Quantitative performance evaluation of Bayesian neural networks

Brian Staber
Safran-Tech
Digital Sciences & Technologies Department
Rue des Jeunes Bois, Châteaufort, 78114 Magny-Les-Hameaux, France
brian.staber@safrangroup.com

Sébastien Da Veiga
Safran-Tech
Digital Sciences & Technologies Department
Rue des Jeunes Bois, Châteaufort, 78114 Magny-Les-Hameaux, France
sebastien.da-veiga@safrangroup.com

Abstract
Due to the growing adoption of deep neural networks in many fields of science and engineering, modeling and estimating their uncertainties has become of primary importance. Various approaches have been investigated including Bayesian neural networks, ensembles, deterministic approximations, amongst others. Despite the growing literature about uncertainty quantification in deep learning, the quality of the uncertainty estimates remains an open question. In this work, we attempt to assess the performance of several algorithms on sampling and regression tasks by evaluating the quality of the confidence regions and how well the generated samples are representative of the unknown target distribution. Towards this end, several sampling and regression tasks are considered, and the selected algorithms are compared in terms of coverage probabilities, kernelized Stein discrepancies, and maximum mean discrepancies.

1 Introduction
Due to their recent achievements in the last decade, deep neural networks have been widely adopted in many research fields and industries. However, they still suffer from several shortcomings that prevent their deployment in fields where decisions involve high stakes. These limitations are mainly due to their inability to provide uncertainty estimates which are crucial for many real-world applications. Therefore, uncertainty quantification in deep learning has attracted a lot of attention and several strategies have been investigated. Thorough overviews of uncertainty quantification in deep learning are provided by the recent review papers of Abdar et al. [2021] and Gawlikowski et al. [2021]. Amongst the available approaches, Bayesian neural networks offer a simple and flexible formulation but raise several challenges. The high-dimensional posterior distribution of the network parameters has a complex structure, it is untractable, and the influence of the prior distribution is not fully understood. The posterior distribution is typically inferred using sampling strategies, variational inference, or parametric/deterministic approximations. When dealing with complex posterior distributions, gradient-based sampling methods such as Markov Chain Monte Carlo (MCMC) methods are often adopted. In particular, Hamiltonian Monte Carlo (HMC) is usually considered as a gold standard sampling algorithm but is unfortunately extremely computationally demanding. Izmailov et al. [2021] recently studied the properties of Bayesian neural networks posterior distributions using HMC thanks to impressive computational resources. Cobb and Jalaian
[2021] proposed a symmetric splitting strategy for scaling HMC to modern deep neural networks. Since the work of [Welling and Teh, 2011], stochastic gradient MCMC (SGMCMC) approaches based on Langevin and Hamiltonian dynamics have been extensively studied (see, e.g., [Ma et al., 2015] and [Nemeth and Fearnhead, 2021]). These methods aim at reducing the computational burden of MCMC algorithms at the cost of introducing asymptotic bias by omitting Metropolis-Hastings correction and using a stochastic approximation to the gradients. In contrast, variational approaches approximate the posterior distribution by minimizing the Kullback-Leibler divergence over a family of tractable distributions [Hoffman et al., 2013]. Several scalable methods have been proposed such as the Bayes by Backprop (BBB) method of Graves [2011] and Blundell et al. [2015], or the multiplicative normalizing flows (MNF) proposed by Louizos and Welling [2017], to name a few. Liu and Wang [2016] proposed the Stein variational gradient descent (SVGD) which can be seen as gradient flow [Liu, 2017, Korba et al., 2021]. Although ensemble methods are originally used to improve the generalization capabilities of neural networks, they can also be used for uncertainty quantification as shown by Lakshminarayanan et al. [2017], where uncertainty is predicted thanks to random initialization and data shuffling. There exist other scalable methods such as Monte Carlo dropout [Gal and Ghahramani, 2016] and stochastic weight averaging Gaussian (SW AG) [Maddox et al., 2019].

Selecting an inference algorithm is challenging given the wide variety of approaches available in the literature. In addition, the underlying hyperparameters of each method strongly affect the resulting approximation of the target distribution. Regardless of the inference approach, evaluating the quality of the predicted uncertainties remains an open issue. Contrary to previous works, we propose here to estimate sensible coverage probabilities using a large number of independently generated datasets for all the considered algorithms, test cases, and hyperparameters values. We argue that a theoretically justified quality measure of the confidence regions is given by coverage probabilities estimated with $n_d > 1$ independent datasets. We also compare the selected algorithms with the help of discrepancy measures, namely, the maximum mean discrepancy (MMD) [Gretton et al., 2006] and the kernelized Stein discrepancy (KSD) [Liu et al., 2016]. As shown in the sequel of this paper, comparing coverage probabilities and discrepancy measures is of particular interest and potentially allows for identifying inference algorithms that provide good approximations to the posterior distributions. Nevertheless, as it will be discussed, it should be kept in mind that the kernelized Stein discrepancy suffers from shortcomings (see, e.g., [Wenliang and Kanagawa, 2020, Korba et al., 2021]) and its underyling hyperparameter(s) should be carefully selected. In contrast to previous works, we consider several sampling methods that include, for instance, variance reduction or preconditioning, with variational and ensembling approaches. To our best knowledge, such a comparison of inference algorithms in deep learning is not yet available.

2 Related works

Quantitative performance evaluation of inference algorithms has been performed by [Yao et al., 2019], where the prediction interval coverage probabilities (PICP) are compared amongst other metrics. However to our understanding, the PICPs computed by [Yao et al., 2019] are estimated with a single dataset and only reflect the proportion of test predictions that fall into the prediction interval. Within the field of computer vision, Gustafsson et al. [2020] compared ensembling, Monte Carlo dropout and sampling algorithms in terms of the area under the sparsification error (AUSE) and the expected calibration error (ECE). The work of [Beluch et al., 2018] compared an ensemble method with Monte Carlo dropout and a geometric approach on image classification tasks. Variational and deterministic approximation methods were also evaluated by [Ovadia et al., 2019] under datashift, where it has been found that ensembling yields good performances compared to other methods.

3 Bayesian neural networks

Let $D = \{(x_i, y_i)\}_{i=1}^N$ denote a dataset made of $N$ observations of input $x_i \in \mathbb{R}^D$ and output $y_i \in \mathbb{R}^M$ pairs. The surrogate model is assumed to take the form $y = f(x; \theta) + \varepsilon(x)$ where $f(\cdot; \theta) : \mathbb{R}^D \to \mathbb{R}^M$ denotes a neural network with parameters $\theta \in \mathbb{R}^d$ and $\varepsilon$ is an additive noise modeling aleatoric uncertainties. The additive noise is generally chosen to be a $\mathbb{R}^M$-valued and centered multivariate Gaussian random variable with a constant covariance matrix $C = \sigma I$ (homoscedastic uncertainties) or spatially-dependent covariance matrix $C(x) = \sigma(x)I$ (heteroscedastic uncertainties).
Assuming that the observations \(y_1, \ldots, y_N\) are i.i.d., the likelihood \(p(\mathcal{D}|\theta)\) is a Gaussian distribution with mean value \(f(\cdot; \theta)\) and covariance matrix \(C\) (or \(C(x)\)). Given a prior distribution \(p(\theta)\) over the network parameters, the posterior distribution \(p(\theta|\mathcal{D})\) can be deduced using Bayes’ formula:

\[
p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})},
\]

where the normalization constant \(p(\mathcal{D})\) is generally untractable. For any new input variable \(x^* \in \mathbb{R}^M\), the predictive distribution is given by

\[
p(y^*|x^*, \mathcal{D}) = \int_{\mathbb{R}^d} p(y^*|x^*, \theta)p(\theta|\mathcal{D}) d\theta.
\]

The above integral is generally untractable as well and approximated by a Monte Carlo estimator using \(S\) samples \(\theta_1, \ldots, \theta_S\) drawn from the posterior distribution \(p(\theta|\mathcal{D})\).

