
Adjusting Model Size in Continual Gaussian Processes: How Big is Big Enough?

Guimar Pescador-Barrios¹ Sarah Filippi¹ Mark van der Wilk²

Abstract

Many machine learning models require setting a parameter that controls their size before training, e.g. number of neurons in DNNs, or inducing points in GPs. Increasing capacity typically improves performance until all the information from the dataset is captured. After this point, computational cost keeps increasing, without improved performance. This leads to the question “How big is big enough?” We investigate this problem for Gaussian processes (single-layer neural networks) in continual learning. Here, data becomes available incrementally, and the final dataset size will therefore not be known before training, preventing the use of heuristics for setting a fixed model size. We develop a method to automatically adjust model size while maintaining near-optimal performance. Our experimental procedure follows the constraint that any hyperparameters must be set without seeing dataset properties, and we show that our method performs well across diverse datasets without the need to adjust its hyperparameter, showing it requires less tuning than others.

1. Introduction

Continual learning aims to train models when the data arrives in a stream of batches, without storing data after it has been processed, and while obtaining predictive performance that is as high as possible at each point in time (Ring, 1997). Selecting the size of the model is challenging in this setting, since typical non-continual training procedures do this by trial-and-error (cross-validation) using repeated training runs, which is not possible under our requirement of not storing any data. This is a crucial parameter to set well,

since if the model is too small, predictive performance will suffer. One solution could be to simply make all continual learning models so large, that they will always have enough capacity, regardless of what dataset and what amount of data they will be given. However, this “worst-case” strategy is wasteful of computational resources.

A more elegant solution would be to grow the size of the model adaptively as data arrives, according to the needs of the problem (see Figure 1 for an illustration). For example, if data were only ever gathered from the same region, there would be diminishing novelty in every new batch. In this case, one would want the model size to stabilise, with growth resuming once data arrives from new regions. In this paper, we investigate a principle for determining how to select the size of a model so that it is sufficient to obtain near-optimal performance, while otherwise wasting a minimal amount of computation. In other words, we seek to answer the question of “how big is big enough?” for setting the size of models throughout continual learning.

We investigate this question for Gaussian processes, since excellent continual learning methods exist that perform very similarly to full-batch methods, but which assume a fixed model capacity that is large enough. We provide a criterion for determining the number of inducing variables (analogous to neurons) that are needed whenever a new batch of data arrives. We show that our method is better able to maintain performance close to optimal full-batch methods, with a slower growth of computational resources compared to other continual methods, which either make models too small and perform poorly, or too large and waste computation. One hyperparameter needs to be tuned to control the balance between computational cost and accuracy. However, a single value works similarly across datasets with different properties, allowing all modelling decisions to be made before seeing any data.

Our approach benefits from Bayesian nonparametric perspectives, by separating the specification of the capacity of the model, which should be able to learn from an unbounded amount of data, from the specification of the approximation, which determines how much computational effort is needed to represent the solution for the current finite dataset. Our approach relies on the variational inducing variable approx-

¹Department of Mathematics, Imperial College London, UK

²Department of Computer Science, University of Oxford, UK.
Correspondence to: Guimar Pescador-Barrios <glp22@ic.ac.uk>.

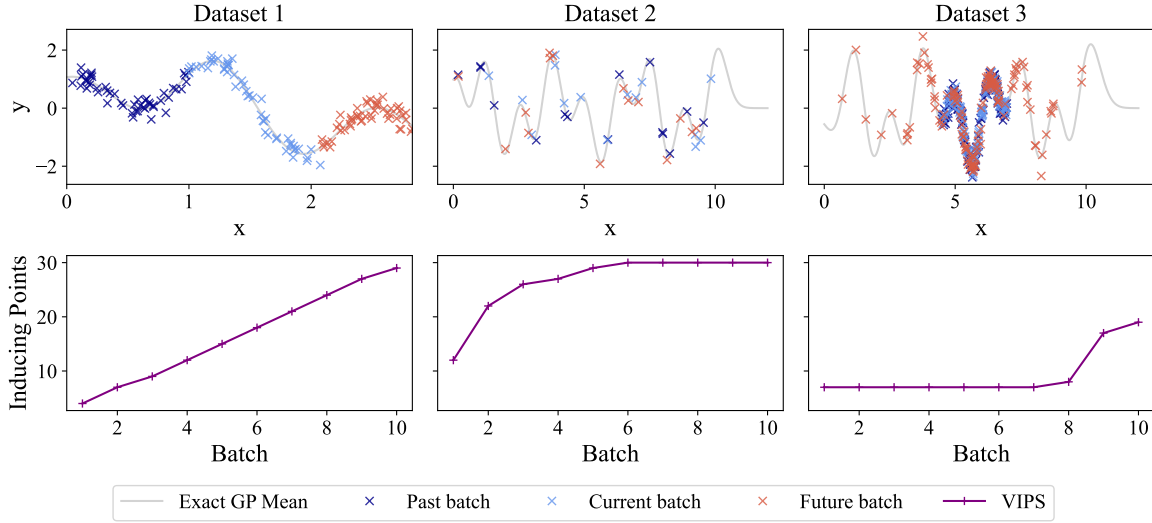


Figure 1. Three continual learning scenarios with different capacity requirements. **Top:** Three consecutive batches for 1) a growing input space, 2) i.i.d. samples from a uniform distribution, and 3) narrow-range samples with occasional outliers. **Bottom:** Number of inducing points selected using the VIPS algorithm at each batch. When selecting model size with our method, we observed: 1) a linear increase, 2) after initial training, we see a halt in growth, and 3) a low model size until it encounters outliers.

imation of Titsias (2009), which leads to an interpretation that our method automatically adjusts the width of a single-layer neural network. Our hope is that such perspectives will be useful for the development of adaptive deep neural networks, which would allow models to be trained on large amounts of data beyond the limitations of storage infrastructure, or learning from streams of data that come directly from interaction with an environment.

2. Related Work

The foundational problem that makes training with gradient descent insufficient for continual learning is *catastrophic forgetting*, where previously acquired knowledge is lost in favour of recent information (McCloskey & Cohen, 1989; Goodfellow et al., 2013). Many solutions have been proposed in the literature, such as encouraging weights to be close to values that were well-determined by past data (Kirkpatrick et al., 2017; Schwarz et al., 2018), storing subsets or statistics of past data to continue to train the neural network in the future (Li & Hoiem, 2017; Lopez-Paz & Ranzato, 2017), and approximate Bayesian methods that balance uncertainty estimates of parameters with the strength of the data (Bui et al., 2017; Nguyen et al., 2018; Rudner et al., 2022; Chang et al., 2023b). Within continual learning, many different settings have been investigated, which vary in difficulty (Farquhar & Gal, 2018). Across these tasks, the gap in performance to a full-batch training procedure therefore also varies, but despite progress, some gap in performance

remains.

Bayesian continual learning methods have been developed because the posterior given past data becomes the prior for future data, making the posterior a sufficient quantity to estimate (Murphy, 2023, §19.7). For the special case of linear-in-the-parameters regression models, the posterior and updates can be calculated in closed form, leading to continual learning giving *exactly* the same result as full-batch training. In most cases (e.g. for neural networks), the posterior cannot be found exactly, necessitating the aforementioned approximations that use an approximate posterior as the sufficient quantity (Nguyen et al., 2018; Rudner et al., 2022; Chang et al., 2023b).

Even with a perfect solution to catastrophic forgetting (e.g. in the case of linear-in-the-parameters regression models), continual learning methods face the additional difficulty of ensuring that models have sufficient capacity to accommodate the continuously arriving information. In continual learning, it is particularly difficult to determine a fixed size for the model, since the number of data or tasks are not yet known, and selecting a model that is too small can significantly hurt performance. Growing models with data can fix this problem. For example, Rusu et al. (2016) extend hidden representations by a fixed amount for each new batch. Yoon et al. (2018) argue that extension by a fixed amount is wasteful and should instead be data dependent, specifically by copying neurons if their value changes too much, and adding new neurons if the training loss doesn't

reach a particular threshold. Kessler et al. (2021) propose to use the Indian Buffet Process as a more principled way to regularise how fast new weights are added with tasks. Maile et al. (2022); Mitchell et al. (2023) develop similar ideas of automatically adjusting model size to the normal batch-learning setting. We agree with the aims of these papers, but note that they rely on thresholds that need to be tuned to the dataset characteristics, e.g. tolerances on the training error which depend on the amount of intrinsic noise in the dataset. We seek a unifying training principle that is broadly applicable to a wide range of datasets, without knowing their properties a priori.

Growing model capacity with dataset size was one of the main justifications for research into (Bayesian) non-parametric models (Ghahramani, 2013; 2015). This approach defines models with infinite capacity, with Bayesian inference naturally using an appropriate finite capacity to make predictions, with finite compute budgets. Gaussian processes (GPs) (Rasmussen & Williams, 2005) are the most common Bayesian non-parametric model for supervised learning, and are equivalent to infinitely-wide deep neural networks (Neal, 1996; Matthews et al., 2018) and linear-in-the-parameters models with an infinite feature space (Mercer, 1909). Their infinite capacity allows them to recover functions perfectly in the limit of infinite data (van der Vaart & van Zanten, 2008), and their posterior can be computed in closed form. These are strong principles for providing high-quality solutions to both catastrophic forgetting and ensuring appropriate capacity, and therefore make GPs an excellent model for studying continual learning.

However, developing a practical continual learning in GPs is not as straightforward as it is in finite dimensional linear models, because (for N datapoints) the posterior requires **1)** an intractable $O(N^3)$ computation to compute it exactly, and **2)** storing the full training dataset, which breaks the requirements of continual learning. Sparse variational inducing variable methods solve these problems (Titsias, 2009), by introducing a small number of M inducing points that control the capacity of the posterior approximation. In certain settings (e.g. bounded inputs), this approximation is near-exact even when $M \ll N$ (Burt et al., 2019; 2020). This property has allowed continual learning methods to be developed for GPs that perform very closely to full-batch methods (Bui et al., 2017; Maddox et al., 2021; Chang et al., 2023a), *provided* M is large enough.

As in neural network models, automatically selecting the capacity M is an open problem, with several proposed solutions. Kapoor et al. (2021) acknowledge the need for scaling the capacity with data size, and propose VAR-GP (Variational Autoregressive GP) which adds a fixed number of inducing points for every batch. However, this number may be too small, leading to poor performance, or too large,

leading to wasted computation. Galy-Fajou & Opper (2021) propose OIPS (online inducing point selection), which determines M through a threshold on the correlation with other inducing points, and is currently the only adaptive method for GPs, although we show that it still relies on tuned hyperparameters to work.

In this work, we propose to select the capacity of the variational approximation by selecting an appropriate tolerance in the KL gap to the true posterior. This criterion works within the same computational constraints as existing GPs continual learning methods, adapts the capacity to the dataset to minimise computational waste while retaining near-optimal predictive performance. Our method has a single hyperparameter that we keep fixed to a single value, which produces similar trade-offs between size and performance across benchmark datasets with different characteristics.

3. Background

3.1. Sparse Variational Gaussian Processes

We consider the typical regression setting, with training data consisting of N input/output pairs $\{\mathbf{x}_n, y_n\}_{n=1}^N$, $\mathbf{x}_n \in \mathbb{R}^D$, $y_n \in \mathbb{R}$. We model y_n by passing $\mathbf{x}_n$ through a function followed by additive Gaussian noise $y_n = f(\mathbf{x}_n) + \epsilon_n$, $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$, and take a Gaussian process prior on $f \sim \mathcal{GP}(0, k_\theta(\cdot, \cdot))$ with zero mean, and a kernel k with hyperparameters θ . We wish to find the posterior (for prediction) and marginal likelihood (for finding θ). We need approximations, despite these quantities being available in closed form (Rasmussen & Williams, 2005), since they have a computational cost of $O(N^3)$ that is too high, and require all training data (or statistics greater in size) to be stored, both of which are prohibitive for continual learning. Variational inference requires a lower $O(NM^2)$ computational and $O(NM)$ memory cost by selecting an approximation from a set of tractable posteriors:

$$q(f(\cdot)) = \int p(f(\cdot) | \mathbf{u}, \theta) q(\mathbf{u}) d\mathbf{u} \quad (1)$$

$$= \mathcal{N}(f(\cdot); \mathbf{k}_{\cdot\mathbf{u}} \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{m}, k(\cdot, \cdot) - \mathbf{k}_{\cdot\mathbf{u}} \mathbf{K}_{\mathbf{uu}}^{-1} (\mathbf{K}_{\mathbf{uu}} - \mathbf{S}) \mathbf{K}_{\mathbf{uu}}^{-1} \mathbf{k}_{\cdot\mathbf{u}}) \quad (2)$$

with $[\mathbf{K}_{\mathbf{uu}}]_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$, $[\mathbf{k}_{\cdot\mathbf{u}}]_i = [\mathbf{k}_{\mathbf{u}}^T]_i = k(\cdot, \mathbf{z}_i)$, $\mathbf{Z} = \{\mathbf{z}_m\}_{m=1}^M$, and $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}; \mathbf{m}, \mathbf{S})$. The variational parameters $\mathbf{m}, \mathbf{S}, \mathbf{Z}$ and hyperparameters θ are selected by maximising the Evidence Lower Bound (ELBO) (Hensman et al., 2013). This simultaneously minimises the KL gap $\text{KL}[q(f) || p(f|y, \theta)]$ between the approximate and true GP posteriors (Matthews et al., 2016; Matthews, 2017), and maximises an approximation to the marginal likelihood of

the hyperparameters:

$$\mathcal{L}_{\text{ELBO}} = \sum_{i=1}^N \mathbb{E}_{q(f(\mathbf{x}_i))} [\log p(y_i | f(\mathbf{x}_i), \theta)] - \text{KL}[q(\mathbf{u}) \parallel p(\mathbf{u} | \theta)] . \quad (3)$$

The variational approximation has the desirable properties (van der Wilk, 2019) of **1**) providing a measure of discrepancy between the finite capacity approximation, and the true infinite capacity model, **2**) arbitrarily accurate approximations if enough capacity is added (Burt et al., 2019; 2020), and **3**) retaining the uncertainty quantification over the infinite number of basis functions. In this work, we will particularly rely on being able to measure the quality of the approximation to help determine how large M should be.

