Structured Stochastic Gradient MCMC: a hybrid VI and MCMC approach

Antonios Alexos*, Alex Boyd*, Stephan Mandt
Department of Computer Science
University of California, Irvine
{aalexos,alexjb,mandt}@uci.edu

Abstract

Stochastic gradient Markov chain Monte Carlo (SGMCMC) is considered the gold standard for Bayesian inference in large-scale models, such as Bayesian neural networks. Since practitioners face speed versus accuracy tradeoffs in these models, variational inference (VI) is often the preferable option. Unfortunately, VI makes strong assumptions on both the factorization and functional form of the posterior. In this work, we propose a new non-parametric variational approximation that makes no assumptions about the approximate posterior’s functional form and allows practitioners to specify the exact dependencies the algorithm should respect or break. The approach relies on a new Langevin-type algorithm that operates on a modified energy function, where parts of the latent variables are averaged over samples from earlier iterations of the Markov chain. This way, statistical dependencies can be broken in a controlled way, allowing the chain to mix faster. This scheme can be further modified in a “dropout” manner, leading to even more scalability. By implementing the scheme on a ResNet-20 architecture, we obtain better predictive likelihoods and larger effective sample sizes than full SGMCMC.

1 Introduction

Bayesian neural networks (BNNs) rely on scalable MCMC approaches based on inexpensive stochastic gradients, of which stochastic gradient Markov chain Monte Carlo (SGMCMC) algorithms are the gold standard (Li et al., 2016; Welling and Teh, 2011; Patterson and Teh, 2013). The main downside of using SGMCMC algorithms in high dimensions is their slow mixing rates since they explore the parameter space in a random walk fashion. Alternative, faster methods include variational inference (VI) algorithms which approximate the posterior distribution with a simpler (often factorized) one. In this work, we derive a fundamentally new SGMCMC approach that takes inspiration from structured VI. While our approach remains a sampling algorithm resembling SGMCMC, we speed up the mixing time by systematically breaking posterior correlations. Furthermore, the resulting algorithm allows users to specify which posterior correlations to keep and which ones to break all while making no assumptions on the functional form of the approximate posterior. We call our approach structured SGMCMC since it relies on a structured (i.e., partially factorized) variational approximation of the posterior (Wainwright and Jordan, 2008).

Related Work

Our work connects both to (stochastic) variational inference (Bishop, 2006; Hoffman et al., 2013; Ranganath et al., 2014; Blei et al., 2017; Zhang et al., 2018) and scalable MCMC (Welling and Teh, 2011; Chen et al., 2014; Ma et al., 2017; Zhang et al., 2020; Leimkuhler et al., 2019; Wenzel et al., 2020; Izmailov et al., 2021). For space limitations, we focus on the most related work at the

*Denotes equal contribution.
intersection of both topics. (de Freitas et al., 2001) constructed a variational proposal distribution in the Metropolis-Hastings step of MCMC. An improved approach to that was introduced in (Habib and Barber, 2018), where they introduce low-dimensional auxiliary variables. Other related advances to MCMC Levy et al. (2017) developed a method to train MCMC kernels with NNs, and Wang et al. (2018); Gong et al. (2018) who leveraged meta learning schemes in SGMCMC methods. Other works aim at explicitly interpolating between MCMC and VI. Domke (2017) proposes a divergence bound for hybridizing VI and MCMC, essentially by running Langevin dynamics on a tempered evidence lower bound (ELBO). Salimans et al. (2015) embody MCMC steps into the variational inference approximation. Ahn et al. (2012) leverage the central limit theorem and use the estimated inverse Fisher information matrix to sample from the approximate posterior distribution. Rezende and Mohamed (2015) interpreted the path of an MCMC algorithm as a variational distribution. Recently, Hoffman and Ma (2020) interpreted (parametric) VI as approximate Langevin dynamics and showed that both algorithms have similar transient dynamics.

2 Proposed Methodology

Structured SGMCMC Given data \( D = \{(x_i, y_i)\}_{i=1,...,N} \), parameters \( \theta \), a proper prior distribution \( p(\theta) \), and a likelihood \( p(D|\theta) = \prod_{i=1}^{N} p(y_i|x_i, \theta) \), suppose we are interested in the corresponding posterior distribution \( p(\theta|D) \propto p(D|\theta)p(\theta) \). A convenient representation of the posterior is as a Boltzmann distribution:

\[
p(\theta|D) \propto \exp\{-U(\theta)\} \quad \text{where} \quad U(\theta) = -\log p(\theta, D) = -\sum_{(x,y) \in D} \log p(y|x, \theta) - \log p(\theta).
\]

(1)

\( U \) is typically referred to as the posterior energy function. The basis of SGMCMC algorithms is using a sampled minibatch of data \( \hat{D} \) from \( D \) to produce an differentiable, unbiased estimate of the posterior energy function:

\[
U(\theta) \approx \hat{U}(\theta; \hat{D}) = \frac{1}{|\hat{D}|} \sum_{(x,y) \in \hat{D}} \log p(y|x, \theta) - \log p(\theta).
\]

(2)

To achieve partial factorization, we must first partition \( \theta \) into \( M > 1 \) distinct, mutually independent groups: \( \theta_1, \ldots, \theta_M \). This partitioning structure is assumed to be known a priori. We will denote the distribution that respects this partitioning structure as \( q(\theta) = \prod_{i=1}^{M} q_i(\theta_i) \). Similar to VI, we would like this distribution \( q(\theta) \) to best approximate the true posterior distribution \( p(\theta|D) \) according to some criteria, such as KL-divergence. This leads to a natural objective function to minimize:

\[
J(q(\theta)) = D_{KL}(q(\theta)||p(\theta|D)) \equiv \mathbb{E}_{\theta \sim q} \left[ \log \frac{q(\theta)}{p(\theta|D)} \right]
\]

(3)

The following Theorem 1 proves that there is a unique solution to the non-parametric KL minimization problem described in Eq. (3). To describe it, we compose \( \theta = \{\theta_i, \tilde{\theta}_{-i}\} \) for any \( i \) where \( \tilde{\theta} \sim q \) and define a structured energy function:

\[
U_i^{(S)}(\theta) = \sum_{i=1}^{M} U_i^{(S)}(\theta_i), \quad \text{with} \quad U_i^{(S)}(\theta_i) := \mathbb{E}_{\theta \sim q} U(\{\theta_i, \tilde{\theta}_{-i}\}) := -\mathbb{E}_{\theta \sim q} \log p(\theta_i, \tilde{\theta}_{-i}, D).
\]

(4)

That is, we first define the marginals \( U_i^{(S)}(\theta_i) \), where we marginalize \( U(\theta) \) with respect to all \( q(\theta) \)-factors except \( q_i(\theta_i) \), and then sum up these marginals to define \( U^{(S)}(\theta) \). Having a well-defined energy function \( U^{(S)}(\theta) \) allows us to use standard SGMCMC methods to approximate the posterior \( q(\theta) \) with samples. This serves as the basis for our proposed algorithm that actually approximates this distribution \( q(\theta) \), which will be discussed shortly.

**Theorem 1.** The unique solution to the KL minimization problem given in Eq. 3 is given by the Boltzmann distribution \( q(\theta) \propto \exp\{-\sum_{i=1}^{M} U_i^{(S)}(\theta_i)\} \).

Our proposed algorithm that actually approximates distribution \( q(\theta) \) is the following:

\[
U^{(S)}(\theta) \approx \hat{U}^{(S)}(\theta; \hat{D}) = \sum_{i=1}^{M} \mathbb{E}_{\theta \sim q} \hat{U}(\{\theta_i, \tilde{\theta}_{-i}\}; \hat{D}),
\]

(5)
where $\tilde{U}(\cdot)$ is defined in Eq. (2). The update step for structured SGLD (S-SGLD) is the following:

$$
\tilde{\theta}^{(t+1)} = \theta^{(t)} - \frac{\epsilon_t}{2} \nabla_{\theta} \tilde{U}^{(S)}(\theta; \tilde{D}) + \xi_t \text{ where } \xi_t \sim \mathcal{N}(0, \epsilon_t I).
$$

**Structured Dropout SGMCMC**

This section presents such a method that both closely resembles the S-SGMC procedure and scales independently from the partitioning scheme., which we call **structured dropout SGMCMC** ($S_d$-SGMC). The main motivation for this technique can be seen by recognizing that the composition $\{\tilde{\theta}_i^{(t)}, \tilde{\theta}_{i=1}^{(t)}\}$ from Eq. (5) can be rewritten as a sum of masked values $r\theta^{(t)} + (1-r)\tilde{\theta}^{(t)}$ where $\tilde{\theta}^{(t)} \sim q^{(t)}$ and $r_i = 1(i = j)$ for $i = 1, \ldots, M$. We can decouple the computational scaling from the number of parameter groups $M$ by replacing the $M$ deterministic masks $r$'s with $K$ stochastically sampled masks $\tilde{r}$.\(^2\) Doing so results in a slightly different energy function and minibatch loss to optimize:

$$
\tilde{U}^{(S_d)}(\theta^{(t)}; \tilde{D}) \approx \frac{M}{K} \sum_{i=1}^M \tilde{U}(\tilde{r}^{(t,k)} \theta^{(t)} + (1 - \tilde{r}^{(t,k)}) \tilde{\theta}^{(t)}; \tilde{D})
$$

where $\tilde{r}^{(t,k)}$ is the $k^{th}$ sample of $\tilde{r}$ for timestep $t$. These energy function approximations lead to the following update step for structured dropout variant of SGLD ($S_d$-SGLD):

$$
\theta^{(t+1)} = \theta^{(t)} - \frac{\epsilon_t}{2} \nabla_{\theta} \tilde{U}^{(S_d)}(\theta; \tilde{D}) + \xi_t \text{ where } \xi_t \sim \mathcal{N}(0, \epsilon_t I).
$$

