Sequential Monte Carlo for Inclusive KL Minimization in Amortized Variational Inference

Declan McNamara
University of Michigan

Jackson Loper
University of Michigan

Jeffrey Regier
University of Michigan

Abstract

For training an encoder network to perform amortized variational inference, the Kullback-Leibler (KL) divergence from the exact posterior to its approximation, known as the inclusive or forward KL, is an increasingly popular choice of variational objective due to the mass-covering property of its minimizer. However, minimizing this objective is challenging. A popular existing approach, Reweighted Wake-Sleep (RWS), suffers from heavily biased gradients and a circular pathology that results in highly concentrated variational distributions. As an alternative, we propose SMC-Wake, a procedure for fitting an amortized variational approximation that uses likelihood-tempered sequential Monte Carlo samplers to estimate the gradient of the inclusive KL divergence. We propose three gradient estimators, all of which are asymptotically unbiased in the number of iterations and two of which are strongly consistent. Our method interleaves stochastic gradient updates, SMC samplers, and iterative improvement to an estimate of the normalizing constant to reduce bias from self-normalization. In experiments with both simulated and real datasets, SMC-Wake fits variational distributions that approximate the posterior more accurately than existing methods.

1 INTRODUCTION

Amortized variational inference (VI) fits an encoder $q_\phi(z \mid x)$ to approximate the posterior for each x in a dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$. The encoder parameters ϕ

are selected to minimize some divergence between the variational posteriors $q_\phi(z \mid x)$ and the exact posteriors $p(z \mid x)$, averaged over observations $x \in \mathcal{D}$. The forward KL divergence (Equation 1), also known as the inclusive KL divergence, is an increasingly popular variational objective because minimizers q_ϕ tend to be mass covering (overdispersed) with respect to the true posterior (Gu, Ghahramani, and Turner, 2015); this property is desirable for applications benefiting from conservative uncertainty quantification. In contrast, reverse KL minimizers tend to be mode-seeking and to underestimate uncertainty (Domke and Sheldon, 2018).

Reweighted Wake-Sleep (RWS) is a popular method for performing amortized variational inference that attempts to minimize the forward KL divergence, but, surprisingly, in practice wake-phase training can result in variational posteriors that are severely underdispersed (Le et al., 2019). We conjecture that this behavior results from a "circular pathology" that arises from fitting q_ϕ with particles proposed from q_ϕ itself (Section 3).

To address these limitations of RWS, we propose SMC-Wake, a method to fit an amortized encoder by minimizing the average forward KL divergence via stochastic gradient descent (SGD) with gradients estimated using sequential Monte Carlo (SMC) samplers (Section 2). We propose three new estimators of the gradient of the forward KL that combine SMC samplers in different ways (Section 4), and we prove that each is asymptotically unbiased and that two are also strongly consistent (Section 4.3). Our method compares favorably to those in related work (Section 5), and we demonstrate these advantages empirically in a variety of experimental settings (Section 6). Compared to other forward KL minimization methods based on importance sampling, SMC-Wake leverages higher-fidelity particle approximations with lower-variance weights, resulting in a stable training regime that avoids weight degeneracy and the circular pathology we identify in RWS.

2 BACKGROUND

2.1 Reweighted Wake-Sleep

Given a dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$, the wake-phase update to the variational parameters in reweighted wake-sleep (RWS) (Bornschein and Bengio, 2015) aims to minimize the average forward KL divergence:

$$\frac{1}{n} \sum_{j=1}^{n} \mathrm{KL}(p(z \mid x_j) \,||\, q_\phi(z \mid x_j)). \quad (1)$$

The gradient of this objective, however, is generally intractable; even a Monte Carlo estimate of it requires sampling from the exact posteriors $p(z \mid x_j)$ for each $x_j \in \mathcal{D}$. Instead, RWS approximates the gradient of each term of Equation 1 using self-normalized importance sampling, with the current iterate of $q_\phi(z \mid x)$ serving as the proposal. With K particles, the resulting gradient estimator is

$$-\sum_{i=1}^{K} w^i \nabla_\phi \log q_\phi(z^i \mid x), \text{ where } w^i = \frac{\tilde{w}^i}{\sum_j \tilde{w}^j}, \quad (2)$$

$\tilde{w}^i = \frac{p(z^i, x)}{q_\phi(z^i \mid x)}$ are unnormalized importance weights, and $z^1, \ldots, z^K \overset{iid}{\sim} q_\phi(z \mid x)$ for a given x. Although biased, this estimator converges almost surely to the gradient of the forward KL divergence as $K \to \infty$, under mild conditions (Owen, 2013). RWS also includes a sleep phase that averages exact gradients of simulated ("dreamt") data in a likelihood-free approach to inference. However, this approach has its own limitations and has been shown to underperform the wake-phase update in some cases (Le et al., 2019).

2.2 Sequential Monte Carlo samplers

Sequential Monte Carlo (SMC) samplers (Del Moral, Doucet, and Jasra, 2006) compute estimates of $\mu = \mathbb{E}_{\gamma(z)} f(z)$ for an integrable test function f and a target distribution γ. SMC samplers generalize self-normalized importance sampling (SNIS) (Owen, 2013): instead of a single target γ, a *sequence* of targets $\gamma_1, \ldots, \gamma_T$ is approximated. Even if only a single target γ is of interest, SMC samplers can improve estimation through annealing. One selects a sequence of distributions such that γ_1 is tractable and $\gamma_T = \gamma$, with intermediate distributions that facilitate moves between these (Chopin and Papaspiliopoulos, 2020).

In a Bayesian setting, where approximations of the posterior are of interest, typically the final target is $\gamma \overset{d}{=} p(z \mid x)$ for some observation x. A particular sequence of targets is given by likelihood-tempered sequential Monte Carlo (LT-SMC) (Chopin and Pa-

paspiliopoulos, 2020). LT-SMC runs SMC samplers using a base distribution, typically the prior $p(z)$, as the initial target γ_1, and then anneals toward $p(z \mid x)$ through the intermediate targets $\gamma_t(z) \propto p(z)p(x \mid z)^{\tau_t}$ with $0 = \tau_1 < \cdots < \tau_T = 1$.

SMC samplers form *discrete* or *empirical* approximations to each distribution $\gamma_t(z)$. The discrete approximation to γ_1 is computed by importance sampling, which produces an initial K-particle approximation $\mathrm{Cat}(z_1^{1:K}, w_1^{1:K})$ to γ_1. Then, at each stage $t = 1, \ldots, T-1$, three distinct steps are performed:

Resample: Draw $z_t^{(i)} \overset{iid}{\sim} \mathrm{Cat}(z_t^{1:K}, w_t^{1:K})$, $i \in [K]$.

Mutate: Propose $z_{t+1}^i \sim M(z_t^{(i)}, dz_{t+1})$, $i \in [K]$.

Update: Recalculate weights $w_{t+1}^{1:K}$ for $z_{t+1}^{1:K}$.

Above, $\mathrm{Cat}(\cdot, \cdot)$ denotes a categorical distribution and $M(\cdot, dz)$ denotes a transition kernel. The weight update depends on the transition kernel. Expectations with respect to the posterior $\gamma_T = p(z \mid x)$ can be approximated by Monte Carlo integration, that is, $\mathbb{E}_{\gamma_T} f(z) \approx \sum_{i=1}^{K} w_T^i f(z_T^i)$. These approximations are biased but consistent as $K \to \infty$ with bias and variance of order $O(\frac{1}{K})$ (Chopin and Papaspiliopoulos, 2020).

The random variables $z_t^{1:K}$ and $w_t^{1:K}$ for $t < T$ are auxiliary: they are not used to approximate $\mathbb{E}_{p(z|x)} f(z)$, but to guide particles toward areas of high mass in the target distribution. Resampling eliminates particles with low weights, while mutation and reweighting provide a means to produce new particles and to weight them according to their quality with respect to the subsequent target $\gamma_{t+1}(z)$.

Like SNIS, SMC only requires the evaluation of an unnormalized density $\tilde{\gamma}_t(z)$ of each target distribution, as is typically required in a Bayesian setting. Compared to Markov chain Monte Carlo (MCMC) methods, SMC typically requires many fewer steps (e.g., $T < 100$) and is more readily adaptable: for example, the mutation kernel need not satisfy detailed balance conditions (Naesseth, Lindsten, and Schön, 2019). Although SMC is more expensive than SNIS due to the $T > 1$ stages, parallelization of operations across the K particles can be highly efficient.

3 MASS CONCENTRATION IN RWS

To fit an amortized encoder network to minimize the forward KL divergence through iterative optimization, we repeatedly evaluate its gradient:

$$\frac{1}{n} \sum_{j=1}^{n} \mathbb{E}_{p(z|x_j)} \nabla_\phi \log q_\phi(z \mid x_j). \quad (3)$$

Doing so requires us to evaluate expectations with respect to the posteriors $p(z \mid x_j)$. Estimating each term of Equation 3 using self-normalized importance sampling (SNIS) with q_ϕ as the proposal, as in wake-phase training, can result in degenerate variational distributions that concentrate mass. Le et al. (2019) first noted this degeneracy for small K, but our case studies show that it persists in practice even for large numbers of particles, such as $K = 1000$, and, surprisingly, it is not resolved by taking a defensive approach to importance sampling (Section 6). We conjecture that this failure mode is due to a "circular pathology" that arises when simultaneously proposing from q_ϕ to optimize the score of q_ϕ itself. Consider the wake-phase gradient

$$-\sum_{i=1}^{K} w^i \nabla_\phi \log q_\phi(z^i \mid x)$$

for fixed x, with w^i denoting the normalized weights. This gradient can be viewed as a Monte Carlo estimate of the gradient of the surrogate objective

$$\mathbb{E}_{z^1,\dots,z^K \overset{iid}{\sim} q_\phi(z|x)} -\sum_{i=1}^{K} w^i \log q_\phi(z^i \mid x) \qquad (4)$$

if one places stop gradient operators on the weights and the law used to generate samples (refer to Appendix B for more details). We now investigate the properties of this objective. Note that wake-phase updates do not directly optimize Equation 4, but the wake-phase gradient estimator is computed identically to a Monte Carlo estimator of the gradient of Equation 4 with stop gradient operations in place. It is illustrative to consider what might go wrong if one attempted to minimize this surrogate objective directly, as wake-phase training dynamics may behave similarly.

Let $\mathcal{L}(q_\phi)$ denote the value of the surrogate functional above for any given proposal distribution q_ϕ. When $K = 1$ we note that this objective can made arbitrarily low by choosing a value of q that is quite peaked (and also arbitrarily different from the true posterior). For any x, a sample $z' \sim q_\phi(z \mid x)$ is drawn and the objective is approximated using the log of its own density, i.e. $\log q_\phi(z' \mid x)$. Thus, a highly peaked q_ϕ can trivially achieve an arbitrarily low value of the objective. This result should not be so surprising: SNIS fails when applied with only a single sample. As is well known, SNIS also fails when the target is not absolutely continuous with respect to the proposal. As a result, even taking K to be arbitrarily large, it is possible to find peaked proposals that are arbitrarily dissimilar from the true posterior and yet lead to arbitrarily low values of the surrogate objective.

Proposition 1. Let $\mathcal{L}(q)$ denote the surrogate objective defined above for fixed x and fixed $K \in \mathbb{N}$. Let p denote the posterior $p(z \mid x)$. Then there exists $q(z) \neq p(z \mid x)$ such that $\mathcal{L}(q) < \mathcal{L}(p)$.

Appendix B provides a proof. Proposition 1 implies that even for arbitrarily large K, there exists a q with a lower surrogate objective value than the exact posterior has. Our proof constructs $q \sim \text{Unif}(0, \delta)$ for small δ.

In practice, the same issue arises even when $p(z \mid x)$ is absolutely continuous with respect to q_ϕ, as demonstrated by the following toy example. Take $z \sim \mathcal{N}(0, 10^2)$ and $x \mid z \sim \mathcal{N}(z, 1^2)$. The exact posterior is $\mathcal{N}(\frac{100}{101}x, \frac{100}{101})$. Using $K = 10,000$ importance samples, we estimate the surrogate objective for each of the highly peaked Gaussian variational distributions in Table 1 for a fixed, simulated draw x. The estimated objective for each peaked Gaussian is lower than the estimated objective for the exact posterior.

Proposal Distribution	Wake Objective
$q \sim \mathcal{N}(0, .0001^2)$	-4.690 (1.471)
$q \sim \mathcal{N}(0, .00001^2)$	-6.841 (1.947)
$q \sim \mathcal{N}(0, .000001^2)$	-9.439 (1.497)
$q \sim \mathcal{N}(0, .0000001^2)$	-11.798 (1.585)
$q \sim \mathcal{N}(\frac{100}{101}x, \frac{100}{101})$	1.415 (0.008)

Table 1: Avg. surrogate objective values for highly-peaked Gaussian proposals (standard errors in parentheses).

Using q_ϕ as a proposal for SNIS (as in RWS) is common, perhaps because there is a known case in which doing so results in unbiased SNIS estimates: if $q_\phi(z \mid x_j) = p(z \mid x_j)$ for all $x_j \in \mathcal{D}$, the SNIS estimate of Equation 3 is unbiased. However, this special case is likely to be irrelevant in real-world settings: firstly, there may be no parameters for which this equality holds due to the variational gap and the "amortization gap" (Cremer, Li, and Duvenaud, 2018), and secondly, the circular pathology itself may prevent optimization trajectories from converging to such parameter values. The circular pathology can be remedied by using an SNIS proposal that does not depend on q_ϕ, such as the prior $p(z)$. However, the prior is largely uninformative, so most of the particles sampled from it will be of poor quality.

4 SMC-WAKE

To address the deficiencies of RWS, we propose a method called SMC-Wake for fitting $q_\phi(z \mid x)$ to minimize the average forward KL divergence (Equation 1). Our method is detailed in Algorithm 1. SMC-Wake generates discrete particle approximations $\hat{P}$ to the posterior $p(z \mid x_j)$ for each point $x_j \in \mathcal{D}$, and fits

the encoder $q_\phi(z \mid x)$ by using these to estimate the gradient of the forward KL divergence. The proposed method uses sequential Monte Carlo samplers to construct discrete particle approximations to the posterior, rather than importance samplers. SMC-Wake also uses a gradient estimator that is consistent as the number of SMC sampler runs tends to infinity (with a fixed number of SMC particles per sampler).