3.2. Sparse Gaussian Processes are Equivalent to Single-Layer Neural Networks

The equations of the predictor (eq. (2)) show the strong connection between Sparse Gaussian processes and neural networks. The kernel $k(\cdot, z_i)$ forms the nonlinearity, with the weights being parameterised as $\mathbf{K}_{\text{uu}}^{-1} \mathbf{m}$. For inner product kernels $k(\mathbf{x}, \mathbf{Z}) = \sigma(\mathbf{Z}\mathbf{x})$ like the arc-cosine kernel (Cho & Saul, 2009), the connection becomes stronger, with $\mathbf{Z}$ as the input weights. This construction also arises from other combinations of kernels and inter-domain inducing variables (Dutordoir et al., 2020; Sun et al., 2020), and has also showed equivalences between deep Gaussian processes and deep neural networks with activations similar to the ReLU (Dutordoir et al., 2021). As a consequence, our method for determining the number of inducing variables in a sparse GP, equivalently finds the number of neurons needed in a single-layer neural network. As such, we hope the ideas presented in this work can inspire similar mechanisms for adaptive neural networks.

3.3. Online Sparse Gaussian Processes

In this work, we use the extension of the sparse variational GP approximation to the continual learning case developed by Bui et al. (2017). Here, we present a modified derivation that clarifies **1**) how the online ELBO provides an estimate to the full-batch ELBO, and **2**) when this approximation is accurate.

We update our posterior and hyperparameter approximations after each batch of new data $\{\mathbf{X}_n, \mathbf{y}_n\}$. While we do not have access to data from older batches $\{\mathbf{X}_o, \mathbf{y}_o\}$, the parameters specifying the approximate posterior $q_o(f) = p(f_{\neq \mathbf{a}} | \mathbf{a}, \theta_o) q_o(\mathbf{a})$ are passed on. We construct this approximate posterior as in eq. (1) but with $\mathbf{a} = f(\mathbf{Z}_o)$ and the old hyperparameters θ_o . Given the “old” $q_o(f)$, online sparse GPs construct a “new” approximation $q_n(f)$ of the posterior for all observed data $p(f | \mathbf{y}_o, \mathbf{y}_n, \theta_n)$, which can be written

as:

$$\begin{aligned} p(f | \mathbf{y}_o, \mathbf{y}_n, \theta_n) &= \frac{p(f | \theta_n) p(\mathbf{y}_n | f) p(\mathbf{y}_o | f)}{p(\mathbf{y}_n, \mathbf{y}_o | \theta_n)} \\ &= \frac{p(f | \theta_n) p(\mathbf{y}_n | f)}{p(\mathbf{y}_n, \mathbf{y}_o | \theta_n)} \frac{p(f | \mathbf{y}_o, \theta_o) p(\mathbf{y}_o | \theta_o)}{p(f | \theta_o)} . \end{aligned}$$

We denote the new variational distribution as $q_n(f) = p(f_{\neq \mathbf{b}} | \mathbf{b}, \theta_n) q_n(\mathbf{b})$ where $\mathbf{b} = f(\mathbf{Z}_n)$ and θ_n is the new hyperparameter setting. The KL divergence between the exact and approximate posterior at the current batch is given by:

$$\begin{aligned} \text{KL}[q_n(f) \parallel p(f | \mathbf{y}_o, \mathbf{y}_n, \theta_n)] &= \log \frac{p(\mathbf{y}_n, \mathbf{y}_o | \theta_n)}{p(\mathbf{y}_o | \theta_o)} \\ &\quad - \int q_n(f) \log \frac{p(f | \theta_n) p(\mathbf{y}_n | f) p(f | \mathbf{y}_o, \theta_o)}{q_n(f) p(f | \theta_o)} df . \end{aligned}$$

The posterior distribution $p(f | \mathbf{y}_o, \theta_o)$ is not available, however by multiplying its approximation $q_o(f)$ in both sides of the fraction inside the log, we obtain:

$$\begin{aligned} \text{KL}[q_n(f) \parallel p(f | \mathbf{y}_o, \mathbf{y}_n, \theta_n)] &= \log \frac{p(\mathbf{y}_n, \mathbf{y}_o | \theta_n)}{p(\mathbf{y}_o | \theta_o)} \\ &\quad - \underbrace{\int q_n(f) \log \frac{p(f | \theta_n) p(\mathbf{y}_n | f) q_o(f)}{q_n(f) p(f | \theta_o)} df}_{=\hat{\mathcal{L}}} + \Phi \end{aligned} \quad (4)$$

where $\Phi = - \int q_n(f) \log \frac{p(f | \mathbf{y}_o, \theta_o)}{q_o(f)} df$. We cannot compute Φ due to its dependence on the exact posterior, so we drop it and use the remaining term as our “online ELBO” training objective, which simplifies as:

$$\hat{\mathcal{L}} := \int q_n(f) \left[\log \frac{p(\mathbf{b} | \theta_n) q_o(\mathbf{a}) p(\mathbf{y}_n | f)}{q_n(\mathbf{b}) p(\mathbf{a} | \theta_o)} \right] df, \quad (5)$$

since $q_o(f) = p(f_{\neq \mathbf{a}} | \mathbf{a}, \theta_o) q_o(\mathbf{a})$ and similarly for $q_n(f)$.

Maximising $\hat{\mathcal{L}}$ will accurately minimise the KL to the true posterior when Φ is small, which is the case when the old approximation is accurate, i.e. $q_o(f) \approx p(f | \mathbf{y}_o, \theta_o)$ for all values of $f(\mathbf{X}_o)$ (with $\Phi = 0$ in the case of equality). In our continual learning procedure, we will keep our sequence of approximations accurate by ensuring they all have enough inducing points, which will keep Φ small.

To get our final bound, we perform a change of variables for the variational distribution $q_o(\mathbf{a}) = \mathcal{N}(\mathbf{a}; \mathbf{m}_a, \mathbf{S}_a)$ to use the likelihood parametrisation (Panos et al., 2018):

$$\begin{aligned} q_o(\mathbf{a}) &= \frac{\mathcal{N}(\mathbf{a}; \tilde{\mathbf{m}}_a, \mathbf{D}_a) \mathcal{N}(\mathbf{a}; 0, \mathbf{K}'_{aa})}{\int \mathcal{N}(\mathbf{a}; \tilde{\mathbf{m}}_a, \mathbf{D}_a) \mathcal{N}(\mathbf{a}; 0, \mathbf{K}'_{aa}) d\mathbf{a}} \\ &= \frac{l(\mathbf{a}) p(\mathbf{a} | \theta_o)}{\mathcal{N}(\mathbf{a}; 0, \mathbf{D}_a + \mathbf{K}'_{aa})}, \end{aligned} \quad (6)$$

where $\mathbf{D}_a = (\mathbf{S}_a^{-1} - \mathbf{K}'_{aa})^{-1}$ and $\tilde{\mathbf{m}}_a = \mathbf{K}'_{aa}^{-1} \mathbf{m}_a$ are the variational parameters, $\mathbf{K}'_{aa}$ is the covariance for the prior distribution $p(\mathbf{a} | \theta_o)$ and $l(\mathbf{a}) := \mathcal{N}(\mathbf{a}; \tilde{\mathbf{m}}_a, \mathbf{D}_a)$. In this formulation, the variational parameters $\tilde{\mathbf{m}}_a, \mathbf{D}_a$ effectively form a dataset that produce the same posterior as the original dataset, but which we have chosen to be smaller in size, $M < N$. This makes our online ELBO from eq. (5)

$$\begin{aligned} \hat{\mathcal{L}} = & \mathbb{E}_{q_n(f)} [\log p(\mathbf{y}_n | f)] + \mathbb{E}_{q_n(f)} [\log l(\mathbf{a})] \\ & - \text{KL}[q_n(\mathbf{b}) || p(\mathbf{b} | \theta_n)] - \log \mathcal{N}(\mathbf{a}; 0, \mathbf{K}'_{aa} + \mathbf{D}_a), \end{aligned} \quad (7)$$

which has the nice interpretation of being the normal ELBO, but with an additional term that includes the approximate likelihood $l(\mathbf{a})$ which summarises the effect of all previous data.

While $\hat{\mathcal{L}}$ is all that is needed to train the online approximation, it differs from the true marginal likelihood by the term $\log p(\mathbf{y}_o | \theta_o)$. To approximate it, we could drop the term $\log \mathcal{N}(\mathbf{a}; 0, \mathbf{K}'_{aa} + \mathbf{D}_a)$ from $\hat{\mathcal{L}}$, since this term also approximates $\log p(\mathbf{y}_o | \theta_o)$, with equality when the posterior is exact, but with no guarantee of being a lower bound.

Although $\hat{\mathcal{L}}$ is a useful training objective for general likelihoods, the regression case we consider allows us to analytically find $q(\mathbf{b})$ (Bui et al., 2017) (see App. E), resulting in the lower bound

$$\begin{aligned} \hat{\mathcal{L}} = & \log \mathcal{N}(\hat{\mathbf{y}}; \mathbf{0}, \mathbf{K}_{fb} \mathbf{K}_{bb}^{-1} \mathbf{K}_{bf} + \Sigma_{\hat{\mathbf{y}}}) + \Delta \\ & - \frac{1}{2} \text{tr}[\mathbf{D}_a^{-1}(\mathbf{K}_{aa} - \mathbf{Q}_{aa})] - \frac{1}{2\sigma^2} \text{tr}(\mathbf{K}_{ff} - \mathbf{Q}_{ff}) \end{aligned} \quad (8)$$

where $\mathbf{K}_{fb} = [\mathbf{K}_{fb} \quad \mathbf{K}_{ab}]^\top$, $\Sigma_{\hat{\mathbf{y}}} = \text{diag}([\sigma_y^2 \mathbf{I}, \mathbf{D}_a])$, $\hat{\mathbf{y}} = [\mathbf{y}_n \quad \mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a]^\top$, and

$$\begin{aligned} \Delta = & -\frac{1}{2} \log \frac{|\mathbf{S}_a|}{|\mathbf{K}'_{aa}| |\mathbf{D}_a|} + \frac{M_a}{2} \log(2\pi) \\ & - \frac{1}{2} \mathbf{m}_a^\top \mathbf{S}_a^{-1} \mathbf{m}_a + \frac{1}{2} \mathbf{m}_a^\top \mathbf{S}_a^{-1} \mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a, \end{aligned}$$

with $\mathbf{Q}_{ff} = \mathbf{K}_{fb} \mathbf{K}_{bb}^{-1} \mathbf{K}_{bf}$, $\mathbf{Q}_{aa} = \mathbf{K}_{ab} \mathbf{K}_{bb}^{-1} \mathbf{K}_{ba}$. All covariances are computed using the new hyperparameters θ_n , except for $\mathbf{K}'_{aa}$ which is the covariance for the prior distribution $p(\mathbf{a} | \theta_o)$. Finally, $M_a = |\mathbf{a}|$ is the number of inducing points used at the previous batch. For calculating $\hat{\mathcal{L}}$ in each batch, the computational complexity is $O(N_n M_b^2 + M_b^3)$ and the memory requirements are $O(M_b^2)$ where M_b is the total number of inducing points for the current batch.

To create a fully black-box solution, we still need to specify how to select the hyperparameters θ_n , the number of inducing variables M_b , and the inducing inputs $\mathbf{Z}_n$. We will always select θ_n by maximising $\hat{\mathcal{L}}$ using L-BFGS. To select the locations $\mathbf{Z}_n$, we use the ‘‘greedy variance’’ criterion (Fine & Scheinberg, 2001; Foster et al., 2009; Burt et al., 2020). This leaves only the number of inducing variables M_b to be selected.

4. Automatically Adapting Approximation Capacity

We propose a method for adjusting the capacity of the approximation M_b to maintain accuracy. We propose to keep inducing points from old batches fixed, and select new inducing points from each incoming batch, with their locations set using the ‘‘greedy variance’’ criterion (Fine & Scheinberg, 2001; Foster et al., 2009; Burt et al., 2020). While optimising all inducing points does lead to a strictly better approximation, we avoid this for the sake of simplicity. The question remains: To achieve a level of accuracy, ‘‘how big is big enough?’’ To answer this, we consider the online ELBO as a function of the capacity $\hat{\mathcal{L}}(M_b)$, and propose a threshold after which to stop adding new inducing variables.

4.1. Online Log Marginal Likelihood (LML) Upper Bound

The problem of selecting a sufficient number of inducing variables is also still open in the batch setting. One possible strategy is to use an upper bound to the marginal likelihood (Titsias, 2014) to bound $\text{KL}[q(f) || p(f | \mathbf{y})] \leq \mathcal{U} - \mathcal{L}$, and stop adding inducing variables once this is below a tolerance α . To extend this strategy to online learning, we begin by deriving an online upper bound, as a counterpart to the online ELBO from eq. (8). We follow the same strategy as Titsias (2014), by considering the highest possible value that our lower bound can attain. While in full-batch inference this is equal to the true LML, in our case this is obtained by keeping the inducing inputs from the previous iteration, and adding each new datapoint to the inducing set:

$$\mathcal{L}^* := \hat{\mathcal{L}}(N_n + M_a) = \log \mathcal{N}(\hat{\mathbf{y}}; \mathbf{0}, \mathbf{K}_{ff} + \Sigma_{\hat{\mathbf{y}}}) + \Delta_a \quad (9)$$

with $\mathbf{K}_{ff} = \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fa} \\ \mathbf{K}_{af} & \mathbf{K}_{aa} \end{bmatrix}$. Using properties of positive semi-definite matrices, we derive an upper bound $\hat{\mathcal{U}}(M)$ to eq. (9):

$$\begin{aligned} & \log \mathcal{N}(\hat{\mathbf{y}}; \mathbf{0}, \mathbf{K}_{ff} + \Sigma_{\hat{\mathbf{y}}}) \\ & \leq -\frac{(N + M_a)}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{K}_{fb} \mathbf{K}_{bb}^{-1} \mathbf{K}_{bf} + \Sigma_{\hat{\mathbf{y}}}| \\ & \quad - \frac{1}{2} \hat{\mathbf{y}}^\top (\mathbf{K}_{fb} \mathbf{K}_{bb}^{-1} \mathbf{K}_{bf} + t + \Sigma_{\hat{\mathbf{y}}})^{-1} \hat{\mathbf{y}} \\ & := \hat{\mathcal{U}}(M), \end{aligned}$$

where $t = \text{tr}(\mathbf{K}_{ff} - \mathbf{Q}_{ff})$ and $\mathbf{Q}_{ff} = \mathbf{K}_{fb} \mathbf{K}_{bb}^{-1} \mathbf{K}_{bf}$ and M is the number of inducing points used to calculate the bound (which can be unequal to M_b).