4 Considered algorithms

We selected the 8 sampling algorithms that belong to the family of SGMCMC algorithms. The latter are based on the resolution of a generic Itô stochastic differential equation (ISDE) of the form [Ma et al., 2015]

\[
d\zeta = \frac{1}{2}b(\zeta) \, dt + \sqrt{D(\zeta)} \, dW(t),
\]

where \(\{\zeta(t), t > 0\}\) is a Markov process with values in \(\mathbb{R}^L, L \geq d\). For any fixed \(t > 0\), the random variable \(\zeta_t = (\theta_t, r_t)\) gathers the network parameters \(\theta_t \in \mathbb{R}^d\) and eventually additional degrees of freedom \(r_t \in \mathbb{R}^{L-d}\) such as the momentum in Hamiltonian dynamics. The drift vector \(b : \mathbb{R}^L \rightarrow \mathbb{R}^L\) is given by

\[
b(\zeta) = -(D(\zeta) + Q(\zeta))\nabla H(\zeta) + \Gamma(\zeta), \quad \Gamma_i(\zeta) = \sum_j \partial_{\zeta_j} (D_{ij}(\zeta) + Q_{ij}(\zeta)),
\]

where \(D\) is the positive semidefinite diffusion matrix and \(Q\) is a skew-symmetric matrix. In Equation (4), \(H\) is a Hamiltonian of the form \(H(\zeta) = U(\theta) + h(\theta, r)\) where the potential function \(U\) is given by

\[
U(\theta) = -\log(p(\mathcal{D}|\theta)p(\theta)) = -\sum_{i=1}^N \log(p(y_{i|}x_{i}, \theta)) - \log(p(\theta)).
\]

Particular choices of the function \(b\) and the matrices \(D, Q\) lead to well-known dynamics such as the Langevin and Hamiltonian dynamics (see, e.g., Nemeth and Fearnhead [2021]). Ma et al. [2015] showed that the generic ISDE admits a stationary distribution of the form \(\pi(\zeta) \propto \exp(-H(\zeta))\), so that \(\zeta_t\) converges in distribution towards the target posterior distribution. Equation (3) is solved with an explicit Euler scheme. In addition, the gradient of the potential function \(U\) is approximated with a stochastic gradient \(\nabla_U(\zeta)\) using mini-batching, i.e.,

\[
\hat{\nabla}U(\theta) = -\frac{N}{|B|} \sum_{i \in B} \nabla \log(p(y_{i|}x_{i}, \theta)) - \nabla \log(p(\theta)),
\]

where \(B\) denotes a random subset of the available \(N\) observations. Let then \(\zeta^k := \zeta(t_k)\) where \(t_k = k\epsilon\) is the \(k\)-th timestep and \(\epsilon > 0\) is a constant step size. The generic SGMCMC algorithms read as

\[
\zeta^{k+1} = \zeta^k - \epsilon \left( (D(\zeta^k) + Q(\zeta^k)) \hat{\nabla}U(\zeta^k) + \Gamma(\zeta^k) \right) + \sqrt{\epsilon D(\zeta^k)} \Delta W^{k+1},
\]

where \(\Delta W^{k+1}\) is a standard centered Gaussian random variable. Herein, we consider the SGLD [Welling and Teh, 2011] and SGHMC [Chen et al., 2014] algorithms. We also consider the variants SGLD-CV, SGLD-SVRG, SGHMC-CV, SGHMC-SVRG that include variance reduction techniques for the stochastic gradients [Baker et al., 2019, Dubey et al., 2016]. These variants attempt to reduce the variance of the stochastic gradient by relying on control variates. The stochastic gradient \(\hat{\nabla}U\) is replaced by the following estimation:

\[
\hat{\nabla}U(\theta) = \nabla U(\theta)|_{\theta=\eta} + \hat{\nabla}U(\theta) - \hat{\nabla}U(\theta)|_{\theta=\eta},
\]
where \( \nabla U(\theta) \) denotes the true gradient and \( \eta \) denotes the control variable. \( \text{Baker et al. [2019]} \) proposed to use a MAP estimate and \( \text{Dubey et al. [2016]} \) proposed to set \( \eta \) to the current value of \( \theta \) every \( m \) iterations. Amongst the family of SGMCMC algorithms, we also consider the preconditioned SGLD (pSGLD) of \( \text{Li et al. [2016]} \) and the scale-adaptive SGHMC (SGHMC-SA) of \( \text{Springenberg et al. [2016]} \) that rely on adaptive preconditioning strategies. Finally, we also consider the Monte Carlo dropout \( \text{[Gal and Ghahramani, 2016]} \) and deep ensembles methods \( \text{[Lakshminarayanan et al., 2017]} \). For each algorithm, we investigate the influence of the hyperparameter \( \epsilon > 0 \) that either corresponds to the step size for the SGMCMC algorithms, and to the learning rate for the MC dropout and deep ensembles methods. More details about the selected algorithms and their implementations can be found in the Appendix B.

5 Metrics

The selected algorithms are compared in terms of three discrepancy measures: maximum mean discrepancy \( \text{[Gretton et al., 2006]} \), kernelized Stein discrepancy \( \text{[Liu et al., 2016]} \) and the sliced kernelized Stein discrepancy \( \text{[Gong et al., 2020]} \). In the case of regression problems, we also estimate coverage probabilities.

5.1 Coverage probability for regression tasks

For a given regression problem with a target function \( x \mapsto y(x) \), we first generate a collection of \( n_d \) independent train datasets \( D_1, \ldots, D_{n_d} \), where \( D_j \) consists in a set of inputs and targets \( \{x_i^j, y_i^j\}_{i=1}^N \), as well as an additional test dataset \( D^* = \{x_i^*, y_i^*\}_{i=1}^{N^*} \) for some chosen \( N^* \). Then, for a given inference algorithm with fixed hyperparameters, a neural network \( f(\cdot; \theta) \) with parameters \( \theta \), and a risk level \( \alpha \in [0, 1] \):

(a) The inference algorithm is applied for each dataset \( D_j \) and \( S \) samples \( \theta_1^j, \ldots, \theta_S^j \) of the network parameters are stored. We let \( \hat{Q}_j = (1/S) \sum_{s=1}^S \delta_\theta(\theta^j_s) \) be the approximation of the target posterior obtained with the \( j \)-th dataset \( D_j \).

(b) For each approximation \( \hat{Q}_j \) and all test inputs \( x_i^* \), the \((1 - \alpha)/2\% \) and \((1 + \alpha)/2\% \) quantiles denoted by \( y_i^{\text{ld}}(\alpha) \) and \( y_i^{\text{ul}}(\alpha) \) are computed using the samples \( f(x_i^*, \theta_1), \ldots, f(x_i^*, \theta_S) \). Let then \( R_i^j(\alpha) = [y_i^{\text{ld}}(\alpha), y_i^{\text{ul}}(\alpha)] \) be the associated confidence regions for \( i = 1, \ldots, N^* \) and \( j = 1, \ldots, n_d \).

(c) For each test input \( x_i^* \), the frequency \( F_i(\alpha) \) at which \( y_i^* \) falls into the confidence regions \( R_i^1(\alpha), \ldots, R_i^{n_d}(\alpha) \) is computed, i.e.,

\[
F_i(\alpha) = \frac{1}{n_d} \sum_{j=1}^{n_d} I(R_i^j(\alpha))(y_i^*), \quad i = 1, \ldots, N^* .
\]  

(9)

(d) The coverage probability is then defined as \( C(\alpha) = (1/N^*) \sum_{i=1}^{N^*} F_i(\alpha) \).

For a good inference algorithm, we expect the mapping \( \alpha \mapsto C(\alpha) \) to be close to identity.

5.2 Measuring similarities between the selected algorithms with maximum mean discrepancy

For a given experiment, we propose to compare the selected algorithms with the maximum mean discrepancy (MMD) \( \text{[Gretton et al., 2006]} \) in order to establish possible similarities between each approximation. More specifically, by denoting \( \{\hat{Q}_i\}_{i=1}^{100} \) the set of approximations obtained with 10 algorithms and 10 values of the hyperparameter \( \epsilon \), we compute the MMD between each pair of approximations \( \{\hat{Q}_i, \hat{Q}_j\}, j = 1, \ldots, 100, j = 1, \ldots, 100 \). Let \( k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \) be a kernel function and let \( \mathcal{H}(k) \) be the reproducing kernel Hilbert space with kernel \( k \). The MMD between two distributions \( P \) and \( Q \) is defined as \( \text{[Gretton et al., 2006]} \):

\[
\text{MMD}(P, Q) = \|\mu_P - \mu_Q\|_{\mathcal{H}(k)}, \quad \mu_P = \int k(\cdot, \theta) \, dP, \quad (10)
\]
where \( \mu_p \) is called the kernel mean embedding of \( P \) in \( \mathcal{H}(k) \). Using the properties of the reproducing kernel Hilbert space \( \mathcal{H}(k) \), the squared MMD can be written as

\[
\text{MMD}^2(P, Q) = \mathbb{E}_{x \sim P} \mathbb{E}_{x' \sim P} k(x,x') + \mathbb{E}_{x \sim Q} \mathbb{E}_{x' \sim Q} k(x,x') - 2 \mathbb{E}_{x \sim P} \mathbb{E}_{x' \sim Q} k(x,x')
\]  

Here \( k \) is chosen as the distance-based kernel function \( k(x,x') = \| x \|_2^2 + \| x' \|_2^2 - \| x - x' \|_2^2 \) proposed by Sejdinovic et al. [2013]. The MMD is estimated with a V-statistics and computed between every samples generated by each algorithm and each step size in order to identify algorithms and step sizes that yield similar approximations. For visualisation purposes, the matrix \( \{\text{MMD}^2(\hat{Q}_i, \hat{Q}_j)\}_{i,j=1}^{100} \) of pairwise squared MMDs is used to construct a two-dimensional representation obtained by multidimensional scaling (see, e.g., Figure 3).

### 5.3 Distances with respect to the untractable posterior distribution

The maximum mean discrepancy between an approximation \( \hat{Q}_i \) and the target posterior distribution \( P \) is intractable due to the mathematical expectations with respect to \( P \) itself. However, if the kernel function in Equation (11) is chosen as a kernel \( k_p \) such that \( \mathbb{E}_{x \sim P} k_p(x, \cdot) = 0 \), then the MMD reduces to the kernelized Stein discrepancy [Liu et al., 2016] which bypasses this difficulty. The squared KSD between the target \( P \) and one of its approximation \( \hat{Q}_i \) is defined as

\[
\text{KSD}^2(P, \hat{Q}_i) = \mathbb{E}_{x \sim \hat{Q}_i} \mathbb{E}_{x' \sim \hat{Q}_i} k_p(x,x'),
\]

where \( k_p : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \) denotes the Stein kernel given by

\[
k_p(x,x') = \nabla_x \cdot \nabla_x k(x,x') + s_p(x) \cdot \nabla_x k(x,x') + s_p(x') \cdot \nabla_x k(x,x') + (s_p(x) - s_p(x')) k(x,x').
\]

The function \( s_p : \mathcal{X} \to \mathbb{R}^d \) in Eq. (13) denotes the score function defined as \( s_p(x) := \nabla_x \log p(x) \) with \( p \) the probability density function of \( P \). In our case, \( \hat{Q}_i \) denotes an empirical distribution given by \( \hat{Q}_i(\theta) = (1/m) \sum_{\tau=1}^m \delta_\theta(\theta_m) \), where \( \theta_1, \ldots, \theta_m \) are samples generated by one of the selected algorithms with a given step size, while \( P \) denotes the unknown target posterior distribution. In this setting, the KSD can be used to evaluate how close the approximation \( \hat{Q}_i \) is to the unknown target \( P \). We also estimate the squared sliced kernelized Stein discrepancy defined as [Gong et al., 2020]

\[
\text{SKSD}^2(P, \hat{Q}_i) = \mathbb{E}_{x \sim U \sim U^d \sim U} \left[ \mathbb{E}_{x \sim \hat{Q}_i} \mathbb{E}_{x' \sim \hat{Q}_i} h_p(x,x'; r, g) \right]
\]

where \( U(S^{K-1}) \) denotes the uniform distribution over the hypersphere \( S^{K-1} \). The kernel function \( h_p(x,x'; r, g) \) takes the form

\[
h_p(x,x'; r, g) = (r^T g) s_p^\prime(x') \nabla_{g^T} k(g^T x, g^T x') + (r^T g) s_p^\prime(x) \nabla_{g^T} k(g^T x, g^T x') s_p^\prime(x') + s_p^\prime(x) \nabla_{g^T} k(g^T x, g^T x') s_p^\prime(x') + (r^T g)^2 \nabla_{g^T} \cdot \nabla_{g^T} k(g^T x, g^T x') s_p^\prime(x'),
\]

where \( s_p^\prime(x) := r^T s_p(x) \). The squared sliced KSD is estimated with a standard Monte Carlo estimator for the outer expectation and a V-statistics for the inner expectations. The kernel function \( k : \mathcal{X} \times \mathcal{X} \to \mathbb{R} \) in Equations (13) and (15) is chosen as the inverse multi-quadratic (IMQ) kernel function in order to ensure weak convergence of the KSD [Gorham and Mackey, 2017], i.e., \( k(x,x') = (c^2 + \|x - x'\|_1)^{-\beta} \) with \( c = 1 \), \( \beta = 1/2 \), and \( \Gamma = \tau^2 I \). The hyperparameter \( \tau \) to be carefully chosen and fixed to a single value when the KSD and SKSD are used to compare different algorithms. The hyperparameter \( \tau \) is chosen as the median of the pairwise distances which is estimated using the entire collection of samples generated by all the algorithms for a given experiment:

\[
\tau := \text{median}\{\|\theta_{\sigma(i)} - \theta_{\sigma(j)}\|_2, 1 \leq i, j \leq M\}.
\]

The set of \( M \) indices \( \sigma(1), \ldots, \sigma(M) \) is obtained by concatenating the samples generated by all the algorithms and drawing \( M = 10000 \) subsamples. It should be underlined that the KSD suffers from a few shortcomings as discussed in the works of Wenliang and Kanagawa [2020] and Korba et al. [2021].

### 6 Experiments

The performances of the inference algorithms are evaluated for two types of experiments, namely, posterior distributions that do not involve any neural network (see §6.1) and posterior distributions...
associated to regression tasks (see §6.2). More details about the datasets are available in the Appendix C. For SGMCMC methods, the number of burnin iterations is fixed to 50000 and 2000 samples of the posterior distribution \( p(\theta | D) \) are selected amongst the next 150000 iterations using the thinning strategy described in Appendix B.2. For the deep ensembles method, 2000 models are trained using the Adam optimizer and 50 epochs. For the Monte Carlo dropout method, the dropout rate is fixed to 0.5, the neural network is trained using the Adam optimizer and 400 epochs. Additional details about the algorithms and their hyperparameters can be found in Appendix B. The experiments were carried out on a cluster consisting of 40 nodes with 4 Nvidia A100-80Gb GPUs per node.

### 6.1 Posterior distributions with analytical likelihoods

We first assess the performances of the selected inference algorithms for simple low-dimensional posterior distributions taken from the literature. Here, the likelihood does not involve any neural network and the performance is assessed with the KSD only. Three test cases respectively taken from [Welling and Teh, 2011], [Baker, 2019], and [Baker et al., 2017] are considered. The likelihoods and prior distributions are summarized in Table 1. Additional details about the generation of the associated datasets can be found in Appendix C.1. For this type of posterior distributions, the kernelized Stein discrepancy is estimated for 8 algorithms, 20 equally log-spaced step sizes and three batch sizes. It follows that 480 simulations are performed per experiment. As an illustration, in the case of experiment #1, Figure 1 shows the graphs of the kernelized Stein discrepancy with respect to the step size. Additional results have been reported in Appendix A.1 for brevity. The step sizes that yield the lowest KSDs are reported in Table 2 for each algorithm and every considered batch sizes.

![Figure 1](image.png)

Figure 1: Experiment #1: graphs \( \epsilon \mapsto \hat{\text{KSD}}^2 \) for each algorithm and three batch sizes. Left panel: batch size 1, middle panel: batch size 10, right panel: batch size 100.

### 6.2 Posterior distributions in regression tasks

Three one-dimensional regression tasks are considered where simple feedforward neural networks are used as surrogate models. For brevity, the results obtained for experiments #5 and #6 have been reported in Appendix A.2. The coverage probabilities are estimated using \( n_d = 1000 \) training datasets, and for each training dataset \( D_j \), we also estimate discrepancy measures \( \text{KSD}(P, \hat{Q}_j) \) and
Table 2: Experiment #1. Lowest KSDs obtained for each algorithm and three batch sizes. Bold values correspond to the lowest KSDs over all the algorithms.

| Algorithm  | Batch size 1 | Batch size 10 | Batch size 100 |
|------------|--------------|---------------|---------------|
| SGLD       | $\epsilon$  | $\tilde{\text{KSD}}$ | $\epsilon$  | $\tilde{\text{KSD}}$ | $\epsilon$  | $\tilde{\text{KSD}}$ |
| SGLD-CV    | 0.001        | 0.218288      | 0.002069      | 0.014349      | 0.002637      | 0.0008441      |
| SGLD-SVRG  | 0.001274     | 0.012251      | 0.003360      | 0.007972      | 0.001624      | 0.001436       |
| SGHMC      | 0.0002637    | 0.007631      | 0.005456      | 0.002507      | 0.004281      | 0.003574       |
| SGHMC-CV   | 0.001        | 0.022469      | 0.006952      | 0.002877      | 0.023357      | 0.000477       |
| SGHMC-SVRG | 0.0002637    | 0.011093      | 0.011288      | 0.0000807     | 0.029764      | 0.000694       |
| pSGLD      | 0.0029764    | 0.035384      | 0.037927      | 0.005755      | 0.037927      | 0.000518       |
| SGHMC-SA   | 0.023357     | 0.046828      | 0.029764      | 0.006889      | 0.037927      | 0.000521       |

The kernelized Stein discrepancies $\tilde{\text{KSD}}(P, \hat{Q}_j)$ for $j = 1, \ldots, n_d$. A summary of the considered regression problems is provided in Table 3 and additional details about the datasets can be found in Appendix C.2. For each inference algorithm, 10 equally log-spaced values $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$ for the hyperparameter $\epsilon$ are considered, where $\epsilon_0$ and $\epsilon_{10}$ have been carefully chosen for each algorithm (see Table 7). It follows that 100,000 simulations are performed for each regression problem.

Table 3: Summary of the considered regression problems: target function, noise distribution, batch size, number of training samples $N$, number of test samples $N^*$, number of parameters $d$ in the surrogate model.

| Experiment | Target function $y(x)$ | Noise $\epsilon$ | Batch size $N$ | $N^*$ | $d$ |
|------------|-------------------------|------------------|----------------|-------|-----|
| (#4) Homoscedastic | $\cos(2x) + \sin(x)$ | $\mathcal{N}(0, 0.09)$ | 32 | 100 | 200 | 10401 |
| (#5) Mismatched | $0.1x^3$ | $\mathcal{N}(0, 0.25)$ | 32 | 100 | 200 | 5251 |
| (#6) Matched | $-(1 + x) \sin(1.2x)$ | $\mathcal{N}(0, 0.25)$ | 32 | 82 | 200 | 5251 |

The graphs $\alpha \mapsto C(\alpha)$ of the coverage probabilities obtained for experiment #4 are shown in Figure 2 for $\alpha \in \{0.05, 0.1, \ldots, 0.95\}$. It can be seen that the Monte Carlo dropout and ensembling methods both underestimate the coverage probabilities. On the other hand, the SGMC methods are able to provide good coverages for some values of the step size $\epsilon$. The SGLD algorithm and its variants with variance reduction provide good coverage probabilities for sufficiently high step sizes ($\epsilon_5, \epsilon_6, \epsilon_7$) while the preconditioned SGLD tends to overestimate the coverage probabilities for all the chosen step sizes. The SGHMC-SVRG and SGHMC-SA algorithms provide excellent coverage probabilities for well-chosen step sizes. Similar results were obtained for the remaining regression problems but it should be noted that deep ensembles show better performances in experiment #5 (see Figure 6).

The kernelized Stein discrepancies $\tilde{\text{KSD}}(P, \hat{Q}_j)$ between the target posterior distribution $P$ and its approximations $\hat{Q}_j$ obtained with the $j$ training datasets $D_j$ were also computed. Boxplots of the $n_d = 1000$ KSDs are shown in Figure 4. The sliced KSDs exhibit similar behaviors with respect to the step size and have been reported in Appendix A.2 (see, e.g., Figure 8). It can first be seen that the algorithms that provide good coverage probabilities, such as the SGHMC-SVRG, do not always yield the lowest KSDs. In particular, the deep ensembles method provides the lowest KSDs but has poor coverage probabilities. Given that the KSD is known for being insensitive to distant modes, it can be argued that the selected ensembling approach does not sufficiently explore regions of interest in the posterior distribution, hence leading to lower values of the KSD. The SGLD algorithm and its variants with variance reduction provide the lowest KSD values amongst the SGMC methods. In the case of the SGLD, SGLD-CV, and SGLD-SVRG algorithms, the lowest KSDs are obtained for the step size $\epsilon_3 = 4.64 \times 10^{-7}$. However, this step size yields poor coverage probabilities. For such a step size, the mixing rates of the SGLD algorithm and its variants are slow and the generated samples may not well represent the target posterior distribution. Given that the KSD is insensitive to distant modes, the low KSD values of the SGLD algorithms should be interpreted carefully. In addition, the step sizes yielding the lowest KSDs do not especially coincide with the step sizes that yield the best coverage probabilities, especially in the case of SGLD algorithms. Although the SGHMC
The similarities between the algorithms can be observed in Figure 3. It can be seen that there exist similarities between the SGLD and SGHMC algorithms, while the deep ensembles, pSGLD and SGHMC-SA methods seem to provide very different approximations. In contrast, the SGLD and SGHMC algorithms form two families of approximations that are highly similar for low step sizes and tend to differ for higher step sizes.

Figure 2: Experiment #4. Graphs of coverage probabilities $\alpha \mapsto C(\alpha)$ for each algorithm and 10 increasing step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The values of the step sizes are used to highlight the individual curves in each figure (dark blue: lowest step size, dark red: highest step size).

Figure 3: Experiment #4. Multidimensional scaling applied to the pairwise maximum mean discrepancies between the samples generated by each algorithm and each (converged) step size. The size of the markers is proportional to the value of the step size.

algorithms and its variants yield higher KSD values, the step sizes associated to the lowest KSD values are close to the step sizes providing the best coverage probabilities. For instance, in the case of the SGHMC-SVRG algorithm, the best coverage probabilities are obtained for $\epsilon_6$, while the best KSDs seem to be obtained for $\epsilon_5$. The similarities between the algorithms can be observed in Figure 3. It can be seen that there exist similarities between the SGLD and SGHMC algorithms, while the deep ensembles, pSGLD and SGHMC-SA methods seem to provide very different approximations. In contrast, the SGLD and SGHMC algorithms form two families of approximations that are highly similar for low step sizes and tend to differ for higher step sizes.
Figure 4: Experiment #4. Boxplots of the kernelized Stein discrepancies $\hat{KSD}(P, \hat{Q}_j)$, $j = 1, \ldots, n_d$, obtained for 10 step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The colors of the boxes coincide with the color scheme used in Figure 2.

7 Discussion

For regression problems, we find that sampling methods such as the SGMCMC algorithms, with or without variance reduction, are able to provide good coverage probabilities if appropriate step sizes are chosen. On the other hand, deep ensembles and Monte Carlo dropout seem to struggle regardless of the step sizes (for the chosen hyperparameters, e.g., number of epochs and dropout rate). In terms of the kernelized Stein discrepancy, it is found that low discrepancy values do not especially correspond to good coverage probabilities, as it has been observed for deep ensembles. Furthermore, for a fixed algorithm, the step sizes yielding the lowest KSDs do not especially correspond to those yielding to best coverage probabilities. This indicates that careful use of the KSD as a tool for hyperparameter selection is recommended, at least until its recently observed flaws [Wenliang and Kanagawa, 2020, Korba et al., 2021] are mitigated. However, it is important to note that our conclusions here are restricted to one-dimensional regression problems with shallow neural networks. Such findings should be confirmed on higher-dimensional problems and larger neural networks. While SGMCMC methods exhibit good performances, they remain difficult to scale to deeper neural networks and higher dimensional quantities of interests. Deep ensembles and Monte Carlo dropout do not suffer from this shortcoming and remain interesting candidates for high-dimensional regression tasks if the tuning of their underlying hyperparameters makes it possible to give reliable coverage probabilities.

8 Conclusion

In this work, we compare several inference algorithms for Bayesian neural networks. Assessing the quality of uncertainty estimates is a challenging task and we propose to evaluate the performance of selected algorithms by computing coverage probabilities and discrepancy measures (KSD, SKSD) between the target posterior distribution and its approximation. In contrast to previous works, we estimate coverage probabilities with a large number of independent datasets instead of estimating
prediction interval coverage probabilities with a single dataset. We argue that in order to assess the performance of an inference algorithm for uncertainty quantification, we should rely on coverage probabilities combined with a discrepancy measure such as the KSD. We hope that the source code made available online will be used and improved to better assess the performance of inference algorithms for uncertainty quantification in deep learning.
References

M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, Acharya U.R., V. Makarenkov, and S. Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information Fusion, 76:243–297, 2021.

J. Baker, P. Fearnhead, E.B. Fox, and C. Nemeth. Control variates for stochastic gradient mcmc. Statistics and Computing, 29(3):599–615, 2019.

Jack Baker. Large-Scale Bayesian Computation Using Stochastic Gradient Markov Chain Monte Carlo. Lancaster University (United Kingdom), 2019.

Jack Baker, Paul Fearnhead, Emily B Fox, and Christopher Nemeth. sgmc: An r package for stochastic gradient markov chain monte carlo. arXiv preprint arXiv:1710.00578, 2017.

W.H. Beluch, T. Genewein, A. Nürnberger, and J.M. Köhler. The power of ensembles for active learning in image classification. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 9368–9377, 2018.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In International conference on machine learning, pages 1613–1622. PMLR, 2015.

T. Chen, E. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. In International conference on machine learning, pages 1683–1691, 2014.

A.D. Cobb and B. Jalaian. Scaling hamiltonian monte carlo inference for bayesian neural networks with symmetric splitting. In Uncertainty in Artificial Intelligence, pages 675–685. PMLR, 2021.

K.A. Dubey, S. J. Reddi, S.A. Williamson, B. Poczos, A.J. Smola, and E.P. Xing. Variance reduction in stochastic gradient langevin dynamics. Advances in neural information processing systems, 29:1154–1162, 2016.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning, pages 1050–1059. PMLR, 2016.

J. Gawlikowski, C.R.N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, et al. A survey of uncertainty in deep neural networks. arXiv preprint arXiv:2107.03342, 2021.

M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73(2):123–214, 2011.

W. Gong, Y. Li, and J.M. Hernández-Lobato. Sliced kernelized stein discrepancy. arXiv preprint arXiv:2006.16531, 2020.

J. Gorham and L. Mackey. Measuring sample quality with kernels. In International Conference on Machine Learning, pages 1292–1301. PMLR, 2017.

A. Graves. Practical variational inference for neural networks. Advances in neural information processing systems, 24, 2011.

A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel method for the two-sample-problem. Advances in neural information processing systems, 19, 2006.

F.K. Gustafsson, M. Danelljan, and T.B. Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshaps, pages 318–319, 2020.

M.D. Hoffman, D.M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. Journal of Machine Learning Research, 2013.

P. Izmailov, S. Vikram, M.D. Hoffman, and A.G.W. Wilson. What are bayesian neural network posteriors really like? In International Conference on Machine Learning, pages 4629–4640. PMLR, 2021.
A. Korba, P.-C. Aubin-Frankowski, S. Majewski, and P. Ablin. Kernel stein discrepancy descent. In *International Conference on Machine Learning*, pages 5719–5730. PMLR, 2021.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

C. Li, C. Chen, D. Carlson, and L. Carin. Preconditioned stochastic gradient langevin dynamics for deep neural networks. In *Thirtyeth AAAI Conference on Artificial Intelligence*, 2016.

Q. Liu. Stein variational gradient descent as gradient flow. *Advances in neural information processing systems*, 30, 2017.

Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in neural information processing systems*, pages 2378–2386, 2016.

Q. Liu, J. Lee, and M. Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284. PMLR, 2016.

C. Louizos and M. Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, pages 2218–2227. PMLR, 2017.

Y. Ma, T. Chen, and E. Fox. A complete recipe for stochastic gradient mcmc. *Advances in neural information processing systems*, 28, 2015.

W.J. Maddox, P. Izmailov, T. Garipov, D.P. Vetrov, and A.G. Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

C. Nemeth and P. Fearnhead. Stochastic gradient markov chain monte carlo. *Journal of the American Statistical Association*, 116(533):433–450, 2021.

Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. jie Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

D. Sejdinovic, B. Sriperumbudur, A. Gretton, and K. Fukumizu. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The Annals of Statistics*, pages 2263–2291, 2013.

J.T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian optimization with robust bayesian neural networks. *Advances in neural information processing systems*, 29:4134–4142, 2016.

G.J. Székely, M.L. Rizzo, et al. Testing for equal distributions in high dimension. *InterStat*, 5(16.10):1249–1272, 2004.

O. Teymur, J. Gorham, M. Rjabiz, and C. Oates. Optimal quantisation of probability measures using maximum mean discrepancy. In *International Conference on Artificial Intelligence and Statistics*, pages 1027–1035. PMLR, 2021.

M. Welling and Y.W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

L.K. Wenliang and H. Kanagawa. Blindness of score-based methods to isolated components and mixing proportions. *arXiv preprint arXiv:2008.10087*, 2020.

Omry Y. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

J. Yao, W. Pan, S. Ghosh, and F. Doshi-Velez. Quality of uncertainty quantification for bayesian neural network inference. *arXiv preprint arXiv:1906.09686*, 2019.
C.2.1 Homoscedastic regression ...................................................... 22
C.2.2 Homoscedastic regression with mismatched prior uncertainty .......... 22
C.2.3 Homoscedastic regression with matched prior uncertainty .......... 23

D Code and reproductibility ................................................................. 23

List of Figures

1 Experiment #1: graphs $\epsilon \mapsto \hat{KSD}^2$ for each algorithm and three batch sizes. Left panel: batch size 1, middle panel: batch size 10, right panel: batch size 100. ........ 6
2 Experiment #4. Graphs of coverage probabilities $\alpha \mapsto C(\alpha)$ for each algorithm and 10 increasing step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The values of the step sizes are used to highlight the individual curves in each figure (dark blue: lowest step size, dark red: highest step size). ................................. 8
3 Experiment #4. Multidimensional scaling applied to the pairwise maximum mean discrepancies between the samples generated by each algorithm and each (converged) step size. The size of the markers is proportional to the value of the step size. ........ 8
4 Experiment #4. Boxplots of the kernelized Stein discrepancies $\hat{KSD}(P, \hat{Q}_j)$, $j = 1, \ldots, n_d$, obtained for 10 step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The colors of the boxes coincide with the color scheme used in Figure 2. ................................. 9
5 Experiments #1 - #3: graphs $\epsilon \mapsto \hat{KSD}^2$ for each algorithm, each batch size (columns), and the first three experiments (lines). ................................. 16
6 Experiment #5. Graphs of coverage probabilities $\alpha \mapsto C(\alpha)$ for each algorithm and 10 increasing step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The values of the step sizes are used to highlight the individual curves in each figure (dark blue: lowest step size, dark red: highest step size). ................................. 19
7 Experiment #6. Graphs of coverage probabilities $\alpha \mapsto C(\alpha)$ for each algorithm and 10 increasing step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The values of the step sizes are used to highlight the individual curves in each figure (dark blue: lowest step size, dark red: highest step size). ................................. 20
8 Experiment #4. Boxplots of the KSDs $\hat{KSD}(P, \hat{Q}_j)$ and $\hat{SKSD}(P, \hat{Q}_j)$, $j = 1, \ldots, n_d$, obtained for 10 step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The colors of the boxes coincide with the color scheme used in Figure 2. ................................. 24
9 Experiment #5. Boxplots of the KSDs $\hat{KSD}(P, \hat{Q}_j)$ and $\hat{SKSD}(P, \hat{Q}_j)$, $j = 1, \ldots, n_d$, obtained for 10 step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The colors of the boxes coincide with the color scheme used in Figure 2. ................................. 25
10 Experiment #6. Boxplots of the KSDs $\hat{KSD}(P, \hat{Q}_j)$ and $\hat{SKSD}(P, \hat{Q}_j)$, $j = 1, \ldots, n_d$, obtained for 10 step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The colors of the boxes coincide with the color scheme used in Figure 2. ................................. 26
11 Experiments #5 (top) and #6 (bottom). Multidimensional scaling applied to the pairwise maximum mean discrepancies between the samples generated by each algorithm and each (converged) step size. ................................. 27

List of Tables

1 Likelihood functions and prior distributions. The two-dimensional identity matrix is denoted by $I_2$. ...................................................... 6
2 Experiment #1. Lowest KSDs obtained for each algorithm and three batch sizes. Bold values correspond to the lowest KSDs over all the algorithms. ............. 7
Summary of the considered regression problems: target function, noise distribution, batch size, number of training samples \( N \), number of test samples \( N^{*} \), number of parameters \( d \) in the surrogate model.

Experiments \#1 - \#3. Lowest KSDs obtained for each algorithm and three batch sizes. Bold values correspond to the lowest KSDs over all the algorithms.

Experiments \#4 - \#6. Coverage probabilities obtained with each algorithm and 10 equally log-spaced values \( \epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10} \) for the hyperparameter \( \epsilon \). The target coverage probability is given by \( \alpha = 0.95 \). NA coverage probabilities correspond to simulations that diverged.

Ratios of simulations that have succeeded for every algorithms and step sizes. For each pair of algorithms and step sizes, 1000 simulations were performed for estimating the coverage probabilities.

Selected learning rates for each algorithm.

A Additional results

This section gathers results that are omitted in the main body for brevity. Results for the posterior distributions with analytical likelihoods (Experiments \#1 - \#3) are displayed in section A.1, and results for the regression problems (Experiments \#4 - \#6) can be found in section A.2.

A.1 Posterior distributions with analytical likelihoods

The lowest KSDs obtained for experiments \#1 - \#3 are reported in Table 4. Graphs of the KSDs with respect to the step size \( \epsilon \) are shown in Figure 5. In these low-dimensional experiments, it is found that SGHMC and its variants provide the lowest discrepancy values, and potentially the best approximations of the posterior distributions.

A.2 Posterior distributions in regression problems

This section gathers additional results for the regression problems. The coverage probabilities (level \( \alpha = 0.95 \)) are reported in Table 5 for all the regression experiments, algorithms, and step sizes. Graphs of the coverage probabilities \( \alpha \mapsto C(\alpha) \) are shown in Figures 6 and 7 for experiments \#5 and \#6. Graphs of the KSDs and SKSDs with respect to the step size can be seen in Figures 8, 9, and 10. The multidimensional scaling representation of the similarities between the algorithms are shown in Figure 11 for experiments \#5 and \#6. Note that for some step sizes, some SGCMC did not converge for all train datasets. The rates of succeeded simulations are reported in Table 6.

B Selected algorithms

B.1 Stochastic gradient Markov chain Monte Carlo algorithms

Stochastic gradient Markov chain Monte Carlo (SGMCMC) methods are a family of gradient-based algorithms that rely on the resolution of an Itô stochastic differential equation [Ma et al., 2015] (see also [Nemeth and Fearnhead, 2021] for a review). Amongst this family of methods, we consider the stochastic gradient Langevin dynamics (SGLD) [Welling and Teh, 2011] and the stochastic gradient Hamiltonian Monte Carlo (SGHMC) [Chen et al., 2014] methods, together with other variants that include variance reduction [Dubey et al., 2016, Baker et al., 2019] or preconditioning techniques [Li et al., 2016, Springenberg et al., 2016]. In what follows, the function \( U \) is such that the posterior distribution reads as \( p(\theta|\mathcal{D}) \propto \exp(-U(\theta)) \) where \( U \) denotes the opposite of the unormalized posterior p.d.f

\[
U(\theta) = -\log(p(\mathcal{D}|\theta)p(\theta)) = -\sum_{i=1}^{N} \log(p(y_i|x_i, \theta)) - \log(p(\theta)), \tag{17}
\]

In Eq. (17), \( p(y|x, \theta) \) denotes the likelihood function and \( p(\theta) \) the prior distribution of \( \theta \).
Figure 5: Experiments #1 - #3: graphs $\epsilon \mapsto \hat{\text{KSD}}^2$ for each algorithm, each batch size (columns), and the first three experiments (lines).

B.1.1 Stochastic gradient Langevin dynamics (SGLD)

The stochastic gradient Langevin dynamics algorithm proposed by Welling and Teh [2011] takes the form

$$k \geq 0, \quad \theta^{k+1} = \theta^k - \frac{\epsilon_k}{2} \hat{\nabla} U(\theta^k) + \sqrt{\epsilon_k} \Delta W^{k+1}. \quad (18)$$

where $\hat{\nabla} U$ denotes the following stochastic approximation of the true gradient $\nabla U$

$$\hat{\nabla} U(\theta) = \frac{N}{n} \sum_{i \in B_t} \nabla \log(p(y_i|x_i, \theta)) - \nabla \log(p(\theta)). \quad (19)$$

The set $B_t$ is a subset of size $n$ of the whole dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ that is randomly drawn at each iteration, and $\Delta W^{k+1} \sim \mathcal{N}(0, I)$. Herein, SGLD is implemented with a constant step size $\epsilon_k = \epsilon$ for all $k \geq 0$. 


Table 4: Experiments #1 - #3. Lowest KSDs obtained for each algorithm and three batch sizes. Bold values correspond to the lowest KSDs over all the algorithms.

| Experiment #1 | Batch size 1 | Batch size 10 | Batch size 100 |
|---------------|--------------|---------------|---------------|
| Algorithm     | $\epsilon$  | $KSD^2$       | $\epsilon$   | $KSD^2$       | $\epsilon$ | $KSD^2$ |
| SGLD          | 0.001        | 0.218288      | 0.002069      | 0.014349      | 0.002637   | 0.008441 |
| SGLD-CV       | 0.001274     | 0.012251      | 0.003360      | 0.007972      | 0.001624   | 0.001436 |
| SGLD-SVRG     | 0.002637     | 0.007631      | 0.005456      | 0.002507      | 0.004281   | 0.003574 |
| SGHMC         | 0.001        | 0.022469      | 0.006952      | 0.002877      | 0.002335   | 0.006047 |
| SGHMC-CV      | 0.002637     | 0.011093      | 0.011288      | 0.000807      | 0.029764   | 0.000694 |
| SGHMC-SVRG    | 0.002637     | 0.007631      | 0.005456      | 0.002507      | 0.004281   | 0.003574 |
| pSGLD         | 0.029764     | 0.035384      | 0.037927      | 0.005755      | 0.037927   | 0.005521 |
| SGHMC-SA      | 0.023357     | 0.046828      | 0.029764      | 0.006889      | 0.037927   | 0.005521 |

| Experiment #2 | Batch size 100 | Batch size 500 | Batch size 1000 |
|---------------|-----------------|-----------------|-----------------|
| Algorithm     | $\epsilon$     | $KSD^2$        | $\epsilon$     | $KSD^2$        | $\epsilon$ | $KSD^2$ |
| SGLD          | 0.001000       | 1.158591       | 0.001274       | 0.113849       | 0.001000   | 0.114181 |
| SGLD-CV       | 0.001624       | 0.220757       | 0.001274       | 0.110967       | 0.001000   | 0.056412 |
| SGLD-SVRG     | 0.001000       | 0.064284       | 0.001000       | 0.096341       | 0.001274   | 0.086304 |
| SGHMC         | 0.002069       | 0.034083       | 0.008859       | 0.003341       | 0.008859   | 0.004896 |
| SGHMC-CV      | 0.008859       | 0.021144       | 0.011288       | 0.003341       | 0.023357   | 0.000515 |
| SGHMC-SVRG    | 0.003360       | 0.012893       | 0.006952       | 0.012918       | 0.008859   | 0.011924 |
| pSGLD         | 0.014384       | 0.122160       | 0.023357       | 0.087891       | 0.018330   | 0.106972 |
| SGHMC-SA      | 0.014384       | 0.227717       | 0.014384       | 0.068923       | 0.023357   | 0.020550 |

| Experiment #3 | Batch size 100 | Batch size 400 | Batch size 800 |
|---------------|-----------------|-----------------|----------------|
| Algorithm     | $\epsilon$     | $KSD^2$        | $\epsilon$     | $KSD^2$        | $\epsilon$ | $KSD^2$ |
| SGLD          | 0.001274       | 0.053010       | 0.001274       | 0.019242       | 0.002069   | 0.016444 |
| SGLD-CV       | 0.001624       | 0.030847       | 0.001274       | 0.008404       | 0.002069   | 0.015406 |
| SGLD-SVRG     | 0.001624       | 0.019400       | 0.002069       | 0.020048       | 0.002069   | 0.016537 |
| SGHMC         | 0.004281       | 0.008833       | 0.023357       | 0.005140       | 0.011288   | 0.004008 |
| SGHMC-CV      | 0.004281       | 0.008183       | 0.006952       | 0.001861       | 0.006952   | 0.002040 |
| SGHMC-SVRG    | 0.004281       | 0.005572       | 0.002069       | 0.001830       | 0.006952   | 0.002050 |
| pSGLD         | 0.037927       | 0.097856       | 0.029764       | 0.058781       | 0.023357   | 0.085251 |
| SGHMC-SA      | 0.018330       | 0.051113       | 0.018330       | 0.018232       | 0.023357   | 0.030005 |

**B.1.2 Stochastic gradient Hamiltonian Monte Carlo (SGHMC)**

The stochastic gradient Hamiltonian Monte Carlo algorithm has been proposed by Chen et al. [2014] where one step of the algorithm reads as:

\[
k \geq 0 : \begin{cases} 
    \theta^{k+1} = \theta^k + \epsilon_k M^{-1} r^k, \\
    r^{k+1} = r^k - \epsilon_k \nabla U(\theta^k) - \epsilon_k CM^{-1} r^k + \sqrt{2C\epsilon_k} \Delta W^{k+1}, 
\end{cases}
\]  

(20)

where $r$ denotes the momentum and $\Delta W^{k+1} \sim \mathcal{N}(0, I)$. A constant step size is used, the matrices $M$ and $C$ are chosen as $M = I$ and $C = \alpha I$. In this setting, the algorithm reduces to

\[
k \geq 0 : \begin{cases} 
    \theta^{k+1} = \theta^k + \epsilon_k r^k, \\
    r^{k+1} = (1 - \alpha \epsilon_k) r^k - \epsilon_k \nabla U(\theta^k) + \sqrt{2\alpha \epsilon_k} \Delta W^{k+1}. 
\end{cases}
\]

(21)

The friction term $\alpha$ is chosen as $\alpha = 10$.

**B.1.3 Variants with variance reduction (SGLD-CV, SGHMC-CV, SGLD-SVRG, SGHMC-SVRG)**

We consider two variants of the SGLD and SGHMC algorithms that rely on control variates to reduce the variance of stochastic approximation $\nabla U$ of $\nabla U$. The stochastic gradient $\nabla$ is replaced by the
estimate given $\nabla U$ given by Eq. (8). Following Baker et al. [2019], SGHMC-CV and SGHMC-CV use an estimation of the maximum a posteriori (MAP) for the control variable $\eta$. Hence, the neural network is first trained in a classical fashion using an optimization algorithm, and the resulting values of the network parameters are stored into $\eta$. The remaining variants SGHMC-SVRG and SGHMC-CV-SVRG use the strategy proposed by Dubey et al. [2016] which consists in setting $\eta$ to a MAP estimate as well, but then updating $\eta$ every $m$ iterations as follows: $\eta = \theta^k$ if $\text{mod}(\ell, m) = 0$. The hyperparameter $m$ has been fixed to 100 in every experiment.

B.1.4 Preconditioned SGLD (pSGLD)

Ma et al. [2015] proposed the Riemannian SGLD by building upon the Riemannian manifold Langevin and Hamiltonian Monte Carlo methods Girolami and Calderhead [2011]. The Riemannian SGLD takes the form:

$$k \geq 0, \quad \theta^{k+1} = \theta^k - \frac{\epsilon_k}{2} \left( D(\theta^k) \nabla U(\theta^k) + \Gamma(\theta^k) \right) + \sqrt{\epsilon_k D(\theta^k) \Delta W^{k+1}}, \quad (22)$$
where $D \in \mathbb{R}^{d \times d}$ can be seen as a preconditioning matrix and the vector $\Gamma \in \mathbb{R}^d$ is given by

$$\Gamma_i(\theta) = \sum_j \frac{\partial}{\partial \theta_j} D_{ij}(\theta).$$

Several forms of the preconditioner $D$ can be used, the most common choice being a constant diagonal matrix $D = N^{-1}I$, where $N$ denotes the size of the training dataset. The expected Fisher information matrix is a natural choice for the preconditioner $D$ but it is usually intractable, and a few alternatives have been proposed in the literature. Herein we consider the preconditioned SGLD algorithm (pSGLD) of Li et al. [2016] that neglects the vector $\Gamma$ and uses a preconditioner inspired by the RMSprop algorithm. The gradient is scaled using a moving average of its norm at each iteration:

$$D(\theta^k) = \text{diag} \left( \lambda I + \sqrt{L(\theta^k)} \right)^{-1},$$

$$L(\theta^k) = \alpha L(\theta^{k-1}) + (1 - \alpha) \tilde{\nabla} U(\theta^{k-1}) \circ \tilde{\nabla} U(\theta^{k-1}),$$

where $\alpha$ is a parameter with values in $[0, 1]$, $\lambda$ is a regularization constant, and $\circ$ denotes the Hadamard (element-wise) product.

### B.1.5 Scale-adaptive Stochastic Gradient Hamiltonian Dynamics (SGHMC-SA)

The scale-adaptive SGHMC (referred to as SGHMC-SA) proposed by Springenberg et al. [2016] uses an adaptive strategy that automatically precomputes the matrices $M, C$ and an estimate $\hat{B}$ of the variance of the stochastic gradient during the burn-in phase of the Markov Chain. The inverse matrix $M^{-1}$ is chosen as $M^{-1} = \text{diag}(\hat{L}^{-1/2})$ where $\hat{L}$ is the second-order moment of the stochastic gradient which is estimated with an exponential moving average during the burn-in phase. The estimate $\hat{B}$ is approximated by the diagonal of $\frac{1}{2} \epsilon \hat{L}$ where $\epsilon$ is the step-size of the algorithm. The
Figure 7: Experiment #6. Graphs of coverage probabilities $\alpha \mapsto C(\alpha)$ for each algorithm and 10 increasing step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The values of the step sizes are used to highlight the individual curves in each figure (dark blue: lowest step size, dark red: highest step size).

The underlying kernel function is chosen such that $\hat{L}^{-1/2}C = 0.05I$. The resulting algorithm is given by:

$$
\begin{cases}
\theta^{k+1} = \theta^k + \epsilon^k, \\
\epsilon^{k+1} = -\epsilon^2 \hat{L}^{-1/2} \hat{U}(\theta^k) - \epsilon \hat{L}^{-1/2} C \epsilon^k + \sqrt{2 \epsilon^2 \hat{L}^{-1/2} C \hat{L}^{-1/2} - \epsilon^4 t} \Delta W^{k+1},
\end{cases}
$$

where $\hat{L}$ is estimated during the burnin-phase and fixed afterwards.

### B.2 Thinning of Markov chains with maximum mean discrepancy

Assume that a SGMCMC algorithm generates $T$ samples $\theta_1, \ldots, \theta_T$ and that we want to select $S << T$ samples. This amounts to find a sequence of indices $\pi \in \{1, \ldots, T\}^S$ such that $\theta_{\pi(1)}, \ldots, \theta_{\pi(S)}$ represent the selected samples. The sequence of indices $\pi$ is obtained with a greedy quantization algorithm [Teymur et al., 2021] that minimizes the maximum mean discrepancy (MMD). Let then $\hat{P}_T$ be the empirical distribution of the $T$ samples $\theta_1, \ldots, \theta_T$. At the $(i+1)$-th iteration of the quantization algorithm, the index $\pi(i+1)$ is obtained by minimizing the MMD as follows:

$$
\pi(i+1) = \arg \min_{j \in \{1, \ldots, T\}} \text{MMD}^2(\hat{P}_T, \hat{Q}_{i+1}(j)),
$$

where

$$
\hat{Q}_{i+1}(j) = \frac{1}{i+1} \sum_{\ell=1}^i \delta_\theta(\theta_{\pi(\ell)}) + \frac{1}{i+1} \delta_\theta(\theta_j)
$$

The underlying kernel function $k$ is chosen as the following characteristic distance-based kernel [Sejdinovic et al., 2013]

$$
k(\theta, \theta') = ||\theta||_2 + ||\theta'||_2 - ||\theta - \theta'||_2.
$$

In this setting, the MMD reduces to the energy distance [Székely et al., 2004].
Table 6: Ratios of simulations that have succeeded for every algorithms and step sizes. For each pair of algorithms and step sizes, 1000 simulations were performed for estimating the coverage probabilities.

| Experiment #4 | $\epsilon_1$ | $\epsilon_2$ | $\epsilon_3$ | $\epsilon_4$ | $\epsilon_5$ | $\epsilon_6$ | $\epsilon_7$ | $\epsilon_8$ | $\epsilon_9$ | $\epsilon_{10}$ |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| SGLD          | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 0.994       | 0.210       | NA          |
| SGLD-CV       | 0.997       | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 0.929       | 0.388       | 0.007       | NA          |
| SGLD-SVRG     | 1.0         | 1.0         | 1.0         | 1.0         | 0.986       | 1.0         | 1.0         | 0.988       | 0.122       | NA          |
| pSGLD         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         |
| SGHMC         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         |
| SGHMC-CV      | 1.0         | 1.0         | 1.0         | 1.0         | 0.996       | 1.0         | 1.0         | 1.0         | 1.0         | 0.993       |
| SGHMC-SVRG    | 1.0         | 0.999       | 0.682       | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         |
| SGHMC-SA      | 0.997       | 0.997       | 0.998       | 0.997       | 0.997       | 0.997       | 1.0         | 0.993       | 0.941       | 0.299       |

| Experiment #5 | $\epsilon_1$ | $\epsilon_2$ | $\epsilon_3$ | $\epsilon_4$ | $\epsilon_5$ | $\epsilon_6$ | $\epsilon_7$ | $\epsilon_8$ | $\epsilon_9$ | $\epsilon_{10}$ |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| SGLD          | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 0.888       | NA          | NA          | NA          | NA          |
| SGLD-CV       | 1.0         | 1.0         | 1.0         | 1.0         | 0.994       | 0.716       | 0.090       | NA          | NA          | NA          |
| SGLD-SVRG     | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 0.988       | 0.122       | NA          | NA          |
| pSGLD         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         |
| SGHMC         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         |
| SGHMC-CV      | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 0.982       | 0.388       | 0.007       | 0.007       |
| SGHMC-SVRG    | 1.0         | 0.999       | 0.682       | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         |
| SGHMC-SA      | 0.997       | 0.997       | 0.998       | 0.997       | 0.997       | 0.997       | 1.0         | 0.993       | 0.941       | 0.299       |

| Experiment #6 | $\epsilon_1$ | $\epsilon_2$ | $\epsilon_3$ | $\epsilon_4$ | $\epsilon_5$ | $\epsilon_6$ | $\epsilon_7$ | $\epsilon_8$ | $\epsilon_9$ | $\epsilon_{10}$ |
|---------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| SGLD          | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 0.986       | 0.422       | NA          | NA          | NA          |
| SGLD-CV       | 1.0         | 1.0         | 1.0         | 1.0         | 0.910       | 0.374       | 0.032       | NA          | NA          | NA          |
| SGLD-SVRG     | 1.0         | 1.0         | 1.0         | 1.0         | 0.994       | 0.736       | 0.040       | NA          | NA          | NA          |
| pSGLD         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         |
| SGHMC         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 0.998       |
| SGHMC-CV      | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 0.982       | 0.780       | 0.394       | NA          |
| SGHMC-SVRG    | 1.0         | 1.0         | 0.988       | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 0.824       | NA          |
| SGHMC-SA      | 1.0         | 1.0         | 1.0         | 0.974       | 1.0         | 1.0         | 0.994       | 0.408       | 0.046       | 0.012       |
| ensemble      | 1.0         | 0.994       | 1.0         | 0.998       | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         | 1.0         |

Table 7: Selected learning rates for each algorithm.

| Step size | Ensemble | MC dropout | pSGLD | SGHMC-SA | Others |
|-----------|----------|------------|-------|----------|--------|
| $\epsilon_1$ | 1.00e-05 | 1.00e-05 | 1.00e-03 | 1.00e-05 | 1.00e-07 |
| $\epsilon_2$ | 2.78e-05 | 2.78e-05 | 1.67e-03 | 2.15e-05 | 2.15e-07 |
| $\epsilon_3$ | 7.74e-05 | 7.74e-05 | 2.78e-03 | 4.64e-05 | 4.64e-07 |
| $\epsilon_4$ | 2.15e-04 | 2.15e-04 | 4.64e-03 | 1.00e-04 | 1.00e-06 |
| $\epsilon_5$ | 5.99e-04 | 5.99e-04 | 7.74e-03 | 2.15e-04 | 2.15e-06 |
| $\epsilon_6$ | 1.67e-03 | 1.67e-03 | 1.29e-02 | 4.64e-04 | 4.64e-06 |
| $\epsilon_7$ | 4.64e-03 | 4.64e-03 | 2.15e-02 | 1.00e-03 | 1.00e-05 |
| $\epsilon_8$ | 1.29e-02 | 1.29e-02 | 3.59e-02 | 2.15e-03 | 2.15e-05 |
| $\epsilon_9$ | 3.59e-02 | 3.59e-02 | 5.99e-02 | 4.64e-03 | 4.64e-05 |
| $\epsilon_{10}$ | 1.00e-01 | 1.00e-01 | 1.00e-01 | 1.00e-02 | 1.00e-04 |

C Datasets

C.1 Analytical likelihoods

C.1.1 Two-dimensional bimodal posterior distribution

This test case is taken from the SGLD paper [Welling and Teh, 2011]. The parameter to be inferred is a random variable $\theta = (\theta_1, \theta_2)$ with values in $\mathbb{R}^2$. The likelihood $p(x|\theta)$ is chosen as a Gaussian.
where

\[ p(x|\theta) = \frac{1}{2} \mathcal{N}(x|\theta_1, 2) + \frac{1}{2} \mathcal{N}(x|\theta_1 + \theta_2, 2). \]  

(29)

The prior distributions of \( \theta_1 \) and \( \theta_2 \) are assumed to be centered normals with variances 10 and 2, i.e.,

\[ p(\theta) = p(\theta_1) \times p(\theta_2), \quad p(\theta_1) = \mathcal{N}(\theta_1|0, 10), \quad p(\theta_2) = \mathcal{N}(\theta_2|0, 2). \]  

(30)

The dataset \( D = \{x_i\}_{i=1}^N \) is made of \( N = 100 \) samples of a \( \mathbb{R} \)-valued random variable generated from

\[ X \sim \frac{1}{2} \mathcal{N}(0, 2) + \frac{1}{2} \mathcal{N}(1, 2). \]  

(31)

### C.1.2 Four-dimensional multimodal posterior distribution

This test case is taken from [Baker et al., 2017]. Assume that our dataset \( D = \{x_i\}_{i=1}^N \) is made of \( N = 1000 \) observations from the Gaussian mixture

\[ p(x) = 0.5 \mathcal{N}(x|0, I_2) + 0.5 \mathcal{N}(x|[0.1, 0.1]^T, I_2), \]

where \( I_2 \) denotes the \( 2 \times 2 \) identity matrix. Let \( \theta \) and \( \theta' \) be two parameters to be inferred with values in \( \mathbb{R}^2 \). The likelihood function is chosen as the Gaussian mixture

\[ p(x|\theta, \theta') = \frac{1}{2} \mathcal{N}(x|\theta, I_2) + \frac{1}{2} \mathcal{N}(x|\theta', I_2), \]  

(32)

and the prior distributions of \( \theta \) and \( \theta' \) are both given by \( \mathcal{N}(-0, 10I_2) \).

### C.1.3 Two-dimensional warped Gaussian

This example is taken from the thesis [Baker, 2019]. The parameter to be inferred is a two-dimensional random variable \( \theta \). The likelihood is chosen as

\[ p(x|\theta) = \mathcal{N}(x|\theta_1, \theta_2^2, 9), \]  

(33)

and the prior distributions are given by

\[ p(\theta) = p(\theta_1) \times p(\theta_2), \quad p(\theta_i) = \mathcal{N}(\theta_i|0, 0.1), \quad i = 1, 2. \]  

(34)

The synthetic dataset \( D = \{x_i\}_{i=1}^N \) is constructed by generating \( N = 800 \) samples from \( X \sim \mathcal{N}(0.5, 9) \).

### C.2 Regression problems

#### C.2.1 Homoscedastic regression

This first regression test case is a homoscedastic regression of a one-dimensional function. The dataset \( D = \{x_i, y_i\}_{i=1}^N \) is constructed by generating \( N = 100 \) samples \( x_1, \ldots, x_N \) of the one-dimensional input \( x \) uniformly between \( [-3, 3] \). The associated targets \( y_1, \ldots, y_{100} \) are given by

\[ y_i = \cos(2x_i) + \sin(x_i) + \sqrt{3}u_i, \quad i = 1, \ldots, N, \]  

(35)

where \( u_i \sim \mathcal{N}(0, 1) \). An additional test dataset is generated as well and made of 200 samples with an input \( x \) being uniformly drawn within \( [-5, 5] \) instead of \( [-3, 3] \). A feed-forward neural network with 1 hidden layer and 100 hidden units is used as a surrogate model (\( d = 10401 \) parameters). The prior distribution of its 10400 parameters is chosen as a centered Gaussian distribution with covariance matrix \( I \).

#### C.2.2 Homoscedastic regression with mismatched prior uncertainty

This test case is taken from [Yao et al., 2019] which consists in a homoscedastic regression of a one-dimensional function but with some missing samples in the input space. The \( N = 200 \) targets are given by

\[ y_i = 0.1 x_i^3 + \epsilon, \quad i = 1, \ldots, 200 \]  

(36)

where \( \epsilon \sim \mathcal{N}(0, 0.25) \). The inputs \( x_1, \ldots, x_N \) of a training dataset \( D \) are uniformly sampled from \([-4, -1] \cup [1, 4]\). In contrast, the inputs of a test dataset \( D^* \) are uniformly distributed between \([-4, 4]\).
C.2.3 Homoscedastic regression with matched prior uncertainty

This test case is also taken from [Yao et al., 2019] which consists in a homoscedastic regression of a one-dimensional function but with some missing samples in the input space. The $N = 82$ targets are given by

$$y_i = -(1 + x_i) \sin(1.2x_i) + \epsilon, \quad i = 1, \ldots, 82,$$

(37)

where $\epsilon \sim \mathcal{N}(0, 0.25)$. The first 80 inputs $x_1, \ldots, x_{80}$ of a training dataset $D$ are uniformly sampled in $[-6, -2] \cup [2, 6]$, and the last two samples $x_{81}, x_{82}$ are uniformly sampled in $[-2, 2]$. In contrast, the inputs of a test dataset $D^*$ are all uniformly distributed in $[-6, 6]$. The surrogate model is a feed-forward neural network with two hidden layers with each 50 hidden units ($d = 5251$ parameters).

D Code and reproducibility

All the results presented in this work can be reproduced with the code provided in the supplemental material and which relies on PyTorch [Paszke et al., 2019]. The code will be uploaded online if the present manuscript is accepted. Five pre-implemented pipelines are available to the user to perform: maximum a posteriori (MAP), sampling (SGMCMC methods), deep ensembles, Monte Carlo dropout, or Stein variational gradient descent [Liu and Wang, 2016]. Each pipeline has been implemented with Hydra [Y., 2019] and is launched via the command line by specifying a configuration file that describes the experiment. All the algorithms can also be used within a regular python script without relying on the pre-implemented pipelines. More details and examples can be found in the repository.
Figure 8: Experiment #4. Boxplots of the KSDs $\hat{\mathcal{K}}_{SD}(P, \hat{Q}_j)$ and $\hat{SKSD}(P, \hat{Q}_j)$, $j = 1, \ldots, n_d$, obtained for 10 step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The colors of the boxes coincide with the color scheme used in Figure 2.
Figure 9: Experiment #5. Boxplots of the KSDs $\hat{\text{KSD}}(P, \hat{Q}_j)$ and $\hat{\text{SKSD}}(P, \hat{Q}_j)$, $j = 1, \ldots, n_d$, obtained for 10 step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The colors of the boxes coincide with the color scheme used in Figure 2.
Figure 10: Experiment #6. Boxplots of the KSDs $\hat{KSD}(P, \hat{Q}_j)$ and $\hat{SKSD}(P, \hat{Q}_j)$, $j = 1, \ldots, n_d$, obtained for 10 step sizes $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_{10}$. The colors of the boxes coincide with the color scheme used in Figure 2.
Figure 11: Experiments #5 (top) and #6 (bottom). Multidimensional scaling applied to the pairwise maximum mean discrepancies between the samples generated by each algorithm and each (converged) step size.