

---

# Amortised Learning by Wake-Sleep

---

Li K. Wenliang<sup>1</sup> Theodore Moskovitz<sup>1</sup> Heishiro Kanagawa<sup>1</sup> Maneesh Sahani<sup>1</sup>

## Abstract

Models that employ latent variables to capture structure in observed data lie at the heart of many current unsupervised learning algorithms, but exact maximum-likelihood learning for powerful and flexible latent-variable models is almost always intractable. Thus, state-of-the-art approaches either abandon the maximum-likelihood framework entirely, or else rely on a variety of variational approximations to the posterior distribution over the latents. Here, we propose an alternative approach that we call amortised learning. Rather than computing an approximation to the posterior over latents, we use a wake-sleep Monte-Carlo strategy to learn a function that directly estimates the maximum-likelihood parameter updates. Amortised learning is possible whenever samples of latents and observations can be simulated from the generative model, treating the model as a “black box”. We demonstrate its effectiveness on a wide range of complex models, including those with latents that are discrete or supported on non-Euclidean spaces.

## 1. Introduction

Many problems in machine learning, particularly unsupervised learning, can be approached by fitting flexible parametric probabilistic models to data, often based on “local” latent variables whose number scales with the number of observations. Once the optimal parameters are found, the resulting model may be used to synthesize samples, detect outliers, or relate observations to a latent “representation”. The quality of all of these operations depends on the appropriateness of the model class chosen and the optimality of the identified parameters.

Although many fitting objectives have been explored in the literature, maximum-likelihood (ML) estimation remains prominent and comes with attractive theoretical properties

<sup>1</sup>Gatsby Computational Neuroscinece Unit. Correspondence to: Li K. Wenliang <kevinli@gatsby.ucl.ac.uk>.

including consistency and asymptotic efficiency (Newey & McFadden, 1994). A challenge, however, is that analytic evaluation of the likelihoods of rich, flexible latent variable models is usually intractable. The Expectation-Maximisation (EM) algorithm (Dempster et al., 1977) offers one route to ML estimation in such circumstances, but it in turn requires explicit calculation of (expected values under) the posterior distribution over latent variables, which proves also to be intractable in most cases of interest. Consequently, state-of-the-art ML-related methods almost always rely on approximations, particularly in large-data settings.

Denote the joint distribution of a generative model as  $p_{\theta}(z, x)$  where  $z$  is latent and  $x$  is observed, and  $\theta$  is the vector of parameters. EM breaks the ML problem into an iteration of two sub-problems. Given parameters  $\theta_t$  on the  $t$ th iteration, first find the posterior  $p_{\theta_t}(z|x)$ ; then maximize a lower bound to the likelihood that depends on this posterior to obtain  $\theta_{t+1}$ . This bound is tight when computed using the correct posterior, ensuring convergence to a local mode of the likelihood.

The intractability of  $p_{\theta}(z|x)$  forces some combination of Monte-Carlo estimation and the use of a tractable parametric approximating family which we call  $q(z|x)$  (Bishop, 2006). To avoid repeating the expensive optimisation in finding  $q(z|x)$  for each  $x$ , amortised inference trains an encoding or recognition model, with parameters  $\phi$ , to map from any  $x$  directly to an approximate posterior  $q_{\phi}(z|x)$ . Examples of amortised inference models include the Helmholtz machine (Dayan et al., 1995; Hinton et al., 1995) trained by the wake-sleep algorithm; and the variational auto-encoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) trained using reparameterisation gradient methods. With considerable effort on improving variational inference (reviewed in (Zhang et al., 2018)), complex and flexible generative models have been trained on large, high-dimensional datasets.

However, approximate variational inference poses at least three challenges. First, the parametric form of the approximate posterior  $q(z|x)$ , and particularly any factorisations assumed, must be crafted for each model. Second, methods such as reparameterisation require specific transformations tailored to the type of latent variables, whether they are continuous or discrete, and whether or not the support is Euclidean. Third, in flexible generative models, such as

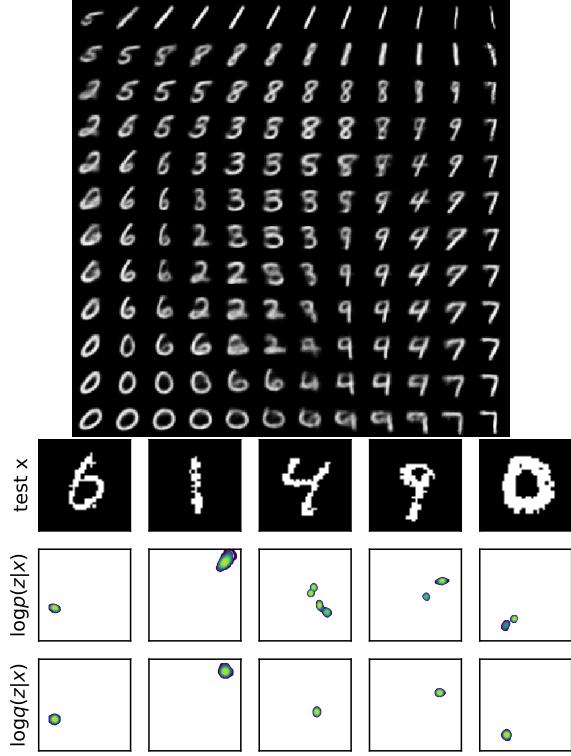


Figure 1. VAE trained on binarised MNIST digits. Top: mean images generated by decoding points on a grid of 2-D latent variables. Bottom three rows show 5 samples of real MNSIT digit (top), the corresponding true posteriors found by histogram (middle) and the approximate posteriors computed by the encoder.

those with conditional dependence modelled using neural networks, true posteriors may be irregular in ways that are difficult to approximate. We illustrate this latter effect using a standard VAE with two-dimensional  $z$  trained on binarised MNIST digits (Figure 1). The exact posterior may be distorted or multi-modal, even though only Gaussian posteriors are ever produced by the encoder.

When inference is approximate, solving the second EM subproblem may not increase the likelihood and so approximate methods usually converge away from the ML parameter values. The dependence of learnt parameters on the quality of the posterior approximation is not straightforward, and the error may not be reduced by (say) approximations with lower Kullback-Leibler (KL) divergence (Turner & Sahani, 2011); indeed errors in posterior statistics that enter the objective function may be unbounded (Huggins et al., 2019).

Here, we propose a novel approach to ML learning in flexible latent variable models that avoids the complications of posterior estimation, instead learns to predict the gradient of the likelihood directly—an approach we call *amortised learning*. The particular realisation we develop here, amortised learning by wake sleep (ALWS), only requires that

sampling from the generative model  $p_\theta(z, x)$  is possible, and that the gradient  $\nabla_\theta \log p_\theta(z, x)$  is available (possibly by automated methods), but otherwise does not make assumptions about the latent variables. We test the performance of ALWS on a wide range of tasks and models, including hierarchical models with heterogeneous priors, nonlinear dynamical systems, and deep models of images. All experiments use the same form of gradient model trained by simple least-squares regression. For image generation, we find that models trained with ALWS can produce samples with considerably better quality when compared with learning algorithms based on variational inference.

## 2. Background

### 2.1. Model definition

Consider a probabilistic generative model with parameters  $\theta$  that define a prior on latents  $p_\theta(z)$  and a conditional on observations  $p_\theta(x|z)$ . In ML learning, we seek parameters that maximise the log (marginal) likelihood

$$\log p_\theta(x) = \log \int p_\theta(z)p_\theta(x|z)dz \quad (1)$$

averaged over data  $\mathcal{D} = \{x_m^*\}_{m=1}^M$ . One approach is to iteratively update  $\theta$  by following the gradient

$$\Delta_\theta(x) := \nabla_\theta \log p_\theta(x) \quad (2)$$

at each iteration.

### 2.2. Variational inference for learning

For many models of interest, the integral in (1) cannot be evaluated analytically, and so direct computation of the gradient is intractable. A popular alternative is to maximise a variational lower bound on the marginal likelihood defined by a distribution  $q(z)$ :

$$\mathcal{F}(q, \theta) := \mathbb{E}_{q(z)}[\log p_\theta(z, x)] + \mathbb{H}[q] \leq \log p_\theta(x), \quad (3)$$

where  $\mathbb{H}[q]$  is the entropy of  $q$ . Thus, the parameter  $\theta$  can be updated by following the gradient of  $\mathcal{F}(q, \theta)$  w.r.t.  $\theta$

$$\begin{aligned} \nabla_\theta \mathcal{F}(q, \theta) &= \nabla_\theta \mathbb{E}_{q(z)}[\log p_\theta(z, x)] \\ &= \mathbb{E}_{q(z)}[\nabla_\theta \log p_\theta(z, x)]. \end{aligned} \quad (4)$$

When  $q(z) = p_\theta(z|x)$ , the lower bound in (3) is tight, and the gradient in (4) is equal to that of the likelihood (see Appendix A.3). Variational approximations attempt to bring  $q$  close to  $p_\theta(z|x)$ , usually by seeking to minimise  $D_{\text{KL}}[q(z)||p_\theta(z|x)]$  (which corresponds to maximising the bound  $\mathcal{F}$  w.r.t.  $q$ ). However, although choosing  $q$  to minimise  $D_{\text{KL}}[q(z)||p_\theta(z|x)]$  ensures consistent optimisation of a single objective, the resulting gradient in (4) will often be a poor approximation to the likelihood gradient (2).

### 2.3. Kernel ridge regression

Our approach is to avoid the difficulties introduced by approximating  $q(\mathbf{z})$  in (4), and instead estimate the expectation directly using least-squares regression (LSR). In this section, we recall the setting of LSR and motivate our use of kernel ridge regression (KRR) in gradient estimation.

Let  $\mathbf{x}$  and  $\mathbf{y}$  be random vectors with a joint distribution  $\rho(\mathbf{x}, \mathbf{y})$  on  $\mathbb{R}^{d_x} \times \mathbb{R}^{d_y}$ . In LSR, we seek a (vector-valued) function  $\mathbf{f}$  that achieves low mean squared error (MSE)  $\mathbb{E}_{\rho(\mathbf{x}, \mathbf{y})}[\|\mathbf{y} - \mathbf{f}(\mathbf{x})\|_2^2]$ . The ideal solution is given by the regression function  $\mathbf{f}_\rho(\mathbf{x}) := \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{y}]$ , as the problem can be cast as the minimisation of the error  $\mathbb{E}_{\rho(\mathbf{x})}[\|\mathbf{f}(\mathbf{x}) - \mathbf{f}_\rho(\mathbf{x})\|_2^2]$ , where  $\rho(\mathbf{x})$  is the marginal distribution of  $\mathbf{x}$  (see Appendix A.1). In practice, the distribution  $\rho(\mathbf{x}, \mathbf{y})$  is known only through an i.i.d. sample  $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N \sim \rho(\mathbf{x}, \mathbf{y})$ ; thus, LSR can be understood to seek a good approximation of  $\mathbf{f}_\rho$  from the sample.

KRR draws the estimated regression function from a flexible class of functions called a reproducing-kernel Hilbert space (RKHS) (Hofmann et al., 2008). The KRR estimator is given by minimising the regularised empirical risk

$$\min_{\mathbf{f} \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^N \|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n)\|_2^2 + \lambda \|\mathbf{f}\|_{\mathcal{H}}^2, \quad (5)$$

where  $\lambda > 0$  is a regularisation parameter and  $\mathcal{H}$  is the RKHS corresponding to a matrix-valued kernel  $\kappa : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_y \times d_y}$  (Carmeli et al., 2006). In this paper, we use a kernel of the form  $\kappa(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') \mathbf{I}_y$ , where  $\mathbf{I}_y$  is the identity matrix and  $k$  is a scalar-valued positive definite kernel; therefore, the matrix-valued kernel  $\kappa$  can be identified with its scalar counterpart  $k$ . In particular, in the scalar output case  $d_y = 1$ , this choice of  $\kappa$  coincides with KRR with the scalar kernel  $k$ . Importantly, the solution  $\hat{\mathbf{f}}_\lambda$  of KRR in (5) can be expressed in closed form

$$\hat{\mathbf{f}}_\lambda(\mathbf{x}^*) = \mathbf{Y}(\mathbf{K} + N\lambda\mathbf{I}_N)^{-1}\mathbf{k}^*, \quad (6)$$

where  $\mathbf{Y}$  is the concatenation of the training targets  $[\mathbf{y}_1, \dots, \mathbf{y}_N] \in \mathbb{R}^{d_y \times N}$ ,  $\mathbf{K} \in \mathbb{R}^{N \times N}$  is the gram matrix whose element is  $(\mathbf{K})_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ ,  $\mathbf{I}_N$  is the identity matrix and  $\mathbf{k}^* = (k(\mathbf{x}_i, \mathbf{x}^*))_{i=1}^N \in \mathbb{R}^N$  for a test point  $\mathbf{x}^*$ .

In the limit of  $N \rightarrow \infty$  and  $\lambda \rightarrow 0$ , the solution  $\hat{\mathbf{f}}_\lambda$  will achieve the minimum MSE in the RKHS (Caponnetto & De Vito, 2007). In general, our target  $\mathbf{f}_\rho$  may not be in the RKHS  $\mathcal{H}$ , i.e.,  $\mathbf{f}_\rho \in \mathcal{L}_\rho^2 \setminus \mathcal{H}$ , where  $\mathcal{L}_\rho^2$  is the set of functions square integrable with respect to the marginal  $\rho(\mathbf{x})$ . Nonetheless, if the RKHS is sufficiently rich (or  $C_0$  universal (Carmeli et al., 2010)), the error made by the estimator  $\mathbb{E}_{\rho(\mathbf{x})}[\|\hat{\mathbf{f}}_\lambda(\mathbf{x}) - \mathbf{f}_\rho(\mathbf{x})\|_2^2]$  will converge to zero (Szabó et al., 2016, Theorem 7).

### 3. Amortised learning by wake-sleep

#### 3.1. Gradient of log likelihood

As stated above and derived in Appendix A.3, the log-likelihood gradient function evaluated on observation  $\mathbf{x}$  at iteration  $t$  (with current parameters  $\theta_t$ ) can be written

$$\begin{aligned} \Delta_{\theta_t}(\mathbf{x}) &= \nabla_{\theta} \log p_{\theta_t}(\mathbf{x})|_{\theta_t} \\ &= \nabla_{\theta} \mathcal{F}(p_{\theta_t}(\mathbf{z}|\mathbf{x}), \theta)|_{\theta_t}, \end{aligned} \quad (7)$$

where the gradient in the second line is taken w.r.t. the second argument; the posterior distribution is for a fixed  $\theta$  at the current  $\theta_t$ .

Here we develop a direct estimate of this gradient, without explicit computation of the posterior. Inserting the definition from (4) into (7) we have,

$$\Delta_{\theta_t}(\mathbf{x}) = \mathbb{E}_{p_{\theta_t}(\mathbf{z}|\mathbf{x})} \left[ \nabla_{\theta} \log p_{\theta}(\mathbf{z}, \mathbf{x})|_{\theta_t} \right] \quad (8)$$

$$\begin{aligned} &= \nabla_{\theta} \mathbb{E}_{p_{\theta_t}(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{z}, \mathbf{x})]|_{\theta_t} \\ &= \nabla_{\theta} J_{\theta}(\mathbf{x})|_{\theta_t}. \end{aligned} \quad (9)$$

where  $J_{\theta}(\mathbf{x}) := \mathbb{E}_{p_{\theta_t}(\mathbf{z}|\mathbf{x})} [\log p_{\theta}(\mathbf{z}, \mathbf{x})]$ . Note that the function  $J_{\theta}(\mathbf{x})$  differs with iteration due to the dependence on  $p_{\theta_t}(\mathbf{z}|\mathbf{x})$ . It can be regarded as an instantaneous objective for ML learning starting from  $\theta_t$ . Neither (8) nor (9) can be computed in closed form, and therefore need to be estimated. We refer to ML learning via the estimation of  $\Delta_{\theta_t}(\mathbf{x})$  either through  $J_{\theta}$  by (9) or directly by (8) as amortised learning. The difference between the two equations lies purely in implementation: The former estimates the high-dimensional  $\Delta_{\theta_t}(\mathbf{x})$  directly, whereas the latter implements the same computation by differentiating  $J_{\theta}(\mathbf{x})$ . We term an estimator of  $J_{\theta}$  a *gradient model*, as it retains information about  $\theta$  and is used to estimate the gradient  $\Delta_{\theta_t}(\mathbf{x})$ . In the next section, we develop a concrete instantiation of amortised learning.

#### 3.2. Training KRR gradient model by wake-sleep

As discussed in Section 2.3, LSR allows us to estimate the conditional expectation of an output variable given an input. Thus, although the gradient in (8) (or in (9)) involves a posterior expectation, by regressing from  $\mathbf{x}$  to  $\nabla_{\theta} \log p_{\theta}(\mathbf{z}, \mathbf{x})$  (or  $\log p_{\theta}(\mathbf{z}, \mathbf{x})$ ), we can obtain an estimate of the gradient  $\Delta_{\theta_t}(\mathbf{x})$ . Any reasonable regression model, e.g., a neural network, could serve this purpose, but here we choose to use KRR introduced in Section 2.3. Other possible forms of gradient model are discussed in Appendix B.1.

The expression in (8) leads to the following LSR problem

$$\min_{\mathbf{f} \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^N \|\nabla_{\theta} (y_{\theta, n})|_{\theta_t} - \mathbf{f}(\mathbf{x}_n)\|_2^2 + \lambda \|\mathbf{f}\|_{\mathcal{H}}^2, \quad (10)$$

where  $y_{\theta, n} = \log p_{\theta}(\mathbf{z}_n, \mathbf{x}_n)$ ,  $\mathcal{H}$  is an RKHS and  $\{(\mathbf{z}_n, \mathbf{x}_n)\}_{n=1}^N \sim p_{\theta}$ . However, the vector-valued

$\nabla_{\theta} \log p_{\theta}$  needs to be evaluated on each  $(z_n, x_n)$ , which requires model-specific derivatives, or a potentially slow loop over all sample pairs in implementation. Alternatively, we can use (9) and find an estimator for the scalar-valued  $J_{\theta}$  that keeps the dependence on  $\theta$  and then evaluate its gradient by automatic differentiation. Thus, we construct an estimator by

$$\hat{J}_{\theta, \gamma} = \arg \min_{f \in \mathcal{H}} \frac{1}{N} \sum_{n=1}^N |y_{\theta, n} - f(x_n)|^2 + \lambda \|f\|_{\mathcal{H}}^2, \quad (11)$$

where  $\mathcal{H}$  is the RKHS induced by a kernel  $k_{\omega}(\cdot, \cdot)$  with hyperparameters  $\omega$ , and  $\gamma = \{\omega, \lambda\}$ . For each data point  $x^* \in \mathcal{D}$ , the estimate of  $J_{\theta}(x^*)$  is

$$\hat{J}_{\theta, \gamma}(x^*) = \alpha_{\theta, \gamma} \cdot k_{\omega}^*, \quad (12)$$

$$\alpha_{\theta, \gamma} = y_{\theta} (\mathbf{K}_{\omega} + \lambda N \mathbf{I}_N)^{-1}, \quad (y_{\theta})_n = \log p_{\theta}(z_n, x_n)$$

$$K_{\omega, i, j} = k_{\omega}(x_i, x_j), \quad k_{\omega, j}^* = k_{\omega}(x_j, x^*)$$

where  $\mathbf{I}_N$  is the identity matrix of size  $N \times N$ . The gradient  $\Delta_{\theta_t}(x)$  is then estimated as

$$\hat{\Delta}_{\theta_t, \gamma}(x) := \nabla_{\theta} \hat{J}_{\theta, \gamma}(x) \Big|_{\theta_t}.$$

In general, a good estimator of  $J_{\theta}$  may not yield a reliable estimate of its gradient  $\nabla_{\theta} J_{\theta}$ ; however, for the KRR estimate, taking the derivative of  $\hat{J}_{\theta, \gamma}$  w.r.t.  $\theta$  is equivalent to replacing  $y_{\theta}$  in (12) with  $\nabla_{\theta}(y_{\theta})|_{\theta_t}$ , which is the solution for the optimisation in (10), with  $\mathcal{H}$  being a vector-valued RKHS given by a kernel  $\kappa_{\omega} = k_{\omega} \mathbf{I}$  (see Section 2.3). We show in Appendix A.2 that, under mild conditions, the target of the regression  $\mathbb{E}_{p_{\theta_t}(z|x)} [\nabla_{\theta} y_{\theta, n}|_{\theta_t}]$  is square-integrable under  $p_{\theta_t}(x)$  for common generative models.

In summary, learning proceeds according to the following wake-sleep procedure: at the  $t$ th step when  $\theta = \theta_t$ , the gradient model is first trained using sleep samples  $(z_n, x_n) \sim p_{\theta_t}$  and evaluations  $\log p_{\theta}(z_n, x_n)$ , keeping the dependence on  $\theta$ ; then the gradient model is applied to real (wake) data  $x^* \in \mathcal{D}$  to produce  $\hat{\Delta}_{\theta_t, \gamma}(x^*)$  by differentiating  $\hat{J}_{\theta, \gamma}$  and evaluating at  $\theta_t$ . See Algorithm 1. Two points are worth emphasis: (a) The algorithm does not require explicit computation or approximation of the posterior, and (b) We only need samples from the model  $p_{\theta}(z, x)$  and differentiable evaluations of  $\log p_{\theta}(z, x)$ .

### 3.3. Exponential family conditionals

The model usually has a likelihood  $p_{\theta}(x|z)$  in the exponential family (e.g. Gaussian, Bernoulli), and we can exploit this structure to simplify the estimation of  $J_{\theta}$ . In this case, the log joint can be written as

$$\begin{aligned} \log p_{\theta}(z, x) &= \log p_{\theta}(x|z) + \log p_{\theta}(z) \\ &= \eta_{\theta}(z) \cdot s(x) - \log Z_{\theta}(z) + \log p_{\theta}(z) \\ &= \eta_{\theta}(z) \cdot s(x) - \Psi_{\theta}(z) \end{aligned}$$

---

**Algorithm 1:** Amortised learning by wake sleep

---

```

input :Dataset  $\mathcal{D}$ , generative model  $\log p_{\theta}(z, x)$ , or
       $\eta_{\theta}$  and  $\Psi_{\theta}$  with parameters  $\theta$ , gradient
      model parameters  $\gamma$ , max epoch
while max epoch not reached do
  Sleep phase: train gradient model
  Sample  $\{z_n, x_n\}_{n=1}^N \sim p_{\theta}$ 
  if  $p(x|z)$  is not in exponential family then
    | Find  $\hat{J}_{\theta, \gamma}(\cdot)$  by computing  $\alpha_{\theta, \gamma}$  in (12)
  else
    | Find  $\hat{h}_{\theta, \gamma}^{\eta}(\cdot)$  and  $\hat{h}_{\theta, \gamma}^{\Psi}(\cdot)$  similar to (12)
    |  $\hat{J}_{\theta, \gamma}(\cdot) = \hat{h}_{\theta, \gamma}^{\eta}(\cdot) \cdot s(x) - \hat{h}_{\theta, \gamma}^{\Psi}(\cdot)$  in (13)

  Sleep phase: update  $\gamma$ 
  Sample  $\{z'_l, x'_l\}_{l=1}^L \sim p_{\theta}$ 
  Compute  $d_l := \log p_{\theta}(z, x)$ 
  Compute  $\mathcal{E}_{\gamma} = \frac{1}{L} \sum_{l=1}^L (\hat{J}_{\theta, \gamma}(x'_l) - d_l)^2$ 
  Update  $\gamma \propto \nabla_{\gamma} \mathcal{E}_{\gamma}$ 

  Wake phase: update  $\theta$ 
  Sample  $\{x_m^*\}_{m=1}^M \in \mathcal{D}$ 
   $\bar{J}_{\theta} = \frac{1}{M} \sum_i^M \hat{J}_{\theta, \gamma}(x_m^*)$ 
  Update  $\theta \propto \nabla_{\theta} \bar{J}_{\theta}$ 
end
return : $\theta$ 

```

---

where  $\eta_{\theta}(z)$ ,  $s(x)$  and  $Z_{\theta}(z)$  are, respectively, the natural parameter, sufficient statistics and normaliser of the likelihood, and  $\Psi_{\theta} := \log Z_{\theta}(z) - \log p_{\theta}(z)$ . By taking the posterior expectation,  $J_{\theta}(x)$  in (9) becomes

$$J_{\theta}(x) = \underbrace{\mathbb{E}_{p_{\theta_t}}[\eta_{\theta}(z)] \cdot s(x)}_{h_{\theta}^{\eta}(x)} - \underbrace{\mathbb{E}_{p_{\theta_t}}[\Psi_{\theta}(z)]}_{h_{\theta}^{\Psi}(x)} \quad (13)$$

where  $p_{\theta_t}$  stands for  $p_{\theta_t}(z|x)$ . Therefore, for exponential family likelihoods, the regression to  $\log p_{\theta}(z, x)$  in (11) can be replaced by two separate regressions to  $\eta_{\theta}(z)$  and  $\Psi_{\theta}(z)$ , which are functions of  $z$  alone. The resulting estimators  $\hat{h}_{\theta, \gamma}^{\eta}$  and  $\hat{h}_{\theta, \gamma}^{\Psi}$  are combined to yield

$$\hat{\Delta}_{\theta_t, \gamma}(x) = \nabla_{\theta} \left[ \hat{h}_{\theta, \gamma}^{\eta}(x) \cdot s(x) \right] \Big|_{\theta_t} - \nabla_{\theta} \hat{h}_{\theta, \gamma}^{\Psi}(x) \Big|_{\theta_t},$$

where Jacobian vector product applies to the first term.

### 3.4. Kernel structure and learning

The kernel  $k_{\omega}$  used in the gradient model affects how well  $\Delta_{\theta_t}(x)$  is estimated. To allow more flexibility,  $k_{\omega}$  can be augmented with a neural network as in (Wilson et al., 2016; Wenliang et al., 2019)

$$k_{\omega}(x, x') = \kappa_{\sigma}(\psi_v(x), \psi_v(x'))$$

where  $\kappa_{\sigma}$  is a standard kernel (e.g. exponentiated quadratic) with parameter  $\sigma$ , and  $\psi_v$  is a neural network with param-

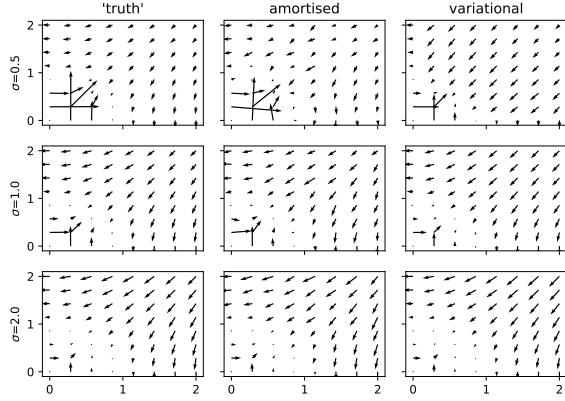


Figure 2. Gradient estimated using amortised learning and variational inference. The true gradients are approximated by importance sampling.

ter  $\mathbf{v}$ , so  $\omega = \{\sigma, \mathbf{v}\}$ . Other details of the kernel structure are described in Appendix B.2.

The gradient model parameter  $\gamma = \{\omega, \lambda\}$  can be learned to further minimise the MSE in (11) using a scheme of cross-validation by gradient descent (Wenliang et al., 2019). Specifically, we generate two sets of sleep samples from  $p_\theta$ ; we use one set to compute  $\alpha_{\theta, \gamma}$  in closed form; then, on the other set, we compute the MSE between the estimator  $\hat{J}_{\theta, \gamma}(\mathbf{x}_l')$  and the ground truth value  $\log p_\theta(\mathbf{z}_l', \mathbf{x}_l')$ , and minimise this by gradient descent on  $\gamma$ . The full ALWS procedure is presented in Algorithm 1.

### 3.5. Dealing with covariate shift

The gradient model is to be used to estimate  $\Delta_{\theta_t}(\mathbf{x})$  on  $\mathbf{x}^*$  drawn from an underlying data distribution  $p^*$ , but it is trained using sleep samples from  $p_\theta$ . This mismatch in input data distribution for training and evaluation is known as covariate shift (Shimodaira, 2000).

Here, to ensure that the gradient model performs reasonably well on  $p^*$ , we initialise  $p_\theta(\mathbf{x})$  to be overdispersed relative to  $p^*$  by setting a large noise in  $p_\theta(\mathbf{x}|\mathbf{z})$ . Since ML estimation minimises  $D_{KL}[p^* \| p_\theta]$ , which penalises a distribution  $p_\theta$  that is narrower than  $p^*$ , we expect the noise to continue to cover the data before the model is well trained. For image data only, we also apply batch normalisation in  $\psi_w$  of the kernel. We find these simple remedies to be effective, though other more principled methods, such as kernel mean matching (Gretton et al., 2009) and binary classification (Gutmann & Hyvärinen, 2010; Goodfellow et al., 2014), may further improve the results.

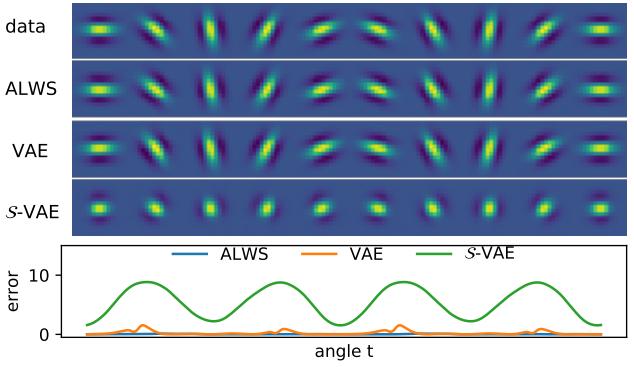


Figure 3. Learning a 1-D circular uniform prior to generate Gabor filters. Top images show samples generated by latents separated by fixed rotation on the circle. For VAE, a 2-D Gaussian prior was used, and the images are generated by latents on the unit circle.  $\mathcal{S}$ -VAEs cannot reliably learn the filters. The errors below show the squared distance between generated images and data at each orientation. For each method, an angle offset and direction are chosen to minimise the total error.

## 4. Experiments

We evaluate ALWS on a wide range of generative models. Details for each experiment can be found in Appendix C. Code is at [github.com/kevin-w-li/al-ws](https://github.com/kevin-w-li/al-ws).

### 4.1. Parameter gradient estimation

First, we demonstrate that KRR can estimate  $\Delta_{\theta_t}(\mathbf{x})$  well on a simple toy generative model described by

$$z_1, z_2 \sim \mathcal{N}(0, 1), \quad x|z \sim \mathcal{N}(\text{softplus}(\mathbf{b} \cdot \mathbf{z}) - \|\mathbf{b}\|_2^2, \sigma_x^2).$$

For a fixed data set, we estimate the gradients of  $\mathbf{b}$  evaluated at different points by ALWS, and compare them to estimates using importance sampling (“truth”) and a factorised Gaussian posterior that minimises the forward KL for each  $\mathbf{x}$ . ALWS tends to estimate better, especially for small  $\mathbf{b}$  (Figure 2). For the smallest  $\sigma_x$ , the KRR estimates are noisier, whereas variational inference introduces greater bias.

### 4.2. Non-Euclidean priors

The prior  $p(\mathbf{z})$  may capture topological structure in the data. For instance, a prior over the hypersphere can be used to describe circular features. Training models with such a prior is straightforward using ALWS, while learning by amortised inference requires special reparameterisation for a posterior on the hypersphere, such as the von-Mises Fisher (vMF) used in the  $\mathcal{S}$ -VAE (Davidson et al., 2018; Xu & Durrett, 2018). We fit a model with uniform circular latent and neural-network output:

$$\begin{aligned} \mathbf{z} &= [\cos(a), \sin(a)], \quad p(a) = \mathcal{U}(a|(-\pi, \pi)), \\ p(\mathbf{x}|\mathbf{z}) &= \mathcal{N}(\mathbf{x}|\text{NN}_w(\mathbf{z}), \sigma_x^2 \mathbf{I}), \end{aligned}$$

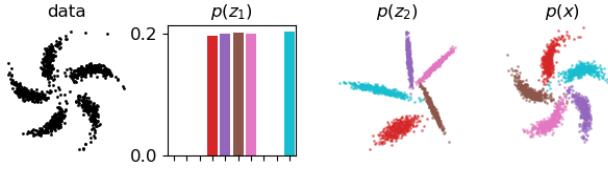


Figure 4. Learning hierarchical model with discrete and continuous latents. From left to right: data sample, component probabilities, samples of the first latent distribution and samples of generated data. Colours correspond to different components

(where  $\mathcal{U}$  is a uniform distribution) on a data set of Gabor wavelets with uniformly distributed orientations. As shown in Figure 3, ALWS learns to generate images that closely resemble the training data. A fixed rotation around the latent circle corresponds to almost a fixed rotation of the Gabor wavelet in the image. The VAE with a 2-D Gaussian latent also generates good filters given latents on the circle, but the length of the filter varies with rotation. Surprisingly, S-VAE is not able to learn on this dataset, the vMF posterior is almost flat for any input image. This hints at potential optimisation issues with the complicated reparameterisation.

### 4.3. Hierarchical models

Rich hierarchical structures in the data can be captured with multiple layers of latents. Provided that samples can be drawn from the hierarchical model and the joint log-likelihood evaluated, ALWS extends straightforwardly to hierarchies, even with mixed discrete and continuous latents. The pinwheel distribution (Johnson et al., 2016; Lin et al., 2018) has five clusters of distorted Gaussian distributions (Figure 4), and can be described by the following model:

$$\begin{aligned} p(\mathbf{z}_1) &= \text{Cat}(\mathbf{z}_1 | \mathbf{m}), \quad p(\mathbf{z}_2 | \mathbf{z}_1 = k) = \mathcal{N}(\mathbf{z}_2 | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \\ p(\mathbf{x} | \mathbf{z}_2) &= \mathcal{N}(\mathbf{x} | \text{NN}_{\mathbf{w}}(\mathbf{z}_2), \boldsymbol{\Sigma}_x). \end{aligned}$$

where Cat is the categorical distribution. The parameters are the logits  $\mathbf{m}$  in 10 dimensions, the means and covariance matrices of the component distributions  $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{10}$ , the weights  $\mathbf{w}$  in NN, and the diagonal covariance  $\boldsymbol{\Sigma}_x$ . The logits  $\mathbf{m}$  are penalised according to a Dirichlet prior, and  $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^{10}$  by a normal-Wishart prior. After training with ALWS, the categorical distribution correctly identifies the five components, and the generated samples match the training data. We compare these samples with those reconstructed from a Bayesian version of the model trained by structured inference network (SIN) (Lin et al., 2018)<sup>1</sup>. A three-way maximum mean discrepancy (MMD) test (Boulifophone et al., 2016) finds that samples from the two models are equally close to the training data ( $p = 0.514$ ,  $N = 1,000$  samples). Details are in Appendix C.3.

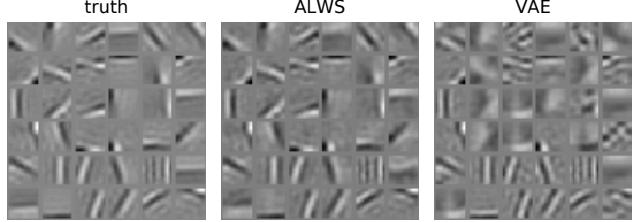


Figure 5. Feature identification. Left, true basis used to generate images. Middle, basis recovered by ALWS. Right, basis recovered by VAE. The filters are arranged according to correlations with the true basis.

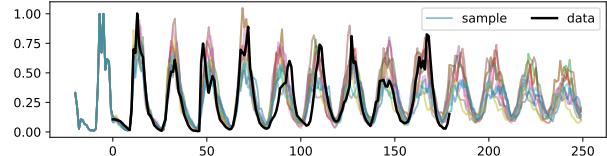


Figure 6. Ecological model for blowfly population. Black, training data. Coloured, samples for an extended time period drawn from trained model.