4.2. Approximation Quality Guarantees

Adding inducing points will eventually increase $\hat{\mathcal{L}}$ until it reaches $\mathcal{L}^*$ (Bauer et al., 2016; Matthews, 2017; Burt et al.,

Algorithm 1 Vegas Inducing Point Selection (VIPS)

Input: $\mathbf{X}_n = \{\mathbf{x}_i\}_{i=1}^{N_n}$, $\mathbf{Z}_o = \{\mathbf{z}_m\}_{m=1}^{M_a}$, $\hat{\mu}$, $\hat{\sigma}$, θ_n , kernel $k(\cdot, \cdot | \theta_n)$, threshold parameter δ .

Output: Updated set $\mathbf{Z}_n = \mathbf{Z}_o \cup \{\mathbf{x}_{m'}\}_{m'=1}^{M'}$, where $|\mathbf{Z}_n| = M_b$.

Initialise $\mathbf{Z}_n = \mathbf{Z}_o$.

while $\hat{\mathcal{U}}(M) - \hat{\mathcal{L}}(M_b) \leq \delta |\hat{\mathcal{U}}(M) - \mathcal{L}_{noise}(\hat{\mu}, \hat{\sigma})|$ **do**

Select $\mathbf{x} = \operatorname{argmax}_{\mathbf{x} \in \mathbf{X}_n} k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_b(\mathbf{x})^\top \mathbf{K}_{bb}^{-1} \mathbf{k}_b(\mathbf{x})$.

Add $\mathbf{x}$ to the set of inducing points: $\mathbf{Z}_n = \mathbf{Z}_n \cup \{\mathbf{x}\}$.

end while

2020). If we add inducing points until $\hat{\mathcal{U}}(M) - \hat{\mathcal{L}}(M_b) \leq \alpha$ we can guarantee the following:

Guarantee. Let M be a fixed integer and M_b be the number of selected inducing points such that $\hat{\mathcal{U}}(M) - \hat{\mathcal{L}}(M_b) \leq \alpha$. Assuming that $\theta_n = \theta_o$, we have two equivalent bounds:

$$\text{KL}[q_n(f) \parallel p(f|\mathbf{y}_o, \mathbf{y}_n, \theta_o)] \leq \alpha + \Psi \quad (10)$$

$$\text{KL}[q_n(f) \parallel q_n^*(f)] \leq \alpha \quad (11)$$

where $\Psi = \int q_n(f) \log \frac{q_n^*(f)}{p(f|\mathbf{y}_n, \mathbf{y}_o)} df$ and $q_n^*(f) = \mathcal{Z}^{-1} q_o(f) p(\mathbf{y}_n | f)$ represents the variational distribution associated with the optimal lower bound $\mathcal{L}^* = \hat{\mathcal{L}}(N_n + M_a)$, with $\mathcal{Z}$ denoting the marginal likelihood that normalises $q_n^*(f)$.

Proof. We cease the addition of points when $\hat{\mathcal{L}}(M_b) > \hat{\mathcal{U}}(M) - \alpha$. Given that $\hat{\mathcal{U}}(M) \geq \mathcal{L}^*$, and assuming $\theta_n = \theta_o$, the rest follows from algebraic manipulation of eq. (4). See App. C for the complete proof. $\square$

The first bound shows that if Ψ is near zero, the KL to the true posterior is bounded by α . While Ψ depends on the true posterior and therefore cannot be computed, if the posterior in the previous iteration was exact, Ψ would be equal to zero. The second bound shows that we are guaranteed to have our actual approximation $q_n(f)$ be within α nats of the best approximation that we can develop, given the limitations of the approximations made in previous iterations.

4.3. Selecting a Threshold

In this final step of our online learning method, we must specify a heuristic for selecting α that does not require knowing any data in advance, while also working in a uniform way across datasets with different properties. A constant value for α does not work well, since the scale of the LML depends strongly on properties such as dataset size, and observation noise. This means that a tolerance of 1 nat (Cover, 1999) may be appropriate for a small dataset, but not for a large one.

As a principle for selecting the threshold, we take loose inspiration from compression and MDL (Grünwald & Roos, 2019), which takes the view of the ELBO being proportional

to negative the code length that the model requires to encode the dataset. Intuitively, our desire to select an α such that our method captures a high proportion (e.g. 95%) of all the information in each batch, so that we can compress to within a small fraction of the optimal variational code. By itself, this is not well-defined since the code length depends on the quantisation tolerance for our continuous variables. To avoid this, we take an independent random noise code as our baseline, and select α to be within some small fraction of the optimal variational code, relative to the random noise code. We want to capture a high proportion of all the additional information that our model provides, relative to the noise model, i.e. we want our threshold to be:

$$\alpha = \delta(\mathcal{L}^* - \mathcal{L}_{noise}), \quad \mathcal{L}_{noise} = \sum_{n=1}^{N_n} \log \mathcal{N}(y_n; \hat{\mu}, \hat{\sigma}^2)$$

where $\hat{\mu}$ and $\hat{\sigma}^2$ are the average and variance of the observations for up to the current task and δ is a user-defined hyperparameter. We validate that this approach leads to values of δ giving consistent behaviour across a wide range of datasets, which allows it to be set in advance without needing much prior knowledge of the dataset characteristics.

Calculating this threshold is intractable for large batch sizes N_n . However, if we change our stopping criterion to the more stringent upper bound

$$\bar{\alpha} = \delta(\hat{\mathcal{U}}(M) - \mathcal{L}_{noise}). \quad (12)$$

and increase M for calculating $\hat{\mathcal{U}}$ as M_b is increased for calculating $\hat{\mathcal{L}}$, we obtain the same guarantees as before but at a lower computational cost. However, this strategy is only worthwhile for very large batch sizes N_n , due to the importance of constant factors in the computational cost. In the common continual learning settings we investigate N_n is small enough to allow computing $\mathcal{L}^*$.

The procedure for our inducing point selection method is detailed in Algorithm 1 and we give further details in App. B.1. We name our approach Vegas Inducing Point Selection (VIPS), drawing an analogy to Las Vegas Algorithms. These methods guarantee the accuracy of the output, however, their computational time fluctuates for every run (Motwani & Raghavan, 1995).

5. Experiments

We evaluate the performance of our adaptive inducing point selection method VIPS in a range of streaming scenarios where we assume the total number of observations is unknown. In all cases, the variational distribution, noise and kernel hyperparameters are optimised using the online lower bound (Eq. (8)). We provide further details in App. D.

Continual learning scenarios pose unique challenges. It is impossible to pre-determine memory allocation due to unknown input space coverage. Additionally, cross-validation for hyperparameter tuning is not feasible as it would require storing all data. Thus, an effective method must **1)** have an adaptive memory that can grow with the demands of the data, **2)** work with hyperparameters that can be set before training. The first part of our experiments demonstrates the benefits of using an adaptive model capacity in continual learning. The second part highlights the necessity of a single, pre-tunable hyperparameter that works across diverse settings. We benchmark VIPS against two other inducing point selection methods on streaming UCI datasets and a real-world robotics application.

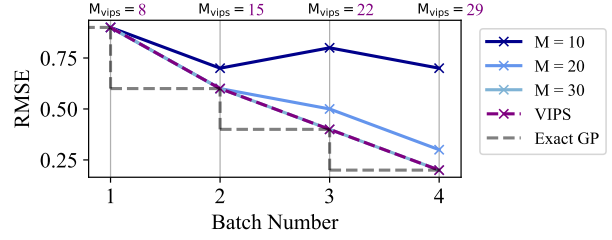
5.1. Model size and data distribution

Figure 1 shows VIPS’s ability to adapt across three datasets with different characteristics, each divided into ten batches, illustrating how input distribution drives model growth as more data is seen. In the first dataset, each batch introduces data from new parts of the input space. Since each batch is equally novel, the model size grows linearly. In the second, the data remain within a fixed interval, leading to diminishing novelty in each batch, and a model size that converges to a fixed value. The third dataset mixes narrow-range uniform samples with occasional batches sampled from a heavy-tailed Cauchy distribution. This leads the model size to converge, with sporadic growth when novel data is observed.

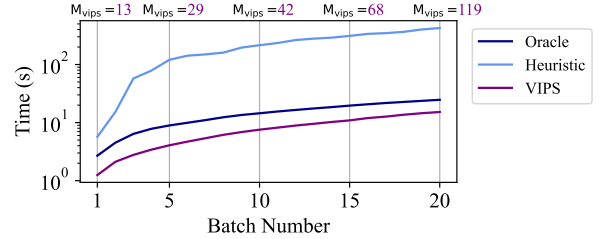
5.2. The impact of model capacity on accuracy and training time

Here, we compare a fixed to dynamic model size, with inducing points selected according to the greedy variance criterion (Burt et al., 2020), to demonstrate the challenges of a fixed model size. Since the total dataset size and data characteristics are not known at the start of training, selecting a fixed size will either lead to a model that is too small and underperforms, or a model that is too large and wastes computational resources. Adaptive size solves these issues.

Performance comparison: We train on a synthetic dataset divided into four batches, using a fixed model size, and our adaptive stopping criterion, VIPS. We test three fixed sizes: $M = 10, 20$, and 30 inducing points. We record



(a) Accuracy on a synthetic dataset.



(b) Training time comparison on “naval” dataset.

Figure 2. (a) Performance comparison of fixed memory approaches (blue curves with $M = 10, 20, 30$ inducing points) and VIPS, with M (shown at the top) inducing points at each batch. (b) (log) Time taken to train the online GP model on the “naval” dataset divided into 20 batches with fixed size (oracle: $M = 100$, heuristic: $M = 1000$) and VIPS, our adaptive method.

the test root mean square error (RMSE) and compare to an exact GP model with access to all current training data as a benchmark. Figure 2a shows that fixed size models $M = 10, 20$ lose accuracy with more data whereas fixed model $M = 30$ and VIPS match exact GP performance. At this point, we can select the model with $M = 30$ with the benefit of hindsight. However, at the start of training, we could not have known that this size would have been sufficient at the point of testing. VIPS not only automatically ends up with this optimal size, but also avoids computational waste by using fewer inducing points in earlier batches.

Training cost comparison: We use the “naval” UCI dataset divided into 20 batches, to compare the training time of models with a fixed and dynamic size. We compare a best-case fixed-size model with the smallest number of inducing points that would still give near-optimal performance. For “naval”, this is $M = 100$, which would have to be set by an “oracle”. We also include a typical heuristic, of $M = 1000$ (around 1/10th of the total data points), that would ensure sufficient capacity to handle different data patterns and complexities. Finally, we test our adaptive method, VIPS, which adjusts the number of inducing points as needed while achieving near-exact performance for the dataset. As shown in Figure 2b, VIPS results in lower computational time by only growing its model size as new data is seen, beating even the oracle method.

Table 1. Mean (std) of inducing points in final batch across training/test splits at operating point selected to achieve 10% RMSE threshold. Cross (X) indicates failure to meet accuracy threshold and "Max." denotes reaching maximum capacity limit. Values for OIPS are in italics, as the operating point was selected based on only five datasets, making it less stringent than other methods.

DATASET	CV	OIPS	VIPS (OURS)
CONCRETE	492(54)	240(102)	234(116)
SKILLCRAFT	739(34)	174(16)	134(9)
KIN8NM	6316(10)	6458(8)	1904(41)
NAVAL	49(1)	X	57(1)
ELEVATORS	3133(140)	298(8)	291(4)
BIKE	MAX. 7000	1964(137)	650(17)

Table 2. Mean (std) of inducing points in final batch across training/test splits at operating point selected to achieve 10% RMSE threshold in all but one dataset. X Gray shows cases exceeding threshold, with percentages showing % difference to full-batch GP relative to a noise model.

DATASET	CV	OIPS	VIPS (OURS)
CONCRETE	184(74)	240(102)	178(95)
SKILLCRAFT	226(22)	174(16)	133(0)
KIN8NM	5334(27)	6458(8)	1563(39)
NAVAL	X 23(8)-323%	X 16(1)-315.28%	48(3)
ELEVATORS	836(27)	298(8)	281(4)
BIKE	4895(28)	1964(137)	X 549(14)-10.8%

5.3. Continual learning of UCI datasets

We compare our method, VIPS, to two other inducing point selection approaches: Conditional Variance (CV) and OIPS (Galy-Fajou & Opper, 2021) (see details in App. B). We use six datasets from the UCI repository (Dua & Graff, 2017), simulating a continual learning scenario by sorting the data along the first dimension and dividing it into batches. When evaluating inducing point selection methods, we must consider the trade-off between accuracy and model size: while selecting more inducing points improves model accuracy, it also increases computational cost.

Our goal is to minimise computational cost, while satisfying the constraint that we reach a fixed target accuracy threshold (e.g. within 10% RMSE of a full-batch GP). To benchmark our methods, we select a single hyperparameter across datasets, so that our accuracy constraint is always met, while otherwise using the fewest inducing points. We refer to this hyperparameter as the operating point. For this operating point, we then measure the number of inducing points that the method chooses for each dataset, with a lower result being better. If across datasets, there is a large variation in hyperparameter values needed to achieve the accuracy constraint, selecting the most stringent criterion will

Table 3. Mean (std) of inducing points in final batch across training/test splits at operating point selected to achieve 10% RMSE threshold. Cross (X) indicates failure to meet accuracy threshold and "Max." denotes reaching maximum capacity limit. Values for OIPS are in italics, as the operating point was selected based on only five datasets, making it less stringent than other methods.

DATASET	CV	OIPS	VIPS (OURS)
CONCRETE	492(54)	383(90)	451(77)
SKILLCRAFT	1091(107)	236(38)	195(18)
KIN8NM	6469(7)	6523(9)	5065(46)
NAVAL	903(8)	X	509(15)
ELEVATORS	3133(140)	386(47)	487(24)
BIKE	MAX. 7000	3916(122)	2794(148)

Table 4. Mean (std) of inducing points in final batch across training/test splits at operating point selected to achieve 10% NLPD threshold in all but one dataset. X Gray shows cases exceeding threshold, with percentages showing % difference to full-batch GP relative to a noise model.

DATASET	CV	OIPS	VIPS (OURS)
CONCRETE	303(78)	383(90)	434(79)
SKILLCRAFT	295(13)	236(38)	177(13)
KIN8NM	5685(26)	6523(9)	4281(69)
NAVAL	X 27(6)-2406%	X 20(1)-133.87%	X 373(19)-11.92%
ELEVATORS	1099(50)	386(47)	438(17)
BIKE	5563(36)	3916(122)	2296(138)

cause large numbers of inducing points to be used for other datasets. This simulates a realistic setting, where one cannot know what the optimal hyperparameter value is for each dataset beforehand, and penalises methods that are hard to tune, due to large variations in the required hyperparameter values across datasets.

Table 1 shows the number of inducing points for each method at the operating point that achieves the accuracy threshold within 10% RMSE of a full-batch GP. VIPS requires fewer inducing points in the majority of datasets. CV often results in larger models, reaching the 7000-point limit on Bike. OIPS performs inconsistently, matching VIPS on smaller datasets but oversizing on larger ones. For the Naval dataset, OIPS failed to meet the accuracy constraint across all tested hyperparameter values. While extending the hyperparameter range might have improved accuracy, it would have led to even larger models. Therefore, we selected an operating point that satisfied our conditions for five out of six datasets, treating Naval as an outlier. Table 2 shows the results when this exception is applied to all methods. We observe a similar trend, with VIPS achieving the smallest model sizes. Tables 3 and 4 report analogous results for the threshold within 10% NLPD of a full-batch GP. The same pattern holds. Figure 3 summarises these outcomes across

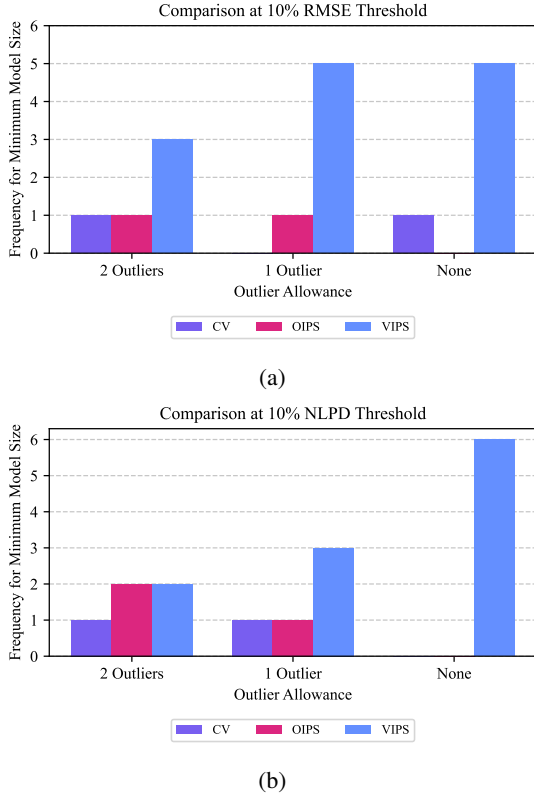


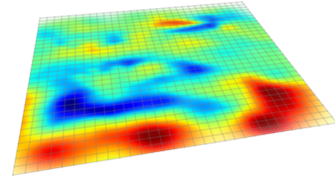
Figure 3. Number of datasets where each method achieves minimal model size at different outlier allowances. A “win” is assigned to the dataset with the smallest model which satisfies the (a) RMSE and (b) NLPD thresholds of 10%. Counts represent absolute wins. Higher counts indicate better method robustness across datasets.

outlier allowances, with VIPS more frequently achieving minimal model size. App. D.3 provides additional results under different thresholds.

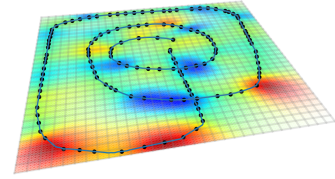
Our results show that VIPS performs consistently across datasets, allowing a single pre-set hyperparameter to be used. In contrast, OIPS and CV require tuning, as no single hyperparameter works well for all datasets. This is a limitation in continual learning, where the whole dataset is unavailable in advance.

5.4. Continual Learning of Magnetic Field

We apply the optimal hyperparameters found in the previous section, selected under the 10% RMSE threshold, to the experimental setup from Chang et al. (2023a) using real-world data from Solin et al. (2018), where a robot maps magnetic field anomalies in a 6m x 6m indoor space. This scenario simulates an expanding domain, where the model needs to continuously learn new areas without uncontrollably increasing its capacity. Our goal is to test whether the methods can form representations by progressively spreading inducing



(a) Learning paths 1, 2, 4, 5 sequentially with VIPS.



(b) Learning path 3 with VIPS. M: 134, RMSE: 7.55.

Figure 4. A small robot is used to perform sequential estimation of magnetic field anomalies. The strength of the magnetic field is given by $10 \mu\text{T}$ to $90 \mu\text{T}$. (a) shows the final estimate of the magnitude field learned sequentially through different paths. (b) show the outcome of learning a single path continuously with the black dots representing the chosen inducing points.

points and learning hyperparameters without prior tuning in two scenarios: 1) sequentially mapping multiple robot trajectories and 2) processing a single path in continuous batches. Figure 4 shows the final magnetic field estimates produced by VIPS in both experimental settings. Comparative results for CV and OIPS are provided in App. D.4. The results show that VIPS was able to grow its memory as the robot moved, allowing for accurate modelling of the path and increasing its capacity dynamically. In contrast, we observe that CV increases its capacity uncontrollably while OIPS provides a less accurate estimate of the magnetic field due to choosing fewer inducing points.

6. Discussion

In this work, we propose a method to dynamically adjust the number of inducing variables in streaming GP regression. Our method achieves a performance close to full-batch approaches while minimising model size. It relies on a single hyperparameter to balance accuracy and complexity, and we demonstrate that our method performs consistently across diverse datasets. This reduces the need for extensive hyperparameter tuning and eliminates the requirement to pre-define model size, thereby addressing a significant bottleneck in traditional methods. While our current application focuses on GPs we hope this is extendable for the development of adaptive neural networks.

Acknowledgements

GPB is supported by EPSRC through the Statistical Machine Learning (StatML) CDT programme, grant no. EP/S023151/1. GPB acknowledges the support of G-Research for her participation in the conference.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283. USENIX Association, 2016.
- Bauer, M., van der Wilk, M., and Rasmussen, C. E. Understanding probabilistic sparse Gaussian process approximations. *Advances in neural information processing systems*, 29, 2016.
- Bui, T. D., Nguyen, C., and Turner, R. E. Streaming sparse Gaussian process approximations. *Advances in Neural Information Processing Systems*, 30:3299–3307, 2017. ISSN 1049-5258.
- Burt, D., Rasmussen, C. E., and Van Der Wilk, M. Rates of convergence for sparse variational Gaussian process regression. In *International Conference on Machine Learning*, pp. 862–871. PMLR, 2019.
- Burt, D. R., Rasmussen, C. E., and van der Wilk, M. Convergence of sparse variational inference in Gaussian processes regression. *The Journal of Machine Learning Research*, 21(1):5120–5182, 2020.
- Chang, P. E., Verma, P., John, S., Solin, A., and Khan, M. E. Memory-Based dual Gaussian processes for sequential learning. In *International Conference on Machine Learning*, pp. 4035–4054. PMLR, June 2023a.
- Chang, P. G., Durán-Martín, G., Shestopaloff, A., Jones, M., and Murphy, K. P. Low-rank extended kalman filtering for online learning of neural networks from streaming data. pp. 1025–1071, 2023b.
- Cho, Y. and Saul, L. Kernel methods for deep learning. *Advances in neural information processing systems*, 22, 2009.
- Cover, T. M. *Elements of information theory*, pp. 14. John Wiley & Sons, 1999.
- Dua, D. and Graff, C. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- Dutordoir, V., Durrande, N., and Hensman, J. Sparse Gaussian processes with spherical harmonic features. In *International Conference on Machine Learning*, pp. 2793–2802. PMLR, 2020.
- Dutordoir, V., Hensman, J., van der Wilk, M., Ek, C. H., Ghahramani, Z., and Durrande, N. Deep neural networks as point estimates for deep Gaussian processes. *Advances in Neural Information Processing Systems*, 34: 9443–9455, 2021.
- Farquhar, S. and Gal, Y. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.
- Fine, S. and Scheinberg, K. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec):243–264, 2001.
- Foster, L., Waagen, A., Aijaz, N., Hurley, M., Luis, A., Rinsky, J., Satyavolu, C., Way, M. J., Gazis, P., and Srivastava, A. Stable and efficient Gaussian process calculations. *Journal of Machine Learning Research*, 10(4), 2009.
- Galy-Fajou, T. and Opper, M. Adaptive inducing points selection for Gaussian Processes. In *Continual Learning Workshop*, July 2021.
- Ghahramani, Z. Bayesian non-parametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110553, 2013.
- Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.
- Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.
- Grünwald, P. and Roos, T. Minimum description length revisited. *International journal of mathematics for industry*, 11(01):1930001, 2019.
- Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 282–290, 2013.

- Kapoor, S., Karaletsos, T., and Bui, T. D. Variational autoregressive Gaussian processes for continual learning. In *International Conference on Machine Learning*, pp. 5290–5300. PMLR, 2021.
- Kessler, S., Nguyen, V., Zohren, S., and Roberts, S. J. Hierarchical indian buffet neural networks for bayesian continual learning. In *Uncertainty in artificial intelligence*, pp. 749–759. PMLR, 2021.
- Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- Li, Z. and Hoiem, D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.
- Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.
- Maddox, W. J., Stanton, S., and Wilson, A. G. Conditioning sparse variational Gaussian processes for online decision-making. *Advances in Neural Information Processing Systems*, 34:6365–6379, 2021.
- Maile, K., Rachelson, E., Luga, H., and Wilson, D. G. When, where, and how to add new neurons to anns. In *International Conference on Automated Machine Learning*, pp. 18–1. PMLR, 2022.
- Matthews, A. G. d. G. *Scalable Gaussian process inference using variational methods*. PhD thesis, 2017.
- Matthews, A. G. d. G., Hensman, J., Turner, R., and Ghahramani, Z. On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In *Artificial Intelligence and Statistics*, pp. 231–239. PMLR, 2016.
- Matthews, A. G. d. G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrà, P., Ghahramani, Z., and Hensman, J. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, apr 2017.
- Matthews, A. G. d. G., Hron, J., Rowland, M., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.
- McCloskey, M. and Cohen, N. J. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.
- Mercer, J. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446, 1909.
- Mitchell, R., Mundt, M., and Kersting, K. Self expanding neural networks. *arXiv preprint arXiv:2307.04526*, 2023.
- Motwani, R. and Raghavan, P. *Randomized Algorithms*. Cambridge University Press, 1995.
- Murphy, K. P. *Probabilistic Machine Learning: Advanced Topics*, chapter 19.7. MIT Press, 2023.
- Neal, R. M. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 1996.
- Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational continual learning. In *International Conference on Learning Representations*, October 2018.
- Panos, A., Dellaportas, P., and Titsias, M. K. Fully scalable Gaussian processes using subspace inducing inputs. *arXiv preprint arXiv:1807.02537*, 2018.
- Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, November 2005. ISBN 9780262182539.
- Ring, M. B. Child: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.
- Rudner, T. G., Smith, F. B., Feng, Q., Teh, Y. W., and Gal, Y. Continual learning via sequential function-space variational inference. In *International Conference on Machine Learning*, pp. 18871–18887. PMLR, 2022.
- Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.
- Schwarz, J., Czarnecki, W., Luketina, J., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pp. 4528–4537. PMLR, 2018.
- Solin, A., Kok, M., Wahlstrom, N., Schon, T., and Sarkka, S. Modeling and interpolation of the ambient magnetic field by Gaussian processes. *IEEE Transactions on Robotics*, 34(4):1112–1127, 2018. ISSN 1552-3098. doi: 10.1109/TRO.2018.2830326.
- Sun, S., Shi, J., and Grosse, R. B. Neural networks as inter-domain inducing points. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2020.

- Titsias, M. K. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5 of *Proceedings of Machine Learning Research*, pp. 567–574. PMLR, 2009.
- Titsias, M. K. Variational inference for Gaussian and determinantal point processes. In *Workshop on Advances in Variational Inference (NIPS)*, 2014.
- van der Vaart, A. and van Zanten, J. Rates of contraction of posterior distributions based on Gaussian process priors. *The Annals of Statistics*, 36(3):1435–1463, 2008.
- van der Wilk, M. *Sparse Gaussian process approximations and applications*. PhD thesis, 2019.
- Yoon, J., Yang, E., Lee, J., and Hwang, S. J. Lifelong learning with dynamically expandable networks. In *6th International Conference on Learning Representations, ICLR 2018*. International Conference on Learning Representations, ICLR, 2018.

A. Code

The methods discussed in this work, along with the code to reproduce our results, are available online at <https://github.com/guionarpescador/vips>. All experiments and figures were done in Python using libraries TensorFlow v.2.12 (Abadi et al., 2016) and GPflow v.2.9 (Matthews et al., 2017). Both packages are under Apache License 2.0. The implementation of the online lower bound (see eq. (8)) by Bui et al. (2017) is available on GitHub under Apache license 2.0. For the “greedy variance” selection criterion and algorithm, we use the implementation provided in Burt et al. (2020) available under Apache license 2.0.

B. Adaptive Inducing Points Selection Methods

In sections 5.3 and 5.4 we compare our method, VIPS, to two other adaptive approaches: Conditional Variance (CV) and OIPS (Galy-Fajou & Oppel, 2021). The key distinction between these approaches lies in their selection strategy of inducing points. OIPS uses a unified selection process (Algorithm 2), where the choice of both location and number of inducing points is determined by a single metric. In particular, each point $\mathbf{x}_i$ in the training set $\mathbf{X}_n$ is evaluated individually, and those that meet the threshold condition are added to the inducing set $\mathbf{Z}_n$. In contrast, both CV and VIPS use a decoupled selection process (Algorithm 3), which separates two decisions: where to place inducing points and how many to include. These methods iteratively select the best candidate for the next inducing point from a pool of potential locations, continuing until a stopping criterion is met, allowing for more control over the model size. VIPS and CV share the same greedy variance selection criterion but differ in their candidate pools and stopping criteria. Table 5 summarises the properties of these methods, including the fixed memory approach of Bui et al. (2017) for comparison. The next three subsections provide detailed descriptions of each method’s specific implementation.

Algorithm 2 Unified inducing set selection process done at each batch.

Input: New batch of data $\mathbf{X}_n$, current set inducing points $\mathbf{Z}_{old}$.

Output: Updated set $\mathbf{Z}_n$

Initialise $\mathbf{Z}_n = \mathbf{Z}_{old}$

for $\mathbf{x}_i \in \mathbf{X}_n$ **do**

if addition_criterion($\mathbf{x}_i$) < threshold **then**

$\mathbf{Z}_n = \mathbf{Z}_n \cup \{\mathbf{x}_i\}$

end if

end for

Algorithm 3 Decoupled inducing set selection process done at each batch.

Input: New batch of data $\mathbf{X}_n$, current set inducing points $\mathbf{Z}_{old}$.

Output: Updated set $\mathbf{Z}_n$

Set candidate pool $\mathbf{X}_{pool}$ based on method (see Table 5) and initialise $\mathbf{Z}_n$ accordingly.

while stopping_criterion($\mathbf{Z}_n$) > threshold **do**

$\mathbf{x}_{best} = \text{selection_criterion}(\mathbf{X}_{pool}, \mathbf{Z})$ {Greedy variance criterion}

$\mathbf{Z}_n = \mathbf{Z}_n \cup \{\mathbf{x}_{best}\}$

end while

Table 5. Properties of inducing points selection method for updating an online GP regression model.

METHOD	MODEL SIZE	SELECTION POOL	SELECTION CRITERION	STOPPING CRITERION
BUI ET AL. (2017)	FIXED	$\{\mathbf{Z}_o, \mathbf{X}_n\}$	GRADIENT OPTIMISATION	M CONSTANT
OIPS (GALY-FAJOU & OPPEL, 2021)	ADAPTIVE	$\{\mathbf{X}_n\}$	$\max \mathbf{k}_u(\mathbf{x}) \leq \rho$	$\max \mathbf{k}_u(\mathbf{x}) \leq \bar{\rho}$
COND. VARIANCE (CV)	ADAPTIVE	$\{\mathbf{Z}_o, \mathbf{X}_n\}$	GREEDY VARIANCE	$\text{tr}(\mathbf{K}_{ff} - \mathbf{Q}_{ff}) \leq \eta$
VIPS	ADAPTIVE	$\{\mathbf{X}_n\}$	GREEDY VARIANCE	$\hat{\mathcal{U}} - \hat{\mathcal{L}} \leq \alpha$

B.1. Vegas Inducing Point Selection (VIPS) Algorithm

To select the location of our new inducing points we use the location selection strategy “greedy variance” proposed in [Burt et al. \(2020\)](#). This strategy iteratively selects points from a set based on a preference criterion until a stopping condition is met. In particular, it chooses the location of the next inducing point to maximise the marginal variance in the conditional prior $p(f_{\mathbf{u}}|\mathbf{u})$. This is equivalent to maximising $\text{diag}[\mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}]$. In continual learning, [Chang et al. \(2023a\)](#) use the “greedy variance” criterion by defining $\{\mathbf{Z}_o, \mathbf{X}_n\}$ as the selection pool from which inducing point locations are selected and maintaining a fixed number of inducing points. Similarly, [Maddox et al. \(2021\)](#) extends the “greedy variance” criterion to heteroskedastic Gaussian likelihoods and also uses a fixed memory approach. In our case, we tested the location strategy with our stopping criterion using both $\{\mathbf{Z}_o, \mathbf{X}_n\}$ and $\{\mathbf{X}_n\}$ as candidates pool for the locations of the inducing points. We did not find a substantial difference between the methods and hence opted for the simpler version where we keep the old inducing point locations fixed and choose the new set of inducing points from among the locations in $\{\mathbf{X}_n\}$.

Algorithm 4 presents an inducing point selection method using our stopping criterion combined with the location selection strategy “greedy variance”. The method takes as input a value for the hyperparameter θ_n . In practice, we will set $\theta_n = \theta_o$ to select the number of inducing points; the hyperparameter θ_n is inferred by optimising $\hat{\mathcal{L}}(M_b)$ once the inducing point locations $\mathbf{Z}_n$ has been chosen. In Algorithm 1, $\hat{\mathcal{U}}(M)$ is used to calculate the stopping criterion. However, in practice, since for the continual learning settings we investigate N_n is small enough, we will use $\hat{\mathcal{U}}(M_a + N_n) = \mathcal{L}^*$. This value is calculated once at the beginning of the process. The algorithm’s complexity depends on the number of inducing points M_b used to compute the lower bound $\hat{\mathcal{L}}(M_b)$ at each iteration. The computational complexity for calculating $\hat{\mathcal{L}}$ at each batch is $O(N_n M_b^2 + M_b^3)$, and the memory requirement is $O(M_b^2)$, where M_b represents the total number of inducing points in the current batch.

Algorithm 4 Vegas Inducing Point Selection (VIPS)

Input: $\mathbf{X}_n = \{\mathbf{x}_i\}_{i=1}^{N_n}$, $\mathbf{Z}_o = \{\mathbf{z}_m\}_{m=1}^{M_a}$, $\hat{\mu}$, $\hat{\sigma}$, θ_n , kernel $k(\cdot, \cdot | \theta_n)$, threshold parameter δ .
Output: Updated set $\mathbf{Z}_n = \mathbf{Z}_o \cup \{\mathbf{x}_{m'}\}_{m'=1}^{M'}$, where $|\mathbf{Z}_n| = M_b$.
 Initialise $\mathbf{Z}_n = \mathbf{Z}_o$.
while $\hat{\mathcal{U}}(M) - \hat{\mathcal{L}}(M_b) \leq \delta |\hat{\mathcal{U}}(M) - \mathcal{L}_{noise}(\hat{\mu}, \hat{\sigma})|$ **do**
 Select $\mathbf{x} = \arg\max_{\mathbf{x} \in \mathbf{X}_n} k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_b(\mathbf{x})^\top \mathbf{K}_{bb}^{-1} \mathbf{k}_b(\mathbf{x})$.
 Add $\mathbf{x}$ to the set of inducing points: $\mathbf{Z}_n = \mathbf{Z}_n \cup \{\mathbf{x}\}$.
end while

B.2. Conditional Variance

The implementation of the Conditional Variance method is presented in Algorithm 5. This method uses the “greedy variance” strategy that iteratively chooses the location of the next inducing point. As a stopping criterion, it uses the quantity $\text{tr}(\mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}})$. In this algorithm, new inducing points are no longer added once $\text{tr}(\mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}})$ falls below a chosen tolerance value η . Although, this approach was mentioned in [Burt et al. \(2020\)](#), this stopping criterion has not yet been tested in the literature. The hyperparameter η is determined by the user.

Algorithm 5 Conditional Variance (CV)

Input: $\mathbf{X}_n = \{\mathbf{x}_i\}_{i=1}^{N_n}$, $\mathbf{Z}_o = \{\mathbf{z}_m\}_{m=1}^{M_a}$, θ_n , kernel, $k(\cdot, \cdot | \theta_n)$, threshold η .
Output: Updated set of inducing points $\mathbf{Z}_n = \{\mathbf{z}_m\}_{m=1}^{M'} \cup \{\mathbf{x}_{m'}\}_{m'=1}^{M_b - M'}$, where $|\mathbf{Z}_n| = M_b$.
 Initialise location selection pool: $\mathbf{X}_{pool} = \mathbf{Z}_o \cup \mathbf{X}_n$.
 Initialise $\mathbf{Z}_n = \arg\max_{\mathbf{x} \in \mathbf{X}_{pool}} k(\mathbf{x}, \mathbf{x})$.
while $\text{tr}(\mathbf{K}_{\mathbf{ff}} - \mathbf{Q}_{\mathbf{ff}}) \leq \eta$ **do**
 Select $\mathbf{x} = \arg\max_{\mathbf{x} \in \mathbf{X}_{pool}} k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_b(\mathbf{x})^\top \mathbf{K}_{bb}^{-1} \mathbf{k}_b(\mathbf{x})$.
 Add $\mathbf{x}$ to the set of inducing points: $\mathbf{Z}_n = \mathbf{Z}_n \cup \{\mathbf{x}\}$.
end while

B.3. Online Inducing Point Selection (OIPS)

Galy-Fajou & Opper (2021) introduced the Online Inducing Points Selection (OIPS) algorithm, which iteratively adds points from $\mathbf{X}_n$ to the set of inducing points. The algorithm assesses the impact of each new point on the existing inducing set, based on a covariance threshold. A point $\mathbf{x}$ is added if the maximum value of $\mathbf{k}_u(\mathbf{x})$ falls below a user-defined threshold ρ . While the original algorithm implicitly assumes unit kernel variance, Algorithm 6 presents our implementation where we scale the threshold ρ by the kernel variance σ_f^2 to $\bar{\rho} = \sigma_f^2 \rho$. This modification ensures the threshold comparison remains valid for arbitrary kernel variances.

Algorithm 6 Online Inducing Point Selection (OIPS)

Input: $\mathbf{X}_n = \{\mathbf{x}_i\}_{i=1}^{N_n}$, $\mathbf{Z}_o = \{\mathbf{z}_m\}_{m=1}^{M_a}$, kernel function $k(\cdot, \cdot | \theta_n)$, kernel hyperparameters θ_n (including variance σ_f^2), acceptance threshold $0 < \rho < 1$.
Output: Updated set $\mathbf{Z}_n = \mathbf{Z}_o \cup \{\mathbf{x}_{m'}\}_{m'=1}^{M'}$, where $|\mathbf{Z}_n| = M_b$.
 Initialise $\mathbf{Z}_n = \mathbf{Z}_o$.
 Initialise $\bar{\rho} = \rho \cdot \sigma_f^2$.
for all $\mathbf{x}_i \in \mathbf{X}_n$ **do**
 $d = \max_j (k(\mathbf{x}_i, \mathbf{z}_j | \theta_n))$, $\forall \mathbf{z}_j \in \mathbf{Z}_n$.
 if $d < \bar{\rho}$ **then**
 Add $\mathbf{x}_i$ to the set of inducing points: $\mathbf{Z}_n = \mathbf{Z}_n \cup \{\mathbf{x}_i\}$.
 end if
end for

C. Proof of Guarantee

Guarantee. Let M be a fixed integer and $M_{\mathbf{b}}$ be the number of selected inducing points such that $\widehat{\mathcal{U}}(M) - \widehat{\mathcal{L}}(M_{\mathbf{b}}) \leq \alpha$. Assuming that $\theta_n = \theta_o$, we have two equivalent bounds:

$$\text{KL}[q_n(f) \parallel p(f|\mathbf{y}_o, \mathbf{y}_n, \theta_o)] \leq \alpha + \Psi \quad (13)$$

$$\text{KL}[q_n(f) \parallel q_n^*(f)] \leq \alpha \quad (14)$$

where $\Psi = \int q_n(f) \log \frac{q_n^*(f)}{p(f|\mathbf{y}_n, \mathbf{y}_o)} df$ and $q_n^*(f) = \mathcal{Z}^{-1} q_o(f) p(\mathbf{y}_n|f)$ represents the variational distribution associated with the optimal lower bound $\mathcal{L}^* = \widehat{\mathcal{L}}(N_n + M_{\mathbf{a}})$, with $\mathcal{Z}$ denoting the marginal likelihood that normalises $q_n^*(f)$.

Proof. We cease the addition of points when $\widehat{\mathcal{U}}(M) - \widehat{\mathcal{L}}(M_{\mathbf{b}}) < \alpha$. Since $\widehat{\mathcal{U}}(M) \geq \mathcal{L}^*$, then $-\widehat{\mathcal{L}}(M_{\mathbf{b}}) < \alpha - \widehat{\mathcal{U}}(M) < \alpha - \mathcal{L}^*$. Eq.(4) can be bounded as:

$$\begin{aligned} \text{KL}[q_n(f) \parallel p(f|\mathbf{y}_o, \mathbf{y}_n, \theta_o)] &= \log \frac{p(\mathbf{y}_n, \mathbf{y}_o|\theta_o)}{p(\mathbf{y}_o|\theta_o)} - \widehat{\mathcal{L}} + \Phi \\ &\leq \log \frac{p(\mathbf{y}_n, \mathbf{y}_o|\theta_o)}{p(\mathbf{y}_o|\theta_o)} + \alpha - \mathcal{L}^* + \Phi \end{aligned} \quad (15)$$

where $\Phi = -\int q_n(f) \log \frac{p(f|\mathbf{y}_o, \theta_o)}{q_o(f)} df$. Let $q_n^*(f) = \mathcal{Z}^{-1} q_o(f) p(\mathbf{y}_n|f)$ be the variational distribution associated with $\mathcal{L}^* = \widehat{\mathcal{L}}(N_n + M_{\mathbf{a}})$. Then, by expanding the true posterior and multiplying by the variational distributions $q_n^*(f)$ on both sides of the fraction inside the log, we obtain:

$$\begin{aligned} \Phi &= \int q_n(f) \log \frac{q_o(f)}{p(f|\mathbf{y}_o, \theta_o)} df \\ &= \int q_n(f) \log \frac{q_o(f) p(\mathbf{y}_o|\theta_o)}{p(\mathbf{y}_o|f) p(f|\theta_o)} df \\ &= \int q_n(f) \log \frac{q_o(f) p(\mathbf{y}_o|\theta_o)}{p(\mathbf{y}_o|f) p(f|\theta_o)} \frac{q_n^*(f)}{q_n^*(f)} df \\ &= \int q_n(f) \log \frac{q_o(f) p(\mathbf{y}_o|\theta_o)}{p(\mathbf{y}_o|f) p(f|\theta_o)} \frac{q_n^*(f)}{\mathcal{Z}^{-1} q_o(f) p(\mathbf{y}_n|f)} df \\ &= \int q_n(f) \log \frac{q_n^*(f)}{p(f|\mathbf{y}_n, \mathbf{y}_o, \theta_o)} df + \log \mathcal{Z} - \log \frac{p(\mathbf{y}_n, \mathbf{y}_o|\theta_o)}{p(\mathbf{y}_o|\theta_o)} \end{aligned} \quad (16)$$

Using the above expansion for Φ , eq. (15) becomes,

$$\begin{aligned} \text{KL}[q_n(f) \parallel p(f|\mathbf{y}_o, \mathbf{y}_n, \theta_o)] &\leq \log \frac{p(\mathbf{y}_n, \mathbf{y}_o|\theta_o)}{p(\mathbf{y}_o|\theta_o)} + \alpha - \mathcal{L}^* + \Phi \\ &\leq \log \frac{p(\mathbf{y}_n, \mathbf{y}_o|\theta_o)}{p(\mathbf{y}_o|\theta_o)} + \alpha - \mathcal{L}^* + \int q_n(f) \log \frac{q_n^*(f)}{p(f|\mathbf{y}_n, \mathbf{y}_o, \theta_o)} df + \log \mathcal{Z} - \log \frac{p(\mathbf{y}_n, \mathbf{y}_o|\theta_o)}{p(\mathbf{y}_o|\theta_o)}. \end{aligned} \quad (17)$$

Assuming that $\theta_n = \theta_o$, the above can be simplified to

$$\text{KL}[q_n(f) \parallel p(f|\mathbf{y}_o, \mathbf{y}_n, \theta_o)] \leq \alpha + \int q_n(f) \log \frac{q_n^*(f)}{p(f|\mathbf{y}_n, \mathbf{y}_o, \theta_o)} df \quad (18)$$

Again by multiplying by $q_n(f)$ both sides of the fraction inside the log, we obtain:

$$\begin{aligned} \text{KL}[q_n(f) \parallel p(f|\mathbf{y}_o, \mathbf{y}_n, \theta_o)] &\leq \int q_n(f) \log \frac{q_n^*(f)}{p(f|\mathbf{y}_n, \mathbf{y}_o, \theta_o)} \frac{q_n(f)}{q_n(f)} df + \alpha \\ \text{KL}[q_n(f) \parallel p(f|\mathbf{y}_o, \mathbf{y}_n, \theta_o)] &\leq \text{KL}[q_n(f) \parallel p(f|\mathbf{y}_o, \mathbf{y}_n, \theta_o)] - \text{KL}[q_n(f) \parallel q_n^*(f)] + \alpha \\ \text{KL}[q_n(f) \parallel q_n^*(f)] &\leq \alpha. \end{aligned} \quad (19)$$

□

D. Further Experimental Details and Results

For all experiments and methods, we use the L-BFGS optimiser.

D.1. Model size and data distribution

For the synthetic dataset, we generate random noisy observations from the test function $f(x) = \sin(2x) + \cos(5x)$. We used a Squared Exponential kernel initialised with lengthscale 0.5 and variance 1. The noise variance was initialised to 0.5. For VIPS, we use $\delta = 0.05$.

Dataset 1: We use $N = 500$ observations uniformly distributed from 0 to 10. The data is ordered and divided into ten batches.

Dataset 2: We simulate a scenario where small batches of data are received but the data is distributed across the input space. We use $N = 150$ observations uniformly distributed from 0 to 10. The data is shuffled and divided into ten batches.

Dataset 3: We simulate a scenario where only outliers are encountered from time to time and the rest of the data is concentrated around a small part of the input space. We use two sets of data: the first set is sampled from a uniform distribution from 4 to 6, with $N = 1000$ and the second set is sampled from a Cauchy distribution with a mean of $\mu = 5$, with $N = 300$. The data is divided into ten batches, where the first few batches only contain observations from the 4 to 6 range, and the Cauchy observations are observed in the latter batches.

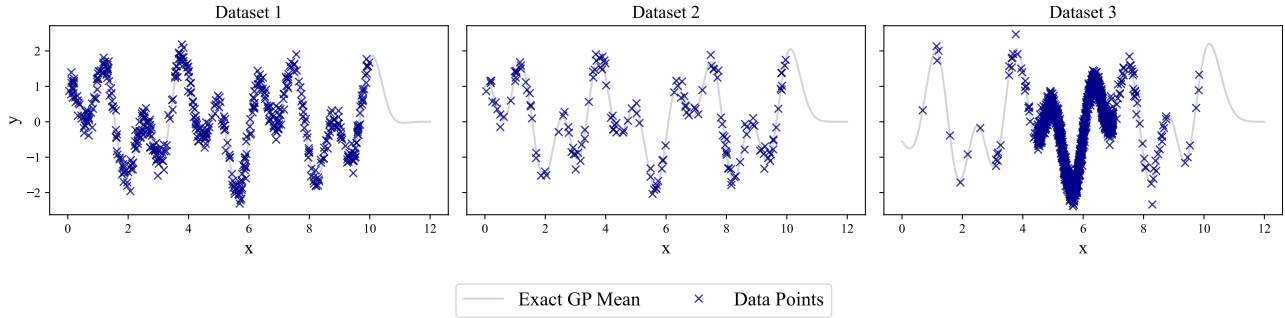


Figure 5. Plot of the three datasets considered in Section D.1.

D.2. The impact of model capacity in accuracy and training time

D.2.1. ACCURACY COMPARISON

For the synthetic dataset, we generate 1000 random noisy observations from the test function $f(x) = \sin(2x) + \cos(5x)$. We used a Squared Exponential kernel initialised with lengthscale 0.5 and variance 1.0. The noise variance was initialised to 0.5. The performance was measured on a test grid of 500 points. For VIPS, we use $\delta = 0.05$.

D.2.2. TRAINING COST COMPARISON

This experiment was performed on an Nvidia RTX 6000's GPU on a high-performance computing cluster. We used a Squared Exponential kernel with hyperparameters initialised to 1. The noise variance was initialised to 0.1. The dataset was divided into 20 batches, and we recorded the time in training per batch. For VIPS, we use $\delta = 0.05$.

D.3. UCI datasets

These experiments were performed on an Nvidia RTX 6000's GPU on a high-performance computing cluster. We used a Squared Exponential kernel with hyperparameters initialised to 1 for all datasets. The noise variance was initialised to 0.1. We consider six UCI (Dua & Graff, 2017) datasets of different characteristics: Concrete (1030, 8), Skillcraft (3338, 19), Kin8nm (8192, 8), Naval (11934, 14), Elevators (16599, 18), and Bike (17379, 17). The smaller datasets (< 12000) were

divided into 20 batches, and the larger (> 12000) into 50 batches. The batches were created by sorting the data points by the first dimension.

Increasing the model size enhances performance until all relevant dataset information is captured; beyond this point, only computational costs increase. Therefore, when comparing inducing point methods, it is essential to consider the size-performance trade-off, rather than focusing solely on performance gains. This trade-off is typically controlled by a hyperparameter of the model. However, in continual learning, traditional cross-validation for tuning is not practical since it would require storing all past data. As a result, a good method needs to use pre-set hyperparameters and still perform well on different datasets. We compare our method, VIPS, with two other adaptive approaches: Conditional Variance (CV) and OIPS. Our goal is to determine if a single hyperparameter for each method can perform consistently well across different datasets. To do this, we evaluate various hyperparameter settings for each method, with ranges $\delta \in [0.005, 0.2]$ for VIPS, $\delta \in [0.005, 1.0]$ for CV, $\rho \in [0.80, 0.999]$ for OIPS.

For each method, we identified hyperparameter values that achieved a particular accuracy threshold (either RMSE or NLPD) within a percentage of 5% or 10% of the full-batch GP relative to a noise model. For example, for 5% RMSE, the accuracy threshold is calculated as: $\text{RMSE}_{\text{exact}} + 0.05 |\text{RMSE}_{\text{noise}} - \text{RMSE}_{\text{exact}}|$. Among the hyperparameter values meeting these criteria, we select the one that results in the smallest model size. For CV and VIPS, this corresponds to the largest hyperparameter value, while for OIPS, it corresponds to the smallest hyperparameter value (see Table 5 for a summary of the methods).

With the optimal hyperparameter selected, all methods achieve the desired performance threshold. Therefore, we can compare the methods based on their model size, where smaller sizes are preferred. As explained in the main part of the paper, we did not find a hyperparameter that satisfied the accuracy thresholds for OIPS in the Naval dataset. For this method, we selected the hyperparameter configuration that satisfied our conditions for five out of six datasets, treating Naval as an outlier. This decision was made considering that OIPS already selected larger models with this less stringent hyperparameter (for example 80% of the data points for the Kin8nm dataset) and extending the hyperparameter range would have led to even larger dataset sizes.

Since such dataset-specific limitations are common in practical applications, and to ensure a fair comparison across methods, we evaluated configurations that allowed for either one or two datasets to be outliers for each method. To report performance, we calculated the difference with the full-batch GP metric relative to a noise model. For each dataset and batch, we computed the relative RMSE percentage as

$$\text{RMSE \%} = \left(\frac{\text{RMSE}_{\text{method}} - \text{RMSE}_{\text{exact}}}{|\text{RMSE}_{\text{noise}} - \text{RMSE}_{\text{exact}}|} \right) \times 100$$

and similarly for the NLPD metric. The test set for each batch consists only of data from the current and previous batches.

Figure 6 shows a summary of the results, presenting the absolute counts for smallest model size. Tables 6, 7, 8 and 9 provide details for each threshold and outlier choice. The results reveal consistent patterns across different accuracy thresholds and metrics. VIPS achieves the smallest model size while maintaining performance within thresholds, particularly when all datasets are considered.

Allowing outlier datasets reduces model sizes across all methods. This comes with performance degradation on the particular outlier. We observe that this degradation is often higher for CV and OIPS. The Naval dataset proved particularly challenging for these methods. This is probably due to its almost noiseless nature. While VIPS came close (missing the threshold by 0.48%), none of the methods achieved the 5% target NLPD threshold within the tested hyperparameter range. Given the dataset’s near-zero noise characteristics, we chose not to expand the hyperparameter range to avoid excessive model sizes.

The results for OIPS and CV suggest that their hyperparameters may need to be tuned for each dataset. As mentioned, this is not possible in continual learning where data is not available in advance. In contrast, across all configurations, VIPS demonstrates consistent behaviour, minimising model size in the majority of cases and hence reducing the need for dataset-specific tuning. Based on our experimental results, we recommend using $\delta = 0.035$ for VIPS, which represents the average of successful parameter values when all datasets were considered. This value should provide near-exact performance across different scenarios while minimising model size.

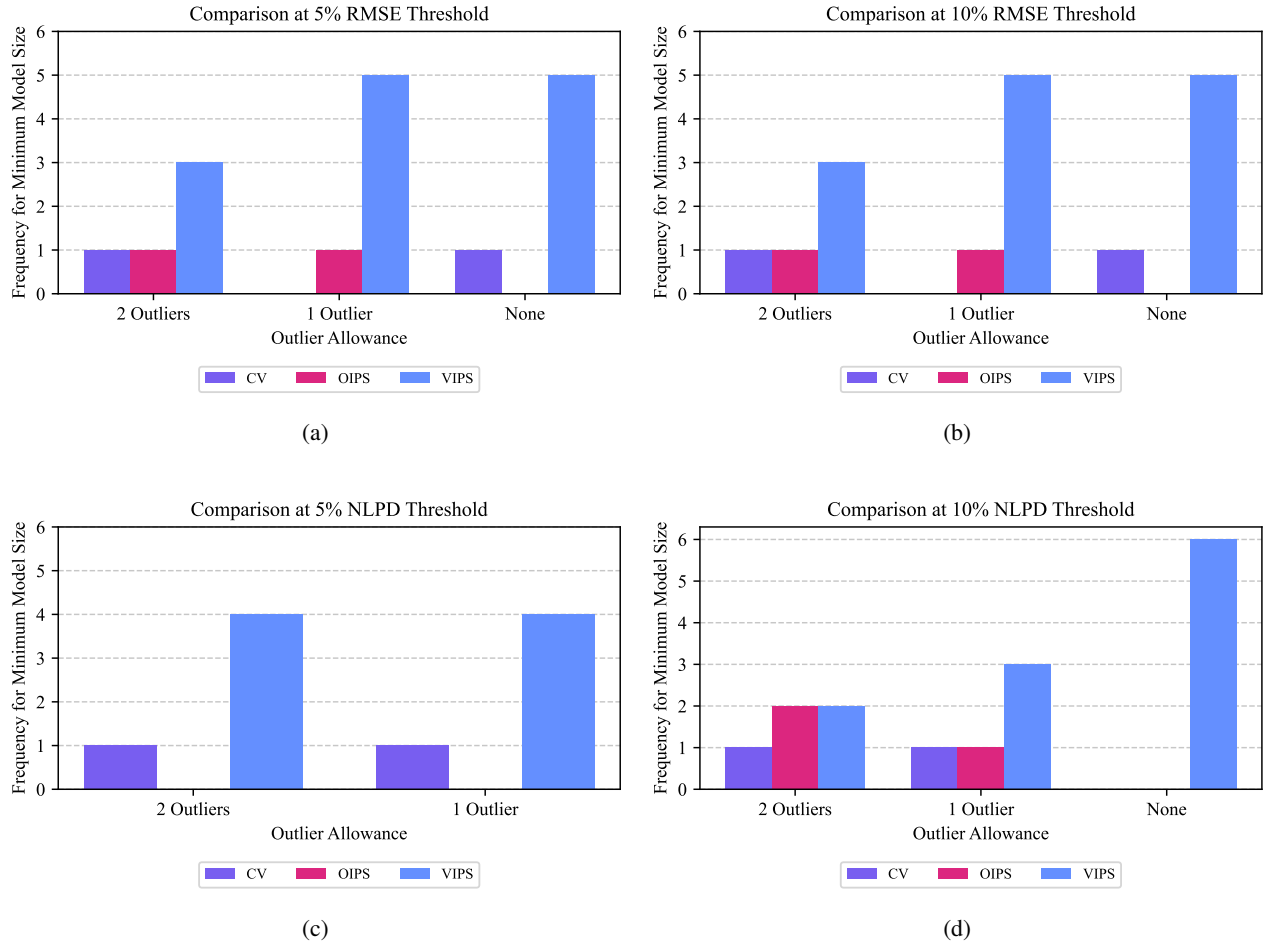


Figure 6. Number of datasets where each method achieves minimal model size at different outlier allowances. A “win” is assigned to the dataset with the smallest model which satisfies the accuracy thresholds. Counts represent absolute wins. Higher counts indicate better method robustness across datasets.

Table 6. Mean (std) of inducing points and RMSE % difference with full-batch GP for the final batch at different outlier allowances at the selected operating point to achieve a 5% RMSE threshold. **X** Gray highlights cases that exceed the threshold, and **bold** text identifies the smallest model size. ‘Max.’ denotes reaching the model’s maximum capacity limit. For none, one, and two outliers, the selected hyperparameters were: VIPS $\delta = 0.02, 0.065, 0.14$, CV $\eta = 0.005, 0.06, 0.5$ and OIPS $\rho = -, 0.955, 0.935$.

DATASET	OUTLIERS	VIPS		CV		OIPS	
		M	RMSE %	M	RMSE %	M	RMSE %
CONCRETE	NONE	434(79)	3.67%	492(54)	3.28%	-	-
	ONE	317(101)	4.24%	321(71)	4.65%	383(90)	4.89%
	TWO	X 163(95)	X 10.43%	X 70(5)	X 13.34%	X 257(106)	X 8.84%
SKILLCRAFT	NONE	177(13)	0.14%	739(34)	0.02%	-	-
	ONE	135(1)	0.21%	312(15)	0.04%	236(38)	0.07%
	TWO	133(0)	0.13%	130(56)	1.45%	185(21)	0.02%
KIN8NM	NONE	4281(69)	0.15%	6316(9)	0.11%	-	-
	ONE	2513(54)	1.35%	5685(26)	0.13%	6523(9)	0.12%
	TWO	1356(42)	4.97%	5334(27)	0.13%	6499(10)	0.12%
NAVAL	NONE	373(19)	0.40%	49(1)	2.39%	-	-
	ONE	86(3)	1.37%	X 23(7)	X 79.08%	X 20(1)	X 407.22%
	TWO	43(2)	2.67%	X 21(5)	X 336.08%	X 20(1)	X 407.22%
ELEVATORS	NONE	438(17)	1.19%	3133(140)	0.31%	-	-
	ONE	312(5)	1.33%	1194(42)	0.85%	386(47)	1.19%
	TWO	276(4)	1.55%	536(22)	1.43%	305(13)	1.49%
BIKE	NONE	2296(138)	4.61%	MAX. 7000	0.58%	-	-
	ONE	X 853(19)	X 8.73%	5563(36)	1.02%	3916(122)	2.19%
	TWO	X 498(10)	X 11.24%	3940(41)	1.80%	2438(130)	4.23%

Table 7. Mean (std) of inducing points and RMSE % difference with full-batch GP for the final batch at different outlier allowances at the selected operating point to achieve a 10% RMSE threshold. **X** Gray highlights cases that exceed the threshold, and **bold** text identifies the smallest model size. ‘Max.’ denotes reaching the model’s maximum capacity limit. For none, one, and two outliers, the selected hyperparameters were: VIPS $\delta = 0.095, 0.12, 0.2$, CV $\eta = 0.005, 0.13, 0.6$ and OIPS $\rho = -, 0.93, 0.845$.

DATASET	OUTLIERS	VIPS		CV		OIPS	
		M	RMSE %	M	RMSE %	M	RMSE %
CONCRETE	NONE	234(116)	7.66%	492(54)	3.28%	-	-
	ONE	178(95)	9.81%	184(74)	9.96%	240(102)	9.03%
	TWO	X 111(37)	X 12.93%	X 3(2)	X 126.81%	X 108(66)	X 20.79%
SKILLCRAFT	NONE	134(0)	0.12%	739(34)	-0.02%	-	-
	ONE	133(0)	0.13%	226(22)	0.20%	174(16)	0.04%
	TWO	133(0)	0.12%	17(0)	7.93%	138(1)	0.06%
KIN8NM	NONE	1904(51)	2.55%	6316(9)	0.11%	-	-
	ONE	1563(39)	3.77%	5334(27)	0.13%	6458(8)	0.11%
	TWO	949(25)	8.17%	3408(32)	0.19%	5053(93)	0.14%
NAVAL	NONE	57(1)	2.25%	49(1)	2.39%	-	-
	ONE	48(3)	2.57%	X 23(8)	X 323.90%	X 16(1)	X 315.28%
	TWO	38(1)	5.07%	X 19(3)	X 553.11%	X 13(1)	X 407.6%
ELEVATORS	NONE	291(4)	1.41%	3133(140)	0.31%	-	-
	ONE	281(4)	1.51%	836(27)	1.11%	298(8)	1.49%
	TWO	267(2)	1.57%	287(18)	2.52%	269(1)	1.59%
BIKE	NONE	650(17)	9.98%	MAX. 7000	0.58%	-	-
	ONE	X 549(14)	X 10.82%	4895(28)	1.29%	1964(137)	5.18%
	TWO	X 400(7)	X 12.50%	2582(66)	4.22%	594(11)	10.00%

Table 8. Mean (std) of inducing points and NLPD % difference with full-batch GP for the final batch at different outlier allowances at the selected operating point to achieve a 5% NLPD threshold. **X** Gray highlights cases that exceed the threshold, and **bold** text identifies the smallest model size. ‘Max.’ denotes reaching the model’s maximum capacity limit. For one, and two outliers, the selected hyperparameters were: VIPS $\delta = 0.005, 0.015$, CV $\eta = 0.025, 0.100$ and OIPS $\rho = 0.975, 0.960$.

DATASET	OUTLIERS	VIPS		CV		OIPS	
		M	NLPD %	M	NLPD %	M	NLPD %
CONCRETE	ONE	514(74)	2.27%	417(63)	4.28%	490(73)	4.72%
	TWO	451(77)	4.43%	X 227(84)	X 15.84%	X 410(88)	X 8.34%
SKILLCRAFT	ONE	291(34)	0.60%	439(29)	0.60%	488(119)	0.53%
	TWO	195(18)	0.78%	260(16)	0.81%	332(82)	0.51%
KIN8NM	ONE	5640(73)	0.85%	6194(12)	0.88%	6547(8)	0.88%
	TWO	5065(46)	0.94%	5334(27)	0.90%	6539(8)	0.88%
NAVAL	ONE	X 1389(18)	X 5.48%	X 27(5)	X 4654.12%	X 26(1)	X 115.29%
	TWO	X 509(15)	X 9.99%	X 28(4)	X 8146.44%	X 20(1)	X 133.87%
ELEVATORS	ONE	740(24)	1.74%	1814(88)	1.23%	1869(257)	1.34%
	TWO	487(24)	2.02%	916(21)	1.91%	642(133)	2.01%
BIKE	ONE	5064(334)	3.99%	MAX. 7000	2.13%	6794(79)	2.55%
	TWO	X 2794(148)	X 8.18%	4895(28)	3.54%	5131(64)	4.05%

Table 9. Mean (std) of inducing points and NLPD % difference with full-batch GP for the final batch at different outlier allowances at the selected operating point to achieve a 10% NLPD threshold. **X** Gray highlights cases that exceed the threshold, and **bold** text identifies the smallest model size. ‘Max.’ denotes reaching the model’s maximum capacity limit. For one, and two outliers, the selected hyperparameters were: VIPS $\delta = 0.015, 0.02, 0.05$, CV $\eta = 0.001, 0.07, 0.5$ and OIPS $\rho = -, 0.95, 0.935$.

DATASET	OUTLIERS	VIPS		CV		OIPS	
		M	NLPD %	M	NLPD %	M	NLPD %
CONCRETE	NONE	451(77)	4.43%	492(54)	2.32%	-	-
	ONE	434(79)	5.68%	303(78)	9.64%	383(90)	9.56%
	TWO	349(86)	9.04%	X 3(2)	X 118.71%	X 247(101)	X 15.73%
SKILLCRAFT	NONE	195(18)	0.78%	1091(107)	-0.34%	-	-
	ONE	177(13)	0.78%	295(13)	0.71%	236(38)	0.57%
	TWO	141(4)	1.10%	X 54(45)	X 8.03%	185(21)	0.68%
KIN8NM	NONE	5065(46)	0.94%	6469(7)	0.87%	-	-
	ONE	4281(69)	0.34%	5685(26)	0.91%	6523(9)	0.88%
	TWO	2953(71)	1.89%	5334(27)	0.90%	6499(10)	0.88%
NAVAL	NONE	509(15)	9.99%	903(8)	6.17%	-	-
	ONE	X 373(19)	X 11.92%	X 27(6)	X 2405.87%	X 20(1)	X 133.87%
	TWO	X 126(6)	X 22.95%	X 22(7)	X 6012.82%	X 20(1)	X 133.87%
ELEVATORS	NONE	487(24)	2.02%	3133(140)	0.82%	-	-
	ONE	438(17)	2.11%	1099(50)	1.73%	386(47)	2.68%
	TWO	332(8)	2.54%	352(29)	3.76%	305(13)	4.84%
BIKE	NONE	2794(148)	8.18%	MAX. 7000	2.04%	-	-
	ONE	2296(138)	9.92%	5563(36)	2.95%	3916(122)	5.50%
	TWO	X 1037(23)	X 17.16%	2582(66)	9.61%	2438(130)	9.66%

D.4. Magnetic anomalies

The data used in this experiment is obtained from [Solin et al. \(2018\)](#) and is available on [GitHub](#). The objective of this task is to detect local anomalies in the Earth’s magnetic field online, caused by the presence of bedrock and magnetic materials in indoor building structures. For this purpose, a small robot with a 3-axis magnetometer moves around an indoor space of approximately 6 meters by 6 meters and measures the magnetic field strength. Out of the 9 available trajectories, we use trajectories 1, 2, 3, 4, and 5 (with $n = 8875, 9105, 9404, 7332, 8313$, respectively) for the experiments. Specifically, we use trajectories 1, 2, 4 and 5 for the first experiment and trajectory 3 for the second.

We use the experimental setup proposed in [Chang et al. \(2023a\)](#). The proposed model applies a GP prior to magnetic field strength, given by $\mathcal{GP}\left(0, \sigma_0^2 + \kappa_{\sigma^2, \ell}^{\text{Mat}}(\mathbf{x}, \mathbf{x}')\right)$ (in μT), where the kernel consist of a constant kernel and a Matérn- $\nu/2$ kernel. The model assumes the spatial domain is affected by Gaussian noise with a variance σ_n^2 . The initial variance for the constant kernel is set to 500, and the Gaussian likelihood is initialised with a noise variance of 0.1.

Our aim for this experiment is to test the optimal hyperparameters identified in the previous section for the 10% RMSE threshold for each adaptive method in a real-world setting. The setting simulates an ever-expanding domain, where the robot is not confined to a predefined area. In this context, the model continuously learns new parts of the space. Therefore, a method that works will need to sufficiently expand the model’s size to accommodate new data without letting it grow uncontrollably.

In the first experiment, we aim to sequentially learn the paths taken by the robot using trajectories 1, 2, 4 and 5, i.e. an entire path will correspond to one batch. We investigate whether the method can adapt to changes in the environment and adjust the number of inducing points accordingly. During this process, we concurrently learn the hyperparameters σ_0^2 , σ^2 , ℓ , and σ_n^2 . As a test set, we use trajectory 3. Figures 7, 9, 11 show the temporally updating field estimate over batches alongside the corresponding path travelled in each batch.

In the second experiment, we focus on the streaming learning of trajectory 3. The trajectory is split into 20 batches. We compare the number of inducing points selected and the estimate obtained by the three methods, Conditional Variance (CV), OIPS and VIPS (ours). Detailed learning of the path for each method is shown in figures 8, 10 and 12. As a test set, we use trajectories 1, 2, 4, and 5. We observed how Conditional Variance chooses an excessive number of inducing points, indicating that its hyperparameter needs tuning, which is impractical in the continual learning setting. OIPS chooses the least inducing points; as seen in the figure, the number of inducing points is more concentrated at the beginning of the path and becomes sparse towards the end. When compared to the learning estimates derived from the paths in the previous experiment, we see that both estimates differ significantly, suggesting that OIPS fails to adequately add enough capacity to capture the changes in the environment. VIPS provides the middle ground, selecting a moderate number of inducing points that effectively balance accuracy and memory size. This choice allows VIPS to maintain a robust estimate of the magnetic field obtained when compared to learning by paths without excessive computational overhead.

In the last two sections, when compared to both alternative approaches, VIPS achieved the best trade-off between performance and model size without requiring hyperparameter tuning, making it the preferred method among the three.

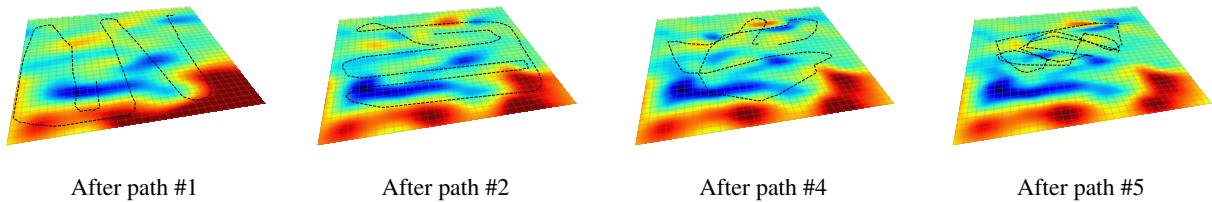


Figure 7. VIPS (Ours). A small robot with wheels is used to perform sequential estimation of magnetic field anomalies. We show the estimate of the magnitude field learned sequentially after travelling the path shown in a dotted line. The degree of transparency represents the marginal variance.

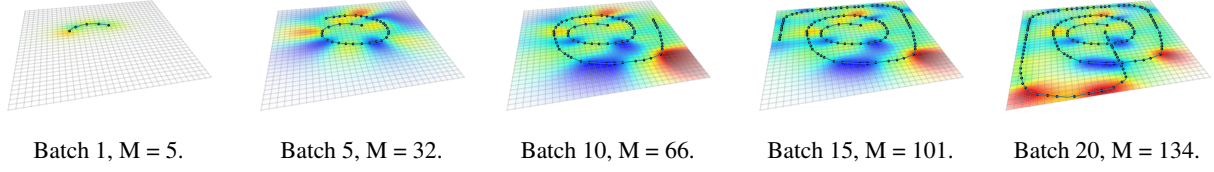


Figure 8. VIPS (Ours). A small robot with wheels is used to perform sequential estimation of magnetic field anomalies. Data is collected continuously as the robot moves along the path. The inducing points are represented as black dots and the line represents the travelled part of the path. We indicate the batch number and number of inducing points (M). Final RMSE = 7.55.

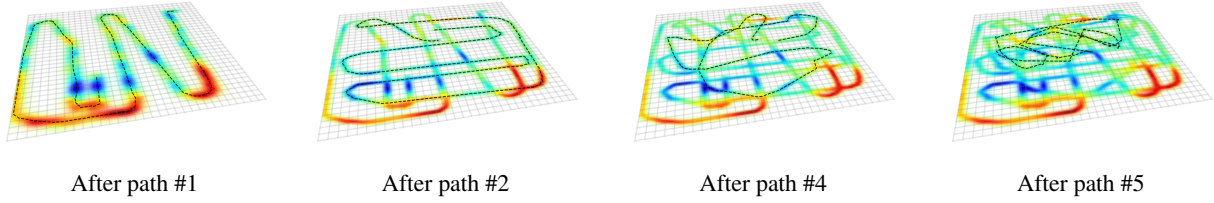


Figure 9. Conditional Variance. A small robot with wheels is used to perform sequential estimation of magnetic field anomalies. We show the estimate of the magnitude field learned sequentially after travelling the path shown in a dotted line. The degree of transparency represents the marginal variance.

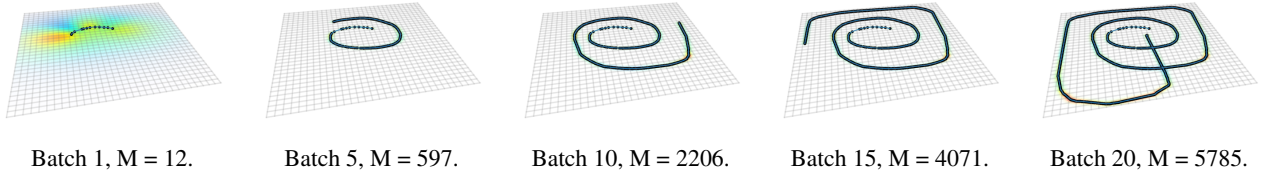


Figure 10. Conditional Variance. A small robot with wheels is used to perform sequential estimation of magnetic field anomalies. Data is collected continuously as the robot moves along the path. The inducing points are represented as black dots and the line represents the travelled part of the path. We indicate the batch number and number of inducing points (M). Final RMSE = 10.66.

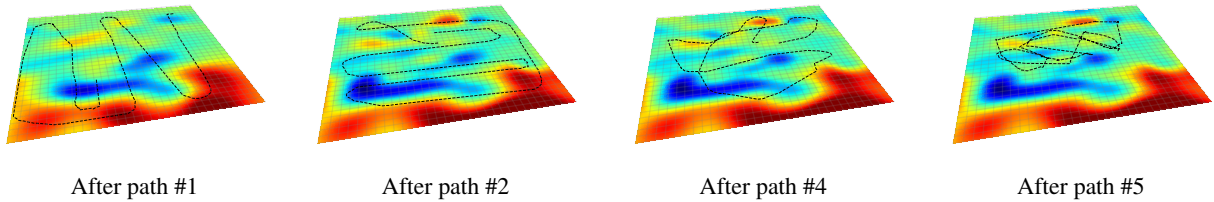


Figure 11. OIPS. A small robot with wheels is used to perform sequential estimation of magnetic field anomalies. We show the estimate of the magnitude field learned sequentially after travelling the path shown in a dotted line. The degree of transparency represents the marginal variance.

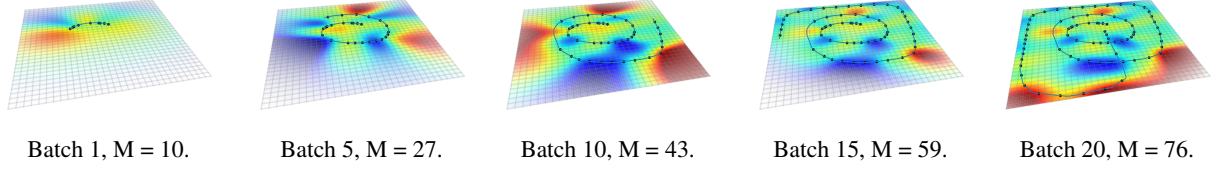


Figure 12. OIPS. A small robot with wheels is used to perform sequential estimation of magnetic field anomalies. Data is collected continuously as the robot moves along the path. The inducing points are represented as black dots and the line represents the travelled part of the path. We indicate the batch number and number of inducing points (M). Final RMSE = 9.42.

D.5. Model size and Data Distribution with Larger Data

We reproduce the three continual learning scenarios from App. D.1, increasing the dataset size to 50 000 data points and the input dimensionality to three, and compare how the three methods (VIPS, OIPS and Conditional Variance) scale in this setting. This setup aims to simulate practical scenarios where an exact GP would be infeasible, and a sequential approach is required instead. We use a Squared Exponential kernel initialised with lengthscale 1.0 and variance 1.0. The noise variance is set to 0.2. We evaluate all methods at the operating points corresponding to the 10% RMSE thresholds determined from the UCI experiments (App. D.3). The data is divided into 50 batches of 1000 points each.

Dataset 1: Data points are sampled uniformly from $[0, 20]^3$, with batches ordered such that new regions of the input space are discovered gradually. Model size is expected to grow linearly as more data is observed.

Dataset 2: Data points are again sampled uniformly from $[0, 20]^3$, but batches are shuffled at random. This represents a setting where the entire input space is observed from the start. As a result, the novelty of each batch decreases over time, and the model size is expected to plateau.

Dataset 3: Most data is concentrated in a narrow region, with occasional outliers. Ninety per cent of the points are drawn from a truncated Cauchy distribution on $[5, 15]^3$ centred at 10, and the remainder uniformly from $[0, 40]^3$. This mimics a scenario where data is dense but includes some outliers. Therefore, the model size should remain small until the region with the outliers is observed.

Figure 13 shows how the number of inducing points evolves as more data is observed for each method across the three datasets. VIPS adapts across all settings, increasing model size only when needed. OIPS shows similar trends but typically adds up to twice as many inducing points as VIPS. CV grows rapidly even when unnecessary (e.g. Dataset 2). Note that for CV we impose a cap of 5000 inducing points.

The datasets were designed to be well-approximated with relatively few inducing points, and all methods attain near-exact accuracy. These results show that even at a larger scale, VIPS maintains a compact model, using fewer than a thousand inducing points in all cases. In contrast, the results for CV and OIPS suggest that these methods may require hyperparameter tuning to control model growth in larger settings.

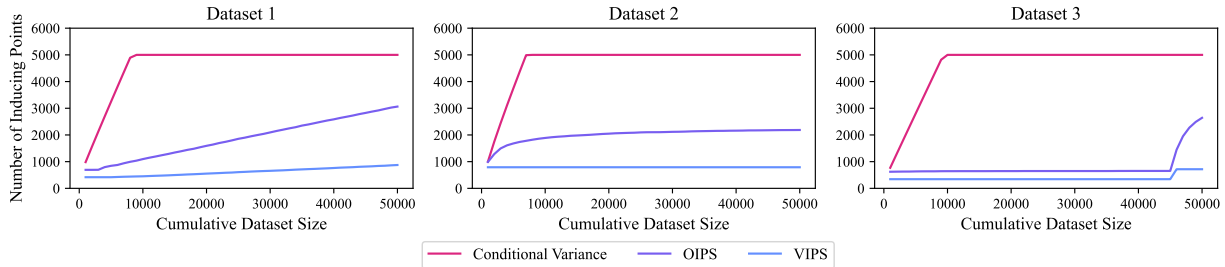


Figure 13. Number of inducing points used by each method as a function of cumulative data size, across the three synthetic datasets. VIPS consistently uses fewer inducing points and only increases model size as needed. OIPS follows a similar trend but typically uses twice as many inducing points as VIPS. CV grows unnecessarily and reaches the imposed cap of 5000 points.

E. Derivation the optimal form of variational distribution

The derivations follow [Bui et al. \(2017\)](#). We start by rewriting the online lower bound,

$$\begin{aligned}\widehat{\mathcal{L}} &= - \int df q_n(f) \left[\log \frac{p(\mathbf{a}|\theta_o)q_n(\mathbf{b})}{p(\mathbf{b}|\theta_n)q_o(\mathbf{a})p(\mathbf{y}_n|f)} \right] \\ &= \int df q_n(f) \log p(\mathbf{y}_n|f) - \int df q_n(f) \log \frac{p(\mathbf{a}|\theta_o)q_n(\mathbf{b})}{p(\mathbf{b}|\theta_n)q_o(\mathbf{a})}.\end{aligned}\quad (20)$$

We can find the optimal q_n by setting the derivative of $\widehat{\mathcal{L}}$ with respect to $q_n(\mathbf{b})$ to zero,

$$\frac{d\widehat{\mathcal{L}}}{dq(\mathbf{b})} = \int df p(f|\mathbf{b}) \left[\log p(\mathbf{y}_n | \mathbf{f}) - \log \frac{p(\mathbf{a} | \theta_{\text{old}}) q_n(\mathbf{b})}{p(\mathbf{b} | \theta_{\text{new}}) q_o(\mathbf{a})} \right] - 1 = 0, \quad (21)$$

this gives us,

$$q_{\text{opt}}(\mathbf{b}) \propto p(\mathbf{b}|\theta_n) \exp \left(\int d\mathbf{a} p(\mathbf{a} | \mathbf{b}, \theta_n) \log \frac{q_o(\mathbf{a})}{p(\mathbf{a} | \theta_{\text{old}})} + \int d\mathbf{f} p(\mathbf{f} | \mathbf{b}, \theta_n) \log p(\mathbf{y}_n | \mathbf{f}) \right).$$

Let $q(\mathbf{a}) = \mathcal{N}(\mathbf{a}; \mathbf{m}_a, \mathbf{S}_a)$ and $p(\mathbf{a} | \theta_{\text{old}}) = \mathcal{N}(\mathbf{a}; \mathbf{0}, \mathbf{K}'_{aa})$, and denoting $\mathbf{D}_a = (\mathbf{S}_a^{-1} - \mathbf{K}'_{aa}{}^{-1})^{-1}$, $\mathbf{Q}_{ff} = \mathbf{K}_{fb}\mathbf{K}_{bb}^{-1}\mathbf{K}_{bf}$, and $\mathbf{Q}_{aa} = \mathbf{K}_{ab}\mathbf{K}_{bb}^{-1}\mathbf{K}_{ba}$, the exponents in the optimal $q_{\text{opt}}(\mathbf{b})$ can be simplified as follows:

$$\begin{aligned}E_1 &= \int d\mathbf{a} p(\mathbf{a} | \mathbf{b}, \theta_n) \log \frac{q(\mathbf{a})}{p(\mathbf{a} | \theta_o)} \\ &= \frac{1}{2} \int d\mathbf{a} \mathcal{N}(\mathbf{a}; \mathbf{K}_{ab}\mathbf{K}_{bb}^{-1}\mathbf{b}, \mathbf{Q}_a) \left(-\log \frac{|\mathbf{S}_a|}{|\mathbf{K}'_{aa}|} - (\mathbf{a} - \mathbf{m}_a)^\top \mathbf{S}_a^{-1} (\mathbf{a} - \mathbf{m}_a) + \mathbf{a}^\top \mathbf{K}'_{aa}{}^{-1} \mathbf{a} \right) \\ &= \frac{1}{2} \int d\mathbf{a} \mathcal{N}(\mathbf{a}; \mathbf{K}_{ab}\mathbf{K}_{bb}^{-1}\mathbf{b}, \mathbf{Q}_a) \left(-\log \frac{|\mathbf{S}_a|}{|\mathbf{K}'_{aa}|} - \mathbf{m}_a^\top \mathbf{S}_a^{-1} \mathbf{m}_a + \mathbf{a}^\top (\mathbf{K}'_{aa}{}^{-1} - \mathbf{S}_a^{-1}) \mathbf{a} + 2 \mathbf{a}^\top \mathbf{S}_a^{-1} \mathbf{m}_a \right) \\ &= \frac{1}{2} \int d\mathbf{a} \mathcal{N}(\mathbf{a}; \mathbf{K}_{ab}\mathbf{K}_{bb}^{-1}\mathbf{b}, \mathbf{Q}_a) \left(-\log \frac{|\mathbf{S}_a|}{|\mathbf{K}'_{aa}|} - \mathbf{m}_a^\top \mathbf{S}_a^{-1} \mathbf{m}_a - \mathbf{a}^\top \mathbf{D}_a^{-1} \mathbf{a} + 2 \mathbf{a}^\top \mathbf{S}_a^{-1} \mathbf{m}_a \right) \\ &= \int d\mathbf{a} \mathcal{N}(\mathbf{a}; \mathbf{K}_{ab}\mathbf{K}_{bb}^{-1}\mathbf{b}, \mathbf{Q}_a) \log \mathcal{N}(\mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a; \mathbf{a}, \mathbf{D}_a) + \Delta_a \\ &= \log \mathcal{N}(\mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a; \mathbf{K}_{ab}\mathbf{K}_{bb}^{-1}\mathbf{b}, \mathbf{D}_a) - \text{tr}[\mathbf{D}_a^{-1} (\mathbf{K}_{aa} - \mathbf{Q}_{aa})] + \Delta_a \\ E_2 &= \int d\mathbf{f} p(\mathbf{f} | \mathbf{b}, \theta_n) \log p(\mathbf{y}_n | \mathbf{f}) \\ &= \int d\mathbf{f} \mathcal{N}(\mathbf{f}; