Amortized variance reduction for doubly stochastic objectives

Ayman Boustati[*]
University of Warwick
Coventry, UK

Sattar Vakili
PROWLER.io
Cambridge, UK

James Hensman[†]
Amazon
Cambridge, UK

ST John
PROWLER.io
Cambridge, UK

Abstract

Approximate inference in complex probabilistic models such as Deep Gaussian Processes requires the optimisation of doubly stochastic objective functions. These objectives incorporate randomness both from mini-batch subsampling of the data and from Monte Carlo estimation of expectations. If the gradient variance is high, the stochastic optimisation problem becomes difficult with a slow rate of convergence. Control variates can be used to reduce the variance, but past approaches do not take into account how mini-batch stochasticity affects sampling stochasticity, resulting in sub-optimal variance reduction. We propose a new approach in which we use a recognition network to cheaply approximate the optimal control variate for each mini-batch, with no additional model gradient computations. We illustrate the properties of this proposal and test its performance on logistic regression and Deep Gaussian Processes.

1 INTRODUCTION

Many machine learning tasks can be formulated as an optimisation problem, where the model parameters $\boldsymbol{\theta}$ are inferred by optimising an objective function $\mathcal{L} = \sum_{n=1}^{N} \ell_n(\boldsymbol{\theta})$ which is a sum over contributions from individual data points n. We focus on objectives containing an analytically intractable expectation, $\ell_n(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{\epsilon})}[f_n(\boldsymbol{\epsilon}, \boldsymbol{\theta})]$, e.g., Black Box Variational Inference (Ranganath et al., 2014), Variational Auto-Encoders (Kingma and Welling, 2014), or Deep Gaussian Processes (Salimbeni and Deisenroth, 2017).

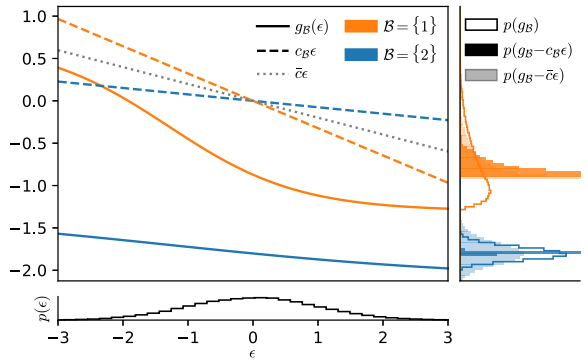


Figure 1: In reparameterized variational inference, the gradient is a function of the randomness sample $\epsilon \sim p(\epsilon)$. This relationship $g_{\mathcal{B}}(\epsilon)$ (solid lines) depends on the mini-batch $\mathcal{B}$ (orange vs blue). Here we show linear control variates with batch-dependent coefficients $c_{\mathcal{B}}\epsilon$ (dashed lines) and the best batch-independent control variate $\bar{c}\epsilon$ (dotted grey line). The right-hand plot shows the distribution of the expectation estimators for each mini-batch: no control variate (outline), batch-independent control variate (shaded), and batch-dependent control variate (filled). The batch-dependent control variate significantly reduces the variance, whereas here the batch-independent control variate increases the variance for the blue mini-batch.

In practice, such objectives are treated using Monte Carlo (MC) sampling. We obtain an unbiased stochastic estimate of the expectation, $\hat{\ell}_n = \frac{1}{S}\sum_{s=1}^{S} f_n(\boldsymbol{\epsilon}_n^{(s)}, \boldsymbol{\theta})$, from S samples $\boldsymbol{\epsilon}_n^{(s)} \sim p(\boldsymbol{\epsilon})$. We can then use Stochastic Gradient Descent (SGD) or other optimisers based on the noisy gradients (Robbins and Monro, 1951). For large N, the evaluation of the full sum in $\mathcal{L}$ is often computationally intractable. This can be addressed by subsampling mini-batches $\mathcal{B} \subset \{1, \ldots, N\}$ of size $|\mathcal{B}|$ from the full data set, introducing additional noise and leading to a doubly stochastic objective function of the form:

$$\hat{\mathcal{L}} := \frac{N}{|\mathcal{B}|S} \sum_{b \in \mathcal{B}} \sum_{s=1}^{S} f_b(\boldsymbol{\epsilon}_b^{(s)}, \boldsymbol{\theta}), \tag{1}$$

with $\mathbb{E}[\hat{\mathcal{L}}] = \mathcal{L}$.

[*]Research conducted while at PROWLER.io. Corresponding author: a.boustati@warwick.ac.uk.

[†]Completed work at PROWLER.io before joining Amazon.

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

The variance of the gradients of $\mathcal{L}$ affects both the convergence rate of the optimisation and how close the optimiser can get to the optimum. This motivates various approaches for reducing either mini-batch variance (e.g., Johnson and Zhang (2013)) or the variance due to MC estimation of the expectation (Ranganath et al., 2014; Roeder et al., 2017). A common approach for variance reduction are control variates (see Section 2.1), which have recently been adopted in the literature (see Section 4 for details). The focus of these works is on deriving and applying control variate schemes to MC objectives, specifically in the context of Variational Inference (VI).

When mini-batching as in Eq. (1), the way each term $\hat{g}_b(\boldsymbol{\epsilon}) = \partial f_b / \partial \boldsymbol{\theta}$ in the stochastic gradient evaluation depends on the randomness $\boldsymbol{\epsilon}$ can vary between data points, and so $\hat{g}_{\mathcal{B}} = \sum_{b \in \mathcal{B}} \hat{g}_b$ varies between batches. Thus a control variate for this stochasticity is affected by the *context points* (e.g., for a regression problem, input and output for the data points in a mini-batch). This dependence is illustrated in Fig. 1 for a Bayesian logistic regression example. The gradient $\hat{g}_{\mathcal{B}}(\epsilon)$ of a doubly stochastic objective is shown for two different mini-batches $\mathcal{B}$. In this simplified case, each batch consists of a single context point. The different context points induce different relationships between randomness and the gradient. This means the two gradients correlate differently with the randomness, yielding different optimal control variates. For comparison, we include a batch-independent control variate which has to average over all contexts. As shown in the right-hand panel in Fig. 1, adapting the control variate to the batch significantly reduces the variance. To the best of our knowledge, this context dependence is not taken into account by the schemes in the literature.

In this work we propose a novel idea for computing control variates that adapt to the context (mini-batch) of the controlled estimators (the gradient). The new formulation takes into account the dependence of the MC estimate on the data by using a recognition network to learn an adaptive control variate coefficient. We derive a low-variance objective to train the network to approximate the optimal control variate coefficient per batch. Additionally, we propose two computationally cheaper, higher-variance alternatives to the network objective. All control variate objectives re-use the already computed model gradients, and hence do not require extra back-propagation steps.

We describe our formulation in the next section, followed in Section 3 by the concrete example of linear and polynomial control variates for Gaussian base randomness. We put our proposal into the context of previous work in Section 4. We empirically test the properties of our proposed method in Section 5.

2 METHODOLOGY

We start by reviewing control variates and highlight the importance of computing the optimal control variate coefficient to maximise variance reduction. In Section 2.2 we introduce the mini-batch dependence of the gradients and the control variates and propose learning context-aware control variate coefficients. Finally, in Section 2.3 we derive objectives for the control variate coefficients that allow amortisation through a recognition network.

2.1 CONTROL VARIATES

Control variates aim to reduce the variance of an unbiased stochastic estimator* $\hat{g}_\theta(\boldsymbol{\epsilon})$ for an intractable expectation $\mathbb{E}[g(\boldsymbol{\epsilon})]$, where $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ is a random variable. We consider a different function $w(\boldsymbol{\epsilon})$ whose expectation is known analytically, $\mathbb{E}[w(\boldsymbol{\epsilon})] = W$. Then $c(w(\boldsymbol{\epsilon}) - W)$ has zero expectation for any c, and its unbiased estimator, $c(\hat{w}(\boldsymbol{\epsilon}) - W)$, can be subtracted from the original estimator,

$$\tilde{g}(\boldsymbol{\epsilon}) := \hat{g}(\boldsymbol{\epsilon}) - c(\hat{w}(\boldsymbol{\epsilon}) - W). \tag{2}$$

This new estimator is unbiased, i.e., it has the same expectation as the original estimator. Minimising its variance $\mathrm{Var}[\tilde{g}]$ gives the optimal $c^* = \mathrm{Cov}[\hat{g}, \hat{w}] / \mathrm{Var}[\hat{w}]$, and $\tilde{g}$ will have lower variance than $\hat{g}$ if $g(\boldsymbol{\epsilon})$ and $w(\boldsymbol{\epsilon})$ are correlated. With the optimal c^*, the variance reduces to

$$\mathrm{Var}[\tilde{g}] = (1 - \rho_{g,w}) \, \mathrm{Var}[\hat{g}], \tag{3}$$

where $\rho_{g,w}$ is the Pearson correlation coefficient between g and w. In practice, computing c^* is not possible, as $\mathrm{Cov}[\hat{g}, \hat{w}]$ and $\mathrm{Var}[\hat{w}]$ cannot be evaluated exactly, and are usually estimated from the optimisation statistics, e.g. running averages (Paisley et al., 2012). Another option is to pre-specify a fixed c (Miller et al., 2017; Grathwohl et al., 2018).

Neither option is convincing for the doubly stochastic case. The first option has very high variance due to the presence of mini-batch stochasticity in addition to sampling stochasticity. The second option is sub-optimal as fixing an arbitrary value for c does not guarantee the optimal variance reduction of Eq. (3).

In the next section we introduce c as a context-dependent adaptive parameter that is learned throughout the optimisation.

*We use the $\hat{}$ symbol (as well as $\tilde{}$) on top of functions of random variables to denote the estimate of this function obtained by evaluating the relevant estimator. In the following we drop the dependence on θ to lighten the notation.

2.2 CONTROLLING MINI-BATCH GRADIENTS

Gradient-based optimisation requires the derivatives of the objective (1) with respect to the model parameters $\{\theta_p\}_{p=1}^P$. The estimated gradient contains a sum over mini-batch elements b,

$$\frac{\partial \hat{\mathcal{L}}}{\partial \theta_p} \propto \sum_{b \in \mathcal{B}} \frac{\partial f_b}{\partial \theta_p}(\boldsymbol{\epsilon}_b, \boldsymbol{\theta}) = \sum_{b \in \mathcal{B}} \hat{g}_{bp}(\boldsymbol{\epsilon}_b) =: \hat{G}_p, \qquad (4)$$

where we chose $S = 1$ to simplify the equations (the extension to multiple MC samples is straightforward). Note that $\mathcal{B}$ is a random subset of $\{1, \ldots, N\}$, i.e., b are indices into the full data set, and each term gets its own realisation $\boldsymbol{\epsilon}_b$ of the randomness. We want to improve the optimisation performance by reducing the variance of this gradient. As demonstrated in Fig. 1, each partial gradient estimator $\hat{g}_{bp}(\boldsymbol{\epsilon}_b)$ may have a different dependence on the randomness. To account for this, we introduce separate control variates for each term (data point within a batch) in the sum in (4). For a single partial gradient, we define the controlled gradient estimator

$$\tilde{g}_{bp}(\boldsymbol{\epsilon}_b) := \hat{g}_{bp}(\boldsymbol{\epsilon}_b) - \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b). \qquad (5)$$

Here and in the following section we subsume the analytic expectation into the definition of the control variate such that $\hat{\mathbf{w}}(\boldsymbol{\epsilon})$ already has zero mean. We use the same type of control variates for all parameters; however, in principle, we could have a different $\hat{\mathbf{w}}_p$ per parameter θ_p. In both cases, the coefficients $\mathbf{c}_{bp}$ are per-parameter. In general, the output dimensionality of the mapping $\hat{\mathbf{w}}(\boldsymbol{\epsilon})$ may be different from that of the randomness $\boldsymbol{\epsilon}$ itself; for simplicity, we assume both $\boldsymbol{\epsilon}$ and $\hat{\mathbf{w}}(\boldsymbol{\epsilon})$ are D-dimensional. Note that $\hat{\mathbf{w}}(\cdot)$ does not depend on the batch element b; the dependence is captured in the coefficients $\mathbf{c}_{bp}$, which is a vector of length D for each index pair b, p.

Specifying the problem this way allows us to explicitly model each control variate coefficient per data point. Under this setting, the new estimator for the gradient is

$$\tilde{G}_p = \sum_{b \in \mathcal{B}} \tilde{g}_{bp}(\boldsymbol{\epsilon}_b) = \sum_{b \in \mathcal{B}} (\hat{g}_{bp}(\boldsymbol{\epsilon}_b) - \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)). \qquad (6)$$

The control variate coefficients $\mathbf{c}_{bp}$ can be set to optimally reduce the variance of $\tilde{G}_p$ by solving

$$\min_C \operatorname{Tr}(\operatorname{Cov}[\tilde{\mathbf{G}}]), \qquad (7)$$

where C is the collection of $\mathbf{c}_{bp}$ and has shape $N \times P \times D$, as separate coefficients are needed for all N data points.

Computing and storing these can be computationally prohibitive for large data sets, hence we propose to amortise the cost of this computation by using a recognition network $r_\phi : \mathcal{Y} \to \mathbb{R}^{P \times D}$ that outputs the coefficients for each batch element throughout the optimisation, where

$$\mathbf{c}_{bp} = [r_\phi(y_b)]_p \qquad (8)$$

is a vector of dimension D and $y_b \in \mathcal{Y}$ are context points (e.g., feature vector and target for the bth data point in a supervised learning problem) and ϕ are the recognition network parameters.

In practice, the recognition network outputs the control variate coefficients per data point, which are then aggregated per batch. This ensures permutation invariance to the order of data points in the batch and allows for the randomisation of batch elements throughout the optimisation procedure.

As the control variate only adds zero-expectation terms to the gradients of the optimisation objective, the minima of the objective remain unchanged. This means that the extra parameters of the recognition network will *not* lead to overfitting. We provide theoretical analysis of the convergence in Appendix B. In the next section we discuss objectives for training the network parameters.

2.3 TRAINING THE RECOGNITION NETWORK

Intuitively, we require the recognition network $r_\phi(\cdot)$ to output coefficients that minimise the variance of the controlled gradient estimator (6). This gives the training objective for the parameters ϕ:

$$\min_\phi \operatorname{Tr}\left(\operatorname{Cov}[\tilde{\mathbf{G}}]\right) = \min_\phi \sum_{p=1}^P \operatorname{Var}[\tilde{G}_p]. \qquad (9)$$

The pth term in the sum in (9) is

$$\begin{aligned}
\operatorname{Var}[\tilde{G}_p] &= \operatorname{Var}\Big[\sum_{b \in \mathcal{B}} \big(\hat{g}_{bp}(\boldsymbol{\epsilon}_b) - \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\big)\Big] \\
&= \sum_{b \in \mathcal{B}} \operatorname{Var}\left[\hat{g}_{bp}(\boldsymbol{\epsilon}_b) - \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right] \\
&= \sum_{b \in \mathcal{B}} \Big(\operatorname{Var}\left[\hat{g}_{bp}(\boldsymbol{\epsilon}_b)\right] + \operatorname{Var}\left[\mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right] \\
&\quad - 2 \operatorname{Cov}\left[\hat{g}_{bp}(\boldsymbol{\epsilon}_b), \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)\right]\Big) \\
&= \text{const} + \sum_{b \in \mathcal{B}} \Big(\mathbb{E}\big[(\mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))^2\big] \\
&\quad - 2\mathbb{E}\big[(\hat{g}_{bp}(\boldsymbol{\epsilon}_b))(\mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))\big]\Big), \qquad (10)
\end{aligned}$$

using $\mathbb{E}[\hat{w}(\boldsymbol{\epsilon}_b)] = 0$. We discard the terms that do not contain $\mathbf{c}_{bp}$ and hence do not give gradients for ϕ. For most problems, the expectations are intractable; we estimate

these with MC sampling (here $S = 1$) and define

$$\tilde{V}_p := \sum_{b \in \mathcal{B}} \big((\mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))^2 - 2(\hat{g}_{bp}(\boldsymbol{\epsilon}_b))(\mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)) \big). \tag{11}$$

We can now learn the optimal recognition network parameters ϕ using SGD (or variants) on

$$\min_{\phi} \sum_{p=1}^{P} \tilde{V}_p. \tag{12}$$

To learn the parameters ϕ, we need to compute gradients of $\sum_p \tilde{V}_p$. Examining the chain rule around the outputs of the recognition network, $\frac{\partial \tilde{V}_p}{\partial c_{bpd}} \frac{\partial c_{bpd}}{\partial \phi}$, the second term is computed by backpropagation through the network, and the cost of computing the first term depends on the form of the estimator $\tilde{V}_p$.

The network objective in (11) depends on the individual partial gradients $\hat{g}_{bp}(\boldsymbol{\epsilon}_b)$ of the model objective *for each data point*; we call this the **partial gradients estimator**. In common reverse-mode automatic differentiation libraries such as TensorFlow and PyTorch, this requires $|\mathcal{B}|$ additional backward passes on the model objective, each at least $O(|\mathcal{B}|)$.[†] This becomes prohibitively expensive for large mini-batches. To overcome this limitation in current implementations, we derive two additional estimators for the recognition network objective that are computationally cheaper, albeit with higher variance.

2.3.1 The Gradient Sum Estimator

To avoid the partial gradients in (11), we return to the pth term of the sum in (9). Instead of taking the sum out of the variance, we separate the sum over partial gradients from the control variates:

$$\begin{aligned} \mathrm{Var}[\tilde{G}_p] &= \mathrm{Var}\left[\sum_{b \in \mathcal{B}} \hat{g}_{bp}(\boldsymbol{\epsilon}_b) - \sum_{b \in \mathcal{B}} \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b) \right] \\ &= \mathrm{Var}\left[\hat{G}_p - \sum_{b \in \mathcal{B}} \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b) \right]. \end{aligned} \tag{13}$$

We can expand the variance of a sum of two terms as

$$\begin{aligned} \mathrm{Var}[\tilde{G}_p] &= \mathrm{Var}[\hat{G}_p] + \mathrm{Var}\left[\sum_{b \in \mathcal{B}} \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b) \right] \\ &\quad - 2\,\mathrm{Cov}\left[\hat{G}_p, \sum_{b \in \mathcal{B}} \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b) \right] \\ &= \mathrm{const} + \sum_{b \in \mathcal{B}} \Big(\mathbb{E}[(\mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))^2] \\ &\quad - 2\mathbb{E}[(\hat{G}_p)(\mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))] \Big), \end{aligned} \tag{14}$$

[†]During the review process, a new PyTorch library, BackPACK (Dangel et al., 2020), was released, which can compute partial gradients with a low computational overhead.

and by replacing the expectations with MC estimates, we arrive at a new estimator

$$\tilde{V}_p^{\mathrm{GS}} := \sum_{b \in \mathcal{B}} \big((\mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b))^2 - 2(\hat{G}_p)(\mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b)) \big). \tag{15}$$

This estimator is similar in form to the partial gradients estimator, replacing the gradient per data point with the sum over the whole mini-batch; we call this the **gradient sum** estimator. As it does not require any additional backward passes, it is much cheaper to compute. One can intuitively see that this estimator has a higher variance than the partial gradients estimator as it additionally includes cross terms that would be zero in expectation.

2.3.2 The Squared Difference Estimator

Alternatively, we can continue from (13) by expanding the variance into moment expectations:

$$\begin{aligned} \mathrm{Var}[\tilde{G}_p] &= \mathbb{E}\big[\big(\hat{G}_p - \sum_{b \in \mathcal{B}} \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b) \big)^2 \big] \\ &\quad - \big(\mathbb{E}\big[\hat{G}_p - \sum_{b \in \mathcal{B}} \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b) \big] \big)^2, \end{aligned} \tag{16}$$

where the control variate term has no contribution inside the second expectation by definition, and $\mathbb{E}[\hat{G}_p]$ is a constant with respect to the recognition network parameters ϕ. Evaluating the remaining expectation using MC gives us the **squared difference** estimator:

$$\tilde{V}_p^{\mathrm{SD}} := \big(\hat{G}_p - \sum_{b \in \mathcal{B}} \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b) \big)^2, \tag{17}$$

which is also cheap to compute. In contrast to $\tilde{V}_p^{\mathrm{GS}}$, it includes the second moment of $\hat{G}_p$. This is similar to a regression problem that uses $\hat{w}_d(\boldsymbol{\epsilon}_b)$ as basis functions to approximate the gradient $\hat{G}_p$.

Both the gradient sum objective (14) and the squared difference objective (17) are *unbiased estimators* and not approximations of the recognition network objective in (7). Compared to the partial gradients objective (11), they have higher variance; we empirically assess this in Appendix E.2.3. We lay out pseudocode for joint optimisation of model and recognition network in Algorithm 1 in Appendix A.

3 ILLUSTRATIVE EXAMPLE: CONTROL VARIATES FOR GAUSSIAN BASE RANDOMNESS

To implement a control variate, we need to specify both the distribution of the base randomness $\boldsymbol{\epsilon}$ and the functional form of the control variate $\mathbf{w}(\boldsymbol{\epsilon}_n)$. In principle, any

functional form for control variates from the literature can be used with this method, e.g. Paisley et al. (2012); Ranganath et al. (2014); Miller et al. (2017). For the sake of simplicity, we illustrate our proposal on a simpler control variate form for the special case of Gaussian base randomness, which is of direct interest to many applications in VI.

We assume $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathrm{I}_D)$ without loss of generality.[‡] In this section we introduce explicit forms for $\mathbf{w}(\boldsymbol{\epsilon}_n)$ for this case, starting with linear control variates, and then extending the discussion to higher-order polynomials.

3.1 LINEAR GAUSSIAN CONTROL VARIATES

The simplest control variate is an element-wise linear function of $\boldsymbol{\epsilon}_n$,

$$\mathbf{w}(\boldsymbol{\epsilon}_n) = \boldsymbol{\alpha} + \boldsymbol{\beta} \circ \boldsymbol{\epsilon}_n, \qquad \boldsymbol{\epsilon}_n \sim \mathcal{N}(\mathbf{0}, \mathrm{I}_D), \tag{18}$$

with $\circ$ representing the element-wise product. Its expectation is $\mathbf{W} = \mathbb{E}[\mathbf{w}(\boldsymbol{\epsilon}_n)] = \boldsymbol{\alpha}$, and the control variate simplifies to $\boldsymbol{\beta} \circ \boldsymbol{\epsilon}_n$. We can also absorb $\boldsymbol{\beta}$ into the control variate coefficient $\mathbf{c}_{np}$, which results in the following controlled version of the gradient component p, for data point n:

$$\tilde{g}_{np}(\boldsymbol{\epsilon}_n) = \hat{g}_{np}(\boldsymbol{\epsilon}_n) - \mathbf{c}_{np}^\top \boldsymbol{\epsilon}_n. \tag{19}$$

Intuitively, one can think of control variates of this form as injecting the estimator with information on the linear dependence of the gradient on the noise. To understand this further, we take a look at the first-order Taylor expansion of the gradient component $g_{np}(\boldsymbol{\epsilon}_n)$ around $\boldsymbol{\epsilon}_n = \mathbf{0}$,

$$g_{np}(\boldsymbol{\epsilon}_n) = g_{np}(\mathbf{0}) + \nabla g_{np}(\mathbf{0})^\top \boldsymbol{\epsilon}_n + O(\boldsymbol{\epsilon}_n^2). \tag{20}$$

If the gradient is sufficiently linear with respect to $\boldsymbol{\epsilon}_n$ (i.e., the $O(\boldsymbol{\epsilon}_n^2)$ terms are negligible), and when $\mathbf{c}_{np}$ is a good approximation to the Jacobian at $\mathbf{0}$, the estimator in (19) will have low variance.

3.2 HIGHER-ORDER POLYNOMIALS

In general, the gradient is unlikely to be linear with respect to the noise, especially for complicated models and objectives. To overcome this, we can use higher-order polynomials to capture some of the non-linear dependence of the gradient on the noise. Consider

$$\mathbf{w}(\boldsymbol{\epsilon}_n) = \sum_{k=1}^{K} \boldsymbol{\alpha}_k \circ \boldsymbol{\epsilon}_n^k, \tag{21}$$

where the kth power is evaluated element-wise. $\mathbf{W}$ can be easily computed and would correspond to the sum of diagonal parts of the first K moment tensors of the multivariate Gaussian distribution, scaled by $\boldsymbol{\alpha}_k$. For instance, for $K = 2$ the control variate is given by

$$\boldsymbol{\alpha}_1 \circ \boldsymbol{\epsilon}_n + \boldsymbol{\alpha}_2 \circ (\boldsymbol{\epsilon}_n^2 - \mathrm{diag}(\mathrm{I}_D)). \tag{22}$$

We can again simplify by absorbing the $\boldsymbol{\alpha}_k$ into the control variate coefficient, with slight adjustments to the controlled gradient estimator.We make the following observation:

Remark 1: A linear combination of control variates is also a valid control variate, i.e., $\tilde{g}_{np}(\boldsymbol{\epsilon}_n) = \hat{g}_{np}(\boldsymbol{\epsilon}_n) - \sum_{k=1}^{K} (\mathbf{c}_{np}^{(k)})^\top (\hat{\mathbf{w}}_k(\boldsymbol{\epsilon}) - \mathbf{W}_k)$ is unbiased. By considering each term in (22) as a separate control variate, we can write the pth component of the controlled gradient at n as

$$\tilde{g}_{np}(\boldsymbol{\epsilon}_n) = \hat{g}_{np}(\boldsymbol{\epsilon}_n) - (\mathbf{c}_{np}^{(1)})^\top \boldsymbol{\epsilon}_n - (\mathbf{c}_{np}^{(2)})^\top (\boldsymbol{\epsilon}_n^2 - \mathrm{diag}(\mathrm{I}_D)). \tag{23}$$

The same construction trivially extends to $K > 2$.

3.3 BRIEF DISCUSSION

The simple examples of the linear and polynomial control variates presented above illustrate the importance of choosing a good control variate coefficient. For instance, in the linear case in (19) the control variate function $\mathbf{w}(\boldsymbol{\epsilon}_n) = \boldsymbol{\epsilon}_n$ does not provide any extra information on the estimator on its own, as it essentially just adds noise to the MC estimate. However, with the selection of a good control variate coefficient $\mathbf{c}_n$ for data point n, we introduce structure to the noise that contains information about the behaviour of the controlled quantity with respect to the Gaussian noise in the form of the Jacobian in (20). Indeed the optimal coefficient $\mathbf{c}_n^*$ for the linear control variate contains the Jacobian term.

4 RELATED WORK

Control variates are widely used to reduce the gradient variance of stochastic objectives, mainly motivated by VI. A comprehensive review can be found in Geffner and Domke (2018). Here, we highlight some relevant work and compare it to our contribution.

Paisley et al. (2012) first introduce the idea of using control variates to reduce the gradient variance in VI. They propose using a bound on the objective or an approximation of the model as control variates. Ranganath et al. (2014) build on this work, using the score function of the approximate posterior to control the gradient of Black Box Variational Inference objectives.

Inspiration for our work comes from Grathwohl et al. (2018), where they use a recognition network to approximate the model and its gradient as a control variate. Miller

[‡]In the general case where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, we can simply apply the location-scale reparameterisation $\boldsymbol{\epsilon} = \boldsymbol{\mu} + \mathrm{Cholesky}(\Sigma)\boldsymbol{\epsilon}_0$ with $\boldsymbol{\epsilon}_0 \sim \mathcal{N}(\mathbf{0}, \mathrm{I}_D)$.

et al. (2017) approximate the reparameterisation gradient for Gaussian variational distributions by its first-order Taylor expansion, using this approximation as a control variate. Our work is related to this construction where the recognition network can be viewed as a cheap approximation to the linear term in the Taylor expansion of the gradient (i.e., the Hessian of the model objective) in the case of the linear construction in Section 3.

The unifying work of Geffner and Domke (2018) categorises different control variate schemes for VI objectives. Additionally, they propose combining them to achieve greater variance reduction. They derive an optimal rule for this combination based on Bayesian risk minimisation.

These related works do not consider the effect of mini-batching on the proposed control variates; therefore, our work *complements* the methods mentioned above. Indeed, Geffner and Domke (2018) show that a combination of control variates is usually more desirable that a single scheme. The method we proposed can be considered an extra addition to the control variate toolkit for doubly stochastic objectives, to take the effect of mini-batch stochasticity on the control variates into account. Our method can also be combined with other variance reduction methods such as extra sampling.

5 EXPERIMENTS

Our discussion thus far applied to the general class of doubly stochastic objectives. For our experiments we focus on objectives from VI problems. Amortising the computation of the control variate coefficients in this setting is advantageous since context arises naturally from the data in the underlying models.

In this section, we aim to answer three questions: a) To what extent can amortisation with a recognition network reduce the variance compared to a fixed context-free control variate coefficient? b) How well can we train the recognition network in an online setting? c) What difference can an amortised control variate make in practice?

5.1 SETUP

We investigate (a), (b) and (c) on a classification task on the *titanic* dataset using a Bayesian logistic regression model and on a regression task on the *airfoil* dataset using a Deep Gaussian Process (DGP) (Salimbeni and Deisenroth, 2017). Detailed description of the models and datasets can be found in Appendix C.

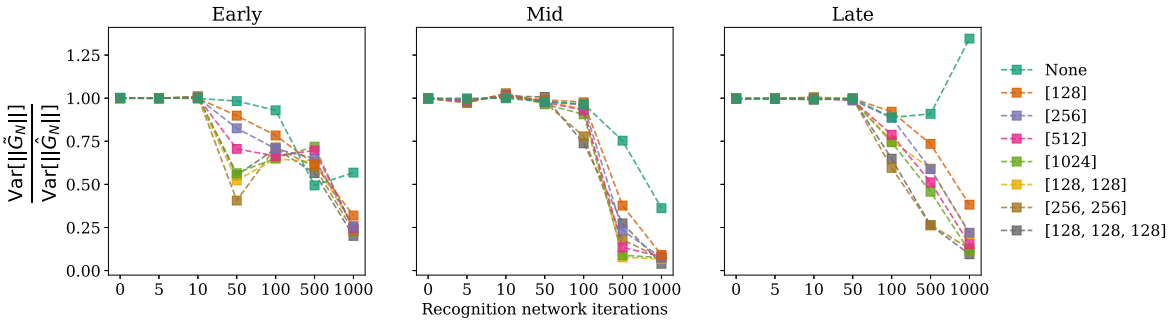
Throughout, we use Adam (Kingma and Ba, 2015) for optimising both the model objective function and the recognition network objective. We use a single-sample MC estimate of the gradients and control these when stated, applying the linear and quadratic control variates introduced in Section 3. We initialise the recognition network with Xavier initialisation (Glorot and Bengio, 2010) and use ReLU activations in the hidden layers.

We compare our proposal to a context-free control variate. In this instance, this is implemented as an optimisable quantity that does not depend on data and uses the same optimisation objectives ((14) & (17)) as the recognition network, i.e., $\mathbf{c}$ is independent of the mini-batch $\mathcal{B}$ in these objectives. This is equivalent to approximating the coefficient with an exponentially weighted moving average of the empirical covariance of the gradient and the control variate estimates.

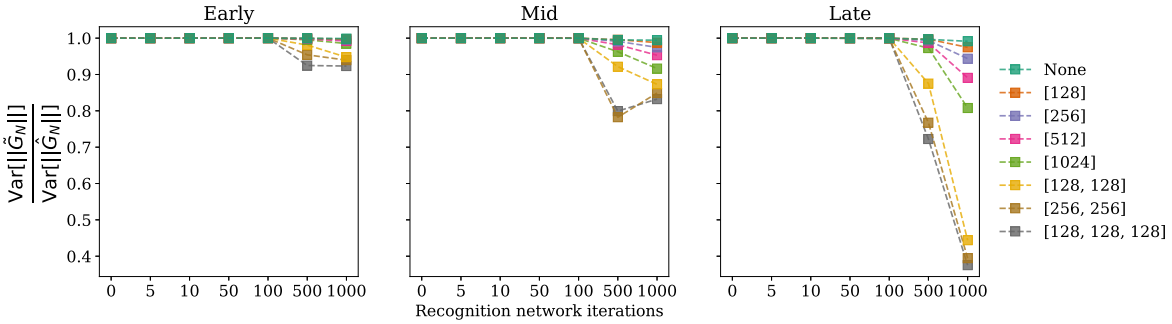
5.2 VERIFICATION OF VARIANCE REDUCTION

We first consider whether the recognition network has the capacity to amortise the control variate coefficients and how well it can learn these versus a context-free coefficient. To test this, we freeze the model parameters at three points in the optimisation – early (10 steps), mid (200 steps), and late (1000 steps) – then optimise the control variate coefficients only. For each setting of model parameters, we optimise the recognition network for 1000 steps and record the variance reduction at different steps. The variance reduction is measured by the ratio $\mathrm{Var}[\|\tilde{G}_N\|]/\,\mathrm{Var}[\|\hat{G}_N\|]$, where $\tilde{G}_N$ and $\hat{G}_N$ are the controlled and uncontrolled gradients, respectively, over the mini-batch (size 10), and $\|\cdot\|$ is the gradient norm. We compare different network sizes to see the effect this has on variance reduction.

Fig. 2 shows that amortising the control variate coefficients induces greater variance reduction than context-free coefficients (labelled as *None* in the figure). The variance reduction does not occur immediately, as the control variate coefficients need to be optimised in all cases to reduce the variance. Also notable is that the amount of variance reduction depends on the optimisation stage of the model; at later stages of the model optimisation, the variance reduction is more pronounced. This is likely a property of both the model and the control variate where the gradients in the beginning of the optimisation have more pronounced non-linearity with respect to the noise. This can also be seen in the amount of variance reduction in logistic regression compared to the DGP. The gradients in the logistic regression models are approximately linear with respect to the noise, while the DGP gradients have more complex dependency on the noise. Finally, we can see that the variance reduction potential depends on the capacity of the network, where wider and deeper networks learn better control variate coefficients. Deeper networks reduce the variance more strongly than wider



(a) Logistic regression on *titanic*.



(b) DGP on *airfoil*.

Figure 2: Variance reduction at different points in the objective optimisation (lower is better); early: 10 steps, mid: 200 steps, late: 1000 steps. The results are shown for the *linear* control variate from Section 3. Recognition network training uses the *squared difference* objective optimised with Adam with learning rate of 10^{-2} for the logistic regression and 10^{-3} for the DGP. For a small number of iterations on the recognition network, it struggles to learn a good control variate coefficient. Continuing the network optimisation, it is able to learn good control variate coefficients that significantly reduce the variance in comparison to the context-free coefficient. Also notable is that the variance reduction is more pronounced at the later stages of the model optimisation.

networks, corresponding to a highly non-linear mapping from the context points to the control variate coefficient.

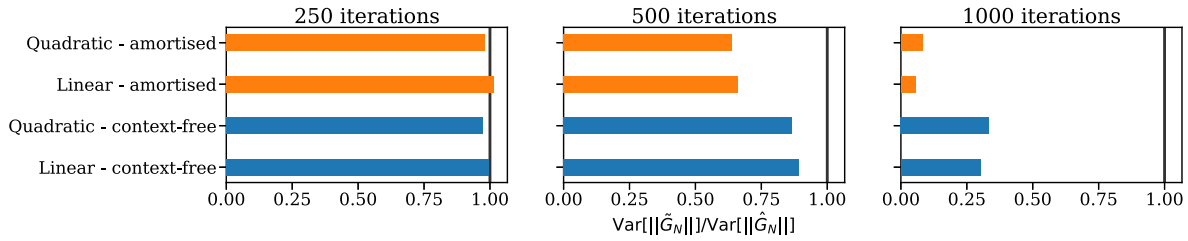
5.3 SIMULTANEOUS OPTIMISATION OF OBJECTIVE FUNCTION AND CONTROL VARIATE COEFFICIENT

In practice, the recognition network must be able to learn the control variate coefficients while the model is being optimised, resulting in a moving target. In this section, we investigate the viability of chasing this target by simultaneously optimising the model objective and recognition network. We use a recognition network with three layers of size 128 each, as this architecture showed the largest variance reduction in Section 5.2. In each step in the optimisation procedure, we compute one gradient estimate of the model objective for a mini-batch of size 10. We take one Adam step on the recognition network, then we apply the control variate correction to the sampled gradient and take an Adam step on the model parameters. We measure the variance of the gradient at different periods in the optimisation by sampling 100 gradient values at each period and taking the empirical variance of their norm.

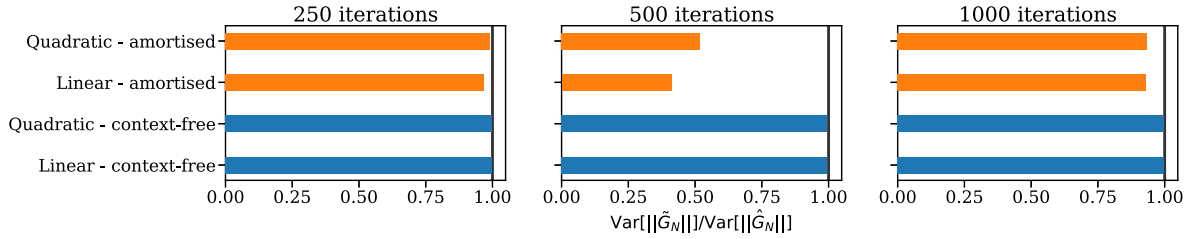
The recognition network is able to learn good control variate coefficients in this dynamic regime, see Fig. 3. The variance reduction improves later on in the optimisation as observed in Section 5.2. We again observe that the amortised control variate results in greater variance reduction than the context-free one.

5.4 PRACTICAL EFFECTIVENESS

To show how our approach works in practice, we use it for training the logistic regression and DGP models. We apply the alternating optimisation procedure described in Section 5.3 on each for 2000 iterations with mini-batches of size 10. We record the mean value of the Negative Evidence Lower Bound (NELBO) from 100 MC samples for the logistic regression and 10 MC samples for the



(a) Logistic regression on *titanic*.



(b) DGP on *airfoil*.

Figure 3: Gradient variance ratio at three different points in the joint optimisation of the model and control variate parameters (lower is better); the vertical line corresponds to a ratio of 1 (i.e. no reduction). Both the model and the control variate objectives are optimised with Adam with learning rate of 10^{-2} for the model and 10^{-2} and 10^{-3} for the logistic regression and DGP control variate coefficients respectively. The recognition network learns a good control variate coefficient and continues to improve throughout the optimisation, outperforming the context-free control variate.

DGP at every iteration computed on the full data sets.

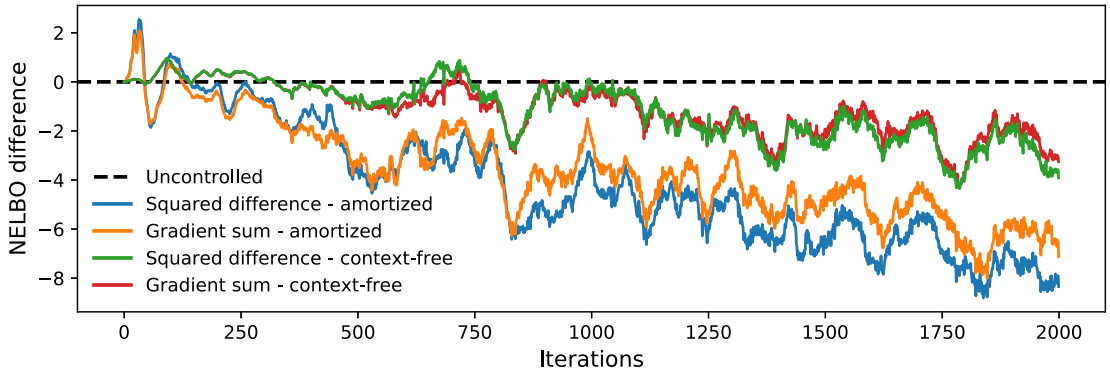
The resulting traces are shown in Fig. 4; in both cases we see that the optimisation with controlled gradients starts off in a worse regime than the uncontrolled gradients (curves on or above the dashed line); however, it improves as better control variate coefficients are learned. The gap between the one-sample MC estimator and the controlled estimators widens later for logistic regression, and fluctuates for the DGP with some instability in the end. This is because the linear control variate sufficiently approximates the dependence of the gradient on the randomness in the case of logistic regression, whereas for the DGP this dependence is more complex. This can be verified in Fig. 3b, where the variance reduction diminishes later on in the optimisation.

For both models, amortising the control variate coefficients results in lower NELBO values on average in comparison to the uncontrolled and the context-free controlled cases. We also see that the optimisation of the control variate coefficients is insensitive to the choice of objective function, with similar behaviour for the gradient sum and squared difference objectives for both the amortised and context-free cases.

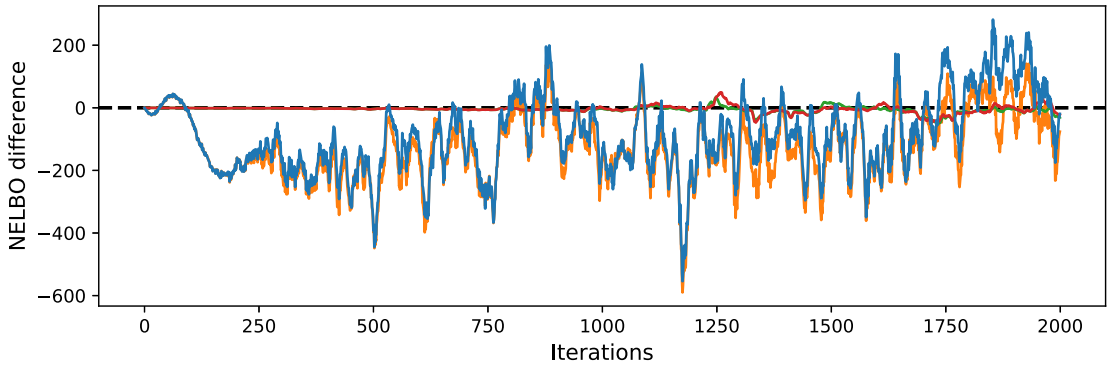
Table 1 shows the average cost for the controlled optimisation steps for the two problems. Amortising the control variate coefficients with a recognition network of size [128, 128, 128] has an additional overhead of around 25% on the context-free coefficient on the CPU, which is a good investment for high-variance objectives. The overhead depends on many factors such as the recognition network size, control variate formulation, mini-batch size and number of gradient components. These should all be taken into account when implementing this scheme.

Table 1: Average optimisation step time in *milliseconds* (on the CPU) for logistic regression and DGP for different *linear* control variate objective functions. Mean figures are presented with the standard error in parentheses. The statistics are computed based on 100 repetitions of 10 runs. The implementation uses TensorFlow 2.0 (Abadi et al., 2016) and GPflow (Matthews et al., 2017; van der Wilk et al., 2020).

Method	Logistic	DGP
Squared diff. - amortised	1.20(0.06)	3.77(0.22)
Grad. sum - amortised	1.25(0.11)	3.78(0.19)
Squared diff. - context-free	0.87(0.09)	3.17(0.13)
Grad. sum - context-free	0.84(0.08)	3.07(0.08)



(a) Logistic regression on *titanic*.



(b) DGP on *airfoil*.

Figure 4: Difference between optimisation traces for different control variate objectives, using the uncontrolled one-sample MC estimate of the gradient as a baseline (lower is better). *Linear* control variates are used in this experiment. The gap between the baseline and controlled models widens throughout the optimisation. Amortised control variate coefficients result in wider gaps indicating better optimisation performance.

6 CONCLUSIONS

We introduced a control variate formulation that exploits the structure of doubly stochastic objectives to reduce Monte Carlo sampling variance from mini-batch gradient estimators. We proposed three objectives for an amortising recognition network that can learn context aware control variate coefficients. Training the network re-uses the gradients of the model objective and does not require additional passes through the model.

Empirical assessment showed that an approximation to the optimal control variate per mini-batch can be performed during optimisation and reduces the gradient variance in practice compared to a context-free global approach. In our experiments we used linear and quadratic control variates for Gaussian base randomness, but our approach is general and can be applied to other control variate formulae and randomness schemes. This is particularly important for complex objectives such as in DGP models, where the simple linear and quadratic control variates show some instability. This opens up a new avenue of research into combining our method with adaptive control variates, e.g. Radial Basis Network control variates.

While our method aims to reduce Monte Carlo sampling variance in doubly stochastic objectives, a promising research direction is to combine it with other variance reduction methods that target the variance due to mini-batching, e.g. Stochastic Variance Reduced Gradient (SVRG) (Johnson and Zhang, 2013) and Stochastic Gradient Recursive Algorithm (SARAH) (Nguyen et al., 2017).

The computational overhead of our proposed control variate could be further reduced with the right compute architecture. Although evaluating the target objective may not benefit from additional compute nodes, one can use a dedicated node for efficiently evaluating the recognition network in parallel. This proposal is detailed in Appendix A.

References

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, 2016.

F. Dangel, F. Kunstner, and P. Hennig. BackPACK: Packing more into backprop. In *8th International Conference on Learning Representations*, 2020.

T. Geffner and J. Domke. Using large ensembles of control variates for variational inference. In *Advances in Neural Information Processing Systems 31*, 2018.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010.

W. Grathwohl, D. Choi, Y. Wu, G. Roeder, and D. Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *6th International Conference on Learning Representations*, 2018.

R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, 2013.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations*, 2014.

A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40), 2017.

A. Miller, N. Foti, A. D' Amour, and R. P. Adams. Reducing reparameterization gradient variance. In *Advances in Neural Information Processing Systems 30*, 2017.

L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

J. Paisley, D. Blei, and M. Jordan. Variational Bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

R. Ranganath, S. Gerrish, and D. Blei. Black Box Variational Inference. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, 2014.

H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951.

G. Roeder, Y. Wu, and D. Duvenaud. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems 30*, 2017.

H. Salimbeni and M. Deisenroth. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems 30*, 2017.

M. van der Wilk, V. Dutordoir, S. John, A. Artemev, V. Adam, and J. Hensman. A framework for interdomain and multioutput Gaussian processes. *arXiv:2003.01115*, 2020.

Supplementary Material

A PSEUDO-CODE

Algorithm 1 describes our proposed amortised control variate scheme in pseudo-code. For simplicity, we illustrate the method on Stochastic Gradient Descent (SGD) (Robbins and Monro, 1951); however, any other gradient-based optimiser could be used instead. The procedure in Algorithm 1 is written in a functional style and represents a single update step for the parameters $\theta = (\theta_1, \ldots, \theta_P)$ of a generic doubly stochastic objective (1), $\texttt{objective}(\theta, \mathcal{B}, \boldsymbol{\epsilon})$, where $\mathcal{B}$ is a set of mini-batch indices and $\boldsymbol{\epsilon}$ is a base randomness. The algorithm also updates the parameters $\phi = (\phi_1, \ldots, \phi_Q)$ of the recognition network $r_\phi(\cdot)$ that amortises the coefficients of the control variate $\hat{\mathbf{w}}(\cdot)$.

This algorithm can be efficiently executed on two separate compute nodes by interleaving the evaluation of the target objective and its gradients on one node with the evaluation of the recognition network and its gradients on another, as suggested below with (O) and (R). The two nodes only need to sync in line 7. The update of the recognition network parameters can still run in parallel with the next evaluation of the target objective gradients.

Algorithm 1 Stochastic Gradient Descent step with amortised control variate gradients

1: **function** AMORTIZEDCVUPDATE($\{y_n\}$, $\texttt{objective}(\theta, \mathcal{B}, \boldsymbol{\epsilon})$, θ, $\hat{\mathbf{w}}(\cdot)$, $r_\phi(\cdot)$, ϕ, learning rates α & β)
2: $\quad\mathcal{B} \sim p(\mathcal{B})$ ▷ draw mini-batch
3: $\quad\boldsymbol{\epsilon}_b \sim p(\boldsymbol{\epsilon}) \quad \forall b \in \mathcal{B}$ ▷ draw base randomness
4: $\quad$(O) $\hat{G}_p \leftarrow \frac{\partial}{\partial \theta_p}(\texttt{objective})(\theta, \mathcal{B}, \{\boldsymbol{\epsilon}_b\}) \quad \forall p$ ▷ uncontrolled model gradients
5: $\quad$(R) $\mathbf{c}_{bp} \leftarrow [r_\phi(y_b)]_p \quad \forall b, p$ ▷ evaluate recognition network on context y_b for batch element b
6: $\quad$(R) $\hat{H}_p \leftarrow \sum_{b \in \mathcal{B}} \mathbf{c}_{bp}^\top \hat{\mathbf{w}}(\boldsymbol{\epsilon}_b) \quad \forall p$ ▷ control variate
7: $\quad$(O+R) $\tilde{G}_p \leftarrow \hat{G}_p - \hat{H}_p \quad \forall p$ ▷ controlled model gradients
8: $\quad$(O) $\theta_p \leftarrow \theta_p - \alpha \tilde{G}_p \quad \forall p$ ▷ SGD updates for the model objective parameters *
9: $\quad$(R) $\tilde{V}_p \leftarrow (\hat{G}_p - \hat{H}_p)^2 \quad \forall p$ ▷ recognition network objective †
10: $\quad$(R) $\delta\phi_q \leftarrow \frac{\partial}{\partial \phi_q} \sum_{p=1}^{P} \tilde{V}_p \quad \forall q$ ▷ recognition network gradients
11: $\quad$(R) $\phi_q \leftarrow \phi_q - \beta \delta_{\phi_q} \quad \forall q$ ▷ SGD updates for recognition network parameters *
12: $\quad$**return** model objective parameters θ, recognition network parameters ϕ
13: **end function**

*For simplicity we show the SGD updates, but in principle any other gradient-based optimiser can be used. The same holds for the recognition network updates.

†We use the squared difference objective (17) for illustrative purposes, but this can also be substituted with the partial gradients objective (11) or the gradient sum objective (14).

B THEORETICAL ANALYSIS

B.1 CONVERGENCE RESULTS

Control variates are introduced to reduce the gradient variance and thereby improve the optimisation behaviour. In this section we show how controlling the gradient can improve the convergence behaviour of Stochastic Gradient Descent (SGD) (Robbins and Monro, 1951), in an idealised scenario.

Consider the function $f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) : \mathbb{R}^D \times \mathbb{R}^P \to \mathbb{R}$, where $\boldsymbol{\epsilon}$ is a random variable distributed according to $p(\boldsymbol{\epsilon})$. We want to solve the following optimisation problem using SGD,

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta})]. \tag{24}$$

We assume $\mathbb{E}_{p(\boldsymbol{\epsilon})}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta})]$ is strongly convex and smooth as defined below. SGD starts with an initial guess for the optimal parameter $\boldsymbol{\theta}_0$. The parameter is then sequentially updated according to

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \hat{\mathbf{g}}(\boldsymbol{\epsilon}_t, \boldsymbol{\theta}_t), \tag{25}$$

where $\boldsymbol{\epsilon}_t$ is an independent realisation (over t) of $\boldsymbol{\epsilon}$ at time t, $\hat{\mathbf{g}}(\boldsymbol{\epsilon}_t, \boldsymbol{\theta}_t) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\epsilon}_t, \boldsymbol{\theta}_t)$, and $\eta_t \in \mathbb{R}$ is a constant. We want to reduce the variance in the update direction by setting

$$\tilde{\mathbf{g}}(\boldsymbol{\epsilon}, \boldsymbol{\theta}) = \hat{\mathbf{g}}(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - \mathbf{c}(\boldsymbol{\epsilon}, \boldsymbol{\theta}), \tag{26}$$

where $\mathbf{c}(\boldsymbol{\epsilon}, \boldsymbol{\theta}) : \mathbb{R}^D \times \mathbb{R}^P \to \mathbb{R}^P$ is a control term designed to reduce the randomness in $\hat{\mathbf{g}}(\boldsymbol{\epsilon}, \boldsymbol{\theta})$. This gives a new update rule

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \tilde{\mathbf{g}}(\boldsymbol{\epsilon}_t, \boldsymbol{\theta}_t). \tag{27}$$

We show that adding a control term as in (26) yields a better convergence rate for SGD under the following assumptions:

Assumption 1 (Smoothness)**.** *$f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$ is L smooth in $\boldsymbol{\theta}$ (for some $L > 0$), i.e.*

$$f(\boldsymbol{\epsilon}, \boldsymbol{\theta}') - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) \leq (\boldsymbol{\theta}' - \boldsymbol{\theta})^\top \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) + \frac{1}{2} L \|\boldsymbol{\theta}' - \boldsymbol{\theta}\|_2^2,$$

Assumption 2 (Strong Convexity)**.** *$f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$ is H strongly convex in $\boldsymbol{\theta}$ (for some $H > 0$), i.e.*

$$f(\boldsymbol{\epsilon}, \boldsymbol{\theta}') - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) \geq (\boldsymbol{\theta}' - \boldsymbol{\theta})^\top \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) + \frac{1}{2} H \|\boldsymbol{\theta}' - \boldsymbol{\theta}\|_2^2,$$

Assumption 3 (Efficient Control Variate)**.** *The norm of the controlled gradient is bounded by a constant M as*

$$\mathbb{E}\big[\|\nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}, \boldsymbol{\theta})\|_2^2\big] \leq M \mathbb{E}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)].$$

Theorem 1 (Convergence Rate)**.** *Under assumptions 1, 2 and 3, with $\eta_t = \eta \leq \frac{1}{2L+M}$ for all t, the update rule given by* (27) *offers a linear convergence rate. Specifically, there exists $0 < c < 1$ (depending on H, L and M as specified in Appendix B.2) such that*

$$\mathbb{E}\left[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2\right] \leq c^t \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2. \tag{28}$$

Proof. See Appendix B.2. □

Although the above assumptions do not necessarily hold in practice, Theorem 1 is useful in giving intuition into the convergence speed of our proposed method. This theorem gives a sufficient condition for the control variate which guarantees the fast convergence rate similar to that of (non-stochastic) gradient descent and stochastic variance reduced gradient descent (SVRG) (Johnson and Zhang, 2013). Specifically, if Assumption 3 holds, our method offers the so called linear convergence rate.

We can relax Assumption 3 as the following assumption.

Assumption 4. *The norm of the controlled gradient is bounded as*

$$\mathbb{E}\big[\|\nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}, \boldsymbol{\theta})\|_2^2\big] \leq \bar{M}.$$

Under assumptions 1, 2 and 4, with $\eta_t = \eta \leq \frac{1}{2L}$ for all t, for the update rule given by (27), there exists $0 < \bar{c} < 1$ (depending on H and L as specified in Appendix B.2) such that

$$\mathbb{E}\bigg[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2\bigg] \leq \bar{c}^t \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2 + \frac{2\eta^2 \bar{M}(1-\bar{c}^t)}{1-\bar{c}}. \tag{29}$$

This result shows that the more efficient the control variate (the smaller $\bar{M}$), the smaller the error in optimisation ($||\boldsymbol{\theta}_t - \boldsymbol{\theta}^*||_2^2$). For the detail on the proof of (29) see Appendix B.2.

B.2 PROOFS FOR CONVERGENCE RESULTS

Proof of Theorem 1. We first establish the following lemma based on the smoothness and strong convexity of $f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$.

Lemma 1. *By smoothness (Assumption 1) and strong convexity (Assumption 2) of $f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$, we have*

$$\mathbb{E}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)] \geq \frac{1}{2L}\mathbb{E}\bigg[\|\nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)\|_2^2\bigg]. \tag{30}$$

Let $h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) = f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*) - \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}^*)$. By convexity of $f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$ we know that $h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) \geq 0$. Let $\boldsymbol{\theta}' = \boldsymbol{\theta} - \eta \nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta}))$. Notice that $\nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) = \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)$.

$$\begin{aligned} h(\boldsymbol{\epsilon}, \boldsymbol{\theta}') - h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) &\leq -\eta \nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta})^\top \nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) + \frac{1}{2} L \eta^2 \|\nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta}))\|_2^2 \\ &= \big(\frac{1}{2} L \eta^2 - \eta\big) \|\nabla h(w, \epsilon))\|_2^2. \end{aligned}$$

With the choice of $\eta = \frac{1}{L}$ and noticing $h(\boldsymbol{\epsilon}, \boldsymbol{\theta}') \geq 0$:

$$-h(\boldsymbol{\epsilon}, \boldsymbol{\theta}) \leq -\frac{1}{2L} \|\nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta}))\|_2^2.$$

Taking expectations, we have

$$\begin{aligned} \frac{1}{2L}\mathbb{E}[\|\nabla h(\boldsymbol{\epsilon}, \boldsymbol{\theta}))\|_2^2] &\leq \mathbb{E}[h(\boldsymbol{\epsilon}, \boldsymbol{\theta})] \\ &= \mathbb{E}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*) - \nabla f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}^*)] \\ &\leq \mathbb{E}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)], \end{aligned}$$

which completes the proof of this lemma.

Now, let $v_t = \nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1}) - c(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1})$. Then

$$\begin{aligned} &\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \\ &\quad= \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2 - 2\eta(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*)^\top \mathbb{E}[v_t] + \eta^2 \mathbb{E}[\|v_t\|_2^2] \\ &\quad= \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2 - 2\eta(\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*)^\top \mathbb{E}[\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1})] + \eta^2 \mathbb{E}[\|v_t\|_2^2] \\ &\quad\leq \|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2 - 2\eta \mathbb{E}[[f(\boldsymbol{\epsilon}, \boldsymbol{\theta}_{t-1}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)] + \eta^2 \mathbb{E}[\|v_t\|_2^2]. \end{aligned} \tag{31}$$

For the last term $\mathbb{E}[\|v_t\|_2^2]$, we have

$$\begin{aligned}
&\mathbb{E}[\|v_t\|_2^2] \\
&= \mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1}) - \nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) + \nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1})\|^2\Big] \\
&\leq 2\mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1}) - \nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*)\|_2^2\Big] \\
&\quad + 2\mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1})\|_2^2\Big] \\
&\leq (4L + 2M)\mathbb{E}[f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1}) - f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*)], \qquad (32)
\end{aligned}$$

where the last inequality holds by Assumption 3.

By strong convexity of $f(\boldsymbol{\epsilon}, \boldsymbol{\theta})$, we have

$$\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2 \leq \frac{2}{H} f(\boldsymbol{\epsilon}, \boldsymbol{\theta}_{t-1}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*). \qquad (33)$$

Combining the last 3 inequalities we get

$$\begin{aligned}
&\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \\
&\leq \mathbb{E}[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2] - 2\eta(1 - \eta(2L + M))\mathbb{E}[f(\boldsymbol{\epsilon}, \boldsymbol{\theta}_{t-1}) - f(\boldsymbol{\epsilon}, \boldsymbol{\theta}^*)] \\
&\leq \Big(1 - \eta H(1 - \eta(2L + M))\Big)\mathbb{E}[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2].
\end{aligned}$$

For $\eta \leq \frac{1}{2L+M}$ and $c = (1 - \eta H(1 - \eta(2L + M))$, we have

$$\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \leq c^t \|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2,$$

which completes the proof of Theorem 1.

When Assumption 3 does not hold and Assumption 4 holds, following the same line of reasoning and replacing the upper bound on $\mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1})\|_2^2\Big]$ with $\bar{M}$, we have

$$\begin{aligned}
&\mathbb{E}[\|v_t\|_2^2] \\
&= \mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1}) - \nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) + \nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1})\|^2\Big] \\
&\leq 2\mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1}) - \nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*)\|_2^2\Big] \\
&\quad + 2\mathbb{E}\Big[\|\nabla f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*) - c(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1})\|_2^2\Big] \\
&\leq 4L\mathbb{E}[f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}_{t-1}) - f(\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\theta}^*)] + 2\bar{M}. \qquad (34)
\end{aligned}$$

Combining inequalities (31), (33) and (34), we have

$$\begin{aligned}
&\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \\
&\leq \Big(1 - \eta H(1 - 2L\eta)\Big)\mathbb{E}[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2] + 2\eta^2\bar{M}.
\end{aligned}$$

For $\eta \leq \frac{1}{2L}$ and $\bar{c} = (1 - \eta H(1 - 2L\eta))$, we have

$$\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \leq \bar{c}\,\mathbb{E}[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2] + 2\eta^2\bar{M}.$$

Equivalently,

$$\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] + \frac{2\eta^2\bar{M}}{\bar{c}-1} \leq \bar{c}\bigg(\mathbb{E}[\|\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}^*\|_2^2] + \frac{2\eta^2\bar{M}}{\bar{c}-1}\bigg),$$

which shows

$$\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] + \frac{2\eta^2\bar{M}}{\bar{c}-1} \leq \bar{c}^t\bigg(\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2 + \frac{2\eta^2\bar{M}}{\bar{c}-1}\bigg).$$

Thus, for the $\mathbb{E}[\|\theta_t - \theta^*\|_2^2]$, we have

$$\mathbb{E}[\|\boldsymbol{\theta}_t - \boldsymbol{\theta}^*\|_2^2] \leq \bar{c}^t\|\boldsymbol{\theta}_0 - \boldsymbol{\theta}^*\|_2^2 + \frac{2\eta^2\bar{M}(\bar{c}^t-1)}{\bar{c}-1}.$$

□

C DESCRIPTION OF EXPERIMENT MODELS

In this section we give detailed descriptions of the models used in the experiments in Section 5.

C.1 LOGISTIC REGRESSION

Summary A classification task on the *titanic* dataset with a Bayesian logistic regression model. We perform inference using the reparameterisation gradient formulation of Variantional Inference, where we select a Gaussian approximate posterior and learn its mean vector and full covariance matrix. We place a unit Gaussian prior on the weights.

Dataset The *titanic* dataset consists of 2201 training examples. Each data point comprises a binary class label and a feature vector of length 4. We perform standard normalisation on the features, i.e., subtracting the mean and dividing by the standard deviation. The dataset can be obtained from the following URL: `http://persoal.citius.usc.es/manuel.fernandez.delgado/papers/jmlr/data.tar.gz`.

Model description In the Bayesian logistic regression model, the output log-odds are modelled as a linear transformation of the inputs, i.e., $\text{logit}(t) = \omega^t x \iff t = s(\omega^t x)$, where t is the target output, x is a feature vector (with a leading element with value 1 to represent the bias), ω is a weight vector and $s(z) = \frac{1}{1+e^{-z}}$ is the logistic sigmoid function. This model can be treated probabilistically by setting a prior on the weights ω and a likelihood model on the targets t. We set the following model:

$$\begin{aligned} p(\omega) &= \mathcal{N}(0, \Sigma), && \text{Gaussian prior on the weight vector,} \\ p(t|\omega) &= \text{Bernoulli}(s(\omega^\top x)), && \text{Bernoulli likelihood model.} \end{aligned}$$

Inference problem We aim to find the posterior distribution of the weight given the targets $T = (t_1, \ldots, t_N)$,

$$p(\omega|T) = \frac{\prod_{n=1}^N p(t_n|\omega)p(\omega)}{\int \prod_{n=1}^N p(t_n|\omega)p(\omega)d\omega}.$$

For this inference task we use Variational Inference where we approximate $p(\omega|T)$ with another distribution $q(\omega) = \mathcal{N}(m, S)$, which can be obtained by minimising the following objective function, known as the Negative Evidence Lower Bound (NELBO):

$$\begin{aligned} \mathcal{L}(\theta) &= -\sum_{n=1}^N \mathbb{E}_{q(\omega)}[\log p(t_n|\omega)] + \text{KL}(q(\omega)||p(\omega)) \\ &\approx -\sum_{n=1}^N \frac{1}{S}\sum_{s=1}^S \log p(t_n|\omega(\boldsymbol{\epsilon}_n^{(s)})) + \text{KL}(q(\omega)||p(\omega)), \qquad \boldsymbol{\epsilon}_n^{(s)} \sim \mathcal{N}(0, I), \end{aligned} \tag{29}$$

where $\theta = (m, L)$ with $S = LL^\top$ and $\omega(\boldsymbol{\epsilon}) = m + L\boldsymbol{\epsilon}$ is overloaded to represent the location-scale transformation (reparameterisation trick).

Instantiation For the experiments in Section 5, we instantiate the model by setting the prior covariance to $\Sigma = I$. For the doubly stochastic objective function in (29), we set the number of Monte Carlo samples to $S = 1$ and sub-sample the N data points to mini-batches of size $|\mathcal{B}| = 10$ (some results for $|\mathcal{B}| = 100$ are also shown in Appendix E.1).

C.2 DEEP GAUSSIAN PROCESSES

Summary A regression task on the *airfoil* dataset using a Deep Gaussian Process (DGP). We use a 2-layer model with an inner-layer dimension of 5, and a Squared Exponential kernel for the GP priors. We use the doubly stochastic formulation of the Variational Inference problem (Salimbeni and Deisenroth, 2017). We learn the parameters of the approximate Gaussian posterior, keeping the hyperparameters fixed. The inducing locations are fixed and selected as the centroids of k-means clusters from the data.

Dataset The *airfoil* dataset consists of 1500 training examples. Each data point comprises a target label in $\mathbb{R}$ and a feature vector of length 5. We perform standard normalisation on the features, i.e., subtracting the mean and dividing by the standard deviation. The dataset can be obtained from the following URL: `https://drive.google.com/file/d/0BxWe_IuTnMFcYXhxdUNwRHBKTlU/view`.

Model description Deep Gaussian processes model the outputs as a non-parametric function of the inputs. Since in our experiments we use a 2-layer model, we will restrict the description to this case. For a target $t \in \mathbb{R}$ with corresponding inputs $x \in \mathcal{F}$, we can model the input-to-output relationship as follows:

$$t = f_2(f_1(x)) + \eta, \qquad \eta \sim \mathcal{N}(0, \sigma^2),$$

i.e., the outputs are noisy perturbations of a function composition of the inputs. f_1 and f_2 are not explicitly defined, instead we set Gaussian process priors on their values. Hence, the full probabilistic model is given by:

$$\begin{aligned}
p(u_1) &= \mathcal{N}(0, K_{u_1u_1}), && \text{GP prior,}\\
p(f_1|u_1) &= \mathcal{N}(\mathrm{id}(X) + K_{f_1u_1}K_{u_1u_1}^{-1}u_1, K_{f_1f_1} - K_{f_1u_1}K_{u_1u_1}^{-1}K_{f_1u_1}^\top), && \text{GP conditional prior,}\\
p(u_2) &= \mathcal{N}(0, K_{u_2u_2}), && \text{GP prior,}\\
p(f_2|u_2) &= \mathcal{N}(K_{f_2u_2}K_{u_2u_2}^{-1}u_2, K_{f_2f_2} - K_{f_2u_2}K_{u_2u_2}^{-1}K_{f_2u_2}^\top), && \text{GP conditional prior,}\\
p(t|f_1, f_2) &= \mathcal{N}(f_2(f_1), \sigma^2 I), && \text{Gaussian likelihood model.}
\end{aligned}$$

Here, Z_1 is a design matrix of M *inducing locations*, i.e., pseudo-inputs that are introduced for computational convenience. $K_{u_1,u_1} = k_1(Z_1, Z_1)$ is an $M \times M$ Gram matrix generated by a kernel function $k_1(\cdot,\cdot)$. Similarly $K_{f_1,f_1} = k_1(X, X)$ is an $N \times N$ Gram matrix, where X is the design matrix of N inputs x_n, and $\mathrm{id}(\cdot)$ is the identity function. $K_{u_2,u_2} = k_2(Z_2, Z_2)$ is also an $M \times M$ Gram matrix generated by $k_2(\cdot,\cdot)$, where Z_2 are the inducing locations for the second layer, and $K_{f_2,f_2} = k_2(f_1(X), f_1(X))$ is an $N \times N$ Gram matrix of the transformation of the inputs by the first layer. Finally, u_1 and u_2 are auxiliary variables introduced for computational tractability. In-depth treatment on the DGP model can be found in (Salimbeni and Deisenroth, 2017).

Inference problem We aim to find the posterior distribution f_1 and f_2 (and by extension u_1 and u_2) given the targets $T = (t_1, \ldots, t_N)$,

$$p(u_1, f_1, u_2, f_2|T) = \frac{\prod_{n=1}^N p(t_n|f_1, f_2)p(f_2|u_2)p(u_2)p(f_1|u_1)p(u_1)}{\int \prod_{n=1}^N p(t_n|f_1, f_2)p(f_2|u_2)p(u_2)p(f_1|u_1)p(u_1)df_2du_2df_1du_1}.$$

This inference problem is generally intractable, hence we use the doubly stochastic formulation of Variational Inference for DGPs (Salimbeni and Deisenroth, 2017). We approximate $p(u_1, f_1, u_2, f_2|T) \approx p(f_2|u_2)q(u_2)p(f_1|u_1)q(u_1)$, where $q(u_1) = \mathcal{N}(m_1, L_1L_1^\top)$ and $q(u_2) = \mathcal{N}(m_2, L_2L_2^\top)$. The parameters $\theta = (m_1, L_1, m_2, L_2)$ of the approximation can be learned by optimising the following NELBO:

$$\begin{aligned}
\mathcal{L}(\theta) &= -\sum_{n=1}^N \mathbb{E}_{q(f_2|f_1)q(f_1)}[\log p(t_n|f_1, f_2)] + \mathrm{KL}(q(u_1)\|p(u_1)) + \mathrm{KL}(q(u_2)\|p(u_2))\\
&\approx -\sum_{n=1}^N \frac{1}{S}\sum_{s=1}^S \log p(t_n|f_1(\boldsymbol{\epsilon}_{1n}^{(s)}), f_2(\boldsymbol{\epsilon}_{2n}^{(s)})) + \mathrm{KL}(q(u_1)\|p(u_1)) + \mathrm{KL}(q(u_2)\|p(u_2)), \qquad \boldsymbol{\epsilon}_{1n}^{(s)}, \boldsymbol{\epsilon}_{2n}^{(s)} \sim \mathcal{N}(0, I),
\end{aligned} \tag{30}$$

where $f_1(\boldsymbol{\epsilon}) = \tilde{m}_1 + \tilde{L}_1\boldsymbol{\epsilon}$ is overloaded to represent the location-scale transformation, and $\tilde{m}_1$ and $\tilde{L}_1$ are the parameters of the marginal $q(f_1) = \int p(f_1|u_1)q(u_1)du_1$. Similarly for $f_2(\boldsymbol{\epsilon})$.

Instantiation For the experiments in Section 5, we instantiate the model by setting $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ as Squared Exponential kernels, with lengthscales fixed to 2 and signal variances also fixed to 2. We set the number of inducing locations to $M = 10$ and their values to the centroids of k-means clusters of the real inputs from the dataset. We set the likelihood variance to $\sigma^2 = 0.01$ and the width of the GP inner layer to 5. For the doubly stochastic NELBO in (30), we set the number of Monte Carlo samples to $S = 1$ and sub-sample the N data points to mini-batches of size $|\mathcal{B}| = 10$ (some results for $|\mathcal{B}| = 100$ are also shown in Appendix E.1).

D VERIFICATION OF VARIANCE REDUCTION

In this section, we provide extra results for the experiment in Section 5.2 for different recognition network objectives and different Adam learning rates on these objectives. The corresponding configurations are in the figure captions.

D.1 LOGISTIC REGRESSION RESULTS ON THE *TITANIC* DATASET

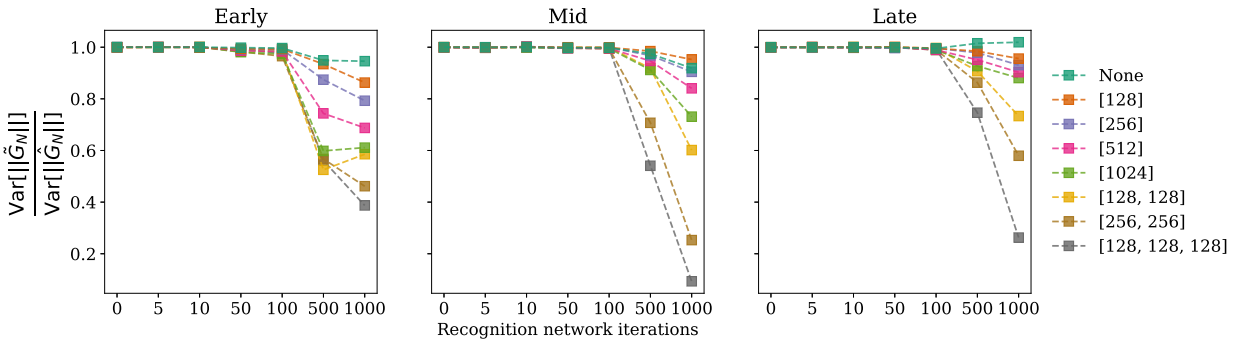


Figure 5: Logistic regression. Squared difference objective. Recognition network learning rate = 10^{-3}.

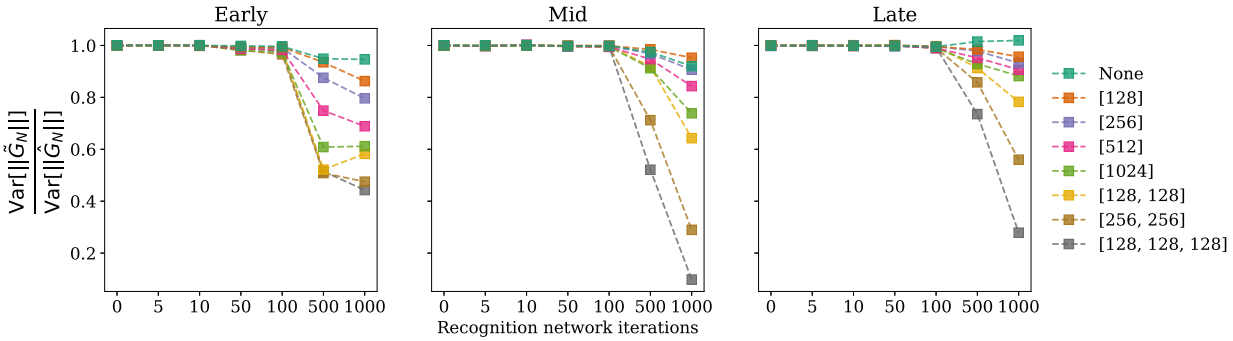


Figure 6: Logistic regression. Gradient sum objective. Recognition network learning rate = 10^{-3}.

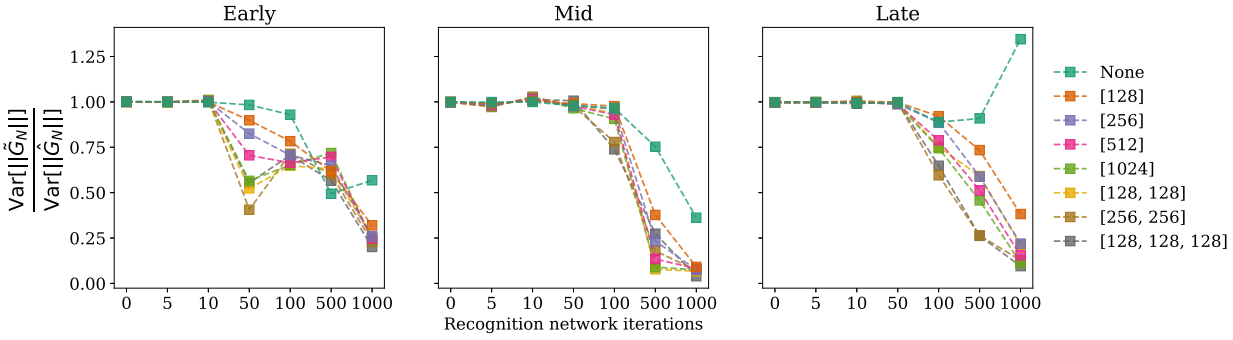


Figure 7: Logistic regression. Squared difference objective. Recognition network learning rate = 10^{-2}.

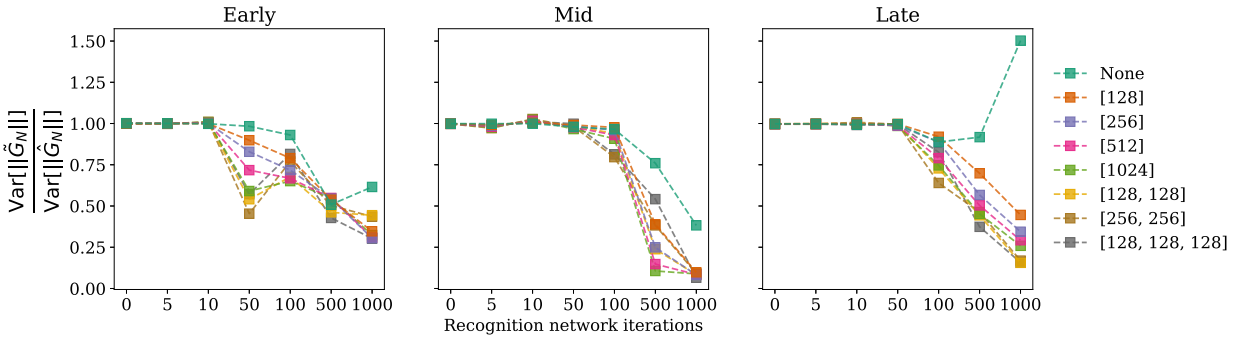


Figure 8: Logistic regression. Squared difference objective. Recognition network learning rate = 10^{-2}.

D.2 DEEP GAUSSIAN PROCESS RESULTS ON THE *AIRFOIL* DATASET

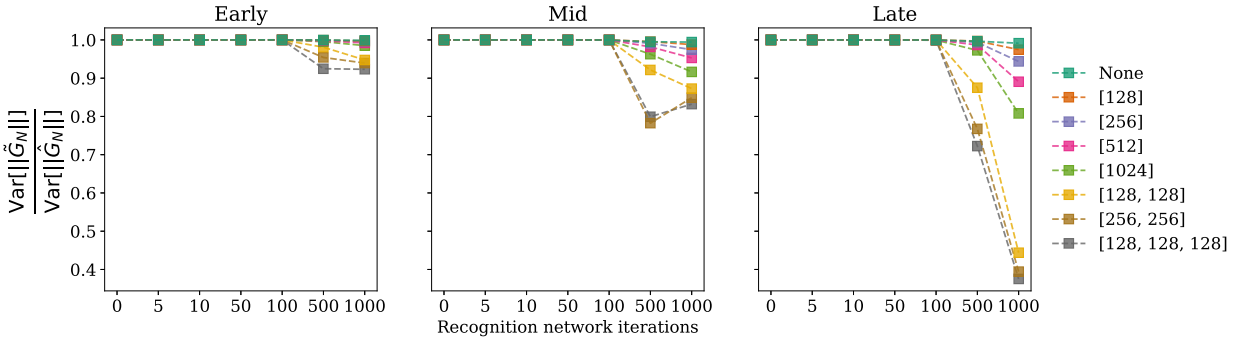


Figure 9: DGP. Squared difference objective. Recognition network learning rate = 10^{-3}.

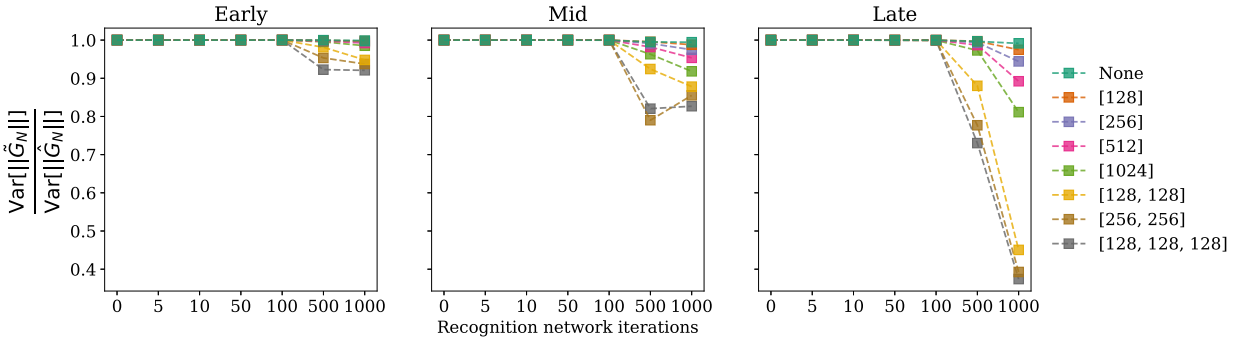


Figure 10: DGP. Gradient sum objective. Recognition network learning rate = 10^{-3}.

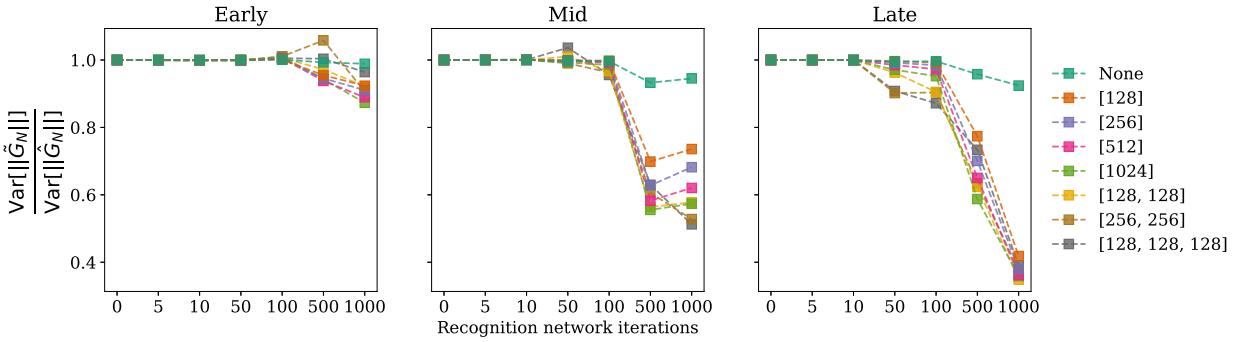


Figure 11: DGP. Squared difference objective. Recognition network learning rate = 10^{-2}.

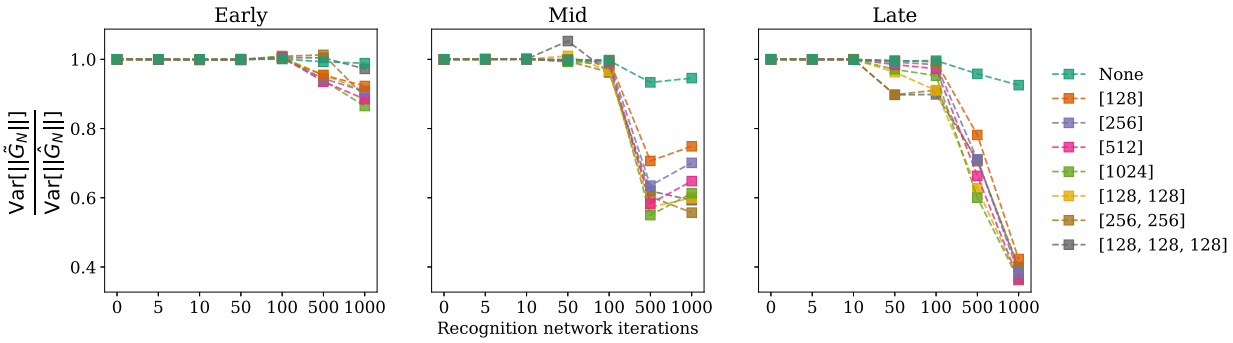


Figure 12: DGP. Gradient sum objective. Recognition network learning rate = 10^{-2}.

E SIMULTANEOUS OPTIMISATION OF MODEL AND RECOGNITION NETWORK

In this section, we provide extra results for the experiment in Section 5.3 for different recognition network objectives and different Adam learning rates on these objectives. We also vary the mini-batch size. The corresponding configurations are in the figure captions.

E.1 LOGISTIC REGRESSION RESULTS ON THE *TITANIC* DATASET

E.1.1 Small mini-batch size, $|\mathcal{B}| = 10$

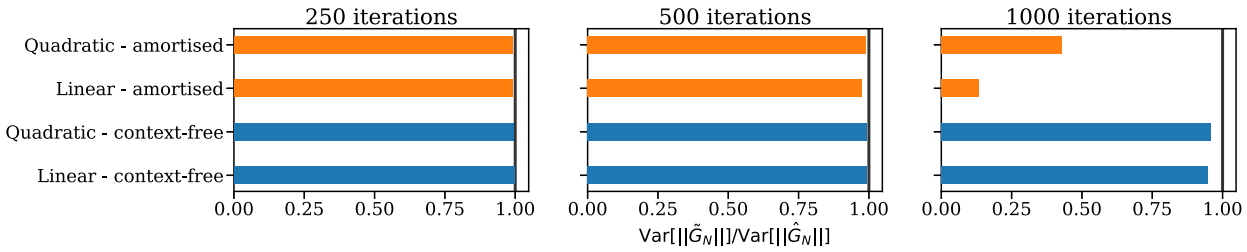


Figure 13: Logistic regression. Squared difference objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

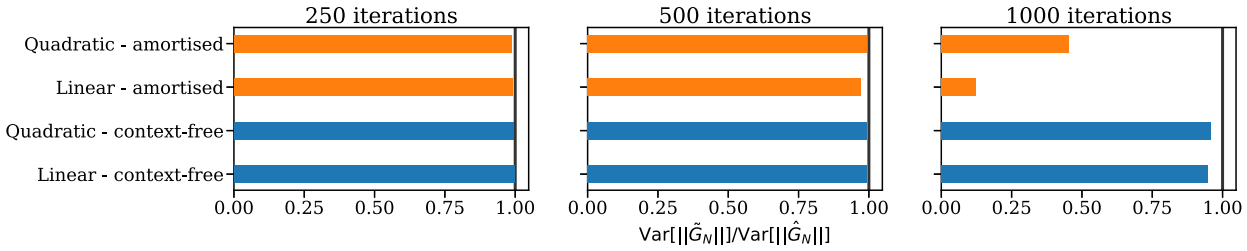


Figure 14: Logistic regression. Gradient sum objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

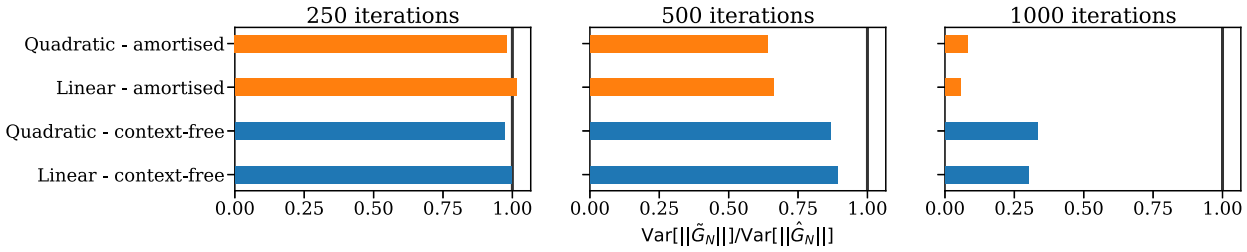


Figure 15: Logistic regression. Squared difference objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-2}.

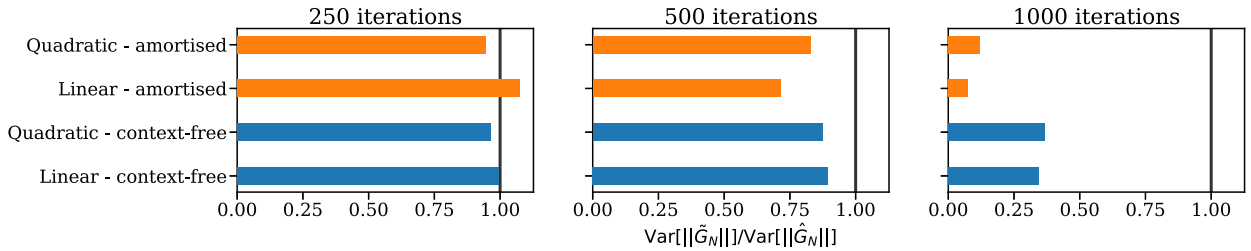


Figure 16: Logistic regression. $|\mathcal{B}| = 10$. Gradient sum objective. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-2}.

E.1.2 Large mini-batch size, $|\mathcal{B}| = 100$

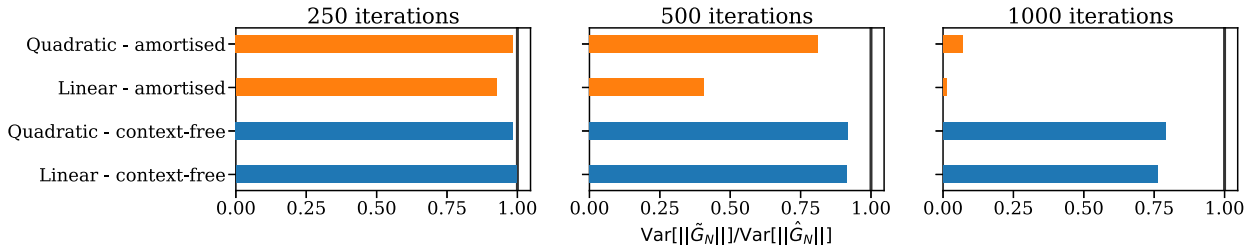


Figure 17: Logistic regression. Squared difference objective. Network of size [128, 128, 128]. $|\mathcal{B}| = 100$. Recognition network learning rate = 10^{-3}.

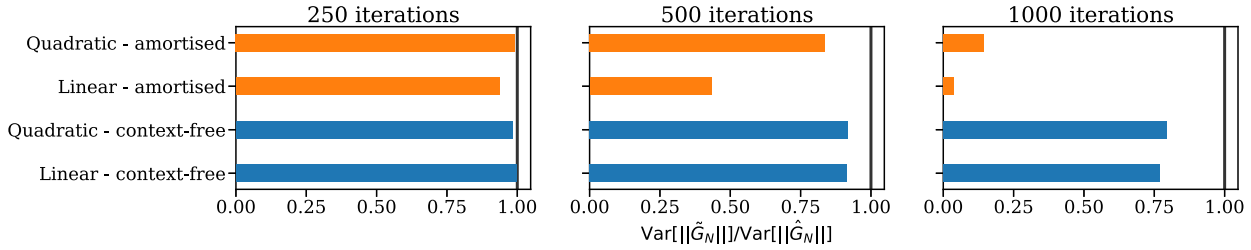


Figure 18: Logistic regression. Gradient sum objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

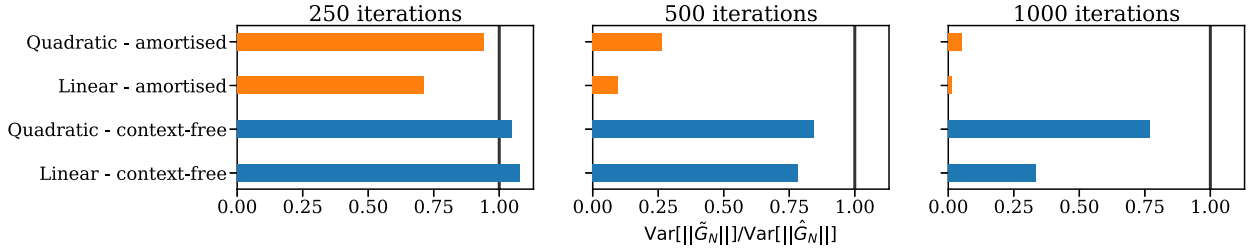


Figure 19: Logistic regression. Squared difference objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-2}.

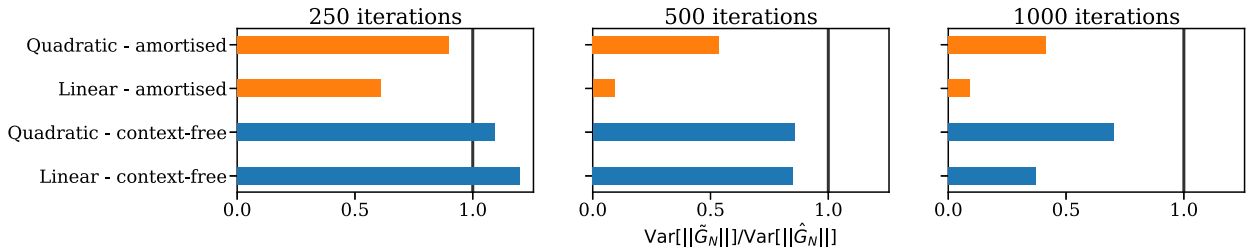


Figure 20: Logistic regression. Gradient sum objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-2}.

E.2 DEEP GAUSIAN PROCESS RESULTS ON THE *AIRFOIL* DATASET

E.2.1 Small mini-batch size, $|\mathcal{B}| = 10$

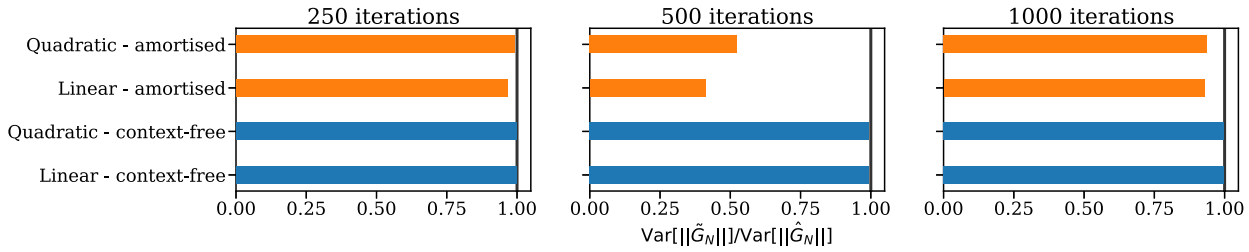


Figure 21: DGP. Squared difference objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

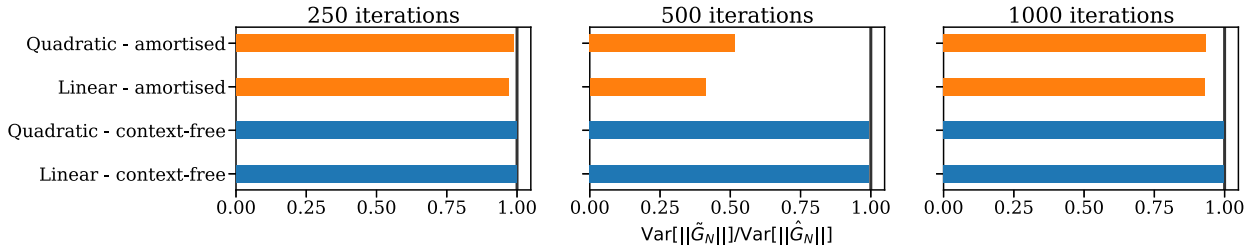


Figure 22: DGP. Gradient sum objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

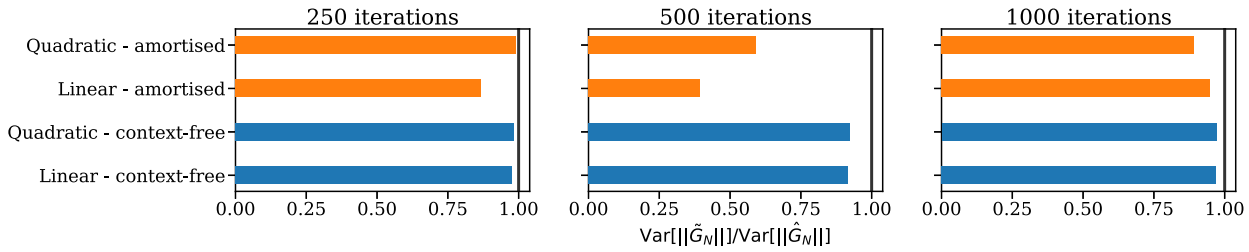


Figure 23: DGP. Squared difference objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-2}.

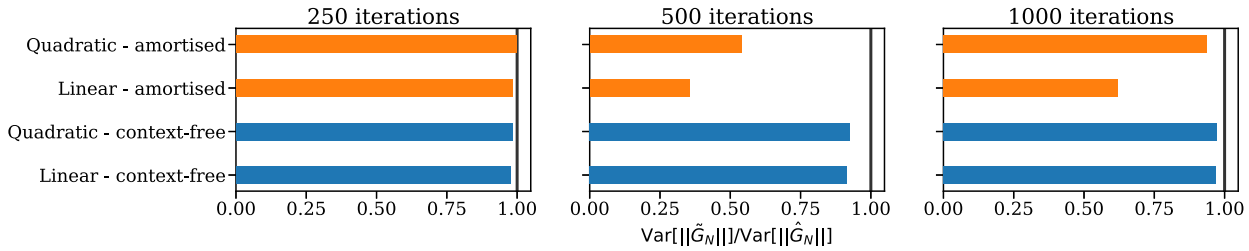


Figure 24: DGP. Gradient sum objective. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-2}.

E.2.2 Large mini-batch size, $|\mathcal{B}| = 100$

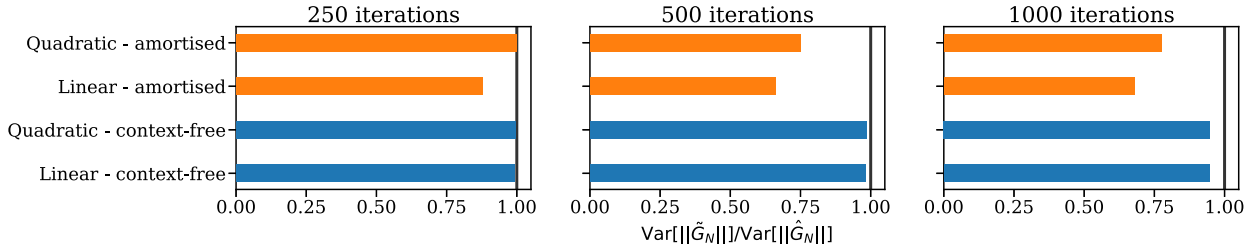


Figure 25: DGP. Squared difference objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

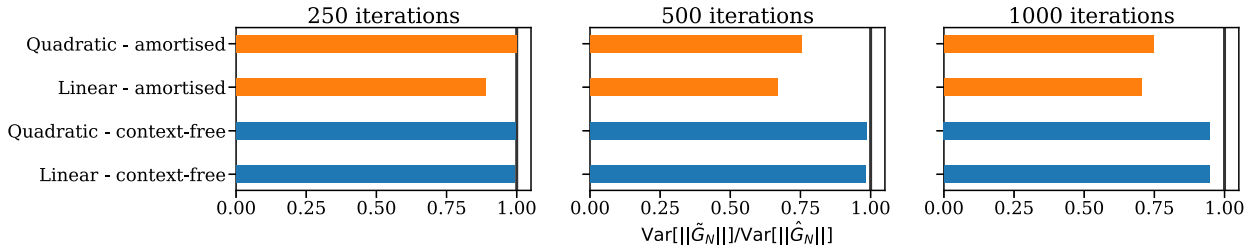


Figure 26: DGP. Gradient sum objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.

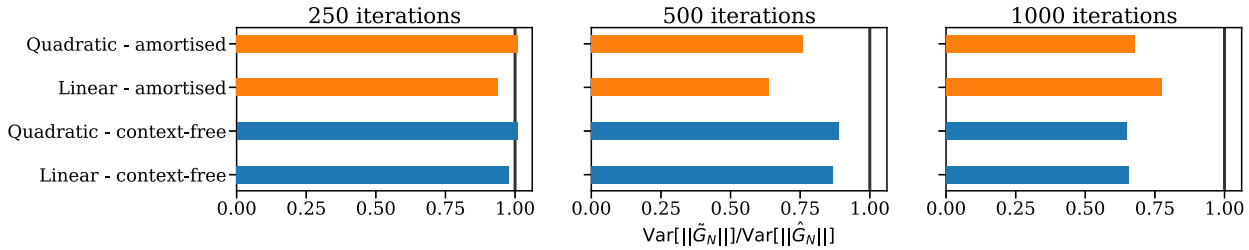


Figure 27: DGP. Squared difference objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-2}.

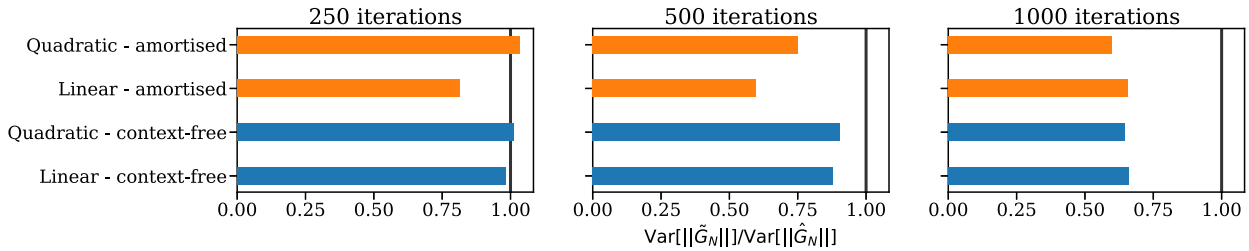


Figure 28: DGP. Gradient sum objective. $|\mathcal{B}| = 100$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-2}.

E.2.3 VARIANCE REDUCTION COMPARISON WITH THE PARTIAL GRADIENTS OBJECTIVE

In this section, we provide results comparing the *partial gradients* objective in (11), with the *gradient sum* (14) and *squared difference* (17) objectives. Fig. 29 shows that the partial gradients objective induces further variance reduction than the alternatives due to its lower variance. However, we note that in practice partial gradients are expensive to compute with current automatic differentiation libraries, therefore it is not feasible to use this objective from a computational point-of-view.

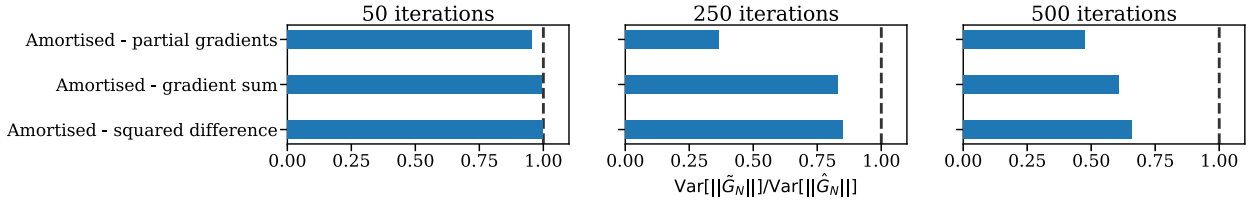


Figure 29: DGP. $|\mathcal{B}| = 10$. Network of size [128, 128, 128]. Recognition network learning rate = 10^{-3}.