pseudo-SGLD	-2.35 ± 0.10	-2.48 ± 0.11	$4.8\% \pm 0.8\%$
PBNN	**-3.91**± 0.07	**-4.83**± 0.09	**0.4**$\% \pm 2.3\%$

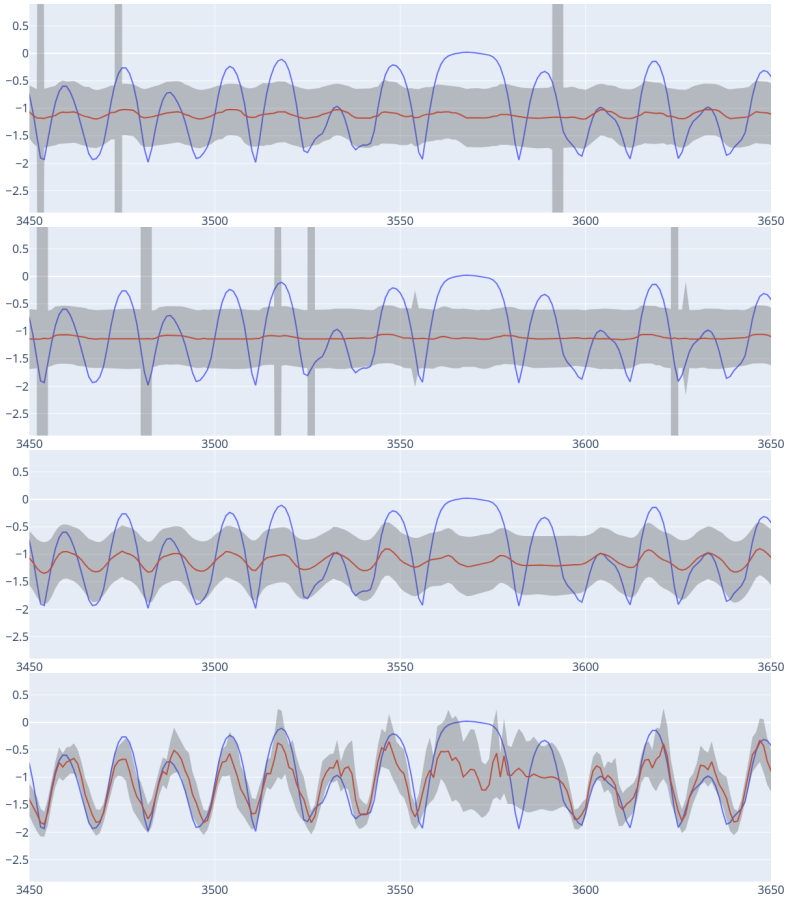

Figure 2: Models predictions over a test data set example extract. The blue line corresponds to one of the Cartesian coordinate of one of the two masses. Mean models predictions are in red and one standard deviation regions are plotted in grey. The horizontal axis corresponds to the the simulated time t. The four figures from top to bottom correspond respectively to the models: Tempered BNN, Batched BNN, pseudo-SGLD and **PBNN**.

N in the Bayes theorem, we do not expect the same prediction between models that use the whole train set, like "vanilla" BNNs and the PBNN as they do not target the same posterior sampling. The vanilla BNN samples a posterior proportional to $e^{-\mathcal{L}_{\mathcal{D}}(\theta)}$ whereas PBNN aims at sampling a posterior proportional to $e^{-\mathcal{L}(\theta)}$ where $\mathcal{L}(\theta)$ is the loss expected for a given size N of mini-batch $\mathcal{D}$ as defined in the equation 3.

Vanilla BNN We call this first model "vanilla" as it corresponds to a standard MCMC random walk based BNN with no mini batches. Vanilla BNN's acceptance writes $A(\theta', \theta_t) = \min(1, e^{-\mathcal{L}_{\mathcal{D}}(\theta') + \mathcal{L}_{\mathcal{D}}(\theta_t)})$ where $\mathcal{D}$ contains all the 2992 available train data points. This model thus uses the whole train set to compute the loss difference for each new proposed state θ'. Note that Langevin algorithms such as MALA or HMC all sample the same posterior. The only difference with this model is in their efficiency as they are designed to maximize the MCMC acceptance

and the ergodicity of the Markov Chain.

The ACE is approximately zero on the training set and significantly different from zero on the test data set. In the limit of a single mini-batch containing all the training data set, the PBNN coincides with this model. In the following we aim at calibrating the PBNN predictive prediction while maintaining good predictive performances.

Tempered BNN In order to provide a comparison between the usual "vanilla" BNN and a PBNN, we adjust the likelihood weight following

$$-\log p(\theta) - \frac{N}{2992}\sum_{i=1}^{2992}\log p(y_i|x_i,\theta) \tag{22}$$

Indeed, we balance the loss to be equal to PBNN's likelihood that uses mini-batches with $N = 60$. This adjusted weight over the likelihood corresponds to the Safe Bayes approach where we vary the weight of the likelihood thanks to a temperature T following $p(\mathcal{D}|\theta)^{1/T}$ as discussed by Wilson (2020) [16]. In our case $T > 1$ is known to help under model misspecification as it is the case in the double pendulum Gaussian prediction example.

The resulting temperature $T = 2992/60$ is however such a high temperature that the prior regularization dominates the likelihood for this model. The resulting predictive performance is unsatisfactory as shown in the table 1.

Batched BNN The acceptance for this model is defined as $A(\delta,\theta',\theta_t) = \min(1, e^{-\delta(\theta',\theta_t)})$ with $\delta(\theta',\theta_t)$ computed with $M = 100$ and $N = 60$. The number of mini-batches used by this model is the same as the number used by the PBNN for a fair comparison. The only difference with the PBNN model is the noise penalty $e^{-\chi^2(\theta',\theta_t)/2}$ in the acceptance. Table 1 therefore demonstrates the impact of the penalty method, strongly improving the overall performance of the model.

pseudo-SGLD The SGLD algorithm is designed to naturally take into account noise from sub-sampled data. Standard SGLD with a weight $N = 2992$, i.e. the number of training samples, in equation 2 leads to results that are similar to the Vanilla BNN both in terms of negative loglikelihood and coverage. In this benchmark we want to test the ability of the algorithm to handle a noisy loss. We therefore set $N = n = 60$ with a constant learning rate $\eta = 10^{-5}$ and call the resulting model a pseudo-SGLD. Comparing to PBNN, we observe in both table 1 and figure 2 that the noise is too high for the SGLD in this setup. The performance of SGLD could probably be improved by decaying the step size η polynomially as suggested in the literature [9]. As a reminder, this model has the great advantage of not requiring a rejection step.

PBNN The noise penalty is estimated following equations 16 and 17 with $M = 100$ and $N = 60$. The random walk acceptance for PBNN writes

$$A(\delta,\theta',\theta_t) = \min\left(1, e^{-\delta(\theta',\theta_t)-\chi^2(\theta',\theta_t)/2}\right)$$

We have adjusted by hand the mini-batch size $N = 60$ in order to calibrate the models to optimize the test data set coverage.

There is a noticeable gap of PBNN's performance between the train and the test data sets that is not intuitively described in the theory of PBNN. This overfitting is probably caused partially by the access to a limited amount of data. Indeed, during the Monte Carlo sampling, all mini-batches $\mathcal{D}$ are part of the same training data and not i.i.d. sampled from a probability distribution $p(\mathcal{D})$. In an ideal situation where we could evaluate both δ and χ^2 from i.i.d. samples $\mathcal{D} \sim p(\mathcal{D})$ as required by the equations 16 and 17, we expect a lower overfitting effect.

5.4 Mini-Batch Size N

For the purpose of the benchmark we have calibrated PBNN error bars in the table 1 by tuning the mini-batch size N. One can wonder what is the optimal value of N in a general purpose outside the scope of calibration and for a given data set size. Figure 3 shows that, as expected, the prediction performance measured as the negative loglikelihood $\overline{\text{NLL}_\mathcal{D}}$ over the test data set increases with N. On the other hand, the acceptance rapidly drops because the number of mini-batches M in equation 17 decreases for a constant data set size. In the figure 3 we indeed notice a linear decrease of the log-acceptance due to the decrease of the number of available mini-batches M. The optimal value of N is therefore a trade-off between reasonable acceptance and good predictive performance.

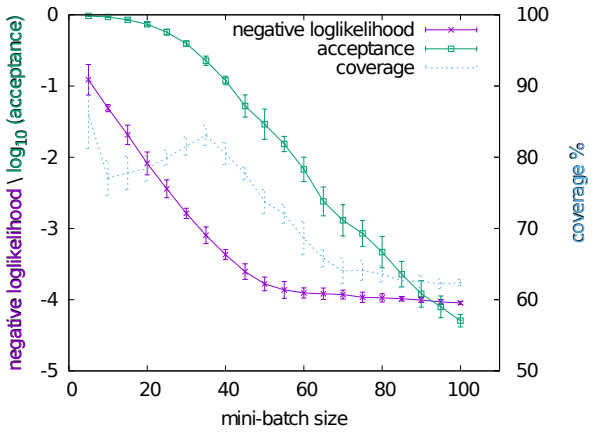

Figure 3: PBNN performance measured by $\overline{\text{NLL}_\mathcal{D}}$ over the test data set, acceptance in base $\log_{10}$ and one sigma coverage in function of the mini-batch size N for a constant prior. The standard deviation continuously decreases in function of the batch size. We notice however that the coverage is not monotonous: it is determined by both the error bars size and by the loglikelihood.

It is important to note in the figure 3 that different batch sizes result in different coverages for a constant uninformative prior. As we have shown the PBNN predictive distribution can be calibrated. In practical setups as discussed by Hermans (2021) [17], it is recommended to compare the expected coverage probability of the predictive distribution defined in equation 21 to the empirical coverage probability as shown in the equation 20.

6 CONCLUSION

Uncertainty quantification for the predictions of large size neural networks remains an open issue. In this work, we have shown a new way to enable data sub-sampling for Bayesian Neural Network independent from gradient based approximation such as Stochastic Gradient Langevin Dynamic.

First, we have demonstrated that a raw estimation of the likelihood based on a noisy loss introduces a bias in the posterior sampling if not taken into account. We then have shown that a

generalization of the Metropolis Hastings algorithm allows us to eliminate the bias and to exactly sample the posterior even with very strong noise. This necessitates an additional "noise penalty" that corresponds to the variance of the noisy loss difference and exponentially suppresses the MCMC acceptance probability.

In practice, the noise penalty corresponds to replacing a single large data set by multiple smaller sub sampled mini batches associated with an uncertainty over their losses. We have shown how to interpret this term as a regularization. Varying the size of the mini-batches enables a natural calibration that we have compared to other techniques such as tempered Safe Bayes approaches.

Based on this calibration principle, we have provided a benchmark that empirically showed good predictive performances of PBNNs. We hope that combining data sub-sampling with other Monte Carlo acceleration techniques such as HMC could allow to compute uncertainties for model sizes not reachable until now.

Lastly, PBNN could be particularly suited in the case when the data sets $\mathcal{D}$ are distributed across multiple decentralized devices as in the typical federated learning setup. Indeed, the noise penalty is determined by the variance of the losses computed on each individual data set. In principle, PBNN should enable the possibility to compute uncertainty with separate data sets without exchanging them.

7 Acknowledgments

This work has been supported by the French government under the "France 2030" program, as part of the SystemX Technological Research Institute. Authors thank Victor Berger for useful comments and discussions.

References

[1] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, Muhammad Shahzad, Wen Yang, Richard Bamler, and Xiao Xiang Zhu. A Survey of Uncertainty in Deep Neural Networks, January 2022. Number: arXiv:2107.03342 arXiv:2107.03342 [cs, stat].

[2] Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What Are Bayesian Neural Network Posteriors Really Like? In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4629–4640. PMLR, July 2021.

[3] Vincent Fortuin. Priors in Bayesian Deep Learning: A Review, March 2022. Number: arXiv:2105.06868 arXiv:2105.06868 [cs, stat].

[4] D. M. Ceperley and M. Dewing. The penalty method for random walks with uncertain energies. *The Journal of Chemical Physics*, 110(20):9812–9820, May 1999.

[5] Carlo Pierleoni, David M. Ceperley, and Markus Holzmann. Coupled Electron-Ion Monte Carlo Calculations of Dense Metallic Hydrogen. *Phys. Rev. Lett.*, 93(14):146402, September 2004. Publisher: American Physical Society.

[6] Rémi Bardenet, Arnaud Doucet, and Chris Holmes. On Markov chain Monte Carlo methods for tall data. *Journal of Machine Learning Research*, 18(47):1–43, 2017.

[7] Anoop Korattikara, Yutian Chen, and Max Welling. Austerity in MCMC Land: Cutting the Metropolis-Hastings Budget. page 10.

[8] Daniel Seita, Xinlei Pan, Haoyu Chen, and John Canny. An Efficient Minibatch Acceptance Test for Metropolis-Hastings. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 5359–5363, Stockholm, Sweden, July 2018. International Joint Conferences on Artificial Intelligence Organization.

[9] Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. page 8.

[10] Radford M. Neal. MCMC Using Hamiltonian Dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010.

[11] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic Gradient Hamiltonian Monte Carlo. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1683–1691, Bejing, China, June 2014. PMLR. Issue: 2.

[12] Nicolas Brosse, Alain Durmus, and Eric Moulines. The promises and pitfalls of Stochastic Gradient Langevin Dynamics. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[13] Adrià Garriga-Alonso and Vincent Fortuin. Exact Langevin Dynamics with Stochastic Gradients. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2021.

[14] Daniel A. De Souza, Diego Mesquita, Samuel Kaski, and Luigi Acerbi. Parallel MCMC Without Embarrassing Failures. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, pages 1786–1804. PMLR, May 2022. ISSN: 2640-3498.

[15] Christopher M. Bishop. Mixture density networks. WorkingPaper, Aston University, 1994. Backup Publisher: Aston University ISBN: NCRG/94/004.

[16] Andrew G Wilson and Pavel Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. In *Advances in Neural Information Processing Systems*, volume 33, pages 4697–4708. Curran Associates, Inc., 2020.

[17] Joeri Hermans, Arnaud Delaunoy, François Rozet, Antoine Wehenkel, and Gilles Louppe. Averting A Crisis In Simulation-Based Inference, October 2021. Number: arXiv:2110.06581 arXiv:2110.06581 [cs, stat].