### 3 Experiments

This section studies our methods with ResNet-20 (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and Fashion MNIST (Xiao et al., 2017) and compares them for their accuracy. Our experiments reveal that the chains in our proposed methods mix faster than SGMCMC and achieve either comparable or even higher accuracies on average. The primary predictive metric of interest used to evaluate our proposal is classification accuracy. We take the average of an ensemble of 100 models whose weights are sampled from the past samples of the parameters chains in order to calculate the accuracy. We deploy pSGLD and SGHMC as optimizers for our experiments. Bernoulli($\rho$) and uniform masking distributions were investigated on either fully joint or fully factorized posteriors. In Fig. 1 we observe how quickly the proposed methods and the baseline SGMCMC methods approach their optimum accuracy over the course of training. As is shown, our proposed methods appear to achieve optimal accuracy values much faster than SGMCMC. Sometimes as in the case of CIFAR10 we achieve a higher accuracy score with our methods than with SGMCMC. **Full paper is in the Appendix.**

\(^2\) $K$ is a hyperparameter that is chosen independent of $M$; however, both $M$ and the distribution of $\tilde{r}$ largely influence how small $K$ can be due to how they affect the variance of the gradient of the associated posterior energy function.
References


4 Introduction

There has been much recent interest in deep Bayesian neural networks (BNN) due to their reliable confidence estimates and generalization properties (Wilson and Izmailov, 2020; Jospin et al., 2020; Cardelli et al., 2019). BNNs rely on ensemble averages over model parameters typically obtained from Markov chain Monte Carlo (MCMC) algorithms, which contrasts to regular neural networks that depend on a single set of parameters. The sheer size of these models requires scalable MCMC approaches based on inexpensive stochastic gradients, of which stochastic gradient Markov chain Monte Carlo (SGMCMC) algorithms are the gold standard (Li et al., 2016; Welling and Teh, 2011; Patterson and Teh, 2013). These algorithms owe their scalability to approximating gradients via mini-batching.

The main downside of using SGMCMC algorithms in high dimensions is their slow mixing rates since they explore the parameter space in a random walk fashion. An often faster alternative is variational inference (VI) algorithms that approximate the posterior with a simpler (typically factorized) distribution. This formulation results in an optimization problem that can be solved more efficiently using stochastic optimization (Blei et al., 2017; Zhang et al., 2018).

One downside of VI approximations is their solid distributional assumptions. A typical choice is to approximate the Bayesian posterior by a product of univariate Gaussian distributions. These distributional assumptions are frequently over-simplistic in high-dimensional models, where the posterior can be highly multi-modal and possibly heavy-tailed. Another downside is that the variational approximation typically underestimates the posterior variance, leading to poorly calibrated uncertainties and overfitting (Ormerod and Wand, 2010; Giordano et al., 2015; Zhang et al., 2018).

In this work, we derive a fundamentally new SGMCMC approach that takes inspiration from structured VI. While our approach remains a sampling algorithm resembling SGMCMC, we speed up the mixing time by systematically breaking posterior correlations. The resulting algorithm furthermore allows users to specify which posterior correlations to keep and which ones to break. It makes no assumptions on the functional form of the approximate posterior. We call our approach structured SGMCMC since it relies on a structured (i.e., only partially factorized) variational approximation of the posterior (Wainwright and Jordan, 2008).

In more detail, we derive the optimal variational distribution for a given posterior subject to factorization constraints by assuming a functional view on variational inference. We show how to sample from this optimal distribution by running SGMCMC on a modified energy function. This energy function is obtained by marginalizing the model’s joint distribution over previously generated samples from the Markov chain, leading to an approximate factorization over user-specified parameter groups. Further, we provide a more robust and computationally efficient approximation to the procedure that allows for interpolation between regular SGMCMC and our structured SGMCMC by taking inspiration from dropout techniques. Both methods are compatible with any Markovian SGMCMC algorithm, including Langevin dynamics and stochastic gradient Hamiltonian Monte Carlo.

In sum, our contributions are as follows:

• We propose a new approximate MCMC scheme running SGMCMC on a modified energy function, trading accuracy for speed. This setup effectively allows sampling from a fully joint posterior, a completely factorized posterior, and any in-between.

• We prove mathematically that the resulting scheme asymptotically generates samples from the best possible posterior approximation subject to user-specified factorization constraints between groups of parameters.

• We extend this scheme further by making it more scalable with a dropout-inspired approximation. This new scheme has a hyperparameter that enables a smooth interpolation between full SGMCMC and a "mean-field" version where all posterior correlations are broken.

• We show in both small and large scale experiments that our method well-approximates posterior marginals and gives improved results over SGMCMC on Resnet-20 architectures on CIFAR-10, Fashion MNIST, and SVHN in terms of both runtime and final accuracy.

Our paper is structured as follows: Section 5 presents the related work to our proposal, Section 6 introduces preliminaries regarding the energy function and the stochastic gradient updates, Sections 7 and 8 derive our proposed methods, Section 9 details experiments and their results, and Section 10 contains our concluding thoughts.
5 Related Work

Our work connects both to (stochastic) variational inference (Bishop, 2006; Hoffman et al., 2013; Ranganath et al., 2014; Blei et al., 2017; Zhang et al., 2018) and scalable MCMC (Welling and Teh, 2011; Chen et al., 2014; Ma et al., 2017; Zhang et al., 2020; Leimkuhler et al., 2019; Wenzel et al., 2020; Izmailov et al., 2021). For space limitations, we focus on the most related work at the intersection of both topics.

Among the earliest works to hybridize both approaches was (de Freitas et al., 2001) who constructed a variational proposal distribution in the Metropolos-Hastings step of MCMC. An improved approach to that was introduced in (Habib and Barber, 2018), where by introducing low-dimensional auxiliary variables they fit a more accurate approximating distribution.

Most recent work focuses on connections between VI and stochastic gradient-based MCMC, or between VI and stochastic gradient descent (SGD). For example, Mandt et al. (2016, 2017) and Duvenaud et al. (2016) consider SGD as a type of variational inference, but their approaches did not attempt to close the gap to exact MCMC. Other works aim at explicitly interpolating between both methods. Domke (2017) proposes a divergence bound for hybridizing VI and MCMC, essentially by running Langevin dynamics on a tempered evidence lower bound (ELBO). Salimans et al. (2015) embody MCMC steps into the variational inference approximation. Ahn et al. (2012) improve stochastic gradient Langevin dynamics by leveraging the central limit theorem and using the estimated inverse Fisher information matrix to sample from the approximate posterior distribution. Rezende and Mohamed (2015) interpreted the path of an MCMC algorithm as a variational distribution, and then fitting parameters to tighten a variational bound. Recently, Hoffman and Ma (2020) interpreted (parametric) VI as approximate Langevin dynamics and showed that both algorithms have similar transient dynamics.

Our approach is different from all these papers in that it is a non-parametric, structured mean-field VI scheme inspired by coordinate VI updates (Bishop, 2006). At the same time, it is also an approximate SGMCMC scheme in that the variational distribution is a Markov Chain and the ELBO is implicit.

6 Preliminaries

Variational inference (VI) approaches differ from MCMC in two regards: (1) they impose a structured (e.g., fully-factorized) approximation of the posterior for tractability, and (2) they often make parametric assumptions. Is it possible to construct a modified scheme that only relies on the assumption (1), inheriting the non-parametric nature of MCMC while breaking posterior correlations in a controlled manner? The answer is affirmative and will be answered as follows. We will first derive a modified energy function for Langevin dynamics that we can sample from and then prove that its negative exponential results in the optimal posterior approximation subject to specified factorization constraints. Running SGMCMC algorithms on this energy function will consequently generate samples from this distribution.

Before we explain our new method, we introduce the setup and common notation. Given data \( D = \{(x_i, y_i)\}_{i=1,...,N} \), parameters \( \theta \), a proper prior distribution \( p(\theta) \), and a likelihood \( p(D|\theta) = \prod_{i=1}^{N} p(y_i|x_i, \theta) \), suppose we are interested in the corresponding posterior distribution \( p(\theta|D) \propto p(D|\theta)p(\theta) \). A convenient representation of the posterior is as a Boltzmann distribution:

\[
p(\theta|D) \propto \exp\{-U(\theta)\} \quad \text{where} \quad U(\theta) = -\log p(\theta, D) = -\sum_{(x,y) \in D} \log p(y|x, \theta) - \log p(\theta). \quad (9)
\]

\( U \) is typically referred to as the posterior energy function. Note that the posterior distribution is typically intractable due to the normalizing constant.

The gold standard for approximating the entire posterior distribution is by deploying Markov chain Monte Carlo (MCMC) algorithms. These methods work by producing an empirical distribution of samples through a random walk in parameter space. While being very accurate and having asymptotic guarantees, these methods are known to not scale well with respect to both data and parameters (Brooks et al., 2011; Geyer, 1992).

Stochastic gradient MCMC (SGMCMC) is a class of scalable MCMC algorithms that can produce posterior samples through gradients on minibatches of data. These algorithms are largely derived from
discretized approximations of continuous-time diffusion processes. Examples of these algorithms include stochastic gradient Langevin dynamics (SGLD) (Welling and Teh, 2011), preconditioned SGLD (pSGLD) (Li et al., 2016), and stochastic gradient Hamiltonian Monte Carlo (SGHMC) (Chen et al., 2014).

As alluded to, the basis of SGMCMC algorithms is using a sampled minibatch of data \( \tilde{D} \) from \( D \) to produce an differentiable, unbiased estimate of the posterior energy function:

\[
U(\theta) \approx \hat{U}(\theta; \tilde{D}) = -\frac{N}{|\tilde{D}|} \sum_{(x,y) \in \tilde{D}} \log p(y|x, \theta) - \log p(\theta).
\]  

(10)

Once \( \hat{U} \) is defined, it is fairly straightforward to generate new samples from the posterior distribution. For instance, the SGLD update is

\[
\theta^{(t+1)} = \theta^{(t)} - \frac{\epsilon_t}{2} \nabla_{\theta} \hat{U}(\theta^{(t)}; \tilde{D}_t) + \xi_t \quad \text{where} \quad \xi_t \sim \mathcal{N}(0, \epsilon_t I).
\]  

(11)

Similar rules for pSGLD and SGHMC can be found in the Supplement. All of these update rules produce a chain of samples up to time step \( t \) that ultimately form an empirical distribution \( \hat{p}^{(t)}(\theta|\mathcal{D}) \). Should the algorithms converge, then \( \lim_{t \to \infty} \hat{p}^{(t)}(\theta|\mathcal{D}) = p(\theta|\mathcal{D}) \).