### 4.4. Feature identification

Learning informative features from complex data can benefit downstream tasks. We use ALWS to identify features from synthetic data generated by

$$p(z_i) = \text{Lap}(z_i | 0, 1), \quad p(\mathbf{x} | \mathbf{z}) = \mathcal{N}(\mathbf{x} | \mathbf{W}\mathbf{z}, \sigma^2 I),$$

where Lap is the Laplace distribution,  $\sigma = 0.1$  and basis  $\mathbf{W}$  contains independent components of natural images (Hateren & Schaaf, 1998) found by the FastICA algorithm (Hyvärinen & Oja, 2000). Since this model is identifiable, we perform model recovery from a random initialisation of  $\mathbf{W}$  using ALWS and compare with a VAE. ALWS clearly finds better features as shown in Figure 5. On generated samples, a three-way MMD test favours ALWS over the Laplace-VAE ( $p < 10^{-5}$ ) based on 10,000 samples. Details are in Appendix C.4.1.

In addition, we demonstrate in Appendix C.4.2 advantage of ALWS in a probabilistic non-negative matrix factorisation model (Ruiz et al., 2016) on reconstructing and denoising MNIST digits.

### 4.5. Neural Processes

The neural process (NP) (Garnelo et al., 2018) is a model that learns to perform inference over function by a fully generative model. We review NPs in more detail and illustrate how they can be trained by ML using ALWS in Appendix C.5. On a toy problem, we compared ALWS with the original variational learning method. NP trained by ALWS produces better prediction and uncertainty estimates on test inputs.

<sup>1</sup>[github.com/emtiyaz/vmp-for-svae](https://github.com/emtiyaz/vmp-for-svae)

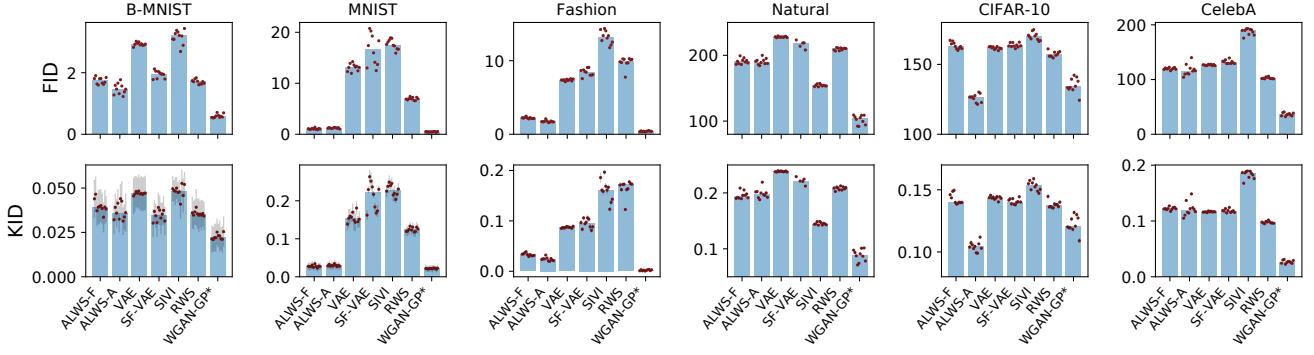


Figure 7. FID and KID scores (lower is better) for different datasets and methods. Red dot is the score for a single run. Bars are medians of the dots for each method. Short bars on KID dots shows standard error of the estimate. \*All models are trained for 50 epochs.

#### 4.6. Dynamical models

In fields such as biology and environmental science, the behaviour of complex systems is often described by simulation-based dynamical models. Finding the good parameters for these models from data is crucial for prediction and policy making. (Lintusaari et al., 2016; Sunnåker et al., 2013; Kypraios et al., 2017) A dynamical model can be expressed, in discrete time, as

$$\mathbf{z}_t = \mathbf{l}_{\theta}(\mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1}, \mathbf{u}_t, \epsilon_t), \quad \mathbf{x}_t = \mathbf{o}_{\theta}(\mathbf{z}_t) + \mathbf{e}_t$$

where  $\mathbf{l}_{\theta}$  describes a latent process that can depend on a control input  $\mathbf{u}_t$ , a noise source  $\epsilon_t$  and the history of latents  $\mathbf{z}_{1:t-1}$  and measurements  $\mathbf{x}_{1:t-1}$ . The function  $\mathbf{o}_{\theta}$  maps the latent  $\mathbf{z}_t$  to measurement with  $\mathbf{n}_t$  being measurement noise. For ALWS, we need that  $p_{\theta}(\mathbf{z}_t, \epsilon_t | \mathbf{z}_{1:t-1}, \mathbf{x}_{1:t-1}, \mathbf{u}_t)$  and  $p_{\theta}(\mathbf{x}_t, \epsilon_t | \mathbf{z}_t)$  are tractable so that  $\nabla_{\theta} \log p(\mathbf{z}_{1:T}, \mathbf{x}_{1:T})$  can be evaluated, where  $T$  is the length of the data. However, learning using approximate inference may be challenging, as there may be complex dependencies between latent variables and across time.

Here, we fit the parameters of two dynamical models: the Hodgkin-Huxley (HH) model (Pospischil et al., 2008) on the membrane potential of a simulated neuron, and an ecological model (ECO) on blowfly data (Wood, 2010). The HH equations describe the membrane potential and three ion-channel state variables of a neuron that follow complicated nonlinear transitions. Details of the experiment are in Appendix C.6, and results in Figure 12. ECO describes the population size of a species following non-linear and non-Gaussian dynamics and has discrete and continuous latent variables. ECO on blowfly data have been used to validate approximate Bayesian computation (ABC) methods (Park et al., 2016). The model trained with ALWS can simulate sequences very close to data Figure 6, and are visibly closer than sequences from the model trained with ABC (Park et al., 2016, Figure 2b).

#### 4.7. Sample quality on benchmark datasets

Finally, we train deep models of images and test sample quality. We chose six benchmark datasets: the binarised and original MNIST (LeCun et al., 1998) (B-MNIST and MNIST, respectively), fashion MNIST (Fashion) (Xiao et al., 2017), natural images (Natural) (Hateren & Schaaf, 1998), CIFAR-10 (Krizhevsky et al., 2009) and CelebA (Liu et al., 2015). The original un-binarised MNIST is known to be difficult for most VAE-based methods (Loaiza-Ganem & Cunningham, 2019). Natural images consist of pre-processed gray-scale images from natural scenes. All images have size  $32 \times 32$  with color channels. For ALWS, we test two variants. In ALWS-F, the gradient model parameters  $\gamma$  are fixed, In ALWS-A,  $\gamma$  is adapted as described in Section 3.4 except for  $\lambda$  which is fixed at 0.1. Fixing  $\lambda$  improved quality for the higher-dimensional CIFAR-10 and CelebA, but lowered quality for Natural and did not affect much on the other datasets.

We compare these methods with four other approaches: the vanilla VAE (Kingma & Welling, 2014), VAE with a Sylvester (orthogonal) flow as an inference network (van den Berg et al., 2018) (Syl-VAE)<sup>2</sup>, semi-implicit variational inference (Yin & Zhou, 2018) (SIVI)<sup>3</sup>, and reweighted wake-sleep (Bornschein & Bengio, 2015). Each algorithm has the same generative network architecture as in DCGAN<sup>4</sup> with the last convolutional layer removed. We also run WGAN-GP (Gulrajani et al., 2017)<sup>5</sup> for reference, although it is not trained by ML methods. Each algorithm is run for 50 epochs 10 times with different initialisations, except for SIVI where we trained for 1000 epochs with a smaller learning rate for stability. To test the generative quality, we compute both the Fréchet Inception Distance (FID)

<sup>2</sup>[github.com/riannedvberg/sylvester-flows](https://github.com/riannedvberg/sylvester-flows)

<sup>3</sup>[github.com/mingzhang-yin/SIVI](https://github.com/mingzhang-yin/SIVI)

<sup>4</sup>[pytorch.org/tutorials/beginner/dcgan\\_faces\\_tutorial.html](https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html)

<sup>5</sup>[github.com/caogang/wgan-gp](https://github.com/caogang/wgan-gp)

(Heusel et al., 2017) and Kernel Inception Distance (KID) (Binkowski et al., 2018) on 10,000 generated images. The results are shown in Figure 7. According to FID, ALWS-A is the best ML method for binarised MNIST, Fashion, and CIFAR-10. Notably, both ALWS-A and ALWS-F has much smaller FID and KID on MNIST and Fashion than other ML methods. WGAN-GP did not produce a good score on CIFAR-10 within 50 epochs, but becomes the best model for all datasets with more epoch. Samples are shown from Figure 15 to Figure 20 in Appendix C.7, along with additional related experiments to show the effectiveness of ALWS.

## 5. Related work

### 5.1. Amortised variational inference

Using  $\mathcal{F}(q, \theta)$  as the objective for learning  $\theta$ , the gradient for  $\theta$  is given by an intractable posterior expectation. The large majority of learning algorithms based on amortised variational inference use Monte Carlo estimators for the gradient. The Variational auto-encoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) parametrises  $q_\phi(z|x)$  by simple distributions using reparameterised samples to obtain gradients for  $\psi$ . Approximate posteriors may also be incorporated into tighter bounds on  $\log p_\theta(x)$  by re-weighting (Burda et al., 2016; Bornschein & Bengio, 2015; Le et al., 2019), although with some loss of gradient signal (Rainforth et al., 2018). More expressive forms of  $q_\phi$  can be formed by invertible transformations (normalizing flows) (Rezende & Mohamed, 2015; Kingma et al., 2016; van den Berg et al., 2018)) that allow  $\mathbb{H}[q_\phi]$  to be computed easily, or by non-invertible mappings (implicit variational inference), which requires estimating  $\mathbb{H}[q_\phi]$  or its gradient w.r.t.  $\phi$ . (Shi et al., 2018; Li & Turner, 2018; Yin & Zhou, 2018; Huszár, 2017). Learning non-Gaussian and discrete posteriors requires specific reparameterisation schemes (Jang et al., 2017; Vahdat et al., 2018; Rolfe, 2017; Ruiz et al., 2016; Figurnov et al., 2018). On the other hand, amortised learning focuses exclusively on estimating the gradient for ML learning, making no assumptions on the type of latent variables.

Our approach is related to at least two other algorithms inspired by the original Helmholtz machine (HM) (Dayan et al., 1995; Hinton et al., 1995). The distributed distributional code HM (DDC-HM) (Vértes & Sahani, 2018) represents posteriors by expectations of predefined and finite nonlinear features, which are used to approximate  $\Delta_{\theta_t}(x)$  by linearity of expectation. ALWS differs from DDC-HM in two ways. First, our gradient model integrates the inferential model and the linear readout for  $\Delta_{\theta_t}(x)$  in DDC-HM using adaptive and more flexible KRR. Second, using (9) avoids explicit computation of  $\nabla_\theta \log p_\theta$  and makes ALWS easily applicable on more complex generative models. Reweighted wake-sleep (RWS) (Bornstein & Bengio, 2015) addressed covariance shift by training an inferential model to increase

the likelihood of not only sleep  $z$  given sleep  $x$  as in the HM, but also weighted posterior samples given data  $x^*$ . ALWS does not make assumptions about the posterior distributions, and we found that simple strategies mitigated covariate shift in practice, but this is a point that deserves further investigation.

### 5.2. Training implicit generative models

Implicit generative models, including generative adversarial networks (GANs) (Goodfellow et al., 2014) and simulation-based models considered by approximate Bayesian computation (ABC) (Tavaré et al., 1997; Marin et al., 2012), do not have an explicitly defined likelihood function, but can be trained using simulated data. Amortised learning requires an explicit joint likelihood function  $p_\theta(x, z)$ , but can also train simulation-based generative models (Section 4.6). In GANs, the generator is improved by a discriminator that is concurrently trained to tell apart real and generated samples. The approach is able to synthesise high-quality samples in high dimensions. However, the competitive setting can be problematic for convergence, and the discriminator needs to be carefully regularised to be less effective at its own task but more informative to the generator. (Arjovsky et al., 2017; Gulrajani et al., 2017; Arbel et al., 2018; Mescheder et al., 2018). In amortised learning, a better gradient model always helps when training the generative model. Furthermore, it is not known whether adversarial methods can learn from a small but complex time series like those in Section 4.6.

ABC estimates the posterior of parameters in a generative model given a prior, rather than performing maximum likelihood estimation. The approximate posterior is formed by prior samples that generate simulations close to real data. Amortised learning can be seen as a method of maximum likelihood based on simulations, since the the gradient model is trained using data from the generative model. In this sense, ALWS is similar to Kernel-ABC (Nakagome et al., 2013) in which the parameter posterior is found by weighting prior samples using KRR on pre-defined summary statistics, although the iterative refinement of ALWS (and dependence on the full data set, rather than summary statistics) is likely to lead to more accurate learning.

## 6. Discussion

Direct estimation of the expected log-likelihood and its gradient in a latent variable model circumvents the challenges and issues posed by explicit approximation of posteriors. The KRR gradient model is consistent, easy to implement, and avoids the need for explicit computation of derivatives. However, its computational complexity limits the number of sleep samples that can be used and thus the quality of approximation. In particular, while the kernel is adapted during training, the KRR prediction is a linear combination of

the finite set  $\{\nabla_{\theta} \log p_{\theta}(z_n, x_n)\}_{n=1}^N$ , and the true gradient function may lie outside this span—an issue that might be compounded by covariate shift. Thus, alternative amortised learning models may be worth future exploration. Nonetheless, we have found here that ALWS based on KRR provides accurate parameter estimates in many settings where approximate-inference-based approaches appear to struggle.

## References

- Arbel, M., Sutherland, D. J., Binkowski, M., and Gretton, A. On gradient regularizers for MMD gans. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 6701–6711, 2018.
- Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223, 2017.
- Binkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying MMD gans. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
- Bishop, C. M. *Pattern recognition and machine learning*. Springer, 2006.
- Bornschein, J. and Bengio, Y. Reweighted wake-sleep. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.
- Bounliphone, W., Belilovsky, E., Blaschko, M. B., Antonoglou, I., and Gretton, A. A test of relative similarity for model selection in generative models. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.
- Burda, Y., Grosse, R. B., and Salakhutdinov, R. Importance weighted autoencoders. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.
- Caponnetto, A. and De Vito, E. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.
- Carmeli, C., De Vito, E., and Toigo, A. Vector valued reproducing kernel hilbert spaces of integrable functions and mercer theorem. *Analysis and Applications*, 4(04):377–408, 2006.
- Carmeli, C., De Vito, E., Toigo, A., and Umanitá, V. Vector valued reproducing kernel hilbert spaces and universality. *Analysis and Applications*, 8(01):19–61, 2010.
- Chatterjee, S., Diaconis, P., et al. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.
- Davidson, T. R., Falorsi, L., Cao, N. D., Kipf, T., and Tomczak, J. M. Hyperspherical variational auto-encoders. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pp. 856–865, 2018.
- Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The Helmholtz machine. *Neural computation*, 7(5):889–904, 1995.
- Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.
- Figurnov, M., Mohamed, S., and Mnih, A. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 439–450, 2018.
- Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.
- Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.
- Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of Wasserstein GANs. In *Advances in neural information processing systems*, pp. 5767–5777, 2017.
- Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pp. 297–304, 2010.
- Hateren, J. H. v. and Schaaf, A. v. d. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings: Biological Sciences*, 265(1394):359–366, Mar 1998.

- Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 6626–6637, 2017.
- Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. The “wake-sleep” algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.
- Hofmann, T., Schölkopf, B., and Smola, A. J. Kernel methods in machine learning. *The annals of statistics*, pp. 1171–1220, 2008.
- Huggins, J. H., Kasprzak, M., Campbell, T., and Broderick, T. Practical posterior error bounds from variational objectives. *CoRR*, abs/1910.04102, 2019.
- Huszár, F. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.
- Hyvärinen, A. and Oja, E. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5): 411–430, 2000.
- Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *ICLR*, 2017.
- Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pp. 2946–2954, 2016.
- Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.
- Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *NIPS*, pp. 4743–4751, 2016.
- Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
- Kypraios, T., Neal, P., and Prangle, D. A tutorial introduction to bayesian inference for stochastic epidemic models using approximate bayesian computation. *Mathematical biosciences*, 287:42–53, 2017.
- Le, T. A., Kosiorek, A. R., Siddharth, N., Teh, Y. W., and Wood, F. Revisiting reweighted wake-sleep for models with stochastic control flow. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*, pp. 373, 2019.
- LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Li, Y. and Turner, R. E. Gradient estimators for implicit models. In *ICLR*, 2018.
- Lin, W., Hubacher, N., and Khan, M. E. Variational message passing with structured inference networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.
- Lintusaari, J., Gutmann, M. U., Dutta, R., Kaski, S., and Corander, J. Fundamentals and Recent Developments in Approximate Bayesian Computation. *Systematic Biology*, 66(1):e66–e82, 09 2016. ISSN 1063-5157. doi: 10.1093/sysbio/syw077.
- Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.
- Loaiza-Ganem, G. and Cunningham, J. P. The continuous Bernoulli: fixing a pervasive error in variational autoencoders. In *Advances in Neural Information Processing Systems 32*, pp. 13266–13276. Curran Associates, Inc., 2019.
- Marin, J.-M., Pudlo, P., Robert, C. P., and Ryder, R. J. Approximate bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.
- Mescheder, L. M., Geiger, A., and Nowozin, S. Which training methods for gans do actually converge? In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 3478–3487, 2018.
- Nakagome, S., Fukumizu, K., and Mano, S. Kernel approximate bayesian computation in population genetic inferences. *Statistical applications in genetics and molecular biology*, 12(6):667–678, 2013.
- Newey, K. and McFadden, D. Large sample estimation and hypothesis. *Handbook of Econometrics, IV, Edited by RF Engle and DL McFadden*, pp. 2112–2245, 1994.
- Park, M., Jitkrittum, W., and Sejdinovic, D. K2-ABC: approximate bayesian computation with kernel embeddings. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, May 9-11, 2016*, pp. 398–407, 2016.

- Pospischil, M., Toledo-Rodriguez, M., Monier, C., Piwkowska, Z., Bal, T., Frégnac, Y., Markram, H., and Destexhe, A. Minimal hodgkin–huxley type models for different classes of cortical and thalamic neurons. *Biological cybernetics*, 99(4-5):427–441, 2008.
- Rainforth, T., Kosiorek, A. R., Le, T. A., Maddison, C. J., Igl, M., Wood, F., and Teh, Y. W. Tighter variational bounds are not necessarily better. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 4274–4282, 2018.
- Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.
- Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pp. 1278–1286, 2014.
- Rolfe, J. T. Discrete variational autoencoders. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.
- Ruiz, F. J. R., Titsias, M. K., and Blei, D. M. The generalized reparameterization gradient. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 460–468, 2016.
- Shi, J., Sun, S., and Zhu, J. Kernel implicit variational inference. In *ICLR*, 2018.
- Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.
- Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. R. Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, 11(Apr):1517–1561, 2010.
- Sunnåker, M., Busetto, A. G., Numminen, E., Corander, J., Foll, M., and Dessimoz, C. Approximate bayesian computation. *PLoS computational biology*, 9(1), 2013.
- Szabó, Z., Sriperumbudur, B. K., Póczos, B., and Gretton, A. Learning theory for distribution regression. *Journal of Machine Learning Research*, 17(152):1–40, 2016.
- Tavaré, S., Balding, D. J., Griffiths, R. C., and Donnelly, P. Inferring coalescence times from dna sequence data. *Genetics*, 145(2):505–518, 1997.
- Turner, R. and Sahani, M. Two problems with variational expectation maximisation for time-series models. *Bayesian Time Series Models*, pp. 109–130, 01 2011.
- Vahdat, A., Macready, W. G., Bian, Z., Khoshaman, A., and Andriyash, E. DVAE++: discrete variational autoencoders with overlapping transformations. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 5042–5051, 2018.
- van den Berg, R., Hasenclever, L., Tomczak, J. M., and Welling, M. Sylvester normalizing flows for variational inference. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pp. 393–402, 2018.
- Vértes, E. and Sahani, M. Flexible and accurate inference and learning for deep generative models. In *Advances in Neural Information Processing Systems*, pp. 4166–4175, 2018.
- Wenliang, L., Sutherland, D. J., Strathmann, H., and Gretton, A. Learning deep kernels for exponential family densities. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 6737–6746, 2019.
- Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In *Artificial Intelligence and Statistics*, pp. 370–378, 2016.
- Wood, S. N. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104, 2010.
- Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.
- Xu, J. and Durrett, G. Spherical latent spaces for stable variational autoencoders. *arXiv preprint arXiv:1808.10805*, 2018.
- Yin, M. and Zhou, M. Semi-implicit variational inference. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 5646–5655, 2018.
- Zhang, C., Butepage, J., Kjellstrom, H., and Mandt, S. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

## A. Mathematical details

### A.1. Solving mean squared error to for conditional expectations

Given  $\mathbf{x}, \mathbf{y} \sim \rho(\mathbf{x}, \mathbf{y})$ , we want to find an estimator  $\hat{\mathbf{f}}(\mathbf{x})$  in some space  $\mathcal{F}$  for the posterior mean function  $\mathbf{f}_\rho : \mathbf{x} \mapsto \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{y}]$ . Assuming that  $\mathcal{F}$  is contained in  $\mathcal{L}_\rho^2$ , the class of squared-integral functions under  $\rho(\mathbf{x})$ , and  $\mathbf{y}$  has finite  $l$ -2 norm under  $\rho(\mathbf{y})$ , a natural cost function to minimise is the averaged squared distance between the two

$$L_E(\mathbf{f}) := \mathbb{E}_{\rho(\mathbf{y}, \mathbf{x})} \left[ \|\mathbf{f}(\mathbf{x}) - \mathbf{y}\|_2^2 \right].$$

We can move  $\mathbf{f}(\mathbf{x})$  inside the inner expectation as it does not depend on  $\mathbf{y}$ , and by Jensen's inequality, we can bound it by

$$L_E(\mathbf{f}) = \mathbb{E}_{\rho(\mathbf{x})} \left[ \left\| \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{f}(\mathbf{x}) - \mathbf{y}] \right\|_2^2 \right] \leq \mathbb{E}_{\rho(\mathbf{x})} \left[ \left\| \mathbf{f}(\mathbf{x}) - \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{y}] \right\|_2^2 \right] = L_R(\mathbf{f}).$$

This shows that the MSE is an upper bound on the expected  $l$ -2 distance between  $\mathbf{f}(\mathbf{x})$  and the posterior mean  $\mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{y}]$ . Further, the optimum of  $L_R$  is the optimum of  $L_E$  through a simple decomposition

$$\begin{aligned} L_E(\mathbf{f}) &= \mathbb{E}_{\rho(\mathbf{x})} \left[ \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})} \left[ \|\mathbf{f}(\mathbf{x}) - \mathbf{y}\|_2^2 \right] \right] \\ &= \mathbb{E}_{\rho(\mathbf{x})} \left[ \|\mathbf{f}(\mathbf{x})\|_2^2 - \mathbf{f}(\mathbf{x}) \cdot \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{y}] + \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})} [\|\mathbf{y}\|_2^2] \right] \\ &\stackrel{(1)}{=} \mathbb{E}_{\rho(\mathbf{x})} \left[ \|\mathbf{f}(\mathbf{x})\|_2^2 - \mathbf{f}(\mathbf{x}) \cdot \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{y}] + \left\| \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{y}] \right\|_2^2 + \text{Tr} [\mathbb{C}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{y}]] \right] \\ &= \mathbb{E}_{\rho(\mathbf{x})} \left[ \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})} \left[ \|\mathbf{f}(\mathbf{x}) - \mathbb{E}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{y}]\|_2^2 \right] \right] + \mathbb{E}_{\rho(\mathbf{x})} [\text{Tr} [\mathbb{C}_{\rho(\mathbf{y}|\mathbf{x})}[\mathbf{y}]]] \\ &= L_R(\mathbf{f}) + \text{term independent of } \mathbf{f} \end{aligned}$$

where  $\mathbb{C}_p$  is the covariance under  $p$ . Equality (1) holds because

$$\mathbb{E}_p [\|\mathbf{a}\|_2^2] = \mathbb{E}_p \left[ \sum_i a_i^2 \right] = \sum_i \mathbb{E}_p [a_i^2] = \mathbb{E}_p [a_i]^2 + \sum_i \mathbb{C}[a_i] = \|\mathbb{E}_p[\mathbf{a}]\|_2^2 + \text{Tr} [\mathbb{C}_p[\mathbf{a}]]$$

for any  $\mathbf{a}$  in  $L_2(p)$ . So  $L_R(\mathbf{f})$  is equal to  $L_E(\mathbf{f})$  up to a constant that depends only on the data itself.

### A.2. Smoothness of the gradient function

To learn to predict  $\mathbf{y}(\mathbf{x}) = \mathbb{E}_{p_{\theta_t}(\mathbf{z}|\mathbf{x})}[\nabla_\theta \log p_\theta(\mathbf{z}, \mathbf{x})]|_{\theta_t}$  using regression as above, the target needs to be square-integrable under  $p_{\theta_t}(\mathbf{x})$ , i.e.  $\mathbf{y}(\mathbf{x}) \in \mathcal{L}_p^2$ . Common likelihood functions are in the exponential family, and has  $\nabla_\theta \log p_\theta(\mathbf{z}, \mathbf{x}) = \nabla_\theta \eta_\theta(\mathbf{z}) \cdot s(\mathbf{x}) - \nabla_\theta \Psi(\mathbf{z})$ . and it suffices to check whether  $\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\nabla_\theta \eta_\theta(\mathbf{z})]|_{\theta_t}$ ,  $\mathbb{E}_{p_{\theta_t}(\mathbf{z}|\mathbf{x})}[\nabla_\theta \Psi(\mathbf{z})]|_{\theta_t}$  and  $s(\mathbf{x})$  are in  $\mathcal{L}_p^2$ . We sketch below that this is indeed the case for common choices of model architectures.

For a simple example, consider a model

$$p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad p_\theta(\mathbf{z}) = \mathcal{N}(\text{NN}_w(\mathbf{z}), \Sigma_x). \quad (14)$$

where  $\mathbf{I}$  is the identity covariance matrix,  $\text{NN}_w$  is a neural network with weights  $w$  and  $\Sigma_x$  is a covariance matrix. In this case, one has that

$$\Psi_\theta(\mathbf{z}) = -\frac{1}{2} \|\mathbf{z}\|_2^2 - \frac{1}{2} \log |\Sigma_x| + \text{constant}, \quad \eta_\theta(\mathbf{z}) = [\Sigma_x^{-1} \text{NN}_w(\mathbf{z}), -\frac{1}{2} \Sigma_x^{-1}], \quad \theta = \{w, \Sigma_x\}, \quad s(\mathbf{x}) = [\mathbf{x}, \mathbf{x}\mathbf{x}^T].$$

Clearly, if all eigenvalues of  $\Sigma_x$  are strictly positive,

$$\mathbb{E}_{p_{\theta_t}(\mathbf{z})} \left[ \left\| \mathbb{E}_{p_{\theta_t}(\mathbf{z}|\mathbf{x})}[\nabla_\theta \Psi_\theta(\mathbf{z})] \right\|_2^2 \right] \leq \mathbb{E}_{p_{\theta_t}(\mathbf{z})} \left[ \mathbb{E}_{p_{\theta_t}(\mathbf{z}|\mathbf{x})} \left[ \left\| \nabla_\theta \Psi_\theta(\mathbf{z}) \right\|_2^2 \right] \right] = -\frac{1}{2} \|\Sigma_x^{-1}\|_2^2 < \infty$$

where we have applied Jensen's inequality. Therefore,  $\mathbb{E}_{p_{\theta_t}(\mathbf{z}|\mathbf{x})}[\nabla_\theta \Psi_\theta(\mathbf{z})]|_{\theta_t}$  is a constant and thus in  $\mathcal{L}_p^2$ .

For  $\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x})}[\nabla_\theta \eta_\theta(\mathbf{z})]|_{\theta_t}$ , it is easy to show that, as long as

1. the neural network  $\text{NN}_{\theta}(\mathbf{z})$  is Lipschitz and differentiable almost everywhere, such as one that is composed of linear projections followed by Lipschitz nonlinearities (e.g. ReLU),
2. spectral norm of  $\mathbf{W}$  in each layer is bounded above by a positive constant,
3. spectral norm of  $\Sigma_{\mathbf{x}}$  is bounded below by a positive constant,

the gradient  $\nabla_{\theta}\eta_{\theta}(\mathbf{z})$  is Lipschitz, i.e. there exists  $C_0, C_1 > 0$  such that  $\left\| \nabla_{\theta}\eta_{\theta}(\mathbf{z}) \right\|_{\theta_t} \leq C_0 + C_1 \|\mathbf{z}\|_2$ . This can be proved by writing out  $\nabla_{\theta}\text{NN}_{\theta}(\mathbf{z})$  using chain rule, which will be a series of product involving  $\mathbf{W}$  in each layer and derivative of Lipschitz functions, and applying the first two conditions above. The third condition is needed as the gradient w.r.t. the covariance can be arbitrarily large as it becomes singular. Then we can bound the quantity below

$$\begin{aligned} \mathbb{E}_{p_{\theta_t}(\mathbf{x})} \left[ \left\| \mathbb{E}_{p_{\theta_t}(\mathbf{z}|\mathbf{x})} \left[ \nabla_{\theta}\eta_{\theta}(\mathbf{z}) \right] \right\|_2^2 \right] &\leq \mathbb{E}_{p_{\theta_t}(\mathbf{x})} \left[ \mathbb{E}_{p_{\theta}(\mathbf{z}|\mathbf{x})} \left[ \left\| \nabla_{\theta}\eta_{\theta}(\mathbf{z}) \right\|_2^2 \right] \right] \leq \mathbb{E}_{p_{\theta_t}(\mathbf{z})} [(C_0 + C_1 \|\mathbf{z}\|_2)^2] \\ &= C_0 + 2C_0C_1 \mathbb{E}_{p_{\theta_t}(\mathbf{z})} [\|\mathbf{z}\|_2] + C_1^2 \mathbb{E}_{p_{\theta_t}(\mathbf{z})} [\|\mathbf{z}\|_2^2] \\ &= C_0 + 2C_0C_1 \mathbb{E}_{p_{\theta_t}(\mathbf{z})} [\|\mathbf{z}\|_2] + C_1^2 \text{Tr}(\mathbf{I}) < \infty, \end{aligned}$$

since  $\mathbb{E}_{p_{\theta_t}(\mathbf{z})} [\|\mathbf{z}\|_2]$  is the mean of a  $\chi$  distribution.

Now for  $\mathbf{s}(\mathbf{x})$ , note that the conditional expectation  $\mathbb{E}_{p_{\theta_t}(\mathbf{x}|\mathbf{z})} [\|\mathbf{s}(\mathbf{x})\|_2^2]$  involves finite sums of products of the second, third and fourth moments of the Gaussian distribution, which is always finite for any  $\mathbf{z}$  almost surely. Its expectation under  $p_{\theta}(\mathbf{z})$  is thus also finite, leading to  $\mathbb{E}_{p_{\theta_t}(\mathbf{x})} [\|\mathbf{s}(\mathbf{x})\|_2^2] < \infty$ .

Therefore, for the generative model defined in (14), the desired target  $\mathbf{y}(\mathbf{x}) = \mathbb{E}_{p_{\theta_t}(\mathbf{z}|\mathbf{x})} [\nabla_{\theta} \log p_{\theta}(\mathbf{z}, \mathbf{x})] \Big|_{\theta_t}$  for regression is in  $\mathcal{L}_p^2$ , which can be approximated arbitrarily well by KRR (see Section 2.3)) with more sleep samples. A similar analysis can show that, for Bernoulli likelihoods whose logits are parametrised by a Lipschitz neural network, the target for the regression is also in  $\mathcal{L}_p^2$ , with the condition that the logits are bounded from above and below.

### A.3. Gradient of the log marginal likelihood w.r.t. parameters

To show the known result (7), we first start from derive free energy (ELBO) lower bound on the log likelihood  $\log p_{\theta}(\mathbf{x})$ .

$$\begin{aligned} \log p_{\theta}(\mathbf{x}) &= \log \frac{p_{\theta}(\mathbf{z}, \mathbf{x})}{p_{\theta}(\mathbf{z}|\mathbf{x})} d\mathbf{z} = \int q(\mathbf{z}) \log \left[ \frac{q(\mathbf{z}) p_{\theta}(\mathbf{z}, \mathbf{x})}{q(\mathbf{z}) p_{\theta}(\mathbf{z}|\mathbf{x})} \right] d\mathbf{z} = \int q(\mathbf{z}) \log \left[ \frac{p_{\theta}(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} \frac{q(\mathbf{z})}{p_{\theta}(\mathbf{z}|\mathbf{x})} \right] d\mathbf{z} \\ &= \int q(\mathbf{z}) \log p_{\theta}(\mathbf{z}, \mathbf{x}) d\mathbf{z} - \int q(\mathbf{z}) \log q(\mathbf{z}) d\mathbf{z} + D_{\text{KL}}[q(\mathbf{z}) \| p_{\theta}(\mathbf{z}|\mathbf{x})] \\ &= \mathcal{F}(q, \theta) + D_{\text{KL}}[q(\mathbf{z}) \| p_{\theta}(\mathbf{z}|\mathbf{x})], \end{aligned} \tag{15}$$

where we have defined

$$\mathcal{F}(q, \theta) = \int q(\mathbf{z}) \log p_{\theta}(\mathbf{z}, \mathbf{x}) d\mathbf{z} - \int q(\mathbf{z}) \log q(\mathbf{z}) d\mathbf{z} = \mathbb{E}_{q(\mathbf{z})} [\log p_{\theta}(\mathbf{z}, \mathbf{x})] + \mathbb{H}[q].$$

The KL term in (15) is non-negative and is zero if  $q(\mathbf{z}) = p_{\theta}(\mathbf{z}|\mathbf{x})$ , suggesting that

$$\log p_{\theta}(\mathbf{x}) = \mathcal{F}(p_{\theta}(\mathbf{z}|\mathbf{x}), \theta)$$

Replacing  $q(\mathbf{z}) = p_{\theta}(\mathbf{z}|\mathbf{x})$  in (15) and take derivative w.r.t.  $\theta$  gives

$$\begin{aligned} \Delta_{\theta}(\mathbf{x}) &:= \nabla_{\theta} \log p_{\theta}(\mathbf{x}) \\ &= \nabla_{\theta} \int p_{\theta}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{z}, \mathbf{x}) d\mathbf{z} - \nabla_{\theta} \int p_{\theta}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\ &= \int \nabla_{\theta} p_{\theta}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{z}, \mathbf{x}) d\mathbf{z} + \int p_{\theta}(\mathbf{z}|\mathbf{x}) \nabla_{\theta} \log p_{\theta}(\mathbf{z}, \mathbf{x}) d\mathbf{z} \\ &\quad - \int \nabla_{\theta} p_{\theta}(\mathbf{z}|\mathbf{x}) \log p_{\theta}(\mathbf{z}|\mathbf{x}) d\mathbf{z} - \int p_{\theta}(\mathbf{z}|\mathbf{x}) \nabla_{\theta} \log p_{\theta}(\mathbf{z}|\mathbf{x}) d\mathbf{z}. \end{aligned} \tag{16}$$

The last term in (16) is zero since it is the expectation of the score function

$$\int p_{\theta}(z|x) \nabla \log p_{\theta}(z|x) dz = \int p_{\theta}(z|x) \frac{1}{p_{\theta}(z|x)} \nabla_{\theta} p_{\theta}(z|x) dz = \nabla_{\theta} \int p_{\theta}(z|x) dz = 0.$$

The first and third terms in (16) combines to give

$$\int \nabla_{\theta} p_{\theta}(z|x) \log \frac{p_{\theta}(z|x)}{p_{\theta}(z|x)} dz = \int \nabla_{\theta} p_{\theta}(z|x) \log p_{\theta}(x) dz = \log p_{\theta}(x) \nabla_{\theta} \int p_{\theta}(z|x) dz = 0.$$

We are left with only the second term in (16)

$$\Delta_{\theta}(x) = \int p_{\theta}(z|x) \nabla_{\theta} \log p_{\theta}(z, x) = \mathbb{E}_{p_{\theta}(z|x)} [\nabla_{\theta} \log p_{\theta}(z, x)] = \nabla_{\theta} \mathcal{F}(p_{\theta}(z|x), \theta) \quad (17)$$

To compute the update at the  $t$ 'th iteration with  $\theta = \theta_t$ , and the expectation above is taken over a fixed posterior distribution  $p_{\theta_t}(z|x)$ . We evaluate the above equation at  $\theta_t$  on both sides, giving

$$\Delta_{\theta_t}(x) := \Delta_{\theta}(x)|_{\theta_t} = \nabla_{\theta} \mathbb{E}_{p_{\theta_t}(z|x)} [\log p_{\theta}(z, x)]|_{\theta_t} = \nabla_{\theta} \mathcal{F}(p_{\theta}(z|x), \theta)|_{\theta_t}$$

which is (7). One can also keep  $\nabla_{\theta}$  and evaluation inside the expectation to obtain

$$\Delta_{\theta_t}(x) = \nabla_{\theta} \mathbb{E}_{p_{\theta_t}(z|x)} [\log p_{\theta}(z, x)]|_{\theta_t} = \mathbb{E}_{p_{\theta_t}(z|x)} [\nabla_{\theta} \log p_{\theta}(z, x)|_{\theta_t}]$$

which (8) used for direct gradient estimation.

In fact, once we know the result above, going from the right hand side to the left is much simpler:

$$\begin{aligned} \mathbb{E}_{p_{\theta_t}(z|x)} [\nabla_{\theta} \log p_{\theta}(z, x)|_{\theta_t}] &= \mathbb{E}_{p_{\theta_t}(z|x)} [\nabla_{\theta} \log p_{\theta}(z|x)|_{\theta_t} + \nabla_{\theta} \log p_{\theta}(x)|_{\theta_t}] \\ &= \nabla_{\theta} \mathbb{E}_{p_{\theta_t}(z|x)} [\log p_{\theta}(z|x)]|_{\theta_t} + \nabla_{\theta} \log p_{\theta}(x)|_{\theta_t} \\ &= 0 + \Delta_{\theta_t}(x). \end{aligned}$$

Additionally, a quicker more direct way to obtain (17) uses the “score trick” as follows

$$\nabla \log p_{\theta}(x) = \frac{1}{p_{\theta}(x)} \nabla_{\theta} \int p_{\theta}(z, x) dz = \frac{1}{p_{\theta}(x)} \int p_{\theta}(z, x) \nabla_{\theta} \log p_{\theta}(z, x) dz = \mathbb{E}_{p_{\theta}(z|x)} [\nabla_{\theta} \log p_{\theta}(z, x)].$$

## B. Method details

### B.1. Alternative gradient models

The gradient model needs to retain dependence on model parameter  $\theta$ . This ensures that the prediction, an estimate of  $J_{\theta}$ , can be used to compute  $\Delta_{\theta_t}(x)$  by differentiating w.r.t.  $\theta$ . KRR satisfies this condition in a attractive way, because its prediction depends on  $\theta$  and gradient parameters  $\gamma$  in a separable way, see (12). However, though theoretically consistent, KRR estimates the gradient at a cost of  $N^3$  in memory and time where  $N$  is the number of sleep samples. We discuss two alternative gradient estimation schemes that could potentially be much faster.

#### B.1.1. GENERIC FUNCTION APPROXIMATOR

For a generic function estimator, such as a neural network, the dependence on  $\theta$  will be encapsulated into  $\gamma$  in a nonlinear fashion. If this estimator is found by stochastic gradient descent procedures on MSE loss (12) over  $\gamma$ , evaluating  $\nabla_{\theta} \hat{J}_{\theta, \gamma}|_{\theta_t}$  will involve differentiating w.r.t.  $\gamma$  that has been updated by this iterative gradient descent procedure. This can be implemented, but is unlikely to be as straightforward as the KRR gradient model.

#### B.1.2. PARTICLE ESTIMATOR

The prediction of the KRR, though consistent in the limit of infinite number of samples, may not be a valid expected quantity. In other words, the estimate given  $x$  may not correspond to the expectation of any valid probability distribution, or

empirical mean of a set of posterior particles. To solve this issue, we can approximate the  $\Delta_{\theta_t}(\mathbf{x})$  through a set of particles  $\mathbf{z}'$  in the space of  $\mathbf{z}'$  generated from a simulator  $S_\gamma : \mathbf{x}, \mathbf{n} \rightarrow \mathbf{z}'$  where  $\gamma$  are parameters,  $\mathbf{x}$  is an observation,  $\mathbf{n}$  is a noise source and  $\mathbf{z}'$  is a particle in the same space as  $\mathbf{z}$ . For any given  $\mathbf{x}$ , we want the simulator to produce particles such that its average  $\mathbb{E}_{p(\mathbf{n})}[\nabla_{\theta} \log p_{\theta}(S(\mathbf{x}, \mathbf{n}), \mathbf{x})] \Big|_{\theta_t}$  is a good estimate of  $\nabla_{\theta} \log p_{\theta}(\mathbf{z} | \mathbf{x})$ . This can be achieved by optimising  $S_\gamma$  by solving

$$\min_{\gamma} \mathbb{E}_{p(\mathbf{z}, \mathbf{x})} \left[ \left\| \mathbb{E}_{p(\mathbf{n})}[\nabla_{\theta} \log p_{\theta}(S_\gamma(\mathbf{x}, \mathbf{n}), \mathbf{x})] \Big|_{\theta_t} - \nabla_{\theta} \log p_{\theta}(\mathbf{z}, \mathbf{x}) \Big|_{\theta_t} \right\|^2 \right],$$

which is equivalent to

$$\min_{\gamma} \mathbb{E}_{p(\mathbf{x})} \left[ \left\| \mathbb{E}_{p(\mathbf{n})}[\nabla_{\theta} \log p_{\theta}(S_\gamma(\mathbf{x}, \mathbf{n}), \mathbf{x})] \Big|_{\theta_t} - \mathbb{E}_{p_{\theta_t}(\mathbf{z} | \mathbf{x})} [\nabla_{\theta} \log p_{\theta}(\mathbf{z}, \mathbf{x}) \Big|_{\theta_t}] \right\|^2 \right].$$

By the argument of MSE regression estimating the posterior mean,  $\mathbb{E}_{p(\mathbf{n})}[\nabla_{\theta} \log p_{\theta}(S_\gamma(\mathbf{x}, \mathbf{n}), \mathbf{x})] \Big|_{\theta_t}$  should be a good estimator of  $\nabla_{\theta} \log p_{\theta}(\mathbf{z} | \mathbf{x})$ . We know that the optimal set of particles is posterior samples, but minimising the cost above does not necessarily drive  $S_\gamma$  to produce posterior samples. Nonetheless, this set of particles is adequate to approximate  $\Delta_{\theta_t}(\mathbf{x})$ .

In the limit of infinite number of parameters in  $\theta$ , minimising the loss above in related to implicit variational inference. The loss above essentially minimises the distance between the posterior expectation of infinitely many function at  $\theta = \theta_t$ . If, for all  $\theta$  and  $\mathbf{x}$ ,  $\nabla_{\theta} \log p_{\theta}(\mathbf{z}, \mathbf{x})$  is contained in some function spaces supported on the union of the spaces of  $\mathbf{z}$  and  $\mathbf{z}'$ , the loss, when averaged by the number of parameters, is upper bounded by integral probability metrics. For examples, if all the dimensions of  $\nabla_{\theta} \log p_{\theta}(\mathbf{z}, \mathbf{x})$  are in an RKHS, the average loss is upper bounded by the corresponding maximum mean discrepancy (Sriperumbudur et al., 2010). On the other hand, if the gradients are 1-Lipschitz, then the average loss is upper bounded by the Wasserstein distance.

However, solving this MSE regression suffers the same issue with solving (10), which requires known  $\nabla_{\theta} \log p_{\theta}(\mathbf{z}_n, \mathbf{x}_n)$  for  $(\mathbf{z}_n, \mathbf{x}_n) \sim p_{\theta}(\mathbf{z}, \mathbf{x})$ . Nonetheless, one may hope to train  $S_\gamma$  to estimate  $J_{\theta}(\mathbf{x})$  by  $\hat{J}_{\theta, \gamma}^I \approx \mathbb{E}_{p(\mathbf{n})}[\log p_{\theta}(S_\gamma(\mathbf{x}, \mathbf{n}), \mathbf{x})] \Big|_{\theta_t}$ , which can be found by minimising the following MSE:

$$\min_{\gamma} \mathbb{E}_{p(\mathbf{x})} \left[ \left\| \mathbb{E}_{p(\mathbf{n})}[\log p_{\theta}(S_\gamma(\mathbf{x}, \mathbf{n}), \mathbf{x})] \Big|_{\theta_t} - \log p_{\theta}(\mathbf{z}, \mathbf{x}) \Big|_{\theta_t} \right\|^2 \right]. \quad (18)$$

and use  $\nabla_{\theta} \hat{J}_{\theta, \gamma}^I(\mathbf{x}) \Big|_{\theta_t}$  as an estimate of  $\Delta_{\theta_t}(\mathbf{x})$ . Note that,  $S_\gamma$  is trained to produce a set of particles that, after evaluating  $\nabla_{\theta} \log p_{\theta}$  and taking the average, yields good estimator of  $\nabla_{\theta} \hat{J}_{\theta, \gamma}^I \Big|_{\theta_t}$ , for all possible  $\mathbf{x} \sim p_{\theta}$ . The simulator  $S_\gamma$  itself is not a gradient model. However, there is no theoretical guarantee that  $\nabla_{\theta} \hat{J}_{\theta, \gamma}^I \Big|_{\theta_t}$  is close to  $\Delta_{\theta_t}(\mathbf{x})$ . We shall refer to this scheme as amortised learning by particles (AL-P), and we show samples trained by this model from Figure 15 to Figure 20 in section Appendix C.7.

#### B.1.3. RELATIONSHIP BETWEEN KRR GRADIENT MODEL AND IMPORTANCE SAMPLING

The KRR gradient model approximates  $\Delta_{\theta_t}(\mathbf{x})$  by linearly weighting a collection of function values evaluated at sleep samples. This is similar to other reweighting schemes, with the most simple one being importance sampling where the proposals are from the prior  $p_{\theta}(\mathbf{z})$  and the weights are normalized density ratios  $p_{\theta}(\mathbf{z}, \mathbf{x})/p_{\theta}(\mathbf{z})$ . However, importance sampling estimation has enormous variance and requires at least exponentially many samples as the KL divergence between the posterior and prior (Chatterjee et al., 2018).

It would appear that the KRR estimate should perform similarly with importance sampling in estimating  $\Delta(\mathbf{x})$ , but in fact KRR uses slightly different information. KRR uses a set of samples  $(\mathbf{z}_n, \mathbf{x}_n) \sim p_{\theta}(\mathbf{z}, \mathbf{x})$ , whereas importance sampling uses  $\mathbf{z}_n \sim p_{\theta}(\mathbf{z})$  and  $p_{\theta}(\mathbf{z}, \mathbf{x})$ . In computing the weights for a particular observation  $\mathbf{x}^*$ , KRR compares  $\mathbf{x}^*$  with all sleep samples  $\{\mathbf{x}_n\}_{n=1}^N$  with the similarity metric determined by the kernel function, taking into account of the similarities between all sleep samples. Importance sampling, though has access to the joint density, uses  $p_{\theta}(\mathbf{z}, \mathbf{x}^*)$  for a given  $\mathbf{x}^*$  and computes the weights for each sample of  $\mathbf{z}$  independently of each other.

In addition, the weights from importance sampling are constrained to be non-negative and sum up to one, whereas the weights in KRR are not constrained and thus can be more flexible.

## B.2. Kernel architecture

In all experiments, we used a squared-exponential kernel  $k(\mathbf{x}, \mathbf{x}') = \exp(-0.5\|\phi_{\mathbf{w}}(\mathbf{x}) - \phi_{\mathbf{w}}(\mathbf{x}')\|_2^2/\sigma^2)$ .  $\phi_{\mathbf{w}}$  can be the identity function, a linear projection, or linear projection followed by batch normalisation, see Table 1 which lists the architecture used for each experiment. The linear projection and batch normalisation are primarily used on high-dimensional benchmark datasets. Nonlinear projections, such as deep nonlinear networks, did not give significant improvement. The length scale  $\sigma$  is initialised as the median of the distance between sleep samples, or projections thereof.

## C. Experimental details

We list the model and training parameters used to run each experiment in Table 1. The scale of the Gaussian kernel is always initialised to be the median distance between the inputs to the (data or output of linear projection). The batch size is 100 except for dynamical models and neural process where the batch size is 1.

### C.1. Gradient estimation

The toy generative model has  $z_1, z_2 \sim \mathcal{N}(0, 1)$ ,  $x|z \sim \mathcal{N}(\text{softplus}(\mathbf{w} \cdot \mathbf{z}) - \|\mathbf{w}\|_2^2, \sigma_2^2)$ . The observations are 100 samples for  $w_1 = w_2 = 1$ ,  $\sigma = 0.1$ . Note that the solution for this synthetic problem is not unique.

For amortised learning, we used 5000 sleep samples to train the gradient model. The length scale is set and fixed by median heuristic for each point on the grid, and regularisation  $\lambda$  is fixed at 0.01. The number of sleep samples is smaller if the kernel is adaptive.

For variational learning, the approximate posterior is factorised Gaussian and optimise ELBO over the mean and variance, and approximated the gradient of ELBO by sampling from the factorised Gaussian. The mean and variances are initialised as the standard Gaussian and are optimised by Adam with step size 0.01 for 300 iterations, which is sufficient for convergence. For ground truth, we estimated the gradient by importance sampling, with  $5 \times 10^4$  samples proposed from the prior.

### C.2. Spherical prior

The Gabors images are of size  $16 \times 16$  generated by sampling its orientation uniformly over 0 to  $\pi$ . The generative network is the DCGAN with just the first two deconvolutional layers so that the output size is  $16 \times 16$ . For VAE, we used the symmetric convolutional neural network and a normal posterior. For  $\mathcal{S}$ -VAE, a von Mises-Fisher distribution is used as the posterior.

### C.3. Hierarchical models

The penalty assigned to  $\mathbf{m}$  is the log pdf of a Dirichlet prior with  $\log p(\mathbf{q}) = (\alpha - 1) \sum_i \log q_i + \text{const}$ , where  $q_i = e^{m_i} / \sum_j e^{m_j}$ . We use  $\alpha = 0.999$ . Similarly, for the  $k$ 'th component in the mixture, the Normal-InverseWishart distribution has log likelihood that penalises  $\|\mu_k\|$ ,  $\log |\Sigma_k|$  and  $\text{Tr}(\Sigma_k^{-1})$ . In addition, we also penalise the  $l_2$  norm of neural network weights. These penalisation strength are set to  $10^{-4}$ .

The three-way maximum mean discrepancy (MMD) test (Bounliphone et al., 2016) is used for model comparison based on generated samples. Denote the set of real data by  $\mathcal{D}$  and the the set of generated samples from model A by  $\mathcal{D}^A$ . The null hypothesis for this test is  $\text{MMD}(\mathcal{D}, \mathcal{D}^{\text{SIN}}) < \text{MMD}(\mathcal{D}, \mathcal{D}^{\text{ALWS}})$ , where MMD is the MMD distance between two sets of samples. The test returns a  $p$ -value of 0.514 for 1500 samples from each of the three distributions, suggesting that the two models perform almost equally well on learning this data distribution. We note that SIN is trained on a full Bayesian version of the model, and the samples are reconstructions given the real dataset using posterior representations, giving an advantage for SIN.

### C.4. Parameter identification

#### C.4.1. INDEPENDENT COMPONENTS

The basis are generated by the FastICA algorithm, with mean subtracted and normalised to have unit length. The synthesised dataset is standardised by subtracting the mean and dividing by the standard deviation. The kernel is augmented with an adaptive linear neural network feature with 300 outputs. Using 200 features produces visibly the same results. The

Experiment		data dim	$M(\text{data})$	$N(\text{sleep})$	$L(\text{val})$	$\lambda$	# proj	batch norm?	gen lr	grad lr	nepoch
gradient estimation		1	100	5,000	–	0.01(f)	–	no	–	–	–
spherical prior	256	10,000	2,000	200	0.01	300	no	0.001	0.001	30	
pinwheel	2	2,500	1,000	200	0.01	–	no	0.001	0.001	2,500	
Independent component	256	100,000	2,000	200	0.001(f)	300	no	0.001	0.01	100	
Matrix factorisation	784	5,000	2,000	150	0.001	300	yes	0.001	0.001	300	
neural process	8	10,000	4,000	200	0.001	–	no	0.0001	0.0001	50	
nonlinear oscillation	600	1	2,000	200	0.001	200	no	0.001	0.001	5,000	
Hodgkin-Huxley	1,000	1	2,000	200	0.001	200	no	0.001	0.001	50,000	
ecology	180	1	2,000	200	0.001	–	no	0.001	0.001	50,000	
B-MNIST	1,024	60,000	2,000	200	0.1 (f)	300	yes	0.001	0.001	50	
MNIST	1,024	60,000	2,000	200	0.1 (f)	300	yes	0.001	0.001	50	
Fashion	1,024	60,000	2,000	200	0.1 (f)	300	yes	0.001	0.001	50	
Natural	1,024	100,000	2,000	200	0.1 (f)	300	yes	0.001	0.001	50	
CIFAR	3,072	50,000	2,000	200	0.1 (f)	300	yes	0.001	0.001	50	
CelebA	3,072	100,000	2,000	200	0.1 (f)	300	yes	0.001	0.001	50	

Table 1. Data properties, and model and training parameters of ALWS for each experiment. The regularisation strength  $\lambda$  is sometimes fixed as indicated by (f). See Appendix B.2 for kernel architectures.

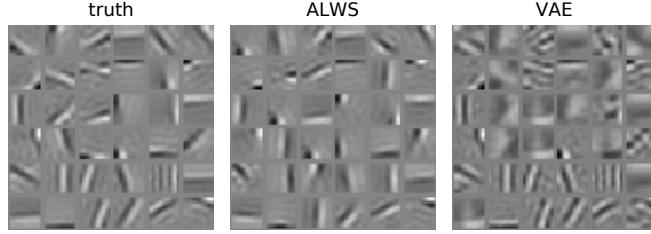


Figure 8. Same as Figure 5 but with  $\lambda$  adaptive.

regularisation strength is fixed at  $\lambda = 0.001$ . Making the filters adaptive results in slightly different filters as shown in Figure 8. This might be that  $\lambda$  is trained to approximate  $J_\theta$  instead of  $\nabla_\theta J_\theta$ , which in this particular problem has different smoothness compared to  $J_\theta$ .

#### C.4.2. BETA-GAMMA MATRIX FACTORISATION

Following (Ruiz et al., 2016), we implemented a probabilistic non-negative matrix factorisation model as follows

$$p(z_i) = \mathcal{U}(z_i|0, 1), \quad p(\mathbf{x}|\mathbf{z}) = \text{Bernoulli}(\mathbf{x}|\text{sigmoid}(\mathbf{W}\text{logit}(\mathbf{z}) + \mathbf{b}))$$

where the logit and sigmoid functions are applied element-wise. We put a penalty on  $\mathbf{W}$  that is consistent with a Gamma(0.9, 0.3) prior on each entry, and learn  $\mathbf{W}$  and  $\mathbf{b}$ . We compare with two other approaches. The first is a Bayesian version of the model learning by variational inference using generalised reparameterisation (G-Rep) Ruiz et al. (2016)<sup>6</sup>. The features  $\mathbf{W}$  are also random variables with the same Gamma prior as above, and  $\mathbf{b} = \mathbf{0}$ . The second is a VAE-like model. The decoder has the generative model as above. The encoder produces independent Beta as posteriors whose parameters are given by a neural network that takes  $\mathbf{x}$  (fully connected, 784  $\rightarrow$  512  $\rightarrow$  512 before feeding into the parameters). The reparameterisation is based on transforming samples from a Gamma distribution (see Ruiz et al. (2016)).

We evaluate the models on reconstructing and denoising handwritten digits from the binarised MNIST dataset. To reconstruct a given test input  $\mathbf{x}^*$ , we find the posterior mode (MAP) by maximising the log-joint likelihood  $\log p(\mathbf{z}, \mathbf{x}^*)$  over  $\mathbf{z}$ , and generate the  $\mathbf{x}$  given the optimal  $\mathbf{z}$ . The results for both tasks are depicted in Figure 9. The leftmost panel shows the histograms of MSE on 1,000 test images, and the other panels show examples of 25 test images and reconstructions by each method. ALWS achieved significantly lower error for both reconstruction and denoising ( $p < 10^{-10}$  for both a two-tailed  $t$ -test and a Wilcoxon signed-rank test).

#### C.5. Neural process

We briefly review the neural processes (NPs). Suppose there is a distribution over function  $f \sim \mathcal{P}(f)$ ,  $f : \mathcal{X} \rightarrow \mathcal{Y}$ . or a given function  $f$ , we observe its potentially noisy values at a set of inputs in the form of input-output pairs  $(\mathbf{x}, \mathbf{y})|f$ . The task is the following: given a set of context pairs  $\mathcal{D} := \{(\mathbf{x}_k^C, \mathbf{y}_k^C)\}_{k=1}^K$ , infer the distribution of the function value at a set of target inputs  $\{\mathbf{x}_m^T\}_{m=1}^M$ . NPs (Garnelo et al., 2018) represent the posterior of  $f$  given  $\mathcal{C}$  by a random variable  $\mathbf{z}$ , which is combined with  $\mathbf{x}_m^T$  to make predictions about the function value.

During training, the training data consists of sets of input-output pairs, and each set is always conditioned on one particular sample of  $f$ . The NP splits the each set into a context set  $\mathcal{C}$ , used to condition the representation  $\mathbf{z}$ , and a target set  $\{(\mathbf{y}_m^T)\}_{m=1}^M$ , used to evaluated the likelihood of  $\mathbf{y}_m^T$  given  $\mathbf{z}$  and  $\mathbf{x}_m^T$ . Formally, the generative model is specified by

$$\begin{aligned} \mathbf{r} &= \frac{1}{K} \sum_{k=1}^K \rho_\theta(\mathbf{x}_k^C, \mathbf{y}_k^C) \\ p_\theta(\mathbf{z}|\mathbf{r}) &= \mathcal{N}(\mathbf{z}|\mu_\theta^C(\mathbf{r}), \Sigma_\theta^C(\mathbf{r})) \\ p_\theta(\{\mathbf{y}_m^T\}|\{\mathbf{x}_m^T\}, \mathbf{z}) &= \prod_{m=1}^M \mathcal{N}(\mathbf{y}_m^T|\mu_\theta^T(\mathbf{z}, \mathbf{x}_m^T), \Sigma_\theta^T(\mathbf{z}, \mathbf{x}_m^T)). \end{aligned}$$

<sup>6</sup>Code kindly provided by the authors

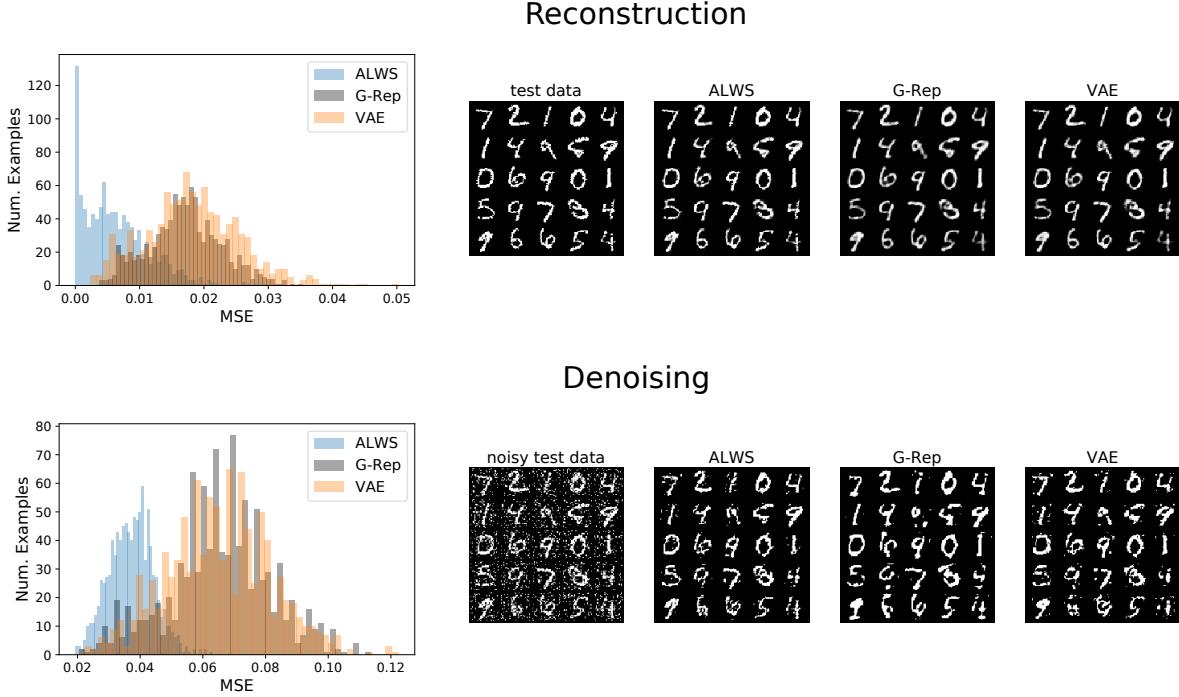


Figure 9. Beta-Gamma Matrix Factorisation. ALWS shows significantly lower MSE across 1,000 test inputs compared to advanced baselines (left), and produces more precise reconstructions (right) for both reconstruction and denoising.

In short, a latent representation of the context  $\mathbf{z}$  is drawn from a normal distribution with parameters formed by an exchangeable function of the context set  $\mathcal{C}$ , and the likelihood on the target outputs are i.i.d. Gaussian conditioned on this  $\mathbf{z}$  and  $\mathbf{x}_m^T$ . The objective for learning is to maximise the likelihood of the target output conditioned on the corresponding context set from the same underlying  $f$  and the target input. Once trained, the neural process is able to produce samples from the distribution of function values (or target output) at context inputs.

The encoding function  $\rho_\theta$  plays the role of an inferential model, but we can view it as a function that parametrises the ‘prior’ distribution on  $\mathbf{z}$  given the context set, and the parameters in  $\rho$  can be treated as in the generative model. The gradient model trained by KRR also needs to be conditioned on each context set, but for simplicity we train a gradient model for a single context followed by  $\theta$  update. Garnelo et al. (2018) trained the neural processes by maximising an ELBO with posteriors of the form

$$q(\mathbf{z}|\mathcal{C}, \mathcal{T}) = p_\theta(\mathbf{z}|\mathbf{r}^{CT}), \quad \mathbf{r}^{CT} = \frac{1}{K} \sum_{k=1}^K \rho_\theta(\mathbf{x}_k^C, \mathbf{y}_k^C) + \frac{1}{M} \sum_{m=1}^M \rho_\theta(\mathbf{x}_m^T, \mathbf{y}_m^T).$$

which is an approximation.

We trained a neural process on functions shown in Figure 10 (left). They are sinusoids with random amplitudes and phase shifts and supported on  $[-\pi, \pi]$ . The observations has additive noisy distributed as Gaussian with standard deviation of 0.1. Conditioning on context points with inputs around  $-\pi$ , 0.0 and  $\pi$  induces large uncertainty over  $f$ , and can be used to probe the representation of uncertainty.

In the NP model, the latent representation  $\mathbf{r}$ , latent  $\mathbf{z}$  are both 50-dimensional. And the encoding and decoding networks are fully connected with ReLU nonlinearities. During training, the number of context pairs  $K = 4$ , and the target set contains the context pairs and additional four pairs, so  $M = 8$ ,  $\mathcal{C} \in \mathcal{T}$ . The gradient model is trained for a each given context set, and hence the batch size is 1. A small learning rate 0.0001 is used for all models and parameters. The gradient model is trained to take sleep samples  $\mathbf{y}_m^T$  evaluated for this single  $\mathcal{C}$  at each  $\mathbf{x}_m^T$ . The kernel used is squared exponential with hyperparameters  $\gamma$ , and treats  $\{\mathbf{y}_m\}_{m=1}^8$  as a single vector. We note that other kernels on sets could be used, and can be extended to a batch size greater than 1 if the kernel also depends on  $\mathbf{r}$ .

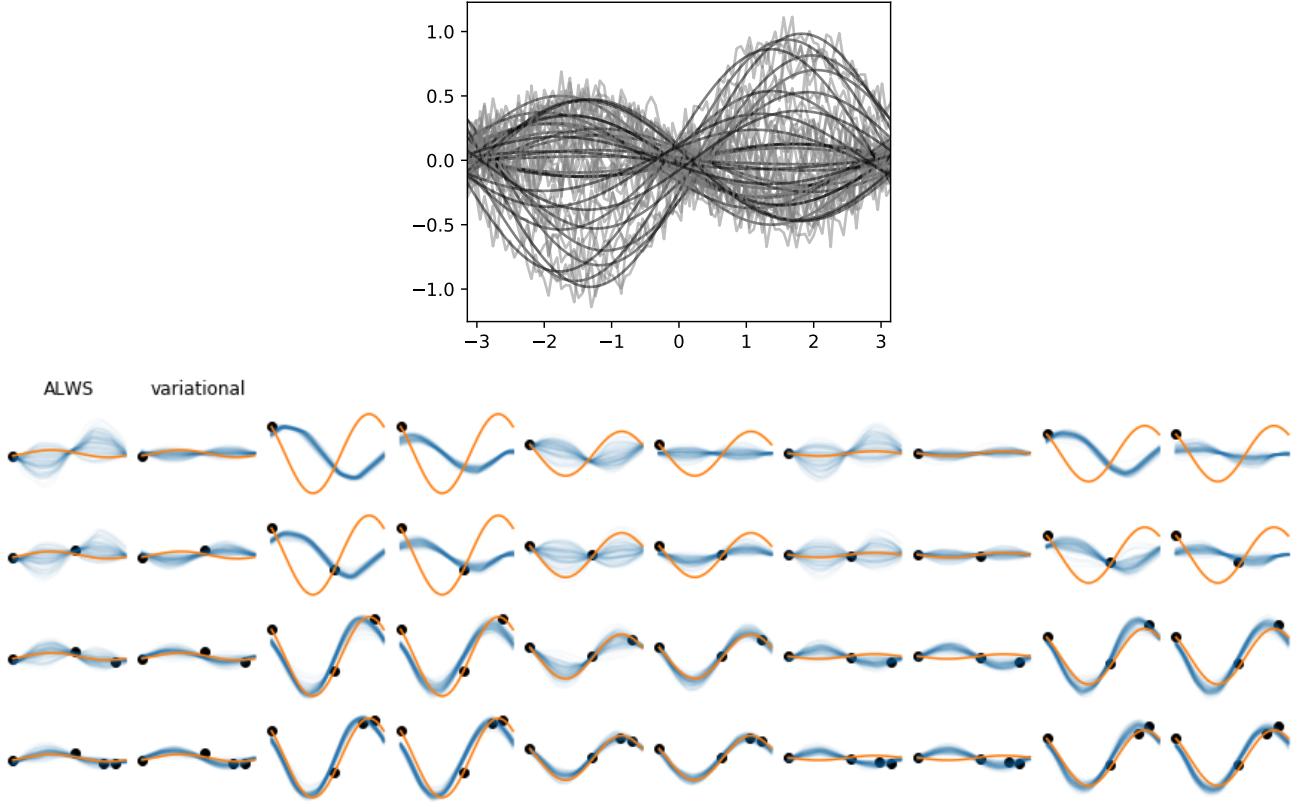


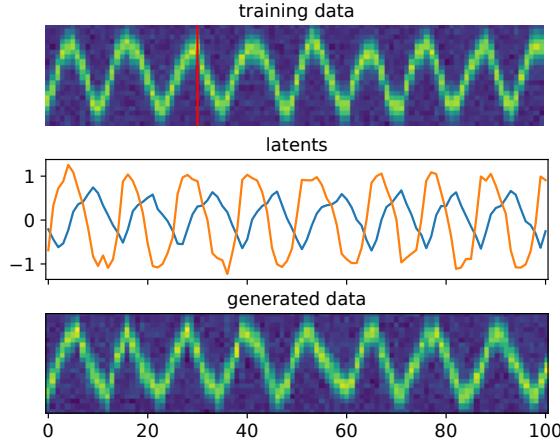
Figure 10. neural processes. Top: samples from prior distribution of functions. Black: Latent function. Gray: noisy observations Bottom: posterior samples (blue lines) from ALWS (odd columns) and the original variational method (even columns) for each true latent  $f$  (orange lines) given 1 to 4 noisy points (black dots) in the context set.

As shown in Figure 10, when the number of context points is small, the model trained with ALWS makes more accurate predictions (e.g. comparing 3rd and 4th columns), and better reflects the uncertainty when the context set is not informative (e.g. 1st and 2nd columns). In the case when there are four context pairs, which is the training condition, we generate 500 test cases and evaluate the how close the posterior samples of  $\mathcal{P}(f|\mathcal{C}, \mathbf{x}_m^T)$  are to the true function at  $M = 100$  target locations.

The errors are measured by mean (log) squared error when either the posterior mean or a random posterior sample is used as the prediction. The random sample prediction takes into account the correlation between the predictions. We find that the errors are significantly smaller for ALWS-trained model based on paired tests for the posterior mean prediction (paired t-test,  $t = -3.47, p = 0.00056$ ; mean of ALWS, -5.11; variational, -4.99. Wilcoxon test,  $W = 44837.0, p = 3.7 \times 10^{-8}$ , median of ALWS, -5.11, variational, -4.98) and the random sample prediction (paired t-test,  $t = -2.09, p = 0.037$ ; mean of ALWS, -4.87; variational, -4.77. Wilcoxon test,  $W = 53762.0, p = 0.0061$ , median of ALWS, -4.91, variational, -4.69).

### C.6. Nonlinear dynamic model

We ran ALWS for a model where  $p(z)$  is defined through a nonlinear dynamical model. In all of the experiments, we treat data at all time steps as a single observation. For longer sequences, the KRR gradient model can be augmented by a recurrent feature extractor.



*Figure 11.* Nonlinear oscillations. Top: a example trajectory. Only the first 30 time steps marked by the red line is used for training. Three such 30 time step traces are used for training. Middle: latent space learned by ALWS. Bottom: Generated trajectory.

#### C.6.1. NONLINEAR OSCILLATIONS

We generated data from a nonlinear sinusoids according to the following equations

$$\begin{aligned} \mathbf{z}_t &= \text{Rot}(\mathbf{z}_{t-1}) + \epsilon_t^{(z)} & \mathbf{x}_t &= \text{Img}(z_{t,0}) + \epsilon_t^{(x)} \\ \text{Rot}(\mathbf{z}_t) &= \mathbf{R}_\alpha \mathbf{z}_t \frac{r(\|\mathbf{z}_t\|_2)}{\|\mathbf{z}_t\|_2} & r(a) &= \text{sigm}(4(a - 0.3)) & \text{Img}(\mathbf{z}) &= \exp(-0.5(z_1 - z^{(0)})^2 / 0.3^2) \end{aligned}$$

where  $\mathbf{R}_\alpha$  is a rotation matrix of  $\alpha$ ,  $\text{Img}()$  maps one of the latent dimensions into a 1-D image through Gaussian bumps, and  $\text{sigm}()$  is the sigmoid function. Intuitively, the latent  $\mathbf{z}$  is rotated by  $\alpha$  and scaled radially so that its length remains close to 1. An example of this trajectory is shown in Figure 11 (top). We train a generative model of the following form

$$p_\theta(\mathbf{z}_t | \mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t | \text{NN}_{\mathbf{w}}^{(z)}(\mathbf{z}_{t-1}), [v_1^{(z)}, v_2^{(z)}]) \quad p_\theta(\mathbf{x}_t | \mathbf{z}_t) = \mathcal{N}(\mathbf{x}_t | \text{NN}_{\mathbf{w}}^{(x)}(\mathbf{z}_t), [v_1^{(x)}, v_2^{(x)}])$$

where the parameters are the weights and biases in the neural networks (NN) and the variances  $v$ 's. The number of units are fully connected with  $2 \rightarrow 20 \rightarrow 2$  neurons for  $\text{NN}^{(z)}$  and  $2 \rightarrow 20 \rightarrow 20$  for  $\text{NN}^{(x)}$ . The tanh is used as the nonlinearity.

#### C.6.2. HODGKIN-HUXLEY (HH) EQUATIONS

The HH equations are described by

$$\begin{aligned} C_m \dot{V}(t) &= -g_l[V(t) - E_l] - \bar{g}_N m^3(t) h(t)[V(t) - E_N] - \bar{g}_K n^4(t)[V(t) - E_K] + I_{in}(t) + \epsilon(t) \\ \dot{e}(t) &= \alpha_e(V(t))[1 - e(t)] - \beta_e(V(t))e(t), \quad e \in \{m, h, n\} \end{aligned}$$

where  $\alpha_e$  and  $\beta_e$  are nonlinear functions of  $V(t)$  involving a parameter  $V_T$  that sets the threshold for action potentials, see (Pospischil et al., 2008) for details.

We used forward-Euler for simulation with  $\Delta t = 0.05ms$ . At each step of simulation we add a small Gaussian noise of standard deviation  $\sigma_z = 0.1$  to  $V$  as process noise so that  $p_\theta(\mathbf{z}_t | \mathbf{z}_{t-1})$  can be calculated. The measurements noise is Gaussian with  $\sigma_x = 1.0(mV)$ , and is added to observed values of  $V$  so that  $p_\theta(\mathbf{x}_t | \mathbf{z}_t)$  can be calculated. Thus, the 10 parameters for the resulting discrete-time state-space model are  $\theta = \{C_m, g_l, E_l, \bar{g}_N, E_N, \bar{g}_K, E_K, V_t, \sigma_z, \sigma_x\}$ .

Before training, we simulated a single trajectory with some true parameters. We then change these parameters such that it the simulated trajectories are not like neurons. The injected currents  $I_{in}$  are independent Gaussian noises with mean 0.0 and std 3.0 for 0-99 steps, mean 10.0 and std 3.0 for 100-899 steps, and mean 0.0 and std 3.0 for 900-999 steps. The results are

shown in Figure 12. After training, we simulate trajectories from the model given learned parameters and the training input current. They simulated trajectories are almost identical and overlap with the training trajectory. We also tested the model given a different test current. The simulated trajectories are very similar to the real trajectories given true parameters, except that the former have less variation between trajectories. Since the model is trained on a single trajectory, it is expected that the learned parameters will be different from the real ones.

#### C.6.3. ECOLOGICAL DATA

To test on real data, we train a model that describes the evolution of blowfly population size under adult food limitation (Wood, 2010), which was used to evaluated K2-ABC (Park et al., 2016). The model is described by

$$\begin{aligned} \tau &\sim \text{Categorical}(\mathbf{m}), \tau \in \{1, \dots, 20\}, \quad e_t \sim \text{Gamma}\left(\frac{1}{\sigma_p^2}, \sigma_p^2\right), \quad \epsilon_t \sim \text{Gamma}\left(\frac{1}{\sigma_d^2}, \sigma_d^2\right), \\ z_t &= P x_{t-\tau} \exp\left(-\frac{x_{t-\tau}}{N_0}\right) + x_t \exp(-\delta \epsilon_t), \quad p(x_t|z_t) = \text{LogNormal}(\log(z_t), \sigma_n^2) \end{aligned}$$

Note that  $\tau$  is a discrete delay time step drawn from a categorical distribution with logit parameters  $\mathbf{m}$ ,  $e_t$  and  $\epsilon_t$  are stochastic variations for births and deaths following Gamma distribution with a common mean 1.0 and standard deviations  $\sigma_p^2$  and  $\sigma_d^2$ , respectively. The variable  $z$  is deterministic given observation,  $\tau$  and the Gamma variables, and observation is noisy with log normal noise so that  $x_t$  remains positive. The log normal noise ensures that a joint density can be defined. Since the first data point depends on some past data that is not observed, we modelled these past data  $x_{-20:-1}$  that are constrained to be between 0 and 1.0. Thus, this model has parameters  $\{\mathbf{m}, \sigma_d, \sigma_p, P, N_0, \delta, \sigma_n, x_{-20:-1}\}$

We fit the model on a data sequence of length 180, normalised to be between 0 and 1.0. The evolution of parameters are in Figure 13. As our training objective is different from that of ABC, we do not make direct quantitative comparison with ABC methods. Nonetheless, compared with the samples from three ABC methods shown in (Park et al., 2016) (Figure 2B), it is clear that our model learns the correct time lag and tracks the trajectory better, especially at the troughs of each cycle.

### C.7. Sample quality on benchmark datasets

#### C.7.1. DATA PROCESSING

All images have  $32 \times 32$  pixels which is either their original size (Natural, CIFAR-10), or by zero-padding (MNIST,F-MNIST) or interpolation (CelebA). The binarised MNIST is statically binarised once before training. Each pixel is set to 1 with probability equal to the pixel value after rescaling to between 0 and 1, and to 0 otherwise. The original MNIST and Fashion MNIST are rescaled to -1.0 and 1.0. The natural images<sup>7</sup> are patches from large natural scenes. It is contrast-normalised within each image and further standardised to zero mean and unit standard deviation along each dimension using. No clipping is applied. CIFAR-10 and CelebA images are rescaled to between -1.0 and 1.0.

#### C.7.2. MODEL AND TRAINING DETAILS

#### C.7.3. RESULTS FOR MODELS WITH RICH ARCHITECTURE

All methods use the same neural network as the DCGAN without the last convolutional layer to make the image size 32. Batch size is 100 for each generative parameter update. We run each algorithm on each dataset with 10 different initialisations. For binary MNIST, a sigmoid nonlinearity is used in the last layer. For natural images, no nonlinear activation is used. For other datasets, a tanh activation is used. Intermediate layers have ReLU nonlinearity. All methods except SIVI are trained for 50 epochs. The optimizer is Adam with fixed learning rate 0.001 and other parameters as default values. For ALWS, we use 2,000 sleep samples for training the gradient model. The kernel is augmented by linear projection to 300 dimensions for all datasets. Higher number of output dimension produced better results, but induces longer runtime. For ALWS-F, a fixed random projection is used throughout training. For ALWS-A, the linear weights are training at each parameter update after 5 epochs, using the two-stage training.

For VAE, we implement the inferential network to be symmetrical to the generative network and a final linear layer for posterior statistics.

For Syl-VAE. We change the gated convolutional layer in the generative network to the same network as all other methods.

<sup>7</sup>[github.com/hunse/vanhateren](https://github.com/hunse/vanhateren)

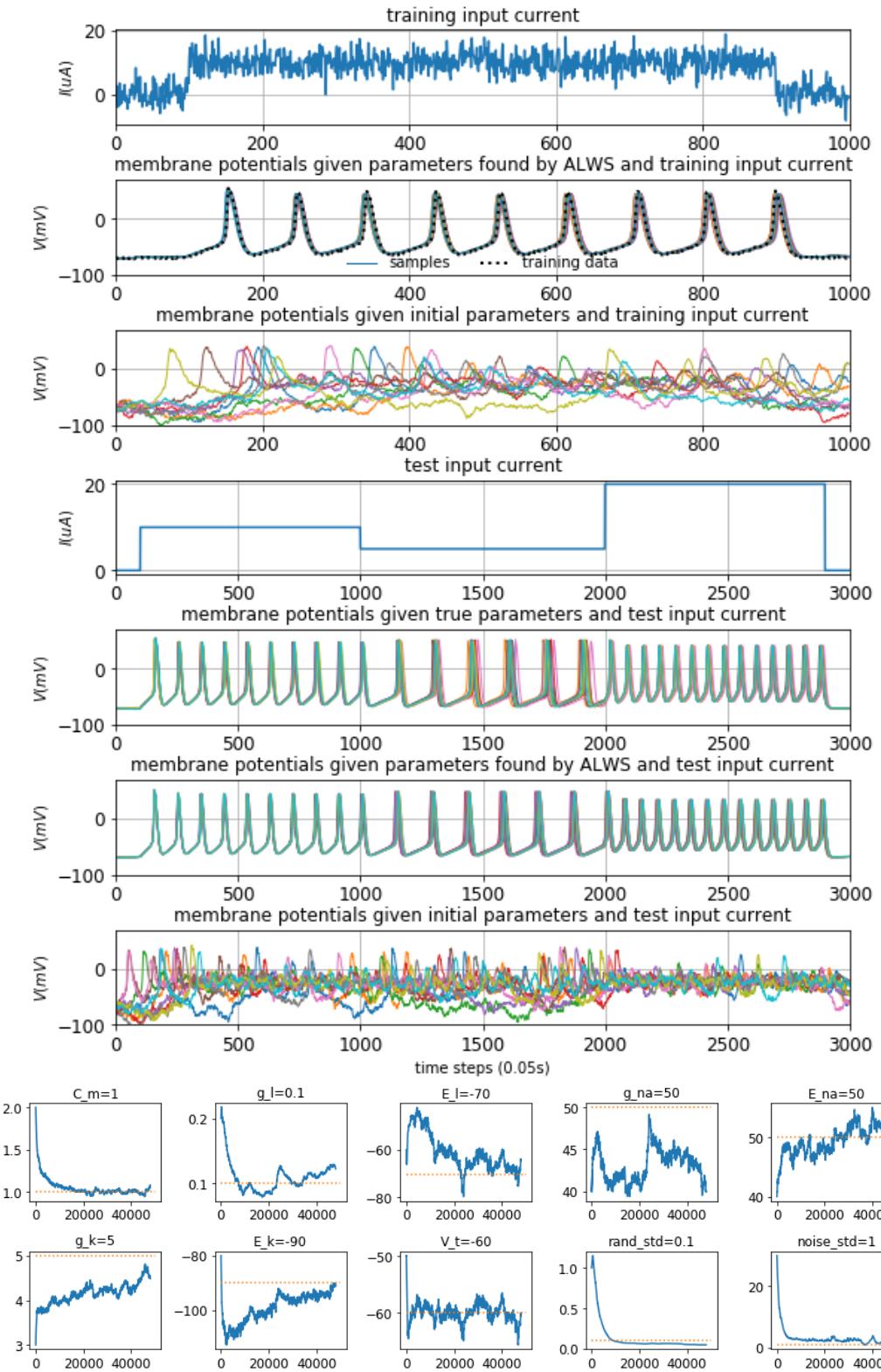


Figure 12. Hodgkin Huxley simulations. Top seven panels: First row, input current  $I_{in}$  during test time. Second to fourth row, trajectories for the learnt, true and initial parameters. Lower 10 panels: Blue: parameter value at each iteration. Yellow dashed: true parameter values

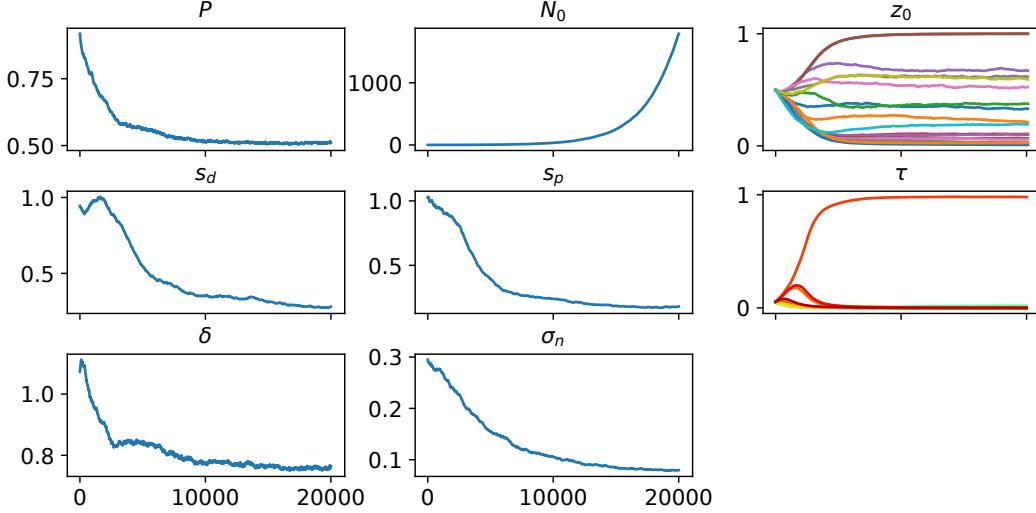


Figure 13. Evolution of parameters in the ecological model for blowfly population.

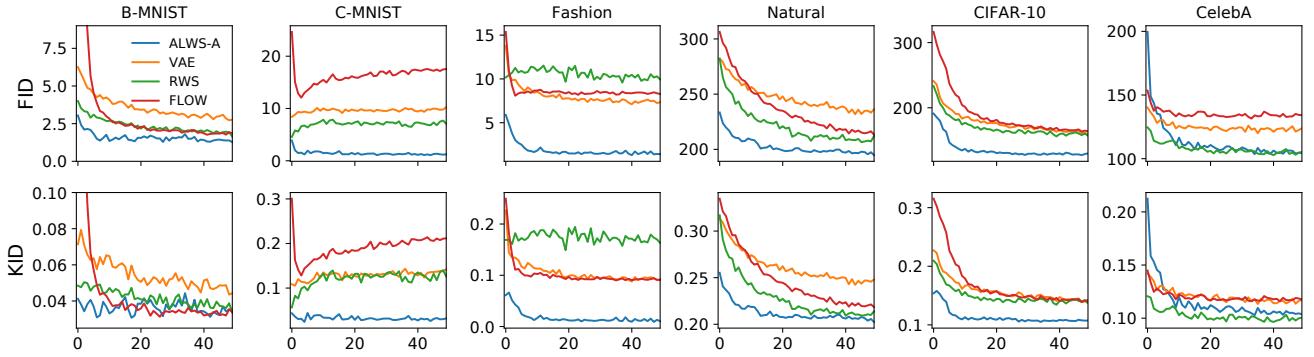


Figure 14. KID and FID scores at the end of each epoch for selected algorithms on convolutional architecture

Other parts of the model remain the same. We use orthogonal flow option in the code. A smaller learning rate of 0.0005 is used. A higher rate causes instability.

For SIVI, we find the model is unstable for learning rate of 0.001, so we change to 0.0001. It also takes more epochs to converge, and we train for 1000 epochs. We used  $J = 10$  proposals from the Gaussian posterior.

For RWS, each parameter update is accompanied with both wake and sleep updates of the inferential parameters, using  $K = 50$  proposals.

For WGAN-GP, learning is unstable for learning rate of 0.001, so we train the model using learning rate of 0.0001 for 50 epochs, which was not sufficient for it to converge. We also ran WGAN-GP for 500 epochs on all datasets and will show the samples from Figure 15 to Figure 20. We show the results of WGAN-GP just for reference, as it is not trained using the maximum likelihood objective.

TO evaluate the quality, we use standard metrics FID and KID, which are computed using features of penultimate layers of neural networks pre-trained on relevant datasets. For both MNISTs, the features are from the LeNet trained to classify MNIST digits. For Fashion, from LeNet trained to classify the objects. For Natural, CIFAR-10 and CelebA, we use inception network trained on ImageNet classification. For natural, we duplicate the image along the channel axis to fill the RGB channels.

ALWS has lower FID and KID than other maximum likelihood methods in most cases, especially on original MNIST and Fashion MNIST, but still does not reach the level of WGAN-GP.

The samples are shown from Figure 15 to Figure 20. These include samples from models presented in the main text, the WGAN-GP for 500 epochs, and the AL-P algorithm introduced in Appendix B.1.2.

## Amortised Learning by Wake-Sleep

real	[AL-F]	[AL-A]	VAE	SF-VAE	SIVI(1000)	RWS	WGAN-GP	WGAN-GP(500)	AL-P
5 5	5 3	1 3	1 3	1 3	2 6	2 6	8 1	6 7	5 3
2 0	9 9	3 4	4 5	4 5	7 3	8 2	6 7	4 1	4 3
0 4	2 3	2 5	3 4	5 3	2 2	2 0	7 3	0 8	2 5
7 7	8 2	3 7	8 7	7 8	8 2	8 2	2 3	2 6	2 7
0 3	3 1	7 7	7 3	5 2	2 5	2 3	7 1	7 0	3 8
4 2	1 6	9 8	4 5	7 4	8 9	9 2	7 8	1 1	3 0
9 5	8 3	7 6	1 2	2 4	8 8	0 9	8 4	8 8	4 8
1 4	7 9	1 6	3 7	1 9	5 2	3 7	3 6	4 1	3 7
9 4	7 7	6 9	5 3	2 8	2 4	7 2	8 5	9 3	3 6
3 1	5 7	0 4	1 4	5 8	7 3	9 3	0 1	0 1	9 1
1 2	1 2	1 8	8 3	5 0	2 5	9 8	5 4	0 9	5 5
6 7	8 3	8 8	7 7	8 0	5 5	8 4	3 6	3 3	0 7
4 9	2 5	4 5	9 7	5 2	7 4	2 9	9 4	9 5	5 5
4 6	4 3	2 9	5 3	9 3	2 6	0 4	6 6	4 3	2 3
3 5	3 8	3 8	5 2	3 0	0 7	2 5	8 3	4 4	3 9

Figure 15. Samples for B-MNIST. Our main algorithms presented in the main text are highlighted in box. Each model is trained for 50 epochs, except otherwise indicated in parenthesis next to algorithm name.

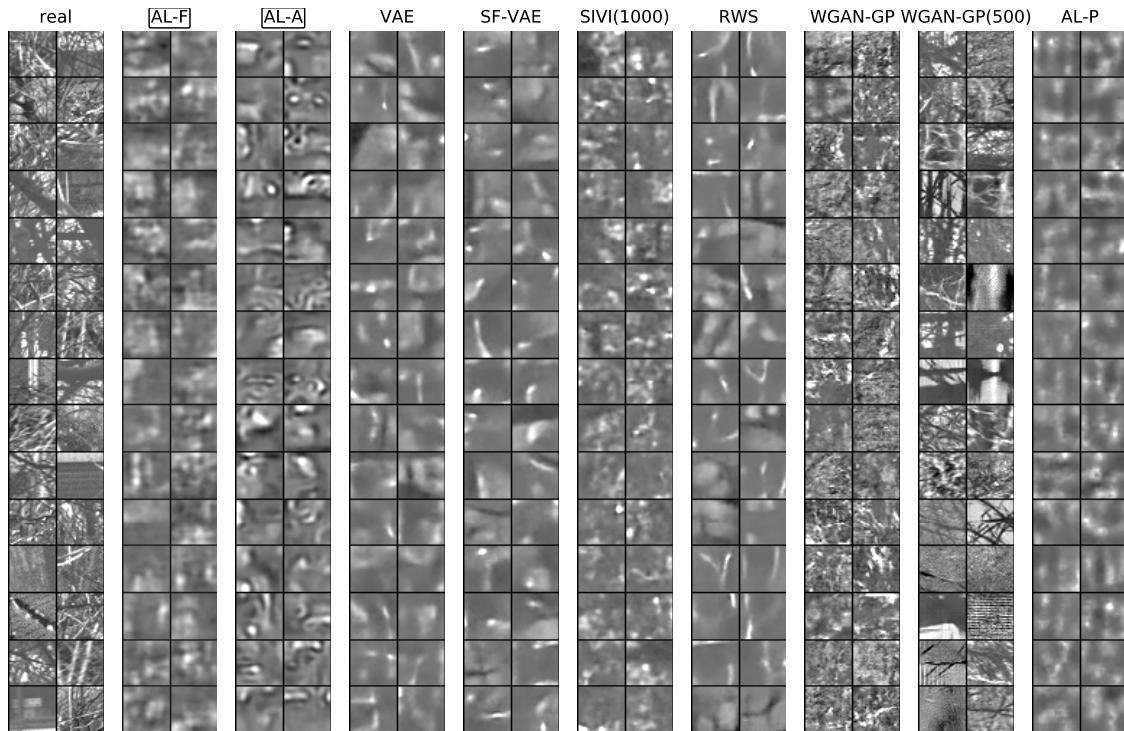
real	[AL-F]	[AL-A]	VAE	SF-VAE	SIVI(1000)	RWS	WGAN-GP	WGAN-GP(500)	AL-P
5 1	3 1	0 0	5 9	0 0	3 3	7 2	6 1	5 3	5 2
3 8	8 8	7 0	3 6	6 0	8 9	3 0	0 9	2 0	5 4
9 4	0 5	1 7	3 5	4 7	6 1	9 6	9 2	7 5	0 2
2 1	9 2	3 0	4 1	7 2	1 0	8 0	9 4	2 6	8 9
2 8	4 0	6 3	0 9	1 0	1 1	7 6	4 9	1 8	1 3
6 2	8 4	9 3	2 0	6 2	7 5	9 1	9 7	2 0	8 5
9 1	5 2	7 2	1 8	2 2	9 5	6 8	5 1	2 1	0 7
1 4	7 5	7 9	0 7	2 0	6 6	4 8	6 1	7 7	4 8
4 0	2 1	0 2	4 8	2 6	6 9	8 9	0 9	6 7	5 3
9 9	6 2	2 6	7 5	3 7	9 0	7 7	6 7	3 9	3 4
4 4	2 2	5 4	0 8	3 9	7 7	5 9	2 1	3 2	8 5
0 0	0 0	7 2	5 6	7 1	2 7	9 1	1 1	2 0	2 9
9 3	8 7	3 9	0 0	0 5	4 3	5 4	7 8	8 4	6 2
7 8	9 2	0 1	1 0	1 0	6 8	0 9	6 5	1 3	3 4
5 3	2 6	0 0	7 4	5 4	2 1	8 5	8 7	7 0	2 8

Figure 16. Samples for MNIST. Our main algorithm is highlighted in box. Each model is trained for 50 epochs, except otherwise indicated in parenthesis next to algorithm name.

### Amortised Learning by Wake-Sleep

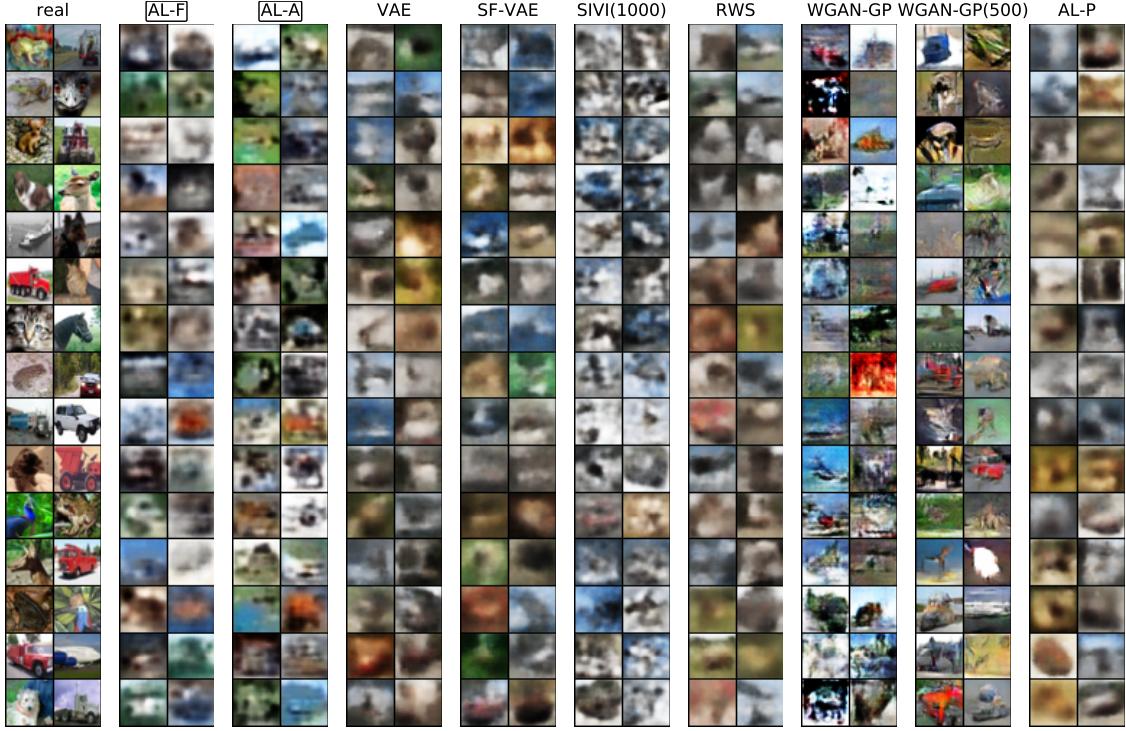


*Figure 17.* Samples for Fashion. Our main algorithms presented in the main text are highlighted in box. Each model is trained for 50 epochs, except otherwise indicated in parenthesis next to algorithm name.

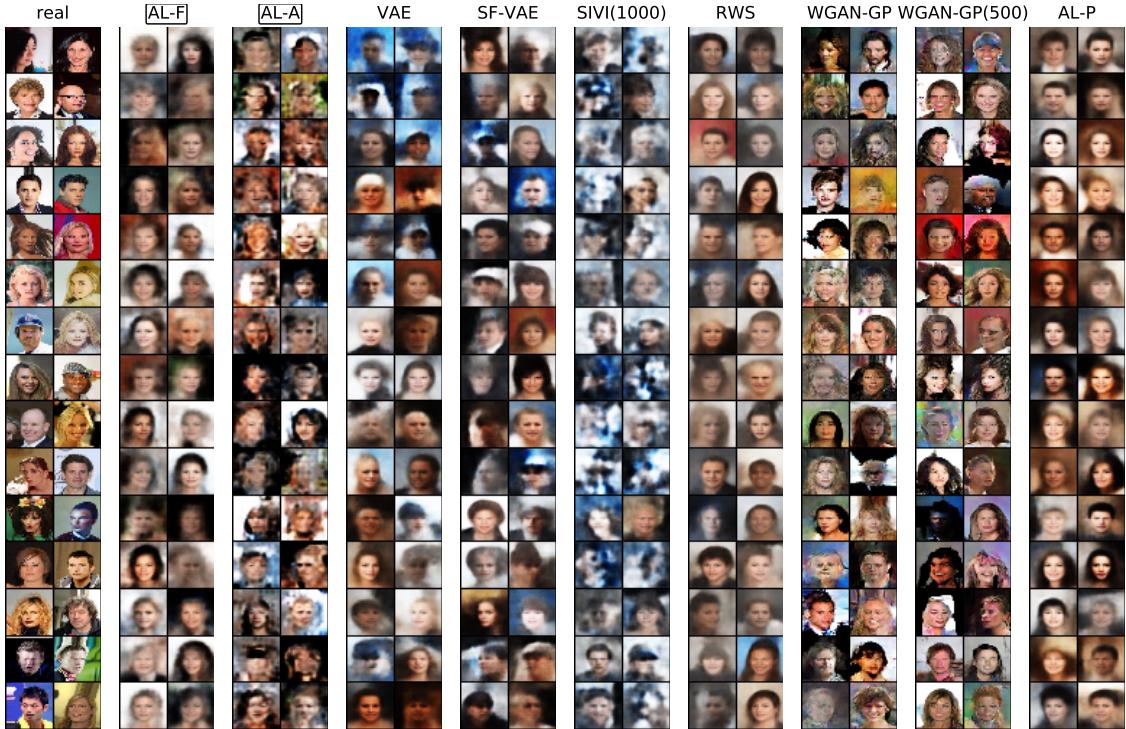


*Figure 18.* Samples for Natural. Our main algorithm is highlighted in box. Each model is trained for 50 epochs, except otherwise indicated in parenthesis next to algorithm name.

### Amortised Learning by Wake-Sleep



*Figure 19.* Samples for CIFAR-10. Our main algorithms presented in the main text are highlighted in box. Each model is trained for 50 epochs, except otherwise indicated in parenthesis next to algorithm name.



*Figure 20.* Samples for CelebA. Our main algorithms presented in the main text are highlighted in box. Each model is trained for 50 epochs, except otherwise indicated in parenthesis next to algorithm name.

## Amortised Learning by Wake-Sleep

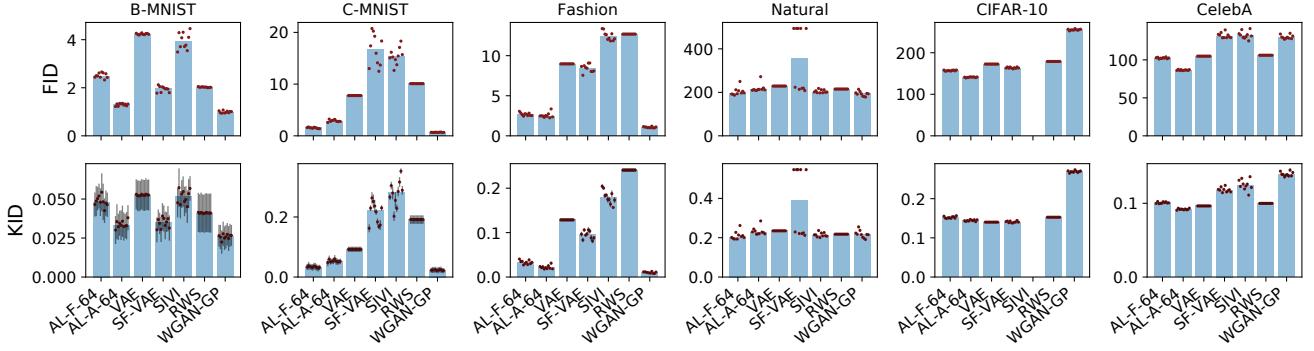


Figure 21. Same as Figure 21 but using fully connected networks. None of the SIVI runs on CIFAR-10 converge.

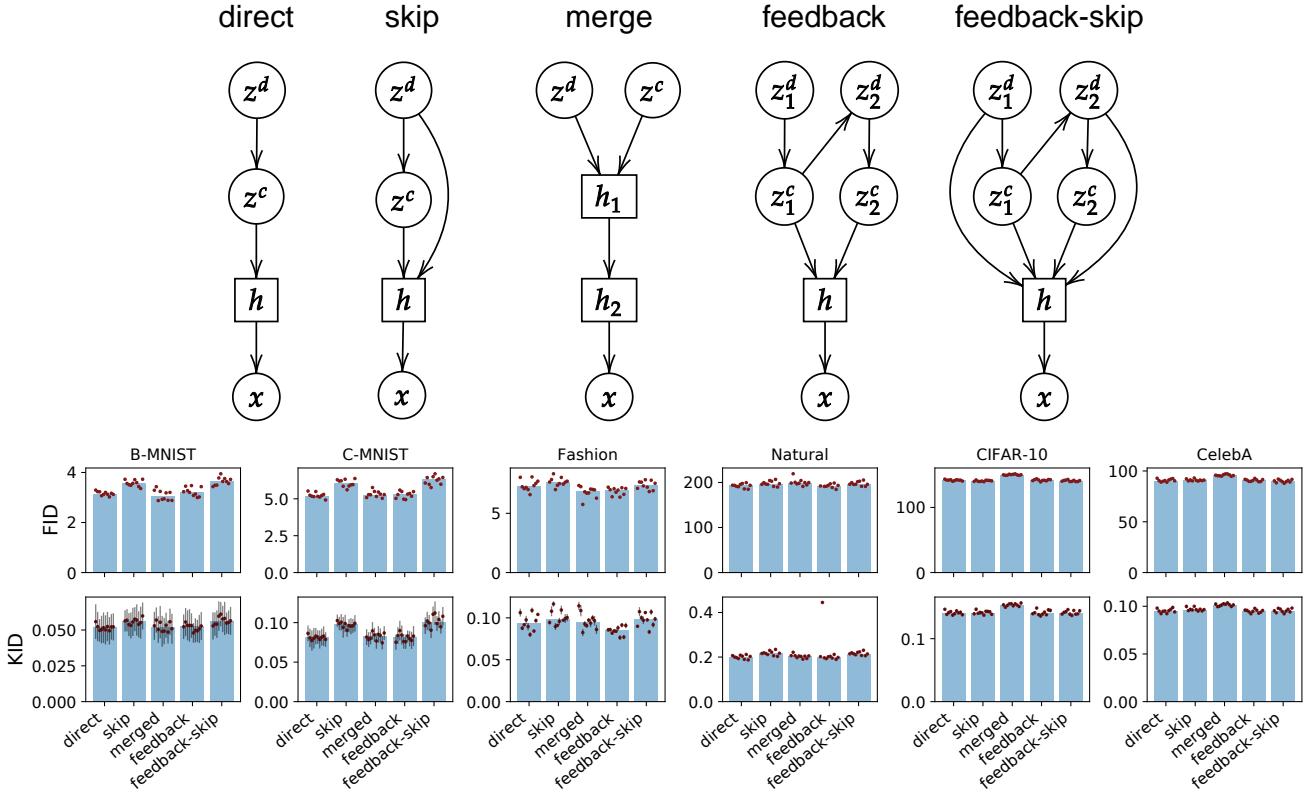


Figure 22. Top, graphical model of generative models. Circles indicate random variables, with discrete Bernoulli  $z^d$  and continuous Gaussian  $z^c$ . Squares indicate deterministic nodes that are ReLU's of fully connected activation from incoming nodes. The dimensionality of  $z^d$  is 10,  $z^c$  is 16,  $z_1^d$  and  $z_2^d$  is 5,  $z_1^c$  and  $z_2^c$  are both 8. The node  $h$  has 512 neurons. Bottom, FID and KID scores of generated images from architecturally complex models

### C.7.4. RESULTS WITH FULLY CONNECTED NETWORKS

We repeat the experiments for fully connected layers, with architecture  $16 \rightarrow 512 \rightarrow 512 \rightarrow$  image dimension, and ReLU nonlinearity. The results are shown in Figure 21. Models trained by ALWS visibly out-performs other maximum likelihood methods on all datasets except Natural according to FID. KID agrees with FID except on CIFAR-10.

### C.7.5. RESULTS WITH COMPLEX GENERATIVE NETWORKS

The goal is to test how model architecture affects the quality of generated samples on different datasets. Discrete variables can be used to capture features such as object category, so its inclusion in the generative model may be beneficial. In order to train models with discrete latent variables, explicit reparameterisation schemes have been developed in the past by continuous relaxation or overlapping transformation (Jang et al., 2017; Vahdat et al., 2018; Rolfe, 2017), and has shown differential performances. On the other hand, amortised learning is agnostic to the discrete or continuous nature of the latents.

We set out to explore different architectures while fixing the number of Bernoulli and Gaussian latent variables, respectively, and keep the number of parameters roughly the same. The different graphs are depicted in Figure 22 and described in the legend. The **direct** model is a simple chain graph. The top Bernoulli layer connects to a Gaussian layer where the mean is a function of the Bernoulli and variance is 1.0. The **skip** model adds an additional connection from the discrete latents to the hidden units. The **merged** model combines the Bernoulli and Gaussian latents at the top layer and goes through a first hidden  $h_1$  layer of 16 units before feeding into the wide  $h_2$  layer. The **feedback** model has an architecture inspired by (Vahdat et al., 2018). The latents  $z_1^c$  parametrises the logits for the  $z_2^d$  layer. The **feedback-skip** model is based on **feedback** and adds a skip connection to  $h$  from the top Bernoulli layer.

We did not find strong effect of model architectures on FID or KID. There is differential performance mainly in the two MNISTs and Fashion where the direct, merge and feedback tend to be better than skip and feedback-skip. On CIFAT-10 and CelebA, the merged model had the worse quality.