4.1 Our LT-SMC subroutine

For each point $x_j \in \mathcal{D}$, SMC-Wake constructs discrete particle approximations $\hat{P}$ to $p(z \mid x_j)$ using LT-SMC (sketched in Algorithm 3, Appendix C). We perform mutation at each stage using a random-walk Metropolis-Hastings kernel $M_t(\cdot, dz_t)$ that is invariant with respect to γ_{t-1}. Our exact implementation (Appendix C) follows that of Chopin and Papaspiliopoulos (2020) and uses adaptive temperature selection and optional resampling. Given an observation x_j and a number of particles K, LT-SMC provides a discrete approximation $\text{Cat}(z_T^{1:K}, w_T^{1:K})$ to the posterior $p(z \mid x_j)$, which we use to estimate a term of the gradient of the forward KL divergence (Equation 3). In our notation, we henceforth suppress the dependence on T, as we always use discrete particle approximations from the final stage T to compute expectations. Importantly, the entire LT-SMC procedure does not depend on q_ϕ. By using the prior as the base distribution γ_1, LT-SMC avoids the circular pathology that we previously identified (Section 3). Although the prior alone is uninformative as a proposal, annealing along a tempering schedule results in discrete particle approximations that approximate the per-observation posteriors $p(z \mid x_j)$ well.

4.2 Consistent gradient estimation

For any fixed number of particles K and observation $x_j \in \mathcal{D}$, the particle approximations from LT-SMC yield gradient estimators $-\sum_{i=1}^{K} w^i \nabla_\phi \log q_\phi(z^i \mid x_j)$, which are biased for $\mu_j = -\mathbb{E}_{p(z|x_j)} \nabla_\phi \log q_\phi(z \mid x_j)$. This is problematic because stochastic gradient descent based on biased estimators is not guaranteed to converge. In this section, we propose three gradient estimators that are asymptotically unbiased, including two that are consistent. Our estimators accomplish this by reducing the bias from self-normalization by averaging the different particle approximations that are computed across the optimizer iterations. Asymptotic results are obtained as the number of particle approximations $M \to \infty$, where the number of particles K remains fixed.

Consider a fixed observation x. The final target $\gamma_T(z) \propto p(z, x)$ in LT-SMC is known up to a normalization constant C such that $p(z, x) = C\gamma_T(z)$, i.e.,

$C = p(x)$. Each individual LT-SMC sampler yields an estimate

$$\hat{C} = \prod_{t=1}^{T} \left(\frac{1}{K} \sum_{i=1}^{K} \tilde{w}_t^i \right), \tag{5}$$

where $\tilde{w}_t^{1:K}$ are the unnormalized weights from Algorithm 3. This estimate satisfies $\mathbb{E}\hat{C} = C$, with the expectation taken over all the random variables generated by the SMC algorithm (Naesseth, Lindsten, and Schön, 2019). SMC-Wake constructs an estimate of C from M LT-SMC runs using the estimator $\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}$, where $\hat{C}_{(m)}$ is the estimate of the normalization constant from iteration $m \in \{1, \ldots, M\}$. By the strong law of large numbers, $\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)} \overset{a.s.}{\to} C$ as $M \to \infty$.

SMC samplers can provide unbiased estimates of $C\mathbb{E}_{p(z|x)} f(z)$ for a test function f. As $C = p(x)$ is a high-dimensional integral, C is usually unknown and so one typically divides by the estimate $\hat{C}$, resulting in bias. Within our iterative fitting procedure for q_ϕ, we instead propose to combine many normalization constant estimates produced by many runs of SMC. These can be averaged with $O(1)$ memory to iteratively refine our estimate of C while simultaneously fitting the variational parameters ϕ. As the number of LT-SMC runs M increases, $\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}$ converges to C, leading to unbiased estimates of $\mathbb{E}_{p(z|x)} f(z)$ as $M \to \infty$.

Table 2 provides three different ratio estimators, constructed with $f(z) = -\nabla_\phi \log q_\phi(z \mid x)$ as the test function for estimating the gradient of the forward KL divergence. In each case, $z_{(m)}^{1:K}$, $w_{(m)}^{1:K}$, and $\hat{C}_{(m)}$ denote the atom positions, weights, and normalizing constant estimates returned by the mth run of LT-SMC. For $\hat{\nabla}^{(b)}$, $\tilde{z}_{(m)}$ refers to a single draw from $\text{Cat}(z_{(m)}^{1:K}, w_{(m)}^{1:K})$. Each of the numerators in Table 2 is an unbiased estimator of $C\mathbb{E}_{p(z|x)} f(z)$. Dividing each by $\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}$ yields an estimator of $\mathbb{E}_{p(z|x)} f(z)$. The different numerators exploit different aggregations of the empirical distributions $\text{Cat}(z_{(m)}^{1:K}, w_{(m)}^{1:K})$ to approximate the posterior $p(z \mid x)$, each with different memory requirements.

Table 2: Forward KL Gradient Estimators

Label	Estimator	Memory
$\hat{\nabla}^{(a)}$	$\dfrac{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)} \left(\sum_{k=1}^{K} w_{(m)}^k f(z_{(m)}^k) \right)}{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}}$	$O(MK)$
$\hat{\nabla}^{(b)}$	$\dfrac{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)} f(\tilde{z}_{(m)})}{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}}$	$O(M)$
$\hat{\nabla}^{(c)}$	$\dfrac{\hat{C}_{(M)} \left(\sum_{k=1}^{K} w_{(M)}^k f(z_{(M)}^k) \right)}{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}}$	$O(K)$

In our amortized setting, because we have a collection of data points $\mathcal{D} = \{x_1, \ldots, x_n\}$, our method keeps

track of the M different LT-SMC runs for each point $x_j \in \mathcal{D}$. For any point $x_j \in \mathcal{D}$, $\hat{\nabla}_j^{(a)}, \hat{\nabla}_j^{(b)}$ and $\hat{\nabla}_j^{(c)}$ are the three gradient estimators from Table 2 for a given iteration of the training procedure. The SMC-Wake procedure is detailed Algorithm 1, where we use the generic gradient estimator $\hat{\nabla}$: any of $\hat{\nabla}^{(a)}, \hat{\nabla}^{(b)}, \hat{\nabla}^{(c)}$ can be used. SMC-Wake interleaves stochastic gradient descent steps that follow the gradient $\frac{1}{n} \sum_{j=1}^{n} \hat{\nabla}_j$ with LT-SMC runs (Algorithm 3). Here we average over all n training observations, but a mini-batch will also suffice. Each $\hat{\nabla}_j$ itself is an aggregate of M samplers for each point. Each additional run of LT-SMC computes new particle approximations and normalizing constant estimates for each point $x_j \in \mathcal{D}$. Each time new particle approximations are computed for a given point x_j, the estimators $\hat{\nabla}_j^{(a)}, \hat{\nabla}_j^{(b)}$ or $\hat{\nabla}_j^{(c)}$ are recomputed and the estimate of the normalizing constant $C_j = p(x_j)$ is refined. Algorithm 1 suggests rerunning LT-SMC samplers for each point $x_j \in \mathcal{D}$ at each iteration, but SMC-Wake can be made less computationally demanding by distributing LT-SMC runs into batches. In our experiments (Section 6), for example, we rerun LT-SMC for only a single observation at each iteration. It is also possible to decouple the process of running LT-SMC with the process of optimization: perform a large number of LT-SMC runs before optimization begins, and then re-use these particles for every step of the optimization.

Algorithm 1: `SMC-Wake`

Inputs: Data $x_1, \ldots, x_n$, encoder q_ϕ, likelihood $p(x \mid z)$, prior $p(z)$, SGD step size η, number of particles K, temperatures τ.

Initialize $\text{SMC}_j = \emptyset$, $j = 1, \ldots, n$

Repeat

  for $j = 1, \ldots, n$ /* (for each datapoint */

  do

    $z^{1:K}, w^{1:K}, \hat{C} \leftarrow \texttt{LT-SMC}(x_j, p(x \mid z), p(z), \tau)$

    $\text{SMC}_j = \text{SMC}_j \cup \{[z^{1:K}, w^{1:K}, \hat{C}]\}$

    Compute $\hat{\nabla}_j$ /* (See Table 2) */

  end

  $\phi \leftarrow \phi - \eta \frac{1}{n} \sum_{j=1}^{n} \hat{\nabla}_j$

Until Convergence

Return q_ϕ.

4.3 Asymptotic analysis

We show that $\hat{\nabla}^{(a)}, \hat{\nabla}^{(b)}$ and $\hat{\nabla}^{(c)}$ are asymptotically unbiased and that the former two estimators are strongly consistent. We begin with a well-known result: the LT-SMC approximations to the posterior become

arbitrarily accurate as the number of particles $K \to \infty$. The following proposition formalizes this result.

Proposition 2. *For an observation x and a joint model with prior $p(z)$ and bounded likelihood density $p(x|z)$, let $0 = \tau_1 < \cdots < \tau_T = 1$ denote a tempering schedule. Let $\gamma_t(z) \propto p(z)p(x \mid z)^{\tau_t}$ denote intermediate targets, and let $M_t(\cdot, dz_t)$ be a sequence of Markov transition kernels that leaves γ_{t-1} invariant. Let $\hat{P} \sim \text{Cat}(w_T^{1:K}, z_T^{1:K})$ be the particle distribution produced by LT-SMC using $\{\tau_t, \gamma_t, M_t\}_{t=1}^{T}$. Then, for any measurable and bounded test function f,*

$$\lim_{K \to \infty} \mathbb{E}\left[\left(\mathbb{E}_{\hat{P}}[f(z)] - \mathbb{E}_{p(z|x))}[f(z)] \right)^2 \right] = 0.$$

A proof follows by applying Proposition 11.3 of Chopin and Papaspiliopoulos (2020) with $G_t(z_{t-1}, z_t) = p(z_t)p(x|z_t)^{\tau_t - \tau_{t-1}}$. Above, the outer expectation is taken with respect to all variables generated by the SMC algorithm. In our case, we are interested in the test function $f(z) = -\nabla_\phi \log q_\phi(z \mid x)$. Proposition 2 implies that gradient estimators based on a single LT-SMC run are asymptotically consistent as $K \to \infty$. In practice, however, finite K must be used, resulting in biased gradient estimators.

The estimators $\hat{\nabla}_j^{(a)}, \hat{\nabla}_j^{(b)}$ and $\hat{\nabla}_j^{(c)}$ (Table 2) resolve this bias. These estimators with fixed K are asymptotically unbiased as the number of LT-SMC runs $M \to \infty$ and the former two are strongly consistent.

Proposition 3. *For an observation x_j, suppose that $f(z) = -\nabla_\phi \log q_\phi(z \mid x_j)$ is a bounded and measurable function for any $\phi \in \Phi$. For $m = 1, \ldots, M$, let the random variables $\hat{C}_{(m)}$, $w_{(m)}^{1:K}$, and $z_{(m)}^{1:K}$ result from an independent run of LT-SMC (Algorithm 3) with a fixed temperature schedule. Then, the gradient estimators $\hat{\nabla}_j^{(a)}$ and $\hat{\nabla}_j^{(b)}$ are strongly consistent for $\mathbb{E}_{p(z|x_j)}f(z)$ in M and the estimator $\hat{\nabla}_j^{(c)}$ is asymptotically unbiased in M.*

Appendix D provides a proof. Strongly consistent estimators may be preferred, but are more costly: computing $\hat{\nabla}_j^{(a)}$ and $\hat{\nabla}_j^{(b)}$ require $O(MK)$ and $O(M)$ memory, respectively, for each data point x_j. However, in our experiments (Section 6), we find that SMC-Wake outperforms wake-phase training even for relatively small M, for which memory usage is not a limiting factor, for example $M = 100$. Moreover, for any fixed number of particles K, the estimator $\hat{\nabla}_j^{(c)}$ is asymptotically unbiased yet requires just $O(K)$ memory (regardless of M). This is advantageous compared to wake-phase training, which uses biased gradient estimators and also requires $O(K)$ memory. We further discuss these three estimators and their memory usage in Appendix D.

4.4 Extension to particle MCMC

Just as Markov chain Monte Carlo methodology can be incorporated into SMC algorithms (e.g., the mutation kernels in Algorithm 3), SMC samplers can be nested within MCMC frameworks. Particle-Independent Metropolis-Hastings (PIMH) (Andrieu, Doucet, and Holenstein, 2010) is one method of constructing a Markov chain that samples the posterior using multiple runs of LT-SMC. Given a state z and a normalizing constant estimate $\hat{C}$, a proposed draw $z_{\text{new}} \sim \hat{P}_{\text{new}}$ is accepted with probability $\alpha = \min(1, \frac{\hat{C}_{\text{new}}}{\hat{C}})$, where $\hat{P}_{\text{new}}, \hat{C}_{\text{new}}$ are quantities from a new LT-SMC run.

For situations where memory is limiting, we propose adding a PIMH outer loop to SMC-Wake to yield a new algorithm: SMC-PIMH-Wake (Algorithm 2). Accepting or rejecting new particles within Metropolis-Hastings steps takes the place of iterative aggregation. For any data point x_j, this procedure results in a convergent Markov chain with $p(z \mid x_j)$ as its stationary distribution under mild conditions. SMC-PIMH-Wake yields asymptotically unbiased estimates of the gradient of the forward KL divergence as the number of Metropolis-Hastings steps tends to infinity with $O(K)$ memory, similar to SMC-Wake with the gradient estimator $\hat{\nabla}^{(c)}$. As the test function $f(z) = -\nabla_\phi \log q_\phi(z \mid x_j)$ changes with every gradient step, consistent estimates cannot be obtained with finite memory. In situations where the variance of the gradient estimators computed by LT-SMC is large, SMC-PIMH-Wake may provide lower-variance gradient updates, e.g. by using the same SMC sampler to estimate gradients for multiple gradient steps consecutively.

Algorithm 2: `SMC-PIMH-Wake`

Initialize $\hat{P}_j = \text{Cat}(z^{1:K}, w^{1:K})$, $\hat{C}_j = \hat{C}$ with
$z^{1:K}, w^{1:K}, \hat{C} \leftarrow \texttt{LT-SMC}(x_j, p(x \mid z), p(z), \tau)$
Repeat
    for $j = 1, \dots, n$ **do**
        $z_{\text{new}}^{1:K}, w_{\text{new}}^{1:K}, \hat{C}_{\text{new}} \leftarrow \texttt{LT-SMC}(x_j, p(x \mid z), p(z), \tau)$
        Compute $\alpha = \min\left(1, \frac{\hat{C}_{\text{new}}}{\hat{C}_j}\right)$
        Set $\hat{P}_j = \text{Cat}(z_{\text{new}}^{1:K}, w_{\text{new}}^{1:K})$, $\hat{C}_j = \hat{C}_{\text{new}}$ w.p. α
    end
    $\phi \leftarrow \phi - \eta \frac{1}{n} \sum_{j=1}^{n} \left(-\mathbb{E}_{\hat{P}_j} \nabla_\phi \log q_\phi(z \mid x_j) \right)$
Until Convergence

---

5 RELATED WORK

Variational Sequential Monte Carlo (VSMC) (Naesseth et al., 2018), Filtering Variational Objectives (FIVO) (Maddison et al., 2017), and Auto-Encoding Sequential Monte Carlo (AESMC) (Le et al., 2018) are closely related to each other; all use SMC to facilitate optimization of bounds on the marginal evidence. In contrast to SMC-Wake, these methods are tailored to sequential data and optimize the reverse KL divergence. Optimizing the reverse KL divergence is particularly difficult within an SMC framework as computing gradients of the reverse KL often relies on the reparameterization trick; however, the resampling step of SMC is not easily reparameterized. By minimizing the forward KL divergence, we eliminate the need for reparameterization.

Markovian Score Climbing (MSC) (Naesseth, Lindsten, and Blei, 2020) is also a method for optimizing the forward KL. It uses an MCMC outer loop to achieve asymptotic guarantees. In this way, MSC differs from SMC-Wake (Algorithm 1), but is similar to SMC-PIMH-Wake. MSC uses a conditional importance sampling (CIS) transition kernel with q_ϕ as the proposal distribution, rather than a Metropolis-Hastings kernel as is used in SMC-PIMH-Wake. We show in Section 6.3 that our PIMH variant outperforms MSC, which mixes slowly because initially q_ϕ is a poor proposal compared to the particles generated by LT-SMC.

Neural Adaptive Sequential Monte Carlo (NASMC) (Gu, Ghahramani, and Turner, 2015) is a non-amortized method for sequential data (e.g., state-space models) that adapts SMC mutation kernels to minimize the forward KL divergence to the intermediate targets. SMC-Wake differs from NASMC in its use of Metropolis-Hastings kernels that do not depend on q_ϕ, and in its gradient calculations that only use particle approximations to the final target γ_T. Like RWS, NASMC gradient estimators are also biased for finite K. Three related methods, namely, Annealed Flow Transport (AFT) (Arbel, Matthews, and Doucet, 2021), Continual Repeated Annealed Flow Transport (Matthews et al., 2022), and Nested Variational Inference (NVI) (Zimmermann et al., 2021), also sample from a target distribution by fitting proposals to minimize a sequence of divergences to intermediate targets. These sampling methods can be incorporated into the SMC-Wake framework: while we propose using LT-SMC samplers, any SMC-based algorithm can be used. The circular pathology is avoided as long as the parameters of the proposals are not shared with the amortized encoder q_ϕ. Using methods such as these in place of the LT-SMC subroutine in Algorithm 1 is a potential direction for future research.

6 EXPERIMENTS

6.1 Two moons

The "two moons" model has been extensively used in the simulation-based inference literature to benchmark existing algorithms (Greenberg, Nonnenmacher, and Macke, 2019; Lueckmann et al., 2021). The generative process for latent $z \in \mathbb{R}^2$ and observed $x \in \mathbb{R}^2$ first draws $z_1, z_2 \overset{iid}{\sim} U(-1, 1)$, and then draws two auxiliary variables $a \sim U\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ and $r \sim \mathcal{N}\left(0.1, 0.01^2\right)$. Finally, the observation

$$x^\top = p + \left[-\frac{|z_1 + z_2|}{\sqrt{2}}, \frac{-z_1 + z_2}{\sqrt{2}}\right],$$

where $p = [r\cos(a) + 0.25, r\sin(a)]$. The posterior distribution on z given x takes the shape of two crescent moons facing each other. Because these regions are disconnected, separated by a zero density region of the posterior, exploring the latent space is potentially difficult and a highly flexible variational family is required to approximate the posterior well. We use the neural spline flow (NSF) as the variational family, conditional on a given observation x (Durkan et al., 2019).

Given 100 points $x_1, \ldots, x_{100}$ generated independently from the two-moons model, we compare SMC-Wake with estimator $\hat{\nabla}^{(a)}$, RWS (wake-phase training only), and a defensive variant of RWS that performs importance sampling using the mixture $\frac{1}{2}p(z) + \frac{1}{2}q_\phi(z \mid x)$ as a proposal. Defensive importance sampling has been found to ameliorate the mass concentration problem that can occur in wake-phase training when q_ϕ is used as the proposal (Le et al., 2019). Additionally, this choice of proposal avoids a divide-by-zero error that arises when the proposed particles all have zero as their posterior densities.

Additional details of the training procedure and results for $\hat{\nabla}^{(b)}$ and $\hat{\nabla}^{(c)}$ are in Appendix E. Figure 1 shows the amortized variational posteriors $q_\phi(z \mid x_{16})$ as an example. SMC-Wake is the only method that captures the shape of the posterior. Wake-phase training severely concentrates mass into a single point-like region, exhibiting the degeneracy described in Section 3. Appendix E shows that problematic mass concentration for RWS occurs quickly, within 1000 gradient steps. While the defensive variant of RWS does better, it is still far worse than the quality of the SMC-Wake approximations. Variational approximations for several additional data points are shown in Appendix E.

6.2 Avoiding mode collapse in MNIST

We consider learning a model of MNIST digits to illustrate that SMC-Wake, though an inference method,
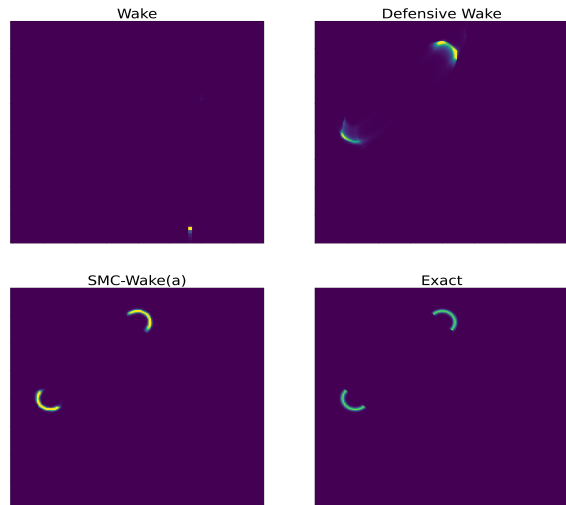

Figure 1: Posterior approximations given x_{16} by SMC-Wake, Wake, and Defensive Wake. The bottom right panel depicts the exact posterior distribution.

can nonetheless aid in model learning. Given 1000 normalized MNIST digits and labels $\{x_i, \ell_i\}_{i=1}^{1000}$, we fit a conditional model $p(x \mid \ell, z)$ to maximize the importance-weighted bound (IWBO) (Appendix A) while simultaneously fitting the encoder q_ϕ. This follows the reweighted wake-sleep (RWS) framework of Bornschein and Bengio (2015). The model is based on a sigmoid belief network (Saul, Jaakkola, and Jordan, 1996) and is given by

$$p_\psi(x \mid \ell, z) \sim \mathcal{N}(\sigma(W_\ell z + b_\ell), \tau^2 I_d), \qquad (6)$$

where $\sigma(\cdot)$ denotes the sigmoid function, $\ell \in \{0, \ldots, 9\}$ denotes the label, and $\psi = \{W_0, \ldots, W_9, b_0, \ldots, b_9\}$ denotes model parameters to be learned. The hyperparameter $\tau = 0.01$ results in a highly peaked likelihood. As the labels ℓ are fixed, this model fitting task is equivalent to fitting a separate model for each digit class. The amortized encoder $q_\phi(z \mid x, \ell)$ is fit jointly with the model using both SMC-Wake and wake-phase training, with alternating gradient updates for ϕ and ψ, respectively. We use the gradient estimator $\hat{\nabla}^{(b)}$ in this example.

Additional details of the implementation are given in Appendix F. Wake-phase gradient updates fit an encoder network that severely concentrates mass, learning similar latent representations z for all digits in a given class ℓ. This results in mode collapse in the generative model, visible by visualizing reconstructions (Figure 2). Given different instances of the zero digit class, for example, the wake-phase reconstructions (middle) look nearly identical. This occurs due to the pathological concentration of $q_\phi(z \mid x, \ell)$ described in Section 3, resulting in nearly identical latent representations for

all instances of a digit ℓ. Appendix F shows similar behavior for the other digit classes as well. SMC-Wake avoids this degeneracy; its reconstructions (bottom) closely match the true images (top).
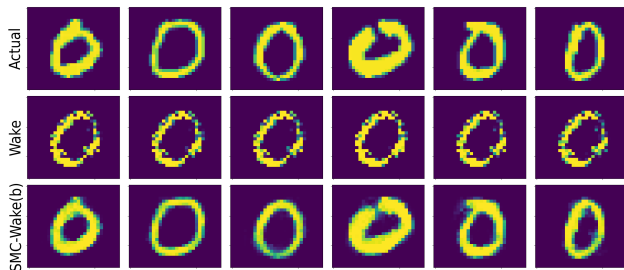

Figure 2: Illustration of mass concentration in wake-phase training. Wake-phase reconstructions (middle) of real MNIST digits (top) with label zero all look nearly identical, unlike SMC-Wake reconstructions (bottom).

6.3 Transformed Gaussian

6.3.1 Nested MCMC approaches

We consider a Gaussian-Gaussian hierarchical model with latent variable $z \sim \mathcal{N}(0, \sigma^2 I_p)$ and data $x \mid z \sim \mathcal{N}(Az, \tau^2 I_d)$ to compare two methods that make use of an MCMC outer loop: SMC-PIMH-Wake (Section 4.4) and Markovian Score Climbing (MSC) (Naesseth, Lindsten, and Blei, 2020). MSC proposes particles for the next step of the Markov chain using q_ϕ. If this proposal is poor, then the Markov chain mixes slowly, and thus may concentrate mass. We study a high-dimensional case with $p = 50$ and $d = 100$, and take standard deviations $\sigma = \tau = 1$. The design matrix $A \in \mathbb{R}^{d \times p}$ is full-rank and fixed.

Both methods fit a variational family $q_\phi(z \mid x) \sim \mathcal{N}(\mu, LL^\top + \epsilon I)$ with a lower-triangular matrix $L \in \mathbb{R}^{p \times p}$. This family is flexible enough to approximate the posterior almost exactly. (The addition of ϵI with $\epsilon = 0.0001$ provides numerical stability.) The variational posterior amortizes over $n = 50$ points $\{x_j\}_{j=1}^{50}$ generated from the above model. The exact posterior distribution can be written in closed form by completing the square (see Appendix G); this enables analytical computation of the forward, reverse, and symmetric KL divergences for comparison. All divergences are averaged over the $n = 50$ observations.

We used $K = 100$ particles for importance sampling in MSC and $K = 100$ particles for LT-SMC in SMC-PIMH-Wake. We find that MSC mixes slowly, and thus performs poorly compared to SMC-PIMH-Wake. Table 3 presents results after 500,000 gradient steps for MSC, corresponding to at least 250,000 MCMC steps for each point $x_j \in \mathcal{D}$ with mini-batching. In

contrast, the results for SMC-PIMH-Wake are computed after only 40,000 gradient steps. As LT-SMC is more expensive than importance sampling, we perform fewer MCMC steps, only 800 MCMC steps for each x_j. (Appendix G provides additional details about our implementation.) SMC-PIMH-Wake leads to lower average forward KL divergence than MSC. The mixing time for MSC is large because initially q_ϕ is a poor proposal, whereas from the first iteration SMC-PIMH-Wake proposes from a high-quality approximation to the posterior computed by LT-SMC.

Table 3: Gaussian Hierarchical Model Learning Results (Lower is better.)

	MSC	SMC-PIMH-Wake
Forward KL	2339	**1387**
Reverse KL	1762	**1287**
Symmetric KL	4102	**2674**

6.3.2 The importance of averaging

We now illustrate that for a fixed computational budget, the approach used in SMC-Wake of taking a small K and using many different samplers outperforms the naive approach of taking large K and using a single sampler. We use a similar setting as above. (For implementation details, see Appendix G.) For each observation, we run LT-SMC with $K = 10,000$ particles and also perform $M = 100$ runs each using $K = 100$ particles. Gradient estimators are constructed from each of these two settings and used to fit q_ϕ. We use a naive estimator for the $M = 100$ case that does not weight according to $\hat{C}_{(m)}$ values. Both settings require a similar number of likelihood evaluations. Figure 3 demonstrates that fitting the encoder with $M = 100$ samplers results in a q_ϕ with significantly lower average forward KL divergence to the posterior.
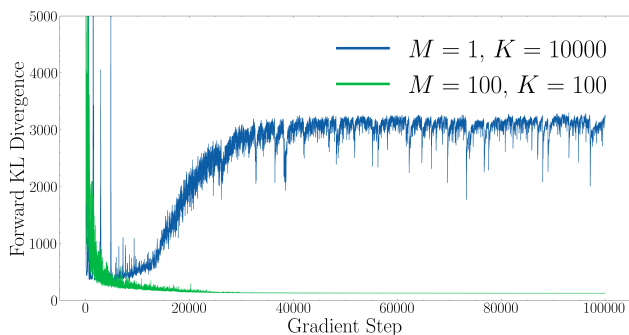

Figure 3: Average forward KL divergence during training for an encoder q_ϕ fit using one $K = 10,000$ sampler per-point (blue) and $M = 100$ different $K = 100$ samplers (green).

The larger K counterintuitively performs worse because it suffers from a lack of sample replenishment (Matthews et al., 2022). If the $K = 10,000$ SMC sampler has a low effective sample size (ESS), as occurs in this case due to the large dimension $p = 50$ of the latent space, the encoder q_ϕ is significantly overfit to only a few particles. Although these particles may be located at a peak of the posterior density, the lack of diversity among them hinders the fitting process. In contrast, even if all $M = 100$ samplers of $K = 100$ have low ESS, the diversity of samples is richer and averaging over many independent samplers reduces variance (at the cost of increased bias due to smaller K).

6.4   Galaxy spectra emulator

The Probabilistic Value-Added Bright Galaxy Survey (PROVABGS) simulator is a state-of-the-art simulator of astronomical spectra (Hahn et al., 2022) that maps parameters θ to spectra x. Each spectrum is a high-dimensional vector of flux measurements for an astronomical object: at each wavelength λ in a grid between 3000 and 10,000 angstroms, the spectrum records the flux, typically measured in units $\text{erg} \cdot \text{cm}^{-2}\text{s}^{-1}\text{Å}^{-1}$ (Donald G. York, J. Adelman et al., 2000; DESI Collaboration, B. Abareshi et al., 2022). We consider the problem of predicting parameters from observed spectra: given noisy samples of the spectra x, we will learn to reconstruct the astronomical parameters θ.

We train a neural network emulator of the PROVABGS simulator for fast likelihood evaluation and work with normalized spectra for ease. Details are given in Appendix H. Our emulator takes 11 scientific input parameters θ of interest as inputs and produces spectra as outputs. (Examples are given in Appendix H.) We use the priors for the 11 parameters suggested by Hahn et al. (2022). As the simulator's output is noiseless, we add Gaussian noise proportional to the signal strength in each dimension to simulate typical measurement error, with signal-to-noise ratio (SNR) set to $1/\sigma$ with $\sigma = 0.1$. The likelihood function is thus multivariate Gaussian and can be evaluated efficiently.

Given 100 spectra from the forward model, we fit an amortized neural spline flow $q_\phi(\theta \mid x)$ using SMC-Wake with estimator $\hat{\nabla}^{(a)}$ and $K = 100$ particles. Additional implementation details are given in Appendix H. To assess the quality of the amortized encoder, as a surrogate for the ground truth we run MCMC using Metropolis-Hastings random walks with 100 walkers for 10,000 steps each following 10,000 burn-in steps. We plot several kernel-smoothed marginal density estimates from both MCMC and SMC-Wake in Figure 4. SMC-Wake generally captures the shape of the posterior well, assigning high density to the ground truth,

while being somewhat overdispersed as a consequence of minimizing the forward KL divergence. This compares favorably to wake-phase training, which exhibits numerical instability on this problem (Appendix H).
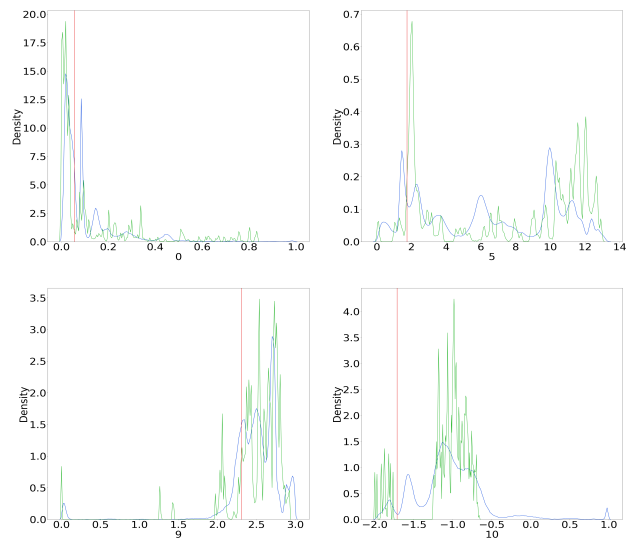

Figure 4: Posterior estimation for one example spectrum x from the training set. We used MCMC (green) and SMC-Wake (blue) to estimate the posterior on θ. Smoothed (marginal) estimates of the posterior for 4 of the 11 parameters θ_i are plotted, with the true value of the parameter indicated with a red vertical line.

7   DISCUSSION

SMC-Wake is a method for minimizing the forward KL divergence in amortized variational inference using likelihood-tempered SMC. SMC-Wake generalizes wake-phase training from Reweighted Wake-Sleep (RWS) and improves on it by providing consistent or asymptotically unbiased gradient estimates for the forward KL divergence. Wake-phase updates have a troublesome circular property: q_ϕ is used to make proposals that are themselves used to update ϕ. Empirical results demonstrate that this entanglement can lead to a degenerate concentration of mass. SMC-Wake avoids this degeneracy by proposing from the prior. Exploring ways to safely incorporate $q_\phi(z \mid x)$ into the proposal step is an interesting direction for future research.

SMC-Wake may not be appropriate in all settings. Each run of SMC is more expensive than the lightweight importance sampling used by wake-phase updates. However, in practice, we find that SMC yields more accurate answers. With further research, we speculate that incorporating proposal kernels based on likelihood gradients, such Metropolis-Adjusted Langevin kernels (Roberts and Rosenthal, 1998) into SMC-Wake may lead to even better performance.

Acknowledgements

References

[1] Luca Ambrogioni et al. "Forward Amortized Inference for Likelihood-Free Variational Marginalization". In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. 2019.

[2] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. "Particle Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 72.3 (2010), pp. 269–342.

[3] Michael Arbel, Alex Matthews, and Arnaud Doucet. "Annealed Flow Transport Monte Carlo". In: *Proceedings of the 38th International Conference on Machine Learning*. 2021.

[4] Michael Betancourt. "Cruising the simplex: Hamiltonian Monte Carlo and the Dirichlet distribution". In: *AIP Conference Proceedings* 1443.1 (2012), pp. 157–164.

[5] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112.518 (2017), pp. 859–877.

[6] Jörg Bornschein and Yoshua Bengio. "Reweighted Wake-Sleep". In: *3rd International Conference on Learning Representations*. 2015.

[7] Yuri Burda, Roger B. Grosse, and Ruslan Salakhutdinov. "Importance Weighted Autoencoders". In: *4th International Conference on Learning Representations*. 2016.

[8] Nicolas Chopin and Omiros Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer, 2020.

[9] Chris Cremer, Xuechen Li, and David Duvenaud. "Inference Suboptimality in Variational Autoencoders". In: *Proceedings of the 35th International Conference on Machine Learning*. 2018.

[10] Pierre Del Moral. "Feynman-Kac Formulae". In: *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer New York, 2004, pp. 47–93.

[11] Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. "Sequential Monte Carlo samplers". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68.3 (2006), pp. 411–436.

[12] DESI Collaboration, B. Abareshi et al. "Overview of the Instrumentation for the Dark Energy Spectroscopic Instrument". In: *The Astronomical Journal* 164.5 (2022), pp. 1–62.

[13] Justin Domke and Daniel R Sheldon. "Importance Weighting and Variational Inference". In: *Advances in Neural Information Processing Systems*. 2018.

[14] Donald G. York, J. Adelman et al. "The Sloan Digital Sky Survey: Technical Summary". In: *The Astronomical Journal* 120.3 (2000), pp. 1579–1587.

[15] Conor Durkan et al. "Neural Spline Flows". In: *Advances in Neural Information Processing Systems*. 2019.

[16] David Greenberg, Marcel Nonnenmacher, and Jakob Macke. "Automatic Posterior Transformation for Likelihood-Free Inference". In: *Proceedings of the 36th International Conference on Machine Learning*. 2019.

[17] Shixiang (Shane) Gu, Zoubin Ghahramani, and Richard E Turner. "Neural Adaptive Sequential Monte Carlo". In: *Advances in Neural Information Processing Systems*. 2015.

[18] ChangHoon Hahn et al. *The DESI PRObabilistic Value-Added Bright Galaxy Survey (PROVABGS) Mock Challenge*. 2022.

[19] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *Foundations and Trends in Machine Learning* 12.4 (2019), pp. 307–392.

[20] Tuan Anh Le et al. "Auto-Encoding Sequential Monte Carlo". In: *International Conference on Learning Representations*. 2018.

[21] Tuan Anh Le et al. "Revisiting Reweighted Wake-Sleep for Models with Stochastic Control Flow". In: *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence*. 2019.

[22] Erich L. Lehmann and George Casella. *Theory of Point Estimation*. 2nd ed. Springer New York, 1998.

[23] Jan-Matthis Lueckmann et al. "Benchmarking Simulation-Based Inference". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. 2021.

[24] Chris J Maddison et al. "Filtering Variational Objectives". In: *Advances in Neural Information Processing Systems*. 2017.

[25] Alex Matthews et al. "Continual Repeated Annealed Flow Transport Monte Carlo". In: *Proceedings of the 39th International Conference on Machine Learning*. 2022.

[26] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[27] Christian Naesseth, Fredrik Lindsten, and David Blei. "Markovian Score Climbing: Variational Inference with $KL(p||q)$". In: *Advances in Neural Information Processing Systems*. 2020.

[28] Christian Naesseth et al. "Variational Sequential Monte Carlo". In: *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. 2018.

[29] Christian A. Naesseth, Fredrik Lindsten, and Thomas B. Schön. "Elements of Sequential Monte Carlo". In: *Foundations and Trends in Machine Learning* 12.3 (2019), pp. 307–392.

[30] Art B. Owen. *Monte Carlo theory, methods and examples*. 2013. Chap. 9.

[31] Rajesh Ranganath, Sean Gerrish, and David Blei. "Black Box Variational Inference". In: *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*. 2014.

[32] Gareth O. Roberts and Jeffrey S. Rosenthal. "Optimal scaling of discrete approximations to Langevin diffusions". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60.1 (1998), pp. 255–268.

[33] Lawrence K. Saul, Tommi S. Jaakkola, and Michael I. Jordan. "Mean Field Theory for Sigmoid Belief Networks". In: *CoRR* (1996).

[34] Alvaro Tejero-Cantero et al. "sbi: A toolkit for simulation-based inference". In: *Journal of Open Source Software* 5.52 (2020), p. 2505.

[35] Martin J. Wainwright and Michael I. Jordan. "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends in Machine Learning* 1.1-2 (2008), pp. 1–305.

[36] Heiko Zimmermann et al. "Nested Variational Inference". In: *Advances in Neural Information Processing Systems*. 2021.

[37] Adam Ścibior and Frank Wood. *Differentiable Particle Filtering without Modifying the Forward Pass*. 2021.

Checklist

1. For all models and algorithms presented, check if you include:

   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]

   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]

   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:

   (a) Statements of the full set of assumptions of all theoretical results. [Yes]

   (b) Complete proofs of all theoretical results. [Yes]

   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:

   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]

   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]

   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]

   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

   (a) Citations of the creator If your work uses existing assets. [Yes]

   (b) The license information of the assets, if applicable. [Yes]

   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

   (d) Information about consent from data providers/curators. [Not Applicable]

   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Not Applicable]

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A OBJECTIVES FOR AMORTIZED INFERENCE

Given a data point x, variational inference seeks to approximate the intractable posterior $p(z \mid x)$ with a tractable distribution $q(z)$ (Blei, Kucukelbir, and McAuliffe, 2017; Wainwright and Jordan, 2008). Learning a different q for every x of interest can be computationally expensive. Amortized variational inference (AVI) leverages deep learning for this task, learning parameters ϕ of a neural network that transforms data x to the distributional parameters of a posterior approximation. AVI thus allows us to create the conditional distribution $q_\phi(z \mid x)$ for any x using a forward pass through this neural network (Ranganath, Gerrish, and Blei, 2014; Ambrogioni et al., 2019; Kingma and Welling, 2019). In many settings, the decoder $p_\psi(x \mid z)$ is also learned along with the inference network, with both fit to an observed dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$.

To learn the model parameters ψ and the variational parameters ϕ, variational autoencoders (VAEs) (Kingma and Welling, 2019) target the reverse KL divergence $\mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\psi(z \mid x))$ via maximization of

$$\mathrm{ELBO}(\psi, \phi) = \mathbb{E}_{z \sim q_\phi(z|x)} \log \frac{p_\psi(x, z)}{q_\phi(z \mid x)}. \tag{7}$$

The importance-weighted autoencoder (IWAE) targets a similar but tighter lower bound on the evidence, a multi-sample generalization of the ELBO called the importance-weighted bound (IWBO):

$$\mathrm{IWBO}(\psi, \phi) = \mathbb{E}_{z_1, \ldots, z_K \overset{iid}{\sim} q_\phi(z|x)} \log \sum_{i=1}^{K} \frac{1}{K} \frac{p_\psi(x, z_i)}{q_\phi(z_i \mid x)}. \tag{8}$$

Both ELBO and IWBO maximization can be viewed as the minimization of a reverse KL divergence. In the case of the ELBO, maximization is equivalent to minimization of $\mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\psi(z \mid x))$, while for the IWBO, the divergence is between two distributions on an extended sampling space (Kingma and Welling, 2019; Le et al., 2018).

To achieve lower variance gradients with respect to ϕ, implementations of both the IWBO and ELBO objectives often rely on the "reparameterization trick", writing a draw $z \sim q_\phi(z \mid x)$ as a deterministic function of an auxiliary random variable that is independent of ϕ (Kingma and Welling, 2019; Burda, Grosse, and Salakhutdinov, 2016). This reliance on reparameterization stems from the fact that expectations are computed with respect to draws from q, which themselves depend on ϕ. On the contrary, Reweighted Wake-Sleep and any other method targeting the forward KL divergence need not use reparameterization, as expectations in the loss function are computed with respect to p (which is constant with respect to ϕ) rather than q_ϕ.

B THE CIRCULAR PATHOLOGY

We provide more detail on the surrogate objective (Equation 4). Let $\perp$ denote a stop gradient operator (Ścibior and Wood, 2021). Note that $\perp$ does not affect evaluation whatsoever; $f(x) = f(\perp x)$ always. However, the $\perp$ notation does alter gradient calculation, i.e. $\perp x$ is always considered as a constant with respect to any gradient operation. Stop gradients are necessary when considering the surrogate objective for wake-phase training because they accurately describe the procedure for implementation of wake-phase training in practice. More precisely, to compute

$$-\sum_{i=1}^{K} w^i \nabla_\phi \log q_\phi(z^i \mid x) \tag{9}$$

one first a) draws $z^{1:K} \overset{iid}{\sim} q_\phi$ and disables gradient tracking with respect to ϕ (e.g., by calling `.sample()` instead of `.rsample()` in PyTorch), and secondly b) ensures the normalized weights do not track gradients either by calling `weights.detach()` or similar. In this fashion, automatic differentiation and back propagation operations function as required, and track correct gradients for wake-phase training. Symbolically, the surrogate objective achieves this same behavior with appropriate stop-gradients, as given below:

$$\mathbb{E}_{z^1, \ldots, z^K \overset{iid}{\sim} q_{\perp \phi}(z|x)} - \sum_{i=1}^{K} (\perp w^i) \log q_\phi(z^i \mid x).$$

We can verify that the Monte Carlo gradient of surrogate objective explicitly matches the Monte Carlo gradient estimator used by wake-phase training. We have

$$\nabla_\phi \mathbb{E}_{z^{1:K} \overset{iid}{\sim} q_{\perp\phi}(z|x)} \sum_{i=1}^{K} (\perp w^i) \log q_\phi(z^i \mid x) = \nabla_\phi \int \left(\sum_{i=1}^{K} (\perp w^i) \log q_\phi(z^i \mid x) \right) q_{\perp\phi}(z^{1:K} \mid x) dz^{1:K} \tag{10}$$

$$= \int \left(\sum_{i=1}^{K} (\perp w^i) \nabla_\phi \log q_\phi(z^i \mid x) \right) q_{\perp\phi}(z^{1:K} \mid x) dz^{1:K} \tag{11}$$

$$= \mathbb{E}_{z^{1:K} \overset{iid}{\sim} q_{\perp\phi}(z|x)} \sum_{i=1}^{K} (\perp w^i) \nabla_\phi \log q_\phi(z^i \mid x) \tag{12}$$

where we just used the basic properties of $\perp$ in the derivation above, namely quantities with $\perp$ are constant with respect to the gradient operator ∇_ϕ. A Monte Carlo draw of the final line is equivalent to the wake-phase SNIS gradient estimator with K particles. Accordingly, wake-phase dynamics in ϕ follow the gradient of this surrogate objective (with the appropriate stop gradient operations in place).

We next prove Proposition 1, reproduced below.

Proposition 1. *Let $\mathcal{L}(q)$ denote the surrogate objective defined above for fixed x and fixed $K \in \mathbb{N}$. Let p denote the posterior $p(z \mid x)$. Then there exists $q(z) \neq p(z \mid x)$ such that $\mathcal{L}(q) < \mathcal{L}(p)$.*

Proof. Fix $c \in \mathbb{R}$. For any $\delta > 0$, consider $\mathcal{L}(q_\delta)$ where $q_\delta \sim \text{Unif}(0, \delta)$. Then, there exists $\delta > 0$ such that $\mathcal{L}(q_\delta) < c$. Because the density of q is given by $\frac{1}{\delta}$, we must have $\mathcal{L}(q_\delta) = -\log(1/\delta)$ because the normalized weights sum to one. For any choice of $c \in \mathbb{R}$, provided $\delta < \exp(c)$, we have $L(q_\delta) < c$ as desired. Clearly, then, for sufficiently small δ we have $\mathcal{L}(q_\delta) < \mathcal{L}(p)$. $\qquad\square$

C SMC-WAKE IMPLEMENTATION

Algorithm 3 below sketches LT-SMC with a fixed temperature schedule. Fixed temperature scheduling is necessary technically for our proofs of asymptotic results to hold. However, either adaptive or non-adapative selection can be used in practice generally: one practical way of exploiting adaptive selection in the fixed temperature framework is to perform exploratory runs to determine a suitable temperature schedule prior to fitting of the encoder network, and thereafter using this fixed schedule throughout.

Algorithm 3: `LT-SMC`

Inputs: Data point x, likelihood function $p(x \mid z)$, prior $p(z)$, temperatures τ.

Initialize particles $z_1^{1:K} \overset{iid}{\sim} p(z)$, unnormalized weights $\tilde{w}_1^{1:K} = 1$, and normalized weights $w_1^{1:K} = \frac{1}{K}$.

for $t = 1$ **to** $T - 1$ **do**

> Resample $z_t^{(i)} \overset{iid}{\sim} \text{Cat}(z_t^{1:K}, w_t^{1:K})$, $i \in [K]$
> Mutate $z_{t+1}^i \sim M(z_t^{(i)}, dz_{t+1})$, $i \in [K]$
> Update $\tilde{w}_{t+1}^i = p(x \mid z_{t+1}^i)^{\tau_{t+1} - \tau_t}$, $i \in [K]$
> Normalize $w_{t+1}^i = \frac{\tilde{w}_{t+1}^i}{\sum_{j=1}^{K} \tilde{w}_{t+1}^j}$, $i \in [K]$

end

Return $z_T^{1:K}, w_T^{1:K}$.

In Algorithm 4, we present the likelihood-tempered SMC procedure that is used in our implementations of SMC-Wake for Section 6[1]. The algorithm is based on Algorithms 17.1 and 17.3 of (Chopin and Papaspiliopoulos, 2020). The algorithm takes a data point x along with a prior density $p(z)$ and a conditional likelihood $p(x \mid z)$ as input. The transition kernel $M(\cdot, dz)$ at stage t leaves invariant the marginal distribution on z with density proportional to $p(z)p(x \mid z)^{\tau_t}$ by constructing a Metropolis-Hastings random-walk kernel. Temperatures are chosen adaptively via Algorithm 5, as discussed below. The primary differences between the implementation

---

[1]Code to reproduce experimental results is publicly available at `https://github.com/declanmcnamara/smc-wake`

and the abbreviated Algorithm 3 are the use of adaptive temperature selection and the optional resampling step, which can reduce variance.

Algorithm 4: `LikelihoodTemperedSMC` (Chopin and Papaspiliopoulos, 2020)

Inputs: Data point x, likelihood function $p(x \mid z)$, prior $p(z)$, invariant kernel $M(\cdot, dz)$, number of particles K, minimum effective sample size $\text{ESS}_{\min}$.

Initialize stage counter $t = 1$, particles $z_1^{1:K} \overset{iid}{\sim} p(z)$, unnormalized weights $\tilde{w}_1^{1:K} = 1$, normalized weights $w_1^{1:K} = \frac{1}{K}$, and $\tau_1 = 0$

for $t = 1$ **to** $T - 1$ **do**

    $\delta \leftarrow \texttt{AdaptiveTempering}(\tau, p(z), p(x \mid z), \{z_t^{1:K}, w_t^{1:K}\})$

    $\tau_{t+1} \leftarrow \tau_t + \delta$

    if $\text{ESS} < \text{ESS}_{\min}$ **then**

        Resample $z_t^{(i)} \overset{iid}{\sim} \text{Cat}(z_t^{1:K}, w_t^{1:K}), \ i \in [K]$

        $\hat{w}_t^{1:K} \leftarrow 1$

    else

        $z_t^{(i)} = z_t^i, \ i \in [K]$

        $\hat{w}_t^{1:K} \leftarrow w_t^{1:K}$

    Mutate $z_{t+1}^i \sim M(z_t^{(i)}, dz_{t+1}), \ i \in [K]$

    Update $\tilde{w}_{t+1}^i = \hat{w}_t^i \cdot p(x \mid z_{t+1}^i)^{\tau_{t+1} - \tau_t}, \ i \in [K]$

    Normalize $w_{t+1}^i = \frac{\tilde{w}_{t+1}^i}{\sum_{j=1}^K \tilde{w}_{t+1}^j}, \ i \in [K]$

end

Return $\hat{P} = \text{Cat}(z_T^{1:K}, w_T^{1:K})$.

Algorithm 5 presents the adaptive tempering algorithm featured in Algorithm 4, reproduced from Chopin and Papaspiliopoulos (2020) §17.2.3. The notation featured in the algorithm comes from writing the target distribution in energy form (Murphy, 2023), i.e. $\gamma(z) \propto \frac{1}{L} p(z) \exp(-V(z))$, which can be easily achieved, for example with the posterior, by taking $V(z) = -\log p(x \mid z)$.

Algorithm 5: `AdaptiveTempering` (Chopin and Papaspiliopoulos, 2020)

Inputs: Current temperature τ, likelihood function $p(x \mid z)$, current particle set $\{z^{1:K}, w^{1:K}\}$, minimum effective sample size $\text{ESS}_{\min}$.

Find δ that solves

$$\frac{\left\{ \sum_{i=1}^K \exp\left\{-\delta V\left(z^i\right)\right\} \right\}^2}{\sum_{i=1}^K \exp\left\{-2\delta V\left(z^i\right)\right\}} = \text{ESS}_{\min},$$

where $V(z_i) = -\log p(x \mid z^i)$.

Return δ.

In practice, we find it unnecessary and computationally expensive to increment the number of SMC sampler runs M for every $x_j \in \mathcal{D}$ each time a gradient step is taken, as is suggested by Algorithm 1. This would require n different LT-SMC runs for every gradient step. As elaborated below in our experimental details, we find it sufficient to increment M for only a single $x_j \in \mathcal{D}$ every gradient step, or perhaps even less often, e.g. every ten gradient steps.

One consequence of this computation-saving scheme is that the number of LT-SMC runs M_j is different for each observation x_j. This complicates vectorized operations when averaging over mini-batches of $x_j \in \mathcal{D}$. To resolve this, our implementation of the gradient estimators $\hat{\nabla}^{(a)}$ and $\hat{\nabla}^{(b)}$ uses a slight alteration of the estimators in Table 2. From the equalities

$$\hat{\nabla}^{(a)} = \sum_{m=1}^M \frac{\hat{C}_{(m)}}{\sum_{m=1}^M \hat{C}_{(m)}} \left(\sum_{k=1}^K w_{(m)}^k f(z_{(m)}^k) \right)$$

and

$$\hat{\nabla}^{(b)} = \sum_{m=1}^{M} \frac{\hat{C}_{(m)}}{\sum_{m=1}^{M} \hat{C}_{(m)}} f(z_{(m)}), \text{ where } z_{(m)} \sim \text{Cat}(w_{(m)}^{1:K}, z_{(m)}^{1:K}),$$

we see that these resemble importance sampling estimators, with normalized weights $\frac{\hat{C}_{(m)}}{\sum_{m=1}^{M} \hat{C}_{(m)}}$, $m = 1, \ldots, M$. To perform vectorized operations, we thus pick a value M^* and for each observation x_j in a minibatch, we sample (with replacement) M^* samplers from the M_j available, each with probability $\frac{\hat{C}_{(m)}}{\sum_{m=1}^{M} \hat{C}_{(m)}}$. The resulting M^* SMC samplers are then used to construct each of these two gradient estimators. This results in consistent dimensions for vectorizing minibatch computations of gradients. Of course, if LT-SMC runs are cheap, the procedure can be performed naively as in Algorithm 1 where n LT-SMC runs are performed at every gradient step. One additional alternative that also saves computation is to increment M less frequently, e.g., running LT-SMC for all n points but only every 10 gradient steps.

In practice, the estimates $\hat{C}_{(m)}$ of the normalization constant may have large variance, especially in high-dimensional problems. This can introduce sample deficiency into the SMC-Wake algorithm, as all weight may concentrate on one $\hat{C}_{(m)}$, and thus effectively particles from a single sampler are used. In implementation, this difficulty can be alleviated by considering only a random subset of SMC samplers (and their estimates of the normalizing constants) of size M' at each gradient step, with $M' < M$. This ensures that samples from many different samplers are used to estimate gradients and retains asymptotic guarantees as long as the size of M' is iteratively increased throughout the procedure.

D ASYMPTOTIC ANALYSIS

In this section, we prove asymptotic properties of the gradient estimators $\hat{\nabla}^{(a)}, \hat{\nabla}^{(b)}$, and $\hat{\nabla}^{(c)}$ and also discuss the memory requirements of each.

Following Naesseth, Lindsten, and Schön (2019), we let $\mathbf{u}$ denote all random variables generated by a run of likelihood-tempered SMC (Algorithm 4) with a fixed temperature schedule $\tau_1, \ldots, \tau_T$. We have the following result from Proposition 7.4.1 of Del Moral (2004), also given in Theorem 2.1 of Naesseth, Lindsten, and Schön (2019), whose notation we follow more closely.

Proposition. *For any integrable test function f and any run of Algorithm 4, we have*

$$\mathbb{E}_{\mathbf{u}}\left[\prod_{s=1}^{T} \left(\frac{1}{K} \sum_{k=1}^{K} \tilde{w}_s^k \right) \sum_{k=1}^{K} w_T^k f(z_T^k) \right] = C \mathbb{E}_{p(z|x)} f(z) \tag{13}$$

with $C = p(x)$, where $\tilde{w}_s$ denotes unnormalized weights and w_s denotes normalized weights at any stage s of the SMC sampler.

The above result The result holds for a general sequence of unnormalized target distributions $\tilde{\gamma}_1, \ldots \tilde{\gamma}_T$ and any general SMC sampling algorithm, but we have specialized to the likelihood-tempered case in the above where these unnormalized targets are given by $p(z)p(x \mid z)^{\tau_1}, \ldots, p(z)p(x \mid z)^{\tau_T}$ for $0 = \tau_1 < \cdots < \tau_T = 1$. The unbiasedness of the estimator $\hat{C}$ (Equation 5) for C results from the above proposition with the special case $f(z) = 1$. We now wish to show the consistency or asymptotic unbiasedness of the estimators below, reproduced from Table 2.

$$\hat{\nabla}^{(a)} = \frac{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)} \left(\sum_{k=1}^{K} w_{(m)}^{k} f(z_{(m)}^{k}) \right)}{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}}$$

$$\hat{\nabla}^{(b)} = \frac{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)} f(z_{(m)})}{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}}, \text{ where } z_{(m)} \sim \text{Cat}(w_{(m)}^{1:K}, z_{(m)}^{1:K})$$

$$\hat{\nabla}^{(c)} = \frac{\hat{C}_{(M)} \left(\sum_{k=1}^{K} w_{(M)}^{k} f(z_{(M)}^{k}) \right)}{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}}$$

We can now consider each of the three estimators in turn, as we prove Proposition 3, reproduced below.

Proposition 3. *For an observation x_j, suppose that $f(z) = -\nabla_\phi \log q_\phi(z \mid x_j)$ is a bounded and measurable function for any $\phi \in \Phi$. For $m = 1, \ldots, M$, let the random variables $\hat{C}_{(m)}$, $w_{(m)}^{1:K}$, and $z_{(m)}^{1:K}$ result from an independent run of LT-SMC (Algorithm 3) with a fixed temperature schedule. Then, the gradient estimators $\hat{\nabla}_j^{(a)}$ and $\hat{\nabla}_j^{(b)}$ are strongly consistent for $\mathbb{E}_{p(z|x_j)} f(z)$ as $M \to \infty$ and the estimator $\hat{\nabla}_j^{(c)}$ is asymptotically unbiased as $M \to \infty$.*

Proof. We first note that by the strong law of large numbers, the denominator $\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)} \overset{a.s.}{\to} C$. This holds immediately because each SMC sampler is generated independently; therefore, the random variables $\mathbf{u}$ generated by each run are independent. In particular, the random variables $\hat{C}_{(m)}$ are independent for all m, which implies the result. Below, we discuss each gradient estimator in turn.

(a) By Equation (13) and the definition of $\hat{C}_{(m)}$ (cf. Equation (5)), the numerator converges almost surely to $C\mathbb{E}_{p(z|x)} f(z)$ by the strong law of large numbers by similar logic to the above. Accordingly, by the continuous mapping theorem the quotient

$$\hat{\nabla}^{(a)} = \frac{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)} \left(\sum_{k=1}^{K} w_{(m)}^{k} f(z_{(m)}^{k}) \right)}{\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}}$$

converges almost surely to $\mathbb{E}_{p(z|x)} f(z)$ and is therefore strongly consistent. An important detail that allows application of the continuous mapping theorem is that the random variables $\hat{C}_{(m)}$ are strictly positive, i.e., $\hat{C}_{(m)} > 0$, avoiding division by zero issues.

(b) First formally define the random variables conditional on $\mathbf{u}_{(m)}$ for the mth sampler by

$$\hat{C}_{(m)} \mid \mathbf{u}_{(m)} = \delta(\hat{C}_{(m)}) \tag{14}$$

$$z_{(m)} \mid \mathbf{u}_{(m)} \sim \text{Cat}(w_{(m)}^{1:K}, z_{(m)}^{1:K}). \tag{15}$$

In other words, conditional on $\mathbf{u}_{(m)}$, the random variable $\hat{C}_{(m)}$ is a point mass at the quantity defined in Equation 5 and the random variable $z_{(m)}$ is a draw from the final particle distribution at stage T for the mth sampler. Then we have by the law of iterated expectation for any m

$$\mathbb{E}\left[\hat{C}_{(m)} f(z_{(m)}) \right] = \mathbb{E}_{\mathbf{u}_{(m)}} \left[\mathbb{E}\left[\hat{C}_{(m)} f(z_{(m)}) \mid \mathbf{u}_{(m)} \right] \right]$$

$$= \mathbb{E}_{\mathbf{u}_{(m)}} \left[\hat{C}_{(m)} \mathbb{E}\left[f(z_{(m)}) \mid \mathbf{u}_{(m)} \right] \right]$$

$$= \mathbb{E}_{\mathbf{u}_{(m)}} \left[\hat{C}_{(m)} \sum_{k=1}^{K} w_{(m)}^{k} f(z_{(m)}^{k}) \right]$$

$$= C\mathbb{E}_{p(z|x)} f(z)$$

by Equation (13), and so again by the strong law of large numbers, the numerator of (b) converges almost surely to $C\mathbb{E}_{p(z|x)}f(z)$. By the same reasoning as above, the quotient that defines the ratio estimator $\hat{\nabla}^{(b)}$ converges almost surely to $\mathbb{E}_{p(z|x)}f(z)$, and so is strongly consistent for $\mathbb{E}_{p(z|x)}f(z)$.

(c) The estimator $\hat{\nabla}^{(c)}$ is asymptotically unbiased, but not consistent because the variance does not tend to zero. Using the fact that $\frac{1}{M}\sum_{m=1}^{M}\hat{C}_{(m)} \overset{a.s.}{\to} C$ and hence $\frac{1}{M}\sum_{m=1}^{M}\hat{C}_{(m)} \overset{p}{\to} C$, an application of Slutsky's theorem yields that the distribution of this estimator tends to that of the random variable $\frac{\hat{C}_{(m)}}{C}\sum_{k=1}^{K}w_{(m)}^{k}f(z_{(m)}^{k})$ which clearly has expectation $\mathbb{E}_{p(z|x)}f(z)$ by Equation (13). This suffices to show asymptotic unbiasedness (cf. Lehmann and Casella (1998) pg. 438, Definition 2.1).

$\square$

The gradient estimators $\hat{\nabla}^{(a)}$ and $\hat{\nabla}^{(b)}$ may be viewed as performing a form of "meta importance sampling" or "distributed importance sampling" (Naesseth, Lindsten, and Schön, 2019), where the "weights" are the evidence estimates $\hat{C}_{(m)}$ across M different samplers, properly normalized to sum to unity by the denominator. Keeping in mind that $\hat{C}_{(m)}$ estimates the evidence $p(x)$ for a given observation x, higher weights thus correspond to particle sets for which the estimator of the evidence is higher—this provides an intuitive understanding of which SMC samplers receive higher weight. The unnormalized weights are computed based on $p(z,x)$, so higher weights indicate particles that better explain the observed data.

The numerator of the gradient estimator $\hat{\nabla}^{(c)}$ has variance that does not tend to zero, and therefore this cannot result in a consistent estimator. However, the variance of this estimator can be reduced at the standard Monte Carlo rate by averaging over some finite number of SMC sampler runs $L < M$ using a rolling window approach, e.g., averaging over the most recent 10 SMC sampler runs.

We conclude with a brief discussion of the memory requirements of the gradient estimators. First, observe that the quantity

$$\frac{1}{M}\sum_{m=1}^{M}\hat{C}_{(m)} = \frac{M-1}{M}\left(\frac{1}{M-1}\sum_{m=1}^{M-1}\hat{C}_{(m)}\right) + \frac{1}{M}\left(\hat{C}_{(M)}\right)$$

and so this average can be maintained with only $O(1)$ memory as $M \to \infty$. Considering the three estimators in turn, for each data point x we thus immediately have that $\hat{\nabla}^{(a)}$ requires $O(MK)$ memory by maintaining a K-particle set for each of the M sampler runs. By resampling a single particle and applying the law of iterated expectation, the estimator $\hat{\nabla}^{(b)}$ just requires $O(M)$ memory for M sampler runs. The estimator $\hat{\nabla}^{(c)}$ is the cheapest—it only uses the K-particle set from the most recent sampler, i.e. the Mth, to compute the numerator, and so thus requires $O(K)$ memory. Of course, to repeatedly construct the samplers for increasing M, the SMC-Wake procedure also requires sufficient memory to run LT-SMC itself which is $O(K)$ for every run.

E   TWO MOONS

We generate 100 points $x_1,\ldots,x_{100}$ independently from the generative model in Section 6.1. The encoder architecture is a neural spline flow (NSF) (Durkan et al., 2019), adapted from the `sbi` Python library (Tejero-Cantero et al., 2020). We use the default NSF settings as the problem is low-dimensional enough to avoid embedding nets, etc. For all experiments, some tensor operations in PyTorch were performed using NVIDIA GeForce RTX 2080 Ti graphical processing units (GPUs).

All methods were trained using a batch size of 16, with a learning rate of 0.0001 for 50,000 gradient steps. We use a large set of $K = 1000$ particles for all the methods. SMC-Wake reruns one SMC sampler (corresponding to one observation) at random every 10 gradient steps, corresponding to about $M = 50$ runs of the SMC sampler per point throughout the fitting procedure. Even relatively small values for M such as those used here yield a significant improvement over wake-phase training. For all SMC runs, the proposal distribution was a 5-step Metropolis-Hastings random-walk kernel with Gaussian steps and $\sigma^2 = 0.1^2$ as its variance. For wake-phase updates, instances where the self-normalized importance sampling estimate of the gradient was undefined were ignored and re-tried.

In the same manner as in Figure 1, we plot the variational approximations for several other points in the training set in Figure 5, Figure 6, and Figure 7. We show results based on training via the three gradient estimators

$\hat{\nabla}^{(a)}, \hat{\nabla}^{(b)}$, and $\hat{\nabla}^{(c)}$, denoted as SMC-Wake(a)-(c), respectively.
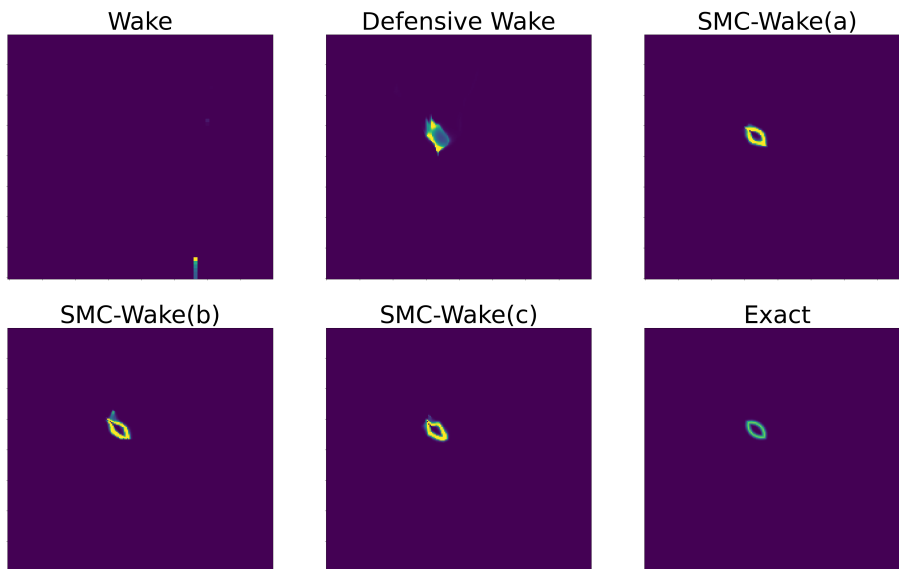

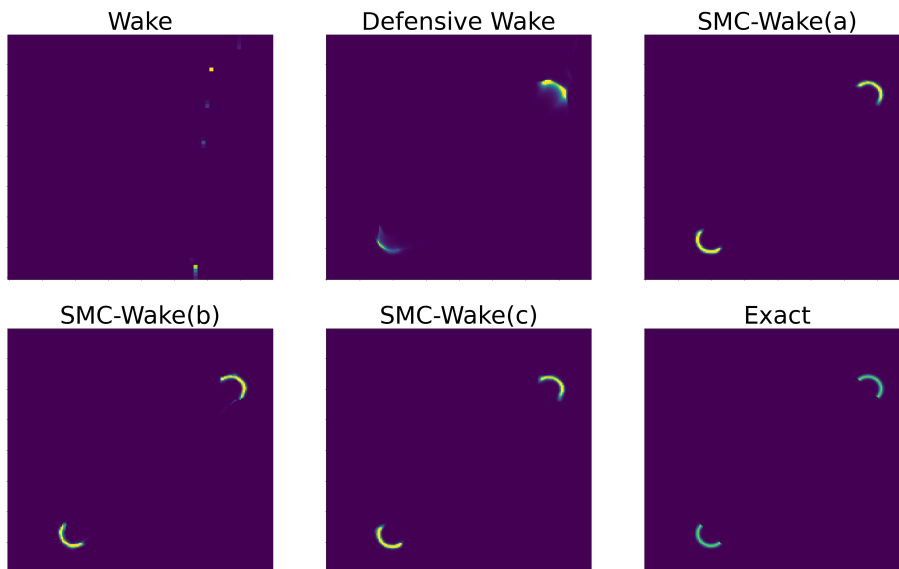Figure 5: Comparison of two-moons variational posteriors.

Figure 6: Comparison of two-moons variational posteriors.

Clearly, wake-phase training concentrates mass. We show that this concentration of mass occurs very rapidly, in this case within one thousand gradient steps, in Figure 8.

F MNIST

Amortized inference is often used to simultaneously train a model (decoder) and a posterior approximator (encoder). In this setting, latent representations can be considered as compressed versions of data, from which reconstructions should nevertheless be accurate (Kingma and Welling, 2019; Burda, Grosse, and Salakhutdinov, 2016). Given 1000 MNIST digits x and labels ℓ denoted by $\{x_i, \ell_i\}_{i=1}^{1000}$, we simultaneously fit an encoder $q_\phi(z \mid x, \ell)$ and a decoder $p_\psi(x \mid \ell, z)$.

Each class ℓ is modeled by a sigmoidal belief network (SBN) (Saul, Jaakkola, and Jordan, 1996), and thus the
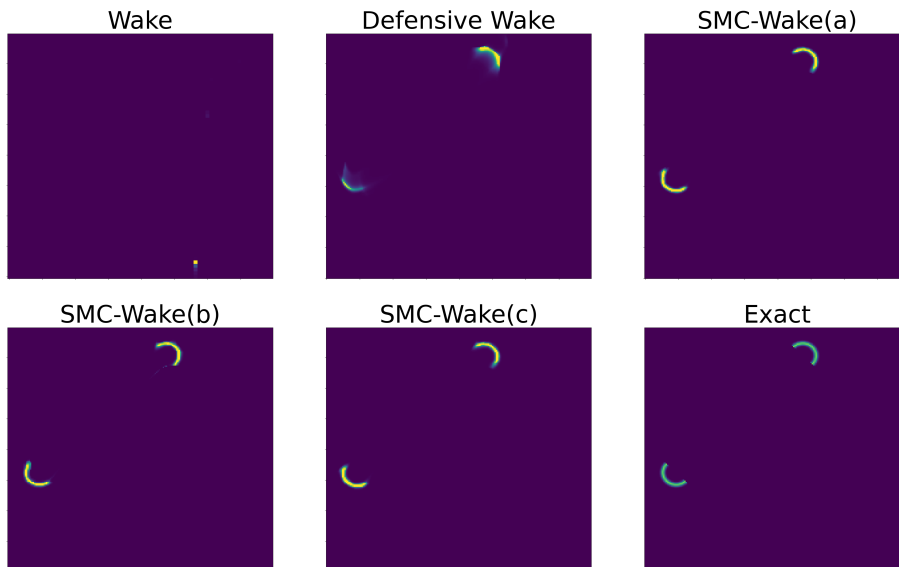
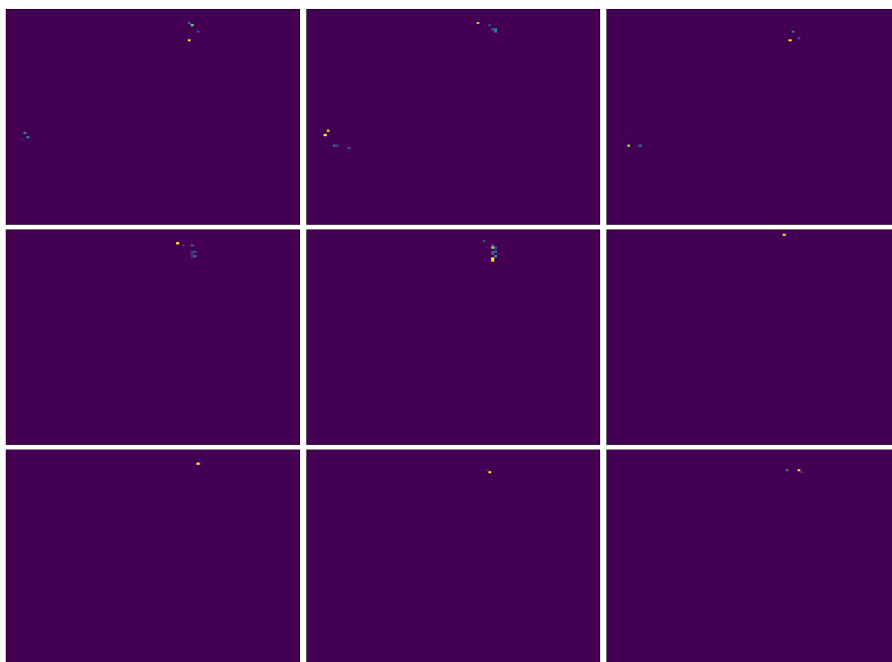Figure 7: Comparison of two-moons variational posteriors.

Figure 8: Visualizations of the importance sampling particle approximations with $K = 1000$ for wake-phase training. The nine panels are sampled from the inference network at every hundred gradient steps, so at step $100, 200, \ldots, 900$. We see that even within the first one thousand gradient steps, mass is concentrated.

model is given by

$$p_\psi(x \mid \ell, z) \sim \mathcal{N}(\sigma(W_\ell z + b_\ell), \tau^2 I_d) \tag{16}$$

where $\sigma(\cdot)$ denotes the sigmoid function. The observations x are normalized to have intensities in $[0, 1]$, and therefore the sigmoidal transformation describes the data well. Setting $\tau = 0.01$ results in a highly peaked likelihood.

We take $p = 16$ as the latent dimension, and thus $W_\ell \in \mathbb{R}^{p \times 784}$ and $b_\ell \in \mathbb{R}^{784}$ for all ℓ. For the inference network, we use a simple amortized diagonal Gaussian distribution $q_\phi(z \mid x) = \mathcal{N}(\mu(x), \text{diag}(\sigma(x)))$ whose parameters are

the output of a three-layer dense network with ReLU activation and hidden dimension of 64. We use logarithmic scales for numerical stability. The prior $p(z)$ is given by a standard multivariate Gaussian distribution. We used a learning rate of 0.0005 to fit both the decoder and encoder. All methods alternated an update to the model parameters followed by an update to the encoder parameters, with 50,000 steps each.

The model parameters ϕ are fit to maximize the IWBO while the encoder is fit by wake-phase training and SMC-Wake, using the gradient estimator $\hat{\nabla}^{(b)}$. In addition to Figure 2, below we plot several similar plots for other digits in Figure 9, Figure 10, and Figure 11.

The procedure is implemented by alternating gradient updates to θ and to ϕ, respectively. This can be done safely without losing the asymptotic guarantees of the estimators $\hat{\nabla}^{(a)}, \hat{\nabla}^{(b)}$, and $\hat{\nabla}^{(c)}$ provided that the model parameters θ converge (stop changing) to some θ^* by some finite $M^* < \infty$. In this case, as $M \to \infty$ the average of normalization constant estimates still tends to the normalization constant for the correct model (i.e. with parameters θ^*). This follows directly from Kolmogorov's generalized LLN for independent (but not identically distributed) r.v.'s: $\left(\frac{1}{M} \sum_{m=1}^{M} \hat{C}_{(m)}\right) - \left(\frac{1}{M} \sum_{m=1}^{M} \mu_{(m)}\right) \overset{a.s.}{\to} 0$, where $\mu_{(m)} = \mathbb{E}\hat{C}_{(m)}$. If $\theta = \theta^*$ for all $M > M^*$, then $\frac{1}{M} \sum_{m=1}^{M} \mu_{(m)} \to C$, the normalization constant for the final model with parameters θ^*. This is true even without discarding early normalization constant estimates, although discarding may be desirable in practice in some applications. Intuitively, the effect of finitely many normalization constant estimates for "wrong" models diminishes as we aggregate infinitely many estimates for the "correct" model.
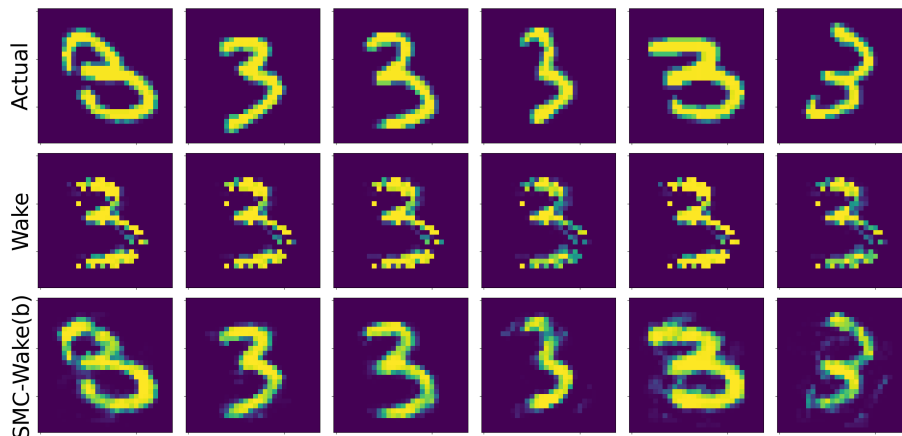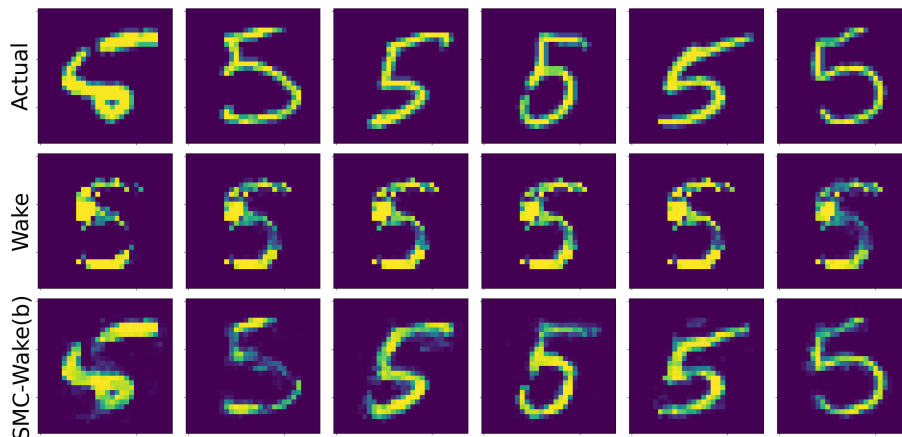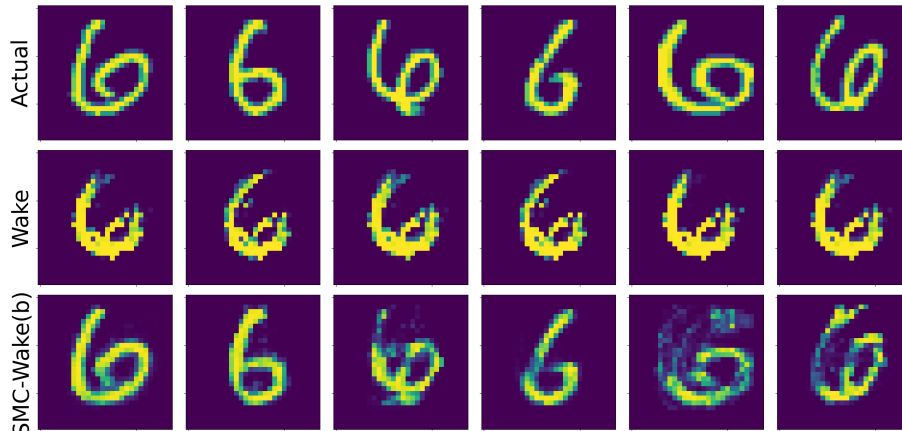

Figure 9

Figure 10

Figure 11

G   GAUSSIAN LINEAR MODEL

For the model

$$z \sim \mathcal{N}(0, \sigma^2 I_p)$$
$$x \mid z \sim \mathcal{N}(Az, \tau^2 I_d)$$

the posterior distribution is proportional to

$$\exp\left(-\frac{1}{2}\left[\frac{z^\top z}{\sigma^2} + \frac{(x - Az)^\top(x - Az)}{\tau^2}\right]\right)$$
$$\propto \exp\left(-\frac{1}{2}\left[z^\top\left(\frac{I_d}{\sigma^2} + \frac{A^\top A}{\tau^2}\right)z - 2\frac{x^\top Az}{\tau^2}\right]\right)$$
$$= \exp\left(-\frac{1}{2}\left[z^\top M z - 2b^\top z\right]\right)$$
$$= \exp\left(-\frac{1}{2}\left[\left(z - M^{-1}b\right)^T M\left(z - M^{-1}b\right) - b^T M^{-1}b\right]\right)$$
$$\propto \exp\left(-\frac{1}{2}\left[\left(z - M^{-1}b\right)^T M\left(z - M^{-1}b\right)\right]\right).$$

by a completing the square argument. Therefore, the posterior distribution is $\mathcal{N}(M^{-1}b, M^{-1})$ where

$$M = \frac{I_d}{\sigma^2} + \frac{A^\top A}{\tau^2} \in \mathbb{R}^{d \times d}$$

and

$$b = \frac{A^\top x}{\tau^2} \in \mathbb{R}^d.$$

The experimental details for the nested MCMC examples are as follows: as stated, we use dimensions $p = 50$ and $d = 100$, as well as $\sigma = \tau = 1$, and the design matrix $A \in \mathbb{R}^{d \times p}$ is fixed. The encoder parameterizes a multivariate Gaussian distribution in p dimensions. The variational family $q_\phi(z \mid x) \sim \mathcal{N}(\mu, LL^\top + \epsilon I)$ is flexible enough to approximate the exact posterior. We add the additional ϵI term with $\epsilon = 0.0001$ for numerical stability, but this does not overly constrain the variational family in this case, as the true posterior covariance has eigenvalues at least this large. The variational parameters given any observation are the outputs of a dense neural network with 4 dense layers each of dimension 64, with ReLU activation. For each run of LT-SMC, each stage t performs the mutation step using 100 steps of a Metropolis-Hastings random-walk kernel with $\sigma = 0.01$, each kernel leaving
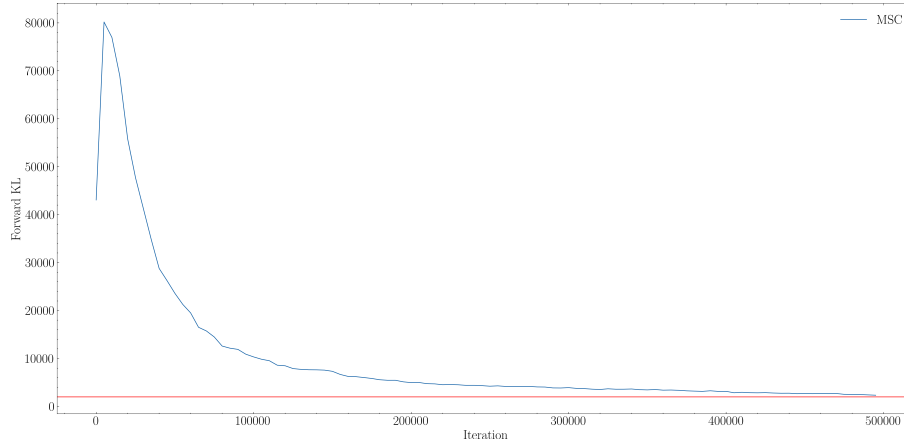
Figure 12: The forward KL divergence averaged across all n observations, plotted throughout the 500,000 gradient steps of fitting by Markovian score climbing (MSC). At convergence, the minimal average forward KL achieved seems to be approximately 5000 or so.

the stage t target γ_t invariant.

We fit the encoder using the SMC-PIMH-Wake objective using a batch size of 32 data points per gradient step (out of $n = 50$ possible in the given dataset), with learning rate 0.0001 for 40,000 gradient steps. The number of particles used is $K = 100$. After each gradient step, we randomly pick just one $x_j \in \mathcal{D}$, rerun LT-SMC for it, and take a Metropolis-Hastings step within the outer PIMH loop for that particular x_j. SMC samplers are more expensive to run than SNIS, and we found that rerunning the SMC samplers for each point in $\mathcal{D}$ at each gradient step was unnecessary to achieve good results. Across 40,000 gradient steps, this averages to about 800 MCMC steps for each $x_j \in \mathcal{D}$.

We compare to Markovian Score Climbing (MSC), using q_ϕ as the proposal distribution within a CIS kernel for MCMC. As IS is more lightweight than SMC, we perform an MCMC step for each of the $n = 50$ Markov chains (one for each datapoint) every time its corresponding observation x_j is in a minibatch for a gradient step. As the batch size is 32 in our implementation, this corresponds to over 250,000 MCMC steps for each of the 50 chains over a total of 500,000 gradient steps. All other hyperparameters for the fitting procedure are the same as those for SMC-PIMH-Wake, including the number of particles K and the learning rate.

In addition to the results of Table 3, in Figure 12 we plot the average forward KL divergence (recorded every 5,000 gradient steps) throughout the MSC fitting procedure. These results suggest that the fitting procedure for q_ϕ is near convergence after 500,000 steps and has converged to an encoder network with an average forward KL of around 2,000 (the red horizontal line in Figure 12 is at 2,000). SMC-PIMH-Wake, on the other hand, fits an encoder with a significantly lower forward KL.

For the example comparing many samplers and a single large sampler, we have the same setup and architectures as above, except we only use 10 data points for ease, as computing the $K = 10,000$ particle sampler for each is computationally expensive. To make the problem more challenging, we also use a different fixed matrix B instead of A that results in a more poorly conditioned posterior, i.e. one that cannot be described exactly by the variational family, and we use a Metropolis-Hastings random-walk kernel with only 10 steps and with a larger σ (0.1) to ensure adequate exploration of the space. For each point, a single SMC sampler is run with $K = 10,000$ particles, along with 100 different runs with $K = 100$ particles. All runs use a 10-step MHRW transition kernel at each stage similar to that described above.

We fit the encoder network using a naive form of the SMC-Wake gradient estimator for this setting, as the $K = 10,000$ case only has $M = 1$ sampler to work with. For the case of $K = 10,000$ particles, the gradient is thus estimated at every step using the same sets of particles $z^{1:K} \in \mathbb{R}^{10,000}$ and weights $w^{1:K} \in \mathbb{R}^{10,000}$, as the sampler is run only once. For the $K = 100$ case, at each gradient step we select one of the $M = 100$ samplers at random, and use the vectors $z^{1:K}, w^{1:K} \in \mathbb{R}^{100}$ to estimate the gradient. This can be thought of as the most naive form of SMC-Wake, as the samplers have even weight and evidence estimates $\hat{C}^{(m)}$ are not used to weight the samplers. Alternatively, this can be considered as the lightweight form of SMC-Wake with estimator $\hat{\nabla}^{(a)}$

focused on sample replenishment with trivial $M' = 1$ (see Appendix C). As Figure 3 illustrates, the $K = 10,000$ sampler is inefficient: because the LT-SMC procedure is difficult to tune in this large dimension, this sampler has ESS ≈ 1, so the larger particle budget does not help much compared to a larger number of SMC samplers.

H   GALAXY SED EMULATOR

The Probabilistic Value-Added Bright Galaxy Survey simulates synthetic spectra for the Dark Energy Spectroscopic Instrument (DESI) Bright Galaxy Survey. Its code is freely available at `https://github.com/changhoonhahn/provabgs` (Hahn et al., 2022; DESI Collaboration, B. Abareshi et al., 2022), and used with permission of the MIT License.

The simulator uses 12 parameters to produce synthetic spectra. We describe these below, reproduced from Hahn et al. (2022).

Name	Description	Prior
$\log M_*$	log galaxy stellar mass	uniform over $[7, 12.5]$
$\beta_1, \beta_2, \beta_3, \beta_4$	NMF basis coefficients for SFH	Dirichlet prior
fburst	fraction of total stellar mass formed in starburst event	uniform over $[0, 1]$
t_{burst}	time of starburst event	uniform over $[10\mathrm{Myr}, 13.2\mathrm{Gyr}]$
γ_1, γ_2	NMF basis coefficients for ZH	log uniform over $[4.5 \times 10^{-5}, 1.5 \times 10^{-2}]$
τ_{BC}	Birth cloud optical depth	uniform over $[0, 3]$
τ_{ISM}	diffuse-dust optical depth	uniform over $[0, 3]$
n_{dust}	Calretti (2001) dust index	uniform over $[-2, 1]$

The simulator is computationally expensive, even using the authors' emulated version; we train our own lightweight neural network emulator of PROVABGS. Because the spectra simulated by PROVABGS are of vastly different magnitudes, we work on the modified problem of normalized spectra, which can be emulated more easily. To train the emulator, we generate batches of synthetic spectra using the prescribed priors, but with fixed $\log M_* = 10.5$, and normalize each to integrate to one. We also resample onto a grid of wavelengths between 3000 and 10,000 Angstrom, with width 5 Angstrom. The emulator is trained with 2000 batches, each consisting of 1000 simulated (θ, x) pairs. The criterion optimized is the mean-squared error (MSE) with learning rate 0.001. After training, the emulator produces draws (Figure 13) that look similar to the (normalized) PROVABGS outputs. Because we fix the stellar mass parameter, our emulator uses 11 input parameters.

The uniform and Dirichlet priors present problems for our likelihood-tempered SMC procedure (Algorithm 4). As we use Metropolis-Hastings random-walk transition kernels, the boundaries of the support complicate using symmetric Gaussian proposals; random-walks on the simplex are similarly challenging. To resolve this issue, we operate on a transformed space of unconstrained random variables $\tilde{\theta}$ that are bijective transformations of the random variables θ of interest. Bijections from any uniform range (a, b) to $\mathbb{R}$ can be constructed using scaled logit transformations, and the inverses computed by scaled sigmoidal transforms. Care must be taken with the Dirichlet parameters; these 4 random variables are first transformed into a 3-dimensional space using the warped manifold transformation of (Betancourt, 2012). The resulting random variables reside in $(0, 1)$; we can apply a subsequent logit transform from here as above. We perform SMC-Wake to learn distributions on the unconstrained space before applying the transformations to the constrained space for evaluation in Figure 4. Overall, SMC thus samples on a 10-dimensional unconstrained space before transforming back into an 11-dimensional unconstrained space.

The data in this example are very high-dimensional; even with resampling onto a coarser wavelength grid, spectra are still 1400-dimensional. Adaptive tempering provides one way of efficiently dealing with the likelihood function of such data in a principled way. Most data choose a low initial temperature (on the order of 10^{-5} or so) to slowly introduce the likelihood function and prevent overpowering of the prior; manual selection of suitable temperatures in this case may prove more difficult.

We generate 100 observations from our emulator and priors described above and add noise proportional to the signal in each wavelength bin. A neural spline flow (NSF) is used as the amortized variational posterior (Durkan et al., 2019) on $\tilde{\theta}$. For all methods, we perform 25,000 gradient steps with learning rate 0.0001 using a mini-batch size of 32.

We fit the encoder using SMC-Wake with gradient estimator $\hat{\nabla}^{(a)}$, taking $K = 100$ for all LT-SMC runs. The
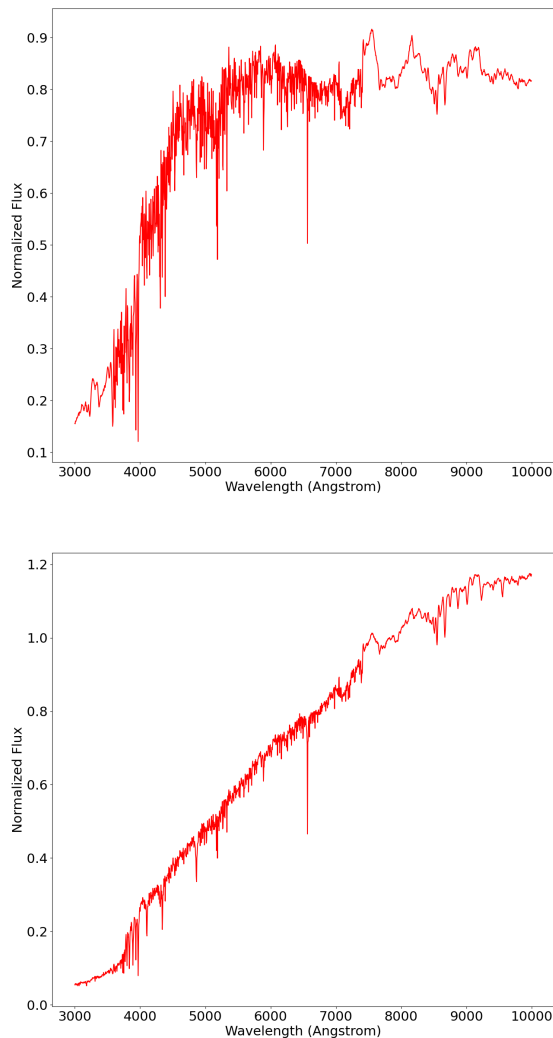
Figure 13: Two examples of draws from our neural network emulator of PROVABGS.

mutation kernel in LT-SMC is a 50-step Metropolis-Hastings kernel with noise parameter $\sigma^2 = 0.1^2$. SMC-Wake performs well with a minimal number of LT-SMC runs—we only rerun LT-SMC for a single point (chosen at random) every 50 gradient steps. Across the 25,000 gradient steps, this averages to only about $M = 5$ LT-SMC runs for each of the $n = 100$ observations. In Figure 4 we only showed posteriors for 4 out of 11 parameters for one of the one hundred observations due to space constraints; here we show all 11 in Figure 14 for this same point. We also performed training by wake-phase updates for this example, but wake-phase training exhibits a troubling numerical stability issue that impairs training, whereby some parameter values sampled from $q_\phi(\theta \mid x)$ produce spectra with NaN values when fed through the emulator. If this is the case for all K particles sampled from q_ϕ in importance sampling, then the wake-phase gradient is undefined. SMC-Wake does not suffer from this issue due to its use of LT-SMC.
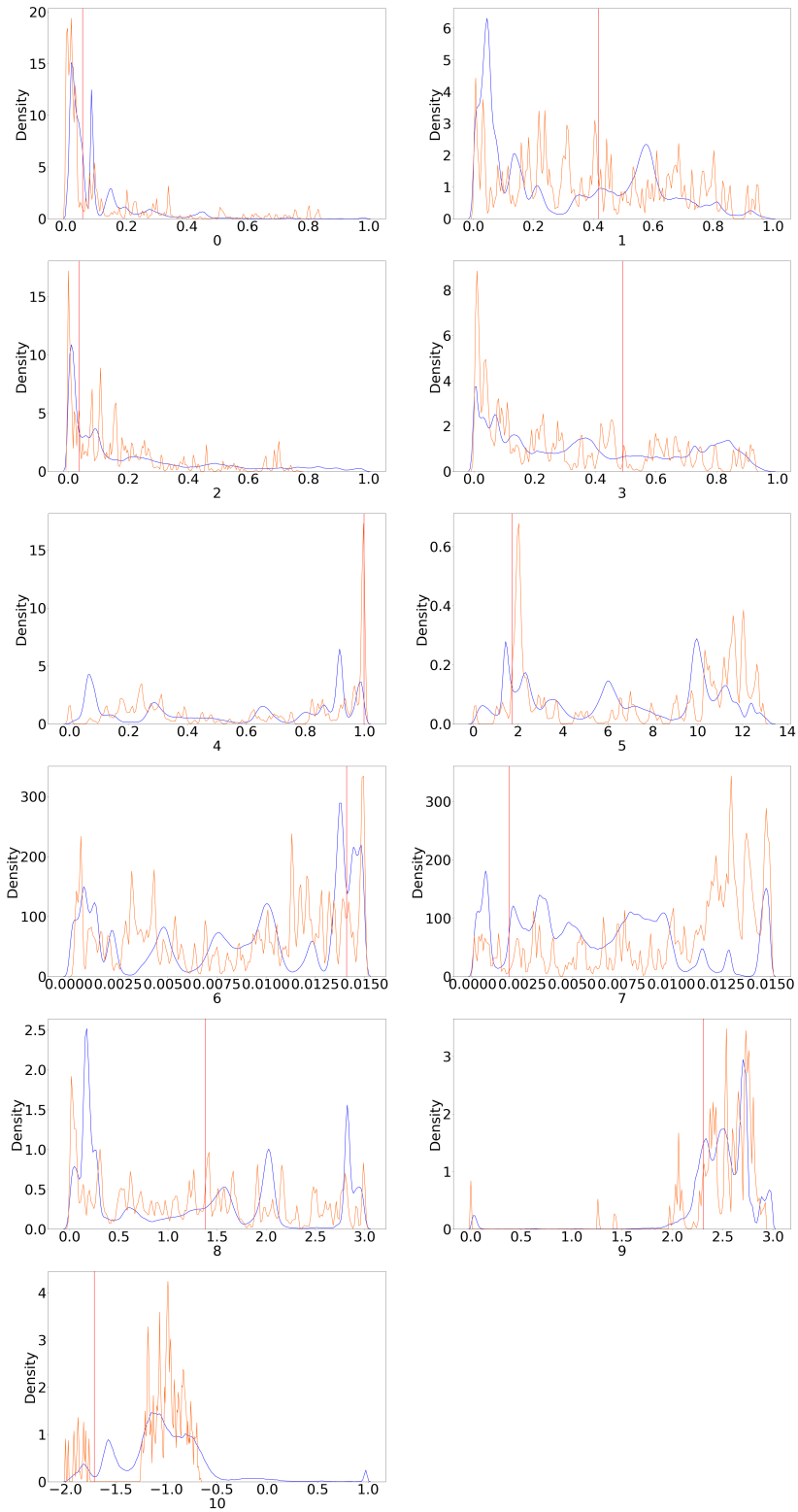
Figure 14: Comparison of SMC-Wake posteriors (blue) to those obtained by MCMC (orange) for a single observation.