7 Structured SGMCMC

By design, SGMCMC methods produce a completely joint posterior distribution over parameters \( \theta \). For models with a large number of parameters, this can lead to various complications due to the curse of dimensionality. This is typically observed with slow convergence times and potentially unexplored parameter spaces. A viable solution is to break dependencies in the posterior distribution by leveraging ideas commonly used in variational inference (VI). This would reduce the number of various potential posterior correlations that the model would need to capture while sampling.

To achieve partial factorization, we must first partition \( \theta \) into \( M \geq 1 \) distinct, mutually independent groups: \( \theta_1, \ldots, \theta_M \). This partitioning structure is assumed to be known a priori. We will denote the distribution that respects this partitioning structure as \( q(\theta) = \prod_{i=1}^{M} q_i(\theta_i) \). Similar to VI, we would like this distribution \( q(\theta) \) to best approximate the true posterior distribution \( p(\theta|\mathcal{D}) \) according to some criteria, such as KL-divergence. This leads to a natural objective function to minimize:

\[
J(q(\theta)) = D_{KL}(q(\theta)|| p(\theta|\mathcal{D})) \equiv \mathbb{E}_{q \sim q} \left[ \log \frac{q(\theta)}{p(\theta|\mathcal{D})} \right] \quad \text{(12)}
\]

The following Theorem 2 proves that there is a unique solution to the non-parametric KL minimization problem described in Eq. (12). To describe it, we compose \( \theta = \{ \theta_i, \tilde{\theta}_{-i} \} \) for any \( i \) where \( \theta \sim q \) and define a structured energy function:

\[
U^{(S)}(\theta) = \sum_{i=1}^{M} U_i^{(S)}(\theta_i), \quad \text{with} \quad U_i^{(S)}(\theta_i) := \mathbb{E}_{\tilde{\theta}_{-i} \sim q} U(\{ \theta_i, \tilde{\theta}_{-i} \}) := -\mathbb{E}_{\tilde{\theta}_{-i} \sim q} \log p(\theta_i, \tilde{\theta}_{-i}, \mathcal{D}).
\]  

(13)

That is, we first define the marginals \( U_i^{(S)}(\theta_i) \), where we marginalize \( U(\theta) \) with respect to all \( q(\theta) \)-factors except \( q_i(\theta_i) \), and then sum up these marginals to define \( U^{(S)}(\theta) \). A similar partial marginalization procedure is carried out for conjugate exponential family distributions in coordinate ascent VI (Bishop, 2006). Having a well-defined energy function \( U^{(S)} \) allows us to use standard SGMCMC methods to approximate the posterior \( q(\theta) \) with samples. This serves as the basis for our proposed algorithm that actually approximates this distribution \( q(\theta) \), which will be discussed shortly.

**Theorem 2.** The unique solution to the KL minimization problem given in Eq. 12 is given by the Boltzmann distribution \( q(\theta) \propto \exp\{ -\sum_{i=1}^{M} U_i^{(S)}(\theta_i) \} \). Please refer to the Supplement for the proof.

In an ideal world, we would be able to use the findings of Theorem 2 directly in conjunction with algorithms like Langevin dynamics and Hamiltonian Monte Carlo to produce empirical distributions for \( q \) using \( U^{(S)} \) (Liu et al., 2019). However, this is intractable for two reasons: (1) these algorithms generally work only well with small amounts of data, and (2) more importantly, the marginals
Figure 2: Sampled posterior distributions between $w_1$ & $w_2$ (top row) and $w_1$ & $w_3$ (bottom row) for a linear regression model. From left to right, (a) the first column has posterior distributions sampled with SGLD, (b) the second and third columns are sampled from S-SGLD, and (c) the fourth and fifth are sampled from $S_{\sigma}$-SGLD. The imposed parameter structure for each are shown above the plots.

$U_i^{(S)}(\theta_i)$ do not have a closed-form solution but need to be approximated via samples from $q$. Luckily, since SGMCMC methods only need access to noisy estimates of $U^{(S)}$, we can run these algorithms on a stochastic estimate of Eq. (13),

$$U^{(S)}(\theta) \approx \hat{U}^{(S)}(\theta; \hat{D}) = \sum_{i=1}^{M} \mathbb{E}_{\hat{\theta} \sim q} \hat{U}((\theta_i, \hat{\theta}_{-i}); \hat{D}),$$

where $\hat{U}(\cdot)$ is defined in Eq. (10). In practice, at timestep $t$ for $i = 1, \ldots, M$ we estimate $\mathbb{E}_{\hat{\theta} \sim q} \hat{U}((\theta_i, \hat{\theta}_{-i}); \hat{D}_t)$ with a Monte Carlo approximation using a single sample $\hat{\theta}^{(t,i)}$ from the current approximate distribution $\hat{q}^{(t)}$ which is composed of samples from previous timesteps. This leads to the following update step for structured SGLD (S-SGLD):

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\epsilon_t}{2} \nabla_{\theta} \hat{U}^{(S)}(\theta; \hat{D}) + \xi_t \text{ where } \xi_t \sim \mathcal{N}(0, \epsilon_t I).$$

Similar rules for structured variants of pSGLD (S-pSGLD) and SGHMC (S-SGHMC) can be found in the Supplement. Additionally, the full procedure for structured SGMCMC (S-SGMCMC) can be seen in Algorithm 1.

**Remark** Since $\nabla_{\theta} \hat{U}^{(S)}$ is an unbiased estimator for $U^{(S)}$, we are guaranteed to converge to $q$ from sampling with S-SGMCMC with sufficiently decreasing learning rates so long as we are in a stationary state. While it is unlikely to have the procedure initialize to a stationary state, we observe in practice that our scheme both tends to converge towards and remain in a stationary state. A general proof of convergence is outside the scope of this work and is left to follow-up research.

An example of S-SGMCMC can be seen in Fig. 2(a-b), which features the approximate posterior distributions of a linear regression model with three coefficients and with various independence structures imposed with S-SGLD: (a) joint dependence between $w_1$, $w_2$, and $w_3$; (b-left) dependence between $w_1$ and $w_2$ but independence between $w_3$ and the other coefficients; (b-right) fully factorized. Of note is that the bivariate posterior distributions appear to respect the imposed independence structure. Interestingly, it also appears that the variance shrinks as we induce these factorizations which is a commonly seen artifact when using VI.

### 8 Structured Dropout SGMCMC

While S-SGMCMC can successfully break dependencies between parameter groups, it does suffer computationally due to each parameter update scaling linearly with respect to $M$. This means that for
We have also conducted experiments on uncertainty visualization, where we tested the proposed method that both closely resembles the S-SGMC procedure and scales independently from the partitioning scheme. This section presents such a method that achieves this, which we call *structured dropout* SGMC (Sd-SGMC), as well as an informal motivation and derivation of the method. More formal details and a theorem proving both SGMC and S-SGMC are limiting cases for Sd-SGMC can be found in the Supplement.

The main motivation for this technique can be seen by recognizing that the composition \{θ(t), q(t)\} from Eq. (14) can be rewrittten as a sum of masked values \( rθ(t) + (1 − r)\tilde{θ}(t) \) where \( \tilde{θ}(t) \sim q(t) \) and \( r_j = 1 (i = j) \) for \( i = 1, \ldots, M \). We can decouple the computational scaling from the number of parameter groups \( M \) by replacing the \( M \) deterministic masks \( r \)'s with \( K \) stochastically sampled masks \( \tilde{r} \).

Doing so results in a slightly different energy function and minibatch loss to optimize:

\[
\hat{U}(S_{d})(θ(t); \tilde{D}) \approx \frac{M}{K \mathbb{E}_{\sum_{i=1}^{M} \rho_i}} \sum_{k=1}^{K} \hat{U}(\tilde{r}_{(t,k)}θ(t) + (1 − \tilde{r}_{(t,k)})\tilde{θ}(t); \tilde{D})
\]  

(16)

where \( \tilde{r}_{(t,k)} \) is the \( k \)th sample of \( \tilde{r} \) for timestep \( t \). A formal justification for Eq. (16) can be found in the Supplement. These energy function approximations lead to the following update step for structured dropout variant of SGMD (Sd-SGLD):

\[
θ(t+1) = \tilde{θ}(t) − \frac{\epsilon_t}{2} \nabla_{θ} \hat{U}(S_{d})(θ; \tilde{D}) + \xi_t \text{ where } ξ_t \sim N(0, \epsilon_t I).
\]  

(17)

The corresponding update rules for the structured dropout variants for pSGMD (Sd-pSGMD) and SGHC (Sd-SGHC) are defined in the Supplement. The exact procedure for generating samples of the approximate posterior \( q(t) \) using structured dropout SGMC (Sd-SGMC) can also be found in the Supplement.

An example of this method (specifically Sd-SGLD with \( \tilde{r}_i \sim \text{Bernoulli}(0.5) \) and \( K = 4 \)) used on a linear regression model can be seen in Fig. 2(c). Of note, we can see that the dropout variant largely respects the independence structure imposed, but maybe not as strictly as the exact S-SGMD method seen in Fig. 2(b). Additionally, the posterior variance also seems to have shrunk similarly to S-SGLD when compared against SGLD.

**Masking Distribution** Should \( \tilde{r}_i \sim \text{Bernoulli}(\rho) \), alongside a structure that factorizes by activation components, then the method starts to resemble dropout with rate \( \rho \) (Srivastava et al., 2014). The main difference being that instead of replacing a parameter value with \( 0 \) it is replaced with a sample from the approximate posterior distribution at time \( t \): \( q(t) \). While a Bernoulli distribution for \( \tilde{r} \) is a natural choice, there are other distributions that can be chosen as well. For instance, \( \tilde{r}_i \sim N(0, 1) \) or \( \tilde{r}_i \sim \text{Beta}(α, β) \) are both viable distributions and can be seen as analog to Gaussian and Beta-dropout respectively (Srivastava et al., 2014; Liu et al., 2019). Our experiments will largely focus on sampling \( \tilde{r} \) from Bernoulli and uniform over \([0, 1]\) (equivalent to Beta(0.5, 0.5)) distributions.

### 9 Experiments

**Overview** In this section we evaluate our proposed approach on various models and datasets. Section 9.1 investigates the impact of the variational approximation on the algorithms’ mixing and autocorrelation times using a fully-connected network architecture on MNIST (LeCun et al., 2010). Section 9.2 studies our methods with ResNet-20 (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and Fashion MNIST (Xiao et al., 2017) and compares them for their accuracy and mixing time. Our experiments reveal that the chains in our proposed methods mix faster than SGMC and achieve either comparable or even higher accuracies on average.

We have also conducted experiments on uncertainty visualization, where we tested the proposed methodology on predictive uncertainty estimation by deploying a two-layer fully connected network.

---

[^1]: \( K \) is a hyperparameter that is chosen independent of \( M \); however, both \( M \) and the distribution of \( \tilde{r} \) largely influence how small \( K \) can be due to how they affect the variance of the gradient of the associated posterior energy function.
on a toy dataset. The uncertainty experimental setup and results, along with more technical details for the other experiments, can be found in the Appendix.

**Metrics** The primary predictive metric of interest use to evaluate our proposal is classification accuracy. We take the average of an ensemble of 100 models whose weights are sampled from the past samples of the parameters chains in order to calculate the accuracy. Additionally, we also monitor the mixing time of the chains of our methods with both integrated autocorrelation time (IAC) (Sokal, 1997; Goodman and Weare, 2010) and effective sample size (ESS) (Geyer, 1992). IAC measures the correlation between samples in a chain and, in turn, describe the inefficiency of a MCMC algorithm. IAC is computed as \( \tau_f = \sum_{n=-\infty}^{\infty} \rho_f(n) \) where \( \rho_f \) is the normalized autocorrelation function of the stochastic process that generated the chain for \( f \) and is calculated as \( \hat{\rho}_f(n) = \frac{\hat{c}_f(0)}{\hat{c}_f(n)} \); where \( \hat{c}_f(n) \) is the autocorrelation. ESS measures how many independent samples would be equivalent to a chain of correlated samples and is calculated as \( n_{\text{eff}} = \frac{n}{1 + \sum_{n=1}^{N} c_f(n)} \) where \( n \) is the number of samples and \( p \) is the autocorrelation. We note that a model with higher ESS and lower IAC have faster mixing time. Please see the Appendix for the detailed implementation details and experimental setup for the metrics and our models.

### 9.1 Dropout Rate & Group Size Investigation

The aim of this set of experiments is to study the effects that the number of independent parameter groups (or alternatively, the amount of allowed posterior correlations) has on accuracy and mixing time when using our proposed methods. We compare pSGLD, S-pSGLD, and S\(_d\)-pSGLD a Bernoulli(\( \rho \)) masking distribution with dropout rates \( \rho \in \{0.1, 0.3, 0.5\} \) on a fully-connected neural network with 2 hidden layers, with 50 hidden units each, trained and evaluated with MNIST using the standard train and test split. The model has 42,200 parameters in total. For S-pSGLD and S\(_d\)-pSGLD, these parameters are evenly distributed into \( M \) groups where \( M \) ranges from 4 to 42,200. Accuracy, IAC, and ESS are reported in Fig. 3 using 100,000 posterior samples after a 150,000 burn in period. More details on the implementation of the model regarding training and evaluation can be found in the Appendix.

As shown in Fig. 3(a), for S-pSGLD we observe that as we increase the number of groups the accuracy drops dramatically whereas S\(_d\)-pSGLD’s accuracy improves slightly and then remains fairly stable. In the best case, S\(_d\)-pSGLD achieves an accuracy of 96.3% with 32 groups and dropout rate of 0.5 which outperforms pSGLD with accuracy of 94.2%. We speculate that the dropout-like behavior

---

4We used the TensorFlow implementation for ESS which uses the direct sum for the autocorrelation.
is beneficial for regularizing the model (much like normal dropout), hence the improved accuracy across all dropout rates. Similarly, a single sample used for the Monte Carlo estimate in S-SGMCMC may not be enough as the number of groups $M$ increase; however, increasing the number of samples in this scenario is infeasible due to S-SGMCMC scaling as $O(M)$.

Fig. 3(b-c) portrays the comparison between number of groups and mixing time metrics IAC and ESS. As the number of groups gradually increase, we note that S-pSGLD mixes faster, as does $S_d$-pSGLD to lesser and lesser degrees as $\rho$ increases. This behavior is to be expected due to Theorem 3, with $S_d$-pSGLD exhibiting mixing times more similar to pSGLD when $\rho = 0.5$ and more similar to S-pSGLD when $\rho = 0.1$.

9.2 Systematic Comparison on Real-World Data

The goal of these experiments is to test the proposed methodology on larger-scale datasets which mimic real-world data: CIFAR-10, SVHN, and FMNIST. We evaluate our methods on performance accuracy and on mixing times of the chains. We employ ResNet-20 for SVHN and FMNIST without any data augmentation to assess our methods. For CIFAR10 we employ the same data augmentation process as proposed in Cubuk et al. (2019). We evaluate the precision of the methods on accuracy over time and the overall mixing time of them on IAC and ESS with 2 base algorithms: pSGLD and SGHMC. For efficiency purposes we limited our scope to models with either fully joint posteriors or fully factorized. As such, for the latter we employed $S_d$-SGMCMC methods as S-SGMCMC would not be feasible with the amount of parameter groups present. Bernoulli($\rho$) and uniform masking distributions were investigated and are denoted as $S_{\text{Bernoulli}}$-SGMCMC and $S_{\text{Uniform}}$-SGMCMC respectively, with $\rho$ varying between datasets as determined by a hyperparameter search (detailed in the Appendix).

In Fig. 4 we observe how quickly the proposed methods and the baseline SGMCMC methods approach their optimum accuracy over the course of training. As is shown, $S_{\text{Bernoulli}}$-SGMCMC and $S_{\text{Uniform}}$-SGMCMC appear to achieve optimal accuracy values much faster than SGMCMC on all datasets and with all base sampling schemes. In some cases, the variational methods achieve better accuracy values than the baseline methods, as seen for CIFAR10 in Fig. 4.

Mixing Time Comparisons We further validated our findings from Section 6.1 by evaluating the IAC and ESS on larger datasets using various methods. Both pSGLD and SGHMC were used as base methods in conjunction with both S-SGMCMC and $S_d$-SGMCMC using a Bernoulli masking distribution. IAC and ESS were calculated for these methods using the latest 5,000 samples after sampling for 300 epochs; the results of which can be found in Table 1. For CIFAR-10, we see that $S_d$-SGMCMC with every parameter in a different group mixes the fastest against all other methods. Likewise, for SVHN and FMNIST, $S_d$-pSGLD with every parameter belonging to its own group mixes faster than all other methods. At times it does appear that increasing the number of parameter groups causes slower mixing time for S-SGMCMC. This could potentially be attributed to large variance in the gradients from using only a single sample per Monte Carlo estimate.
Figure 4: Comparison for (a) CIFAR-10, (b) SVHN and (c) FMNIST using ResNet-20 with (i) pSGLD and (ii) SGHMC sampling algorithms and their proposed variational variants for model averaged accuracy. Grid search was used to determine optimal hyperparameters for each method — more details can be found in the Appendix.

Table 2: MSE for various partitioning schemes on the parameters for a 2-layer fully-connected BNN.

<table>
<thead>
<tr>
<th>Partition Scheme</th>
<th>Wine</th>
<th>Housing</th>
<th>Obesity</th>
<th>Bike</th>
<th>Concrete</th>
<th>Airfoil</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random (M = 3)</td>
<td>0.0454</td>
<td>0.0233</td>
<td>0.0232</td>
<td>0.0242</td>
<td>0.0226</td>
<td>0.0454</td>
</tr>
<tr>
<td></td>
<td>±0.001</td>
<td>±0.003</td>
<td>±0.005</td>
<td>±0.001</td>
<td>±0.003</td>
<td>±0.001</td>
</tr>
<tr>
<td>By Layer</td>
<td>0.0494</td>
<td>0.0236</td>
<td>0.0274</td>
<td>0.0247</td>
<td>0.0243</td>
<td>0.0494</td>
</tr>
<tr>
<td></td>
<td>±0.001</td>
<td>±0.003</td>
<td>±0.002</td>
<td>±0.001</td>
<td>±0.003</td>
<td>±0.001</td>
</tr>
<tr>
<td>By Neurons</td>
<td>0.0496</td>
<td>0.0233</td>
<td>0.0262</td>
<td>0.0247</td>
<td>0.0238</td>
<td>0.0496</td>
</tr>
<tr>
<td></td>
<td>±0.001</td>
<td>±0.003</td>
<td>±0.003</td>
<td>±0.001</td>
<td>±0.003</td>
<td>±0.001</td>
</tr>
<tr>
<td>Fully-Factorized</td>
<td>0.0478</td>
<td>0.0236</td>
<td>0.0227</td>
<td>0.025</td>
<td>0.0238</td>
<td>0.0478</td>
</tr>
<tr>
<td></td>
<td>±0.002</td>
<td>±0.002</td>
<td>±0.002</td>
<td>±0.001</td>
<td>±0.001</td>
<td>±0.002</td>
</tr>
</tbody>
</table>

9.3 Exploring Partitioning Schemes

This part of the study aims to explore the capabilities of the proposed methodology further. Here we explore different parameter partitioning schemes on regression datasets.

Here we present the results with different partitions on various regression datasets. We used 7 different datasets: the wine quality dataset (Cortez et al., 2009), the Boston housing dataset (Harrison Jr and Rubinfeld, 1978), the obesity levels dataset (Palechor and de la Hoz Manotas, 2019), the Seoul bike-sharing dataset (E et al., 2020; E and Cho, 2020), the concrete compressive strength dataset (Yeh, 1998), and the airfoil self-noise dataset (Brooks et al., 1989). For the evaluation we chose a simple fully connected network with two layers with 50 neurons each, and we use SGLD as an optimizer. As a performance metric we chose mean squared error (MSE). We did hyperparameter tuning with different learning rates and the final results are the means with the standard deviations of 5 runs with different seeds. We do not observe any specific systematic trends on the partitions, apart from the fact that in some cases random performs better. In that way the use of either random partitioning or the fully-factorized partitioning, where every parameter is in a different group appears to be a valid choice \textit{a priori}; especially the latter since we have noted earlier the faster mixing times associated with this partitioning scheme. More details about the partitioning schemes experiments can be found in the Appendix.

10 Conclusions

In an attempt to hybridize MCMC and VI, we proposed S-SGMCMC: an approach that produces samples from an structured posterior by running SGMCMC on a modified energy function. The resulting Markov chain becomes asymptotically decoupled across user-specified groups of parameters, resulting in faster convergence. For better computational efficiency, we proposed $S_{d}$-SGMCMC:
We then rewrite Eq. 12 as follows:

\[ p \text{ we are interested in the corresponding posterior distribution} \]

We also write the equation for KL divergence from the main text:

\[ U = \sum_{i \neq j} q_i(\theta_i) \log q_i(\theta_i) - \int \log \frac{p(\theta, D) q_i(\theta_i) d\theta_i}{p(\theta|D)} q_j(\theta_j) d\theta_j + C \]

for some \( i \in \{1, \ldots, M\} \) where \( \theta_i := \{1, \ldots, M\} \setminus \{i\} \) and \( C = \log p(D) \). In order to find the optimal distribution that respects the factorization constraints imposed between parameter groups, we need to minimize this functional over \( q_i \) — or rather every \( q_i \). This is done by taking the functional derivative of \( J \) with respect to \( q_i \), setting it equal to zero, and solving for \( q_i \):

\[ \frac{\delta J(q(\theta))}{\delta q_i(\theta_i)} = \int \log p(\theta, D) \prod_{i \neq j} q_j(\theta_j) d\theta_j - 1 - \log q_i(\theta_i) \]

\[ \Rightarrow \log q_i(\theta_i) = E_{\tilde{\theta}_i^{-i} \sim q_{-i}} \left[ \log p(\theta_i, \tilde{\theta}_i^{-i}, D) \right] - 1 \]

\[ \Rightarrow q_i(\theta_i) \propto \exp \left\{ E_{\tilde{\theta}_i^{-i} \sim q_{-i}} \left[ \log p(\theta_i, \tilde{\theta}_i^{-i}, D) \right] \right\} \]

By defining the energy \( U_i^S(\theta_i) = -E_{\tilde{\theta}_i^{-i} \sim q_{-i}} \left[ \log p(\theta_i, \tilde{\theta}_i^{-i}, D) \right] \), we realize that by minimizing the KL-divergence in Eq. 12, the approximate posterior distribution \( q = \prod_{i=1}^M q_i \) takes the form of a Boltzmann distribution as in Eq. 9 with \( U_i^S(\theta_i) = \sum_{i=1}^M U_i^S(\theta_i) \).

It remains to be shown that the solution is unique. To this end, we refer to the convexity of the KL divergence in function space (Cover and Thomas, 2001). This implies that the stationary point of the KL is indeed a global optimum and unique.
B Deriving \( U^{(S_d)} \)

With just a slight shift in perspective, it is actually possible to further generalize \( U^{(S)} \) (and consequently S-SGMCMC) to produce a broader class of approximate sampling algorithms. This is done by first noting that \( U^{(S)} \) can be represented with a scaled double-expectation:

\[
U^{(S)}(\theta) = -\frac{M}{\mathbb{E}_{r \sim p(\rho)} \left[ \sum_{i=1}^{M} r_i \right]} \mathbb{E}_{r \sim p(\rho)} \mathbb{E}_{\tilde{\theta} \sim q} \left[ \log p(r\theta + (1 - r)\tilde{\theta}, \mathcal{D}) \right]
\]  
\[26\]

where \( p^{(S)}(r) = \text{Cat}(r; M^{-1}, \ldots, M^{-1}) \) and \( (r\theta + (1 - r)\tilde{\theta})_i \) is equal to \( \theta_i \) if \( r_i = 1 \) and \( \tilde{\theta}_i \) otherwise for \( i = 1, \ldots, M \). Note that this is constructed in this manner specifically so that \( U^{(S)} \) remains differentiable with respect to \( \theta \). Also note that though the denominator appears superfluous as \( \mathbb{E}_{r \sim p(\rho)} \left[ \sum_{i=1}^{M} r_i \right] = 1 \), it is necessary for certain theoretic properties, as seen in Theorem 3.

By replacing \( p^{(S)} \) with a more flexible distribution, we can further generalize and encapsulate different energy functions to sample from. One such choice is \( p^{(S_d)}(r; \rho) : \propto \prod_{i=1}^{M} \text{Bern}(r_i; \rho) \mathbb{I}(\sum_{i=1}^{M} r_i > 0) \) with \( \rho \in (0, 1) \). Substituting \( p^{(S)} \) for \( p^{(S_d)} \) in Eq. (26) yields a new energy function that we will refer to as \( U^{(S_d)} \). We note that this choice in distribution leads to a dropout-like behavior (Nalisnick et al., 2019; Srivastava et al., 2014), where the composition of model parameters as \( r\theta + (1 - r)\tilde{\theta} \) leads to each parameter group \( \theta_i \) having a probability of approximately \( \rho \) to be used in a prediction and a \( (1 - \rho) \) probability of being replaced by \( \tilde{\theta}_i \) from the approximate posterior (in traditional dropout, \( \theta_i \) would instead be replaced with 0). Likewise, we will denote methods that use this energy function for sampling as structured dropout SGCMC (\( S_d \)-SGCMC) with different variants all sharing the same \( S_d \) prefix (e.g. \( S_d \)-SGMCMC).

In practice, the double-expectation in \( U^{(S_d)} \) is jointly approximated using a Monte Carlo estimate with \( K \) samples. This leads to Eq. (16) in the main paper. We note that by approximating \( U^{(S_d)} \) in this way, computing a gradient no longer scales on the order of \( O(M) \), but rather \( O(K) \). This means that the choice of structure imposed on the posterior distribution remains independent of computing resources. As such, configurations with large amounts of parameter groups are typically only feasible when using \( S_d \)-SGMCMC as S-SGMCMC would use too much memory and/or compute per sample.

C Theorem 2

Theorem 3. For a given set of parameters \( \theta \) partitioned into \( M \) groups, under minor assumptions (i) \( U^{(S_d)} \to U \) as \( \rho \to 1 \) and (ii) \( U^{(S_d)} \to U^{(S)} \) as \( \rho \to 0 \). Thus, distributions approximated by \( S_d \)-SGCMC lie on a continuum with those generated by S-SGMCMC at one extreme and with those from SGMCMC at the other.

Proof. Assume an arbitrary \( \theta, \mathcal{D}, n \in \mathbb{N} \), and that \( \mathbb{E}_{\tilde{\theta} \sim q} \left[ \log p(r\theta + (1 - r)\tilde{\theta}, \mathcal{D}) \right] \) exists for \( r \in \mathcal{R} \).

As an aside, this proof assumes that \( p^{(S_d)}(r; \rho) : \propto \prod_{i=1}^{M} \text{Bern}(r_i; \rho) \mathbb{I}(\sum_{i=1}^{M} r_i > 0) \) with \( \rho \in (0, 1) \); however, the theorem still holds an arbitrary \( p^{(S_d)} \) so long as the mean approaches 1 and variance approaches 0 as \( n \to \infty \).

(i) Let \( r^{(n)} \sim p^{(S_d)}(\rho_n) \) where \( \forall_n \rho_n \in (0, 1) \) and \( \rho_n \to 1 \). It follows that \( r^{(n)} \to \{1 \}^M \) as \( n \to \infty \) in distribution (see Lemma 1 in Supplement). Due to bounded and finite support \( \mathcal{R} \), we find the following:

\[
U^{(S_d)}(\theta) = -\frac{M}{\mathbb{E}_{r \sim p^{(S_d)}} \left[ \sum_{i=1}^{M} r_i \right]} \mathbb{E}_{r \sim p^{(S_d)}} \mathbb{E}_{\tilde{\theta} \sim q} \left[ \log p(r\theta + (1 - r)\tilde{\theta}, \mathcal{D}) \right]
\]

\[27\]

\[
\to -\frac{M}{M} \sum_{r \in \mathcal{R}} \mathbb{I}(\forall_i r_i = 1) \mathbb{E}_{\tilde{\theta} \sim q} \left[ \log p(\theta, \mathcal{D}) \right] \text{ as } n \to \infty
\]

\[28\]

\[
= -\log p(\theta, \mathcal{D}) = U(\theta)
\]

\[29\]

\(^5\)Other choices of distribution that are well justified include any with support over \([0, 1]^M\) and with measure 0 over \(\{0\}^M\). Exploring the effects these distributions have are an interesting line of future inquiry.
(ii) Let \( r^{(n)} \sim p^{(S_d)}(\rho_n) \) where \( \forall_n \rho_n \in (0, 1) \) and \( \rho_n \to 0 \). It follows that \( r^{(n)} \to r \sim \text{Cat}(M^{-1}, \ldots, M^{-1}) \) as \( n \to \infty \) in distribution (see Lemma 2 in Supplement). Due to bounded and finite support \( \mathcal{R} \), we find the following:

\[
U^{(S_d)}(\theta) = -\frac{M}{\mathbb{E}_{r \sim p^{(S_d)}} \left[ \sum_{i=1}^{M} r_i \right]} \sum_{r \in \mathcal{R}} p^{(S_d)}(r; \rho_n) \mathbb{E}_{\tilde{\theta} \sim q} \left[ \log p(r \theta + (1 - r) \tilde{\theta}, D) \right]
\]

\[
\to -\frac{M}{1} \sum_{r \in \mathcal{R}} \frac{1}{M} \mathbb{E}_{\tilde{\theta} \sim q} \left[ \log p(r \theta + (1 - r) \tilde{\theta}, D) \right] \text{ as } n \to \infty
\]

\[
= -\sum_{i=1}^{M} \mathbb{E}_{\tilde{\theta} \sim q} \left[ \log p(\theta_i, \tilde{\theta}_{-i}, D) \right] = U^{(S)}(\theta)
\]

\( \blacksquare \)

For both Lemmas 1 and 2, let

\[
p^{(S_d)}(r; \rho) = \frac{\rho \sum_{i=1}^{M} r_i (1 - \rho)^{M - \sum_{i=1}^{M} r_i}}{1 - (1 - \rho)^M} \mathbb{1}(\forall_i r_i \in \{0, 1\}) \mathbb{1} \left( \sum_{i=1}^{M} r_i > 0 \right)
\]

**Lemma 1.** For \( r^{(n)} \sim p^{(S_d)}(\rho_n) \), \( \rho_n \in (0, 1) \) and \( n \in \mathbb{N} \), if \( \rho_n \to 1 \) as \( n \to \infty \) then \( r^{(n)} \to r \sim \delta(\{1\}^M) \) in distribution as \( n \to \infty \).

**Proof.**

\[
p^{(S_d)}(r = \{1\}^M; \rho_n) = \frac{\rho_n^M (1 - \rho_n)^0}{1 - (1 - \rho_n)^M}
\]

\[
\to 1 \text{ as } n \to \infty
\]

\( \Rightarrow r^{(n)} \to \delta(\{1\}^M) \) in distribution.

\( \blacksquare \)

**Lemma 2.** For \( r^{(n)} \sim p^{(S_d)}(\rho_n) \), \( \rho_n \in (0, 1) \) and \( n \in \mathbb{N} \), if \( \rho_n \to 0 \) as \( n \to \infty \) then \( r^{(n)} \to r \sim \text{Cat}(M^{-1}, \ldots, M^{-1}) \) in distribution as \( n \to \infty \).

**Proof.** Let \( i \in \{1, \ldots, M\} \).

\[
p^{(S_d)}(r_i = 1, r_{-i} = 0; \rho_n) = \frac{\rho_n (1 - \rho_n)^{M-1}}{1 - (1 - \rho_n)^M}
\]

By l'Hôpital's Rule

\[
\frac{M}{1 - \rho_n} \frac{1 - (1 - \rho_n)^M}{M(1 - r_n)^{M-2}}
\]

\( \to \frac{1}{M} \) as \( n \to \infty \)

Since the resulting probabilities sum to 1, this implies that \( r^{(n)} \to r \sim \text{Cat}(M^{-1}, \ldots, M^{-1}) \) in distribution as \( n \to \infty \). \( \blacksquare \)

### D Deriving \( U^{(S_d)} \)

To derive \( U^{(S_d)} \), we must first start with a shift in perspective on how \( U^{(S)} \) is represented. We will rewrite the function in the following way:

\[
U^{(S)}(\theta) = -\sum_{i=1}^{M} \mathbb{E}_{\tilde{\theta} \sim q} \left[ \mathbb{E}_{r \sim p^{(S)}} \left[ \log p([\theta_i, \tilde{\theta}_{-i}], D) \right] \right]
\]

\[
= -\frac{M}{\mathbb{E}_{r \sim p^{(S)}} \left[ \sum_{i=1}^{M} r_i \right]} \mathbb{E}_{r \sim p^{(S)}} \mathbb{E}_{\tilde{\theta} \sim q} \left[ \log p(r \theta + (1 - r) \tilde{\theta}, D) \right]
\]
where \( p^{(S)} \) is a \( M \)-dimensional categorical distribution with uniform weights \( M^{-1} \) and \( p(r\theta + (1 - r)\tilde{\theta}, \mathcal{D}) \) is the joint probability of parameters taking values of \( r\theta + (1 - r)\tilde{\theta} \) and data \( \mathcal{D} \).

We note that changing the distribution of \( r \) leads to different energy functions to sample from. One such choice is to have \( p^{(S_d)}(r; \rho) \propto \rho \sum_{i=1}^M r_i (1 - \rho)^M - \sum_{i=1}^M r_i \mathbb{1}(\forall_i r_i \in \{0, 1\}) \mathbb{1}(\sum_{i=1}^M r_i > 0) \) for \( \rho \in (0, 1) \). Note that this is identical to \( r_i \overset{iid}{\sim} \text{Bernoulli}(\rho) \) conditional to \( \sum_{i=1}^M r_i > 0 \). Let the support of \( p^{(S_d)} \) be denoted as \( \mathcal{R} = \{0, 1\}^M \setminus \{0\}^M \). This leads to the following energy function:

\[
U^{(S_d)}(\theta) = -\frac{M}{\mathbb{E}_{r \sim p^{(S_d)}}[\sum_{i=1}^M r_i]} \mathbb{E}_{r \sim p^{(S_d)}} \mathbb{E}_{\tilde{\theta} \sim q} \left[ \log p(r\theta + (1 - r)\tilde{\theta}, \mathcal{D}) \right].
\] (42)

In practice, a few approximations are made to compute the corresponding \( U^{(S_d)} \). Firstly, we approximate \( p^{(S_d)} \) with an \( M \)-dimensional Bernoulli(\( \rho \)) distribution as the difference is minute when \( M \rho \) is large. Secondly, the outer expectation in Eq. (42) is approximated with a Monte Carlo estimate of \( K \) samples. The inner expectation is also approximated with a Monte Carlo estimate using the latest approximate posterior \( \hat{q}^{(t)} \). However, just like for S-SGMCMC, only a single sample is used. This further leads to:

\[
U^{(S_d)}(\theta^{(t)}; \mathcal{D}) = -\frac{1}{K \rho} \sum_{k=1}^K U(r^{(t,k)}\theta^{(t)} + (1 - r^{(t,k)})\tilde{\theta}^{(t,k)}, \mathcal{D})
\] (43)

E Algorithm for S_{d}-SGMCMC

The procedure for S_{d}-SGMCMC can be seen in Algorithm 2.

---

**Algorithm 2:** S_{d}-SGMCMC

**Input:** Initial sample \( \theta^{(0)} \); parameter partitions \( \theta_1, \ldots, \theta_M \); data set \( \mathcal{D} \); initial auxiliary statistics \( \xi^{(0)} \); step sizes \{\epsilon_t\}_{t=1,\ldots,T}; masking distribution \( p^{(S_d)} \); dropout iterations \( K \).

**Output:** \( \hat{q}^{(T)}(\theta) := \{\hat{q}(t)\}_{t=1,\ldots,T} \).

1. for \( t = 0 \) to \( T - 1 \) do
2. Sample minibatch \( \mathcal{D}^{(t)} \subset \mathcal{D} \)
3. for \( k = 1 \) to \( K \) do
4. Sample masks \( r^{(t,k)}_1, \ldots, r^{(t,k)}_M \sim p^{(S_d)} \)
5. Sample \( \tilde{\theta}^{(t,k)} \sim q^{(t)} \)
6. \( \theta^{(t,k)} = [r^{(t,k)}_i \theta_i^{(t)} + (1 - r^{(t,k)}_i)\tilde{\theta}^{(t,k)}_i]_{i=1,\ldots,M} \)
7. \( \hat{U}^{(S_d,t)} = \mathbb{E}_\theta[U(\theta^{(t,k)}; \mathcal{D}^{(t)})] \)
8. end
9. \( \nabla_\theta \hat{U}^{(S_d,t)} = \frac{1}{K \rho} \sum_{i=1}^M \sum_{k=1}^K \nabla_\theta \hat{U}^{(S_d,t)}_k \)
10. \( \theta^{(t+1)}, \xi^{(t+1)} = \text{SGMCMC\_step}(\theta^{(t)}, \nabla_\theta \hat{U}^{(S_d,t)}, \xi^{(t)}, \epsilon_t) \)
11. end
12. return \( \hat{q}^{(T)}(\theta) \)

---

*\( r\theta + (1 - r)\tilde{\theta} \) is a slight abuse of notation that is meant to represent masking out \( \theta_i \) when \( r_i = 0 \) and masking out \( \tilde{\theta}_i \) when \( r_i = 1 \).*
F SGMCMC Update Rules

The update rules for SG LD, pSG LD, and SGHMC are defined as follows:

\[
\begin{align*}
\text{SGLD} & \quad \theta^{(t+1)} = \theta^{(t)} - \frac{\epsilon_t}{2} \nabla_\theta \hat{U}(\theta^{(t)}) + \mathcal{N}(0, \epsilon_t I) \\
\text{pSGLD} & \quad \theta^{(t+1)} = \theta^{(t)} - \frac{\epsilon_t}{2} \left[ R(\theta^{(t)}) \nabla_\theta \hat{U}(\theta^{(t)}) + \sum_\theta \nabla_\theta R(\theta^{(t)}) \right] + \mathcal{N}(0, \epsilon_t R(\theta^{(t)})) \\
\text{SGHMC} & \quad \theta^{(t+1)} = \theta^{(t)} + \epsilon_t M^{-1} m^{(t+1)} \\
& \quad m^{(t+1)} = (1 - \gamma \epsilon_t M^{-1}) m^{(t)} - \epsilon_t \nabla_\theta \hat{U}(\theta^{(t)}) + \mathcal{N}(0, 2\gamma - \epsilon_t \hat{V}(\theta^{(t)}))
\end{align*}
\]

where \( \epsilon_t \) is the step size at time step \( t \), \( R(\cdot) \) and \( M \) are preconditioners, \( \gamma \geq 0 \) is a friction term, and \( \hat{V}(\cdot) \) is an estimate of the covariance induced by the stochastic gradient.\(^7\)

The update rules for the S-SGMCMC variants are similarly defined as Eqs. 44-47 but all instances of \( \hat{U}(\theta^{(t)}) \) are replaced with \( \hat{U}^{(S)}(\theta^{(t)}) \). Likewise, replacing with \( \hat{U}^{(S_d)}(\theta^{(t)}) \) yields the \( S_d \)-SGMCMC variants.

G Ablation Study

This subsection aims to further explore the capabilities of the proposed methodology. More specifically, we visualize uncertainty for a two-layer Fully Connected Network and experiment with various parameter partitions.

Uncertainty. We tested the proposed methodology on uncertainty estimation by deploying a two-layer Fully Connected Network on a toy dataset. The Neural Network has 50 neurons in each layer and was trained with \( S_d \)-pSGLD with different parameter partitions, while the artificial dataset consists of 500 data points. In Fig. 5, we compare uncertainty estimates for the pSGLD and \( S_d \)-pSGLD while progressively breaking dependencies between parameters. We observe that as we break dependencies we capture similar uncertainty intervals.

Parameter Partitions. We tested our proposal with four partitioning schemes on a 2 layer with 50 neurons fully connected network on a regression task. The partitioning schemes that we used are the following: (a) the parameters are split into 3 groups randomly, (b) the parameters are split by layer(3 layers, 1 input and 2 hidden), (c) by activating neurons inside the layers and (d) every parameter belongs in each own group. We used 7 different datasets: the wine quality dataset\( \text{Cortez et al. (2009)} \), the concrete compressive strength dataset\( \text{Yeh (1998)} \), the obesity levels dataset\( \text{Palechor and de la Hoz Manotas (2019)} \), the Seoul bike-sharing dataset\( \text{E et al. (2020); E and Cho (2020)} \), the Boston housing dataset\( \text{Harrison Jr and Rubinfeld (1978)} \), the airfoil self-noise dataset\( \text{Brooks et al. (1989)} \). Every dataset was split into 75% training data, 10% validation data, and 15% test data. We trained the model on training set and validated it in the validation set with an early stoppage. For

\(^7\)Note that we abuse notation in Eqs. 44-47 where the addition of \( \mathcal{N}(\mu, \Sigma) \) denotes the addition of a normally distributed random variable with mean \( \mu \) and covariance \( \Sigma \).
every dataset and every partitioning scheme we used the learning rates: 1e-3, 1e-4, 1e-5, 1e-6, 1e-7 for hyperparameter tuning. For each combination of partition and dataset, we chose the learning rate that provides the best accuracy score on the test set. In this case, as an accuracy score, we used the Mean Squared Error. The final learning rates that we used are presented in Table 3.

<table>
<thead>
<tr>
<th>Partition Scheme</th>
<th>Wine</th>
<th>Housing</th>
<th>Obesity</th>
<th>Bike</th>
<th>Concrete</th>
<th>Airfoil</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random (M = 3)</td>
<td>1e-3</td>
<td>1e-5</td>
<td>1e-5</td>
<td>1e-4</td>
<td>1e-5</td>
<td>1e-4</td>
</tr>
<tr>
<td>By Layer</td>
<td>1e-3</td>
<td>1e-5</td>
<td>1e-4</td>
<td>1e-4</td>
<td>1e-5</td>
<td>1e-4</td>
</tr>
<tr>
<td>By Neurons</td>
<td>1e-3</td>
<td>1e-5</td>
<td>1e-4</td>
<td>1e-4</td>
<td>1e-5</td>
<td>1e-5</td>
</tr>
<tr>
<td>Fully-Factorized</td>
<td>1e-5</td>
<td>1e-4</td>
<td>1e-5</td>
<td>1e-4</td>
<td>1e-4</td>
<td>1e-3</td>
</tr>
</tbody>
</table>

Table 3: Best Learning Rates for various partitioning schemes on multiple regression datasets.

H Details on Experiments

H.1 Qualitative Regression Experiments

First, we aim to showcase qualitative differences in the empirical posterior distributions generated by a baseline SGMC algorithm and our proposed variants. To do so, we consider a regression task where 100 randomly sampled three-dimensional covariates \( \{ \vec{x}_i = [x_{i,1}, x_{i,2}, x_{i,3}]^T \}_{i=1,...,100} \) are used to sample response values \( y_i \sim N(\vec{w}^T \vec{x}_i + b, \sigma^2) \) where \( \vec{w} = [w_1, w_2, w_3]^T = [1.5, -0.8, 1.3]^T \), \( b = 0.5 \), and \( \sigma^2 = 1 \). More details on the generation process for \( \vec{x} \) can be found in the Supplement.

We choose to fit a linear regression model of the same form as the generation process. \( \sigma^2 \) is assumed to be known. Thus, \( \theta = [w_1, w_2, w_3, b] \). A standard normal distribution is used as the prior for each parameter. Due to conjugacy, the posterior distribution can be calculated analytically. As such, the MAP is roughly \( \hat{\theta}_{\text{MAP}} \approx [0.52, 0.31, 0.47, 0.84] \).

The approximated posterior distributions for \( \theta \) are found using SGLD, S-SGLD, and Sₚ-SGLD. For the latter two sampling schemes, two parameter partitions are tested: (i) two groups of parameters where \( \theta_1 = [w_1, w_2] \) and \( \theta_2 = [w_3, b] \) and (ii) four groups of parameters where \( \theta_1 = w_1, \theta_2 = w_2, \theta_3 = w_3, \) and \( \theta_4 = b \). For Sₚ-SGLD, \( \rho = 0.5 \) and \( K = 4 \) was used.

The resulting posterior distributions for \( (w_1, w_2) \) and \( (w_1, w_3) \) from all five scenarios, with SGLD in the leftmost column as our baseline, can be seen in Fig. 2. We observe that, as expected, correlations between \( (w_1, w_2) \) still exist when they are allocated to the same parameter group and become apparently independent when assigned to different groups. We also note that the variance of the distributions shrink as the parameter space is partitioned into smaller groups. The underestimation of posterior variance is a commonly reported finding for VI techniques and is interesting to note that our non-parametric methods appear to exhibit this behavior as well. Finally, it appears that the Sₚ-SGLD adequately approximates S-SGLD with just slightly higher variances and very minor correlations between parameter groups being exhibited.

H.2 Real-World Data Experiments

Framework details. In this subsection, we provide more detailed results for our experiments and a grid search for FMNIST, CIFAR10, and SVHN. We note that all the code apart from the metrics was written in PyTorch (Paszke et al., 2019). Regarding the metrics, ESS was adopted from the TensorFlow probability library (Dillon et al., 2017; Abadi et al., 2016) and IAC was calculated in python. For all the experiments, we used a seed of 2. Moreover, we note that we grouped the parameters in an ordered way for Sₚ-pSGLD and S-pSGLD. We denoted previously that \( K_\rho \) is the number of groups. So every parameter will go to the \( i \mod K_\rho \) group where \( i \) is the parameter index. If, for instance, \( K_\rho = 8 \) then parameter 1 will go to group 1, parameter 2 will go to group 2, parameter 9 will go to group 1, etc. If \( K_\rho = 3 \) is the same as the number of parameters, every parameter will go into its own group.

MNIST. Regarding MNIST, we ran all the experiments for 500 epochs with a batch size of 500 and a learning rate of 1e-2. For Sₚ-pSGLD, the \( K \) is set to 300, which is the forward passes that the
model does within 1 epoch. For the grouping of the parameters, for \( S_d \)-pSGLD we used group sizes of 2, 4, 8, 32, 128, 2048, 4096, 8192, 16384, 32768 and 42200; and for S-pSGLD we used groups sizes of 2, 4, 8, 32, 128, 2048, 4096, 8192.

**FashionMNIST.** We ran all experiments for 300 epochs with a batch size of 500. For \( S_d \)-SGHMC the \( K \) is set to 2, which is the forward passes that the model does within 1 epoch. We observed with experimenting with \( K \) that we do not need to set \( K \) very high, and even a small number like 2 that we used here is enough to produce the same results as with an \( K \) of 200 or 300. In this way, we save significant time in training. Regarding the parameter partitioning, for \( S_d \)-SGMCMC, we put every parameter in a different group, and for S-SGMCMC we used groups of 2, 4, and 16. For \( S_d \)-pSGLD, pSGLD, \( S_d \)-SGHMC and SGHMC we tested their performances with learning rates of \( 1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5} \). For S-pSGLD we used a learning rate of \( 1e^{-3} \) and for S-SGHMC a learning rate of \( 1e^{-2} \).

<table>
<thead>
<tr>
<th>Method</th>
<th>dropout</th>
<th>LR</th>
<th>IAC</th>
<th>ESS</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_d )-pSGLD</td>
<td>0.1</td>
<td>1e-05</td>
<td>1018</td>
<td>5.63</td>
<td>0.918</td>
</tr>
<tr>
<td>( S_d )-pSGLD</td>
<td>0.1</td>
<td>1e-04</td>
<td>808</td>
<td>7</td>
<td><strong>0.925</strong></td>
</tr>
<tr>
<td>( S_d )-pSGLD</td>
<td>0.1</td>
<td>1e-03</td>
<td>754</td>
<td>7.48</td>
<td>0.924</td>
</tr>
<tr>
<td>( S_d )-pSGLD</td>
<td>0.1</td>
<td>1e-02</td>
<td>723</td>
<td>8.05</td>
<td>0.911</td>
</tr>
<tr>
<td>( S_d )-pSGLD</td>
<td>0.5</td>
<td>1e-05</td>
<td>778</td>
<td>7.08</td>
<td>0.923</td>
</tr>
<tr>
<td>( S_d )-pSGLD</td>
<td>0.5</td>
<td>1e-04</td>
<td>777</td>
<td>7.15</td>
<td>0.923</td>
</tr>
<tr>
<td>( S_d )-pSGLD</td>
<td>0.5</td>
<td>1e-03</td>
<td>737</td>
<td>7.57</td>
<td>0.924</td>
</tr>
<tr>
<td>( S_d )-pSGLD</td>
<td>0.5</td>
<td>1e-02</td>
<td><strong>677</strong></td>
<td>8.24</td>
<td>0.91</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-05</td>
<td>779</td>
<td>7.09</td>
<td>0.924</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-04</td>
<td>774</td>
<td>7.16</td>
<td>0.911</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-03</td>
<td>770</td>
<td>7.26</td>
<td>0.809</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-02</td>
<td>745</td>
<td>7.48</td>
<td>0.724</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-01</td>
<td>740</td>
<td>7.55</td>
<td>0.918</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-03</td>
<td>751</td>
<td>7.45</td>
<td>0.919</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-02</td>
<td>776</td>
<td>7.24</td>
<td>0.919</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-03</td>
<td>855</td>
<td>6.64</td>
<td>0.916</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>dropout</th>
<th>LR</th>
<th>IAC</th>
<th>ESS</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_d )-SGHMC</td>
<td>0.1</td>
<td>1e-05</td>
<td><strong>782</strong></td>
<td>7.08</td>
<td>0.412</td>
</tr>
<tr>
<td>( S_d )-SGHMC</td>
<td>0.1</td>
<td>1e-04</td>
<td>888</td>
<td>6.41</td>
<td>0.796</td>
</tr>
<tr>
<td>( S_d )-SGHMC</td>
<td>0.1</td>
<td>1e-03</td>
<td>793</td>
<td>6.98</td>
<td>0.92</td>
</tr>
<tr>
<td>( S_d )-SGHMC</td>
<td>0.1</td>
<td>1e-02</td>
<td>1113</td>
<td>5.06</td>
<td>0.922</td>
</tr>
<tr>
<td>( S_d )-SGHMC</td>
<td>0.5</td>
<td>1e-05</td>
<td>790</td>
<td>6.93</td>
<td>0.207</td>
</tr>
<tr>
<td>( S_d )-SGHMC</td>
<td>0.5</td>
<td>1e-04</td>
<td>789</td>
<td>6.9</td>
<td>0.758</td>
</tr>
<tr>
<td>( S_d )-SGHMC</td>
<td>0.5</td>
<td>1e-03</td>
<td>796</td>
<td>6.81</td>
<td>0.92</td>
</tr>
<tr>
<td>( S_d )-SGHMC</td>
<td>0.5</td>
<td>1e-02</td>
<td>923</td>
<td>5.7</td>
<td>0.927</td>
</tr>
<tr>
<td>SGHMC</td>
<td>-</td>
<td>1e-05</td>
<td>791</td>
<td>6.93</td>
<td>0.206</td>
</tr>
<tr>
<td>SGHMC</td>
<td>-</td>
<td>1e-04</td>
<td>789</td>
<td>6.9</td>
<td>0.751</td>
</tr>
<tr>
<td>SGHMC</td>
<td>-</td>
<td>1e-03</td>
<td>795</td>
<td>6.83</td>
<td>0.92</td>
</tr>
<tr>
<td>SGHMC</td>
<td>-</td>
<td>1e-02</td>
<td>920</td>
<td>5.72</td>
<td>0.928</td>
</tr>
<tr>
<td>S-SGHMC</td>
<td>-</td>
<td>1e-02</td>
<td>928</td>
<td>5.67</td>
<td><strong>0.928</strong></td>
</tr>
<tr>
<td>S-SGHMC</td>
<td>-</td>
<td>1e-02</td>
<td>915</td>
<td>5.77</td>
<td>0.927</td>
</tr>
<tr>
<td>S-SGHMC</td>
<td>-</td>
<td>1e-02</td>
<td>1142</td>
<td>4.87</td>
<td>0.919</td>
</tr>
<tr>
<td>S-SGHMC</td>
<td>-</td>
<td>1e-02</td>
<td>1121</td>
<td>4.92</td>
<td>0.906</td>
</tr>
</tbody>
</table>

**CIFAR10.** The setup is similar to the one we used in FashionMNIST as we ran all experiments for 300 epochs with a batch size of 128. For \( S_d \)-SGHMC, the \( K \) is set to 2, which \( K \) is the forward
passes that the model does within 1 epoch. Regarding the parameter partitioning, for $S_{d}$-SGMCMC, we put every parameter in a different group, and for S-SGMCMC we used groups of 2,4,8, and 16. For $S_{d}$-pSGLD, pSGLD, $S_{d}$-SGHMC and SGHMC we tested their performances with learning rates of $1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}$. For S-pSGLD, we used a learning rate of $1e^{-3}$, and for S-SGHMC, a learning rate of $1e^{-2}$. We focused our strategy on evaluating the accuracy of the different combinations of hyperparameters with the proposed methods, as can be seen in Figs. 6 and 7. Quantitative results on IAC, ESS and maximum accuracy are depicted in Tables 6 and 7.

### Table 6: Evaluation Metrics on CIFAR10 with pSGLD, $S_{d}$-pSGLD and S-pSGLD

<table>
<thead>
<tr>
<th>Method</th>
<th>dropout</th>
<th>LR</th>
<th>IAC</th>
<th>ESS</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.1</td>
<td>1e-02</td>
<td>623</td>
<td>7.23</td>
<td>0.191</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.1</td>
<td>1e-03</td>
<td>572</td>
<td>7.6</td>
<td>0.896</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.1</td>
<td>1e-04</td>
<td>692</td>
<td>6.45</td>
<td>0.921</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.1</td>
<td>1e-05</td>
<td>922</td>
<td>4.88</td>
<td>0.922</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.5</td>
<td>1e-02</td>
<td>546</td>
<td>8.01</td>
<td>0.768</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.5</td>
<td>1e-03</td>
<td>582</td>
<td>7.88</td>
<td>0.918</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.5</td>
<td>1e-04</td>
<td>691</td>
<td>6.85</td>
<td>0.926</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.5</td>
<td>1e-05</td>
<td>620</td>
<td>7.22</td>
<td>0.927</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-02</td>
<td>716</td>
<td>8.01</td>
<td>0.666</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-03</td>
<td>740</td>
<td>7.87</td>
<td>0.866</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-04</td>
<td>780</td>
<td>7.41</td>
<td>0.914</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-05</td>
<td>831</td>
<td>6.89</td>
<td>0.926</td>
</tr>
<tr>
<td>S-pSGLD</td>
<td>1e-3</td>
<td>1e-02</td>
<td>600</td>
<td>7.44</td>
<td>0.894</td>
</tr>
<tr>
<td>S-pSGLD</td>
<td>1e-3</td>
<td>1e-03</td>
<td>599</td>
<td>7.4</td>
<td>0.905</td>
</tr>
<tr>
<td>S-pSGLD</td>
<td>1e-3</td>
<td>1e-04</td>
<td>709</td>
<td>6.41</td>
<td>0.881</td>
</tr>
<tr>
<td>S-pSGLD</td>
<td>1e-3</td>
<td>1e-05</td>
<td>767</td>
<td>5.93</td>
<td>0.836</td>
</tr>
</tbody>
</table>

### Table 7: Evaluation Metrics on CIFAR10 with SGHMC, $S_{d}$-SGHMC and S-SGHMC

<table>
<thead>
<tr>
<th>Method</th>
<th>dropout</th>
<th>LR</th>
<th>IAC</th>
<th>ESS</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.1</td>
<td>1e-02</td>
<td>608</td>
<td>7.16</td>
<td>0.91</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.1</td>
<td>1e-03</td>
<td>975</td>
<td>4.6</td>
<td>0.922</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.1</td>
<td>1e-04</td>
<td>654</td>
<td>6.63</td>
<td>0.869</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.1</td>
<td>1e-05</td>
<td>652</td>
<td>6.65</td>
<td>0.724</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.5</td>
<td>1e-02</td>
<td>584</td>
<td>7.7</td>
<td>0.918</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.5</td>
<td>1e-03</td>
<td>751</td>
<td>6.23</td>
<td>0.927</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.5</td>
<td>1e-04</td>
<td>679</td>
<td>6.73</td>
<td>0.886</td>
</tr>
<tr>
<td>$S_{d}$-pSGLD</td>
<td>0.5</td>
<td>1e-05</td>
<td>772</td>
<td>6.01</td>
<td>0.778</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-02</td>
<td>727</td>
<td>7.94</td>
<td>0.86</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-03</td>
<td>832</td>
<td>6.84</td>
<td>0.926</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-04</td>
<td>862</td>
<td>6.57</td>
<td>0.885</td>
</tr>
<tr>
<td>pSGLD</td>
<td>-</td>
<td>1e-05</td>
<td>858</td>
<td>6.6</td>
<td>0.746</td>
</tr>
<tr>
<td>S-pSGLD</td>
<td>1e-3</td>
<td>1e-02</td>
<td>583</td>
<td>7.49</td>
<td>0.913</td>
</tr>
<tr>
<td>S-pSGLD</td>
<td>1e-3</td>
<td>1e-03</td>
<td>624</td>
<td>7.03</td>
<td>0.919</td>
</tr>
<tr>
<td>S-pSGLD</td>
<td>1e-3</td>
<td>1e-04</td>
<td>904</td>
<td>4.97</td>
<td>0.908</td>
</tr>
<tr>
<td>S-pSGLD</td>
<td>1e-3</td>
<td>1e-05</td>
<td>822</td>
<td>5.47</td>
<td>0.774</td>
</tr>
</tbody>
</table>

**SVHN.** We also ran all of the experiments for 300 epochs with a batch size of 128. Here for $S_{d}$-SGHMC, the $K$ is set to 2, which is the forward passes that the model does within 1 epoch. We note that $K$ here is less than on CIFAR10 and FashionMNIST, but as we mentioned before, this does not make a difference for our results, as we have tested. Regarding the parameter partitioning, for $S_{d}$-SGMCMC, we put every parameter in a different group, and for S-SGMCMC we used groups of 2,4,8, and 16. For $S_{d}$-pSGLD, pSGLD, $S_{d}$-SGHMC and SGHMC we tested their performances with learning rates of $1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}, 1e^{-6}$. For S-pSGLD we used a learning rate of $1e^{-4}$, and for S-SGHMC, a learning rate of $1e^{-2}$. Same as in CIFAR10, we conducted a grid search for learning rate, dropout rate, and optimizers to find the best performing models and test them for their accuracy.
Figure 6: Model comparison with different hyperparameters for the pSGLD versions of our baselines and proposed algorithms. We evaluated expected accuracy over iterations on CIFAR-10. Subscript "d" denotes the dropout version of our approach. LR refers to the learning rate, and $\rho$ denotes the dropout rate.

We can observe these results in Figure 3 in the main paper. The strategy that we followed is the same as in CIFAR10 and is presented in Figs. 8 and 9.
Figure 7: Model comparison with different hyperparameters for the SGHMC versions of our baselines and proposed algorithms. We evaluated expected accuracy over iterations on CIFAR-10. Subscript "d" denotes the dropout version of our approach. LR refers to the learning rate, and $\rho$ denotes the dropout rate.
Figure 8: Model comparison with different hyperparameters for the pSGLD versions of our baselines and proposed algorithms. We evaluated expected accuracy over iterations on SVHN. Subscript "d" denotes the dropout version of our approach. LR refers to the learning rate, and $\rho$ denotes the dropout rate.
Figure 9: Model comparison with different hyperparameters for the SGHMC versions of our baselines and proposed algorithms. We evaluated expected accuracy over iterations on SVHN. Subscript "d" denotes the dropout version of our approach. LR refers to the learning rate, and $\rho$ denotes the dropout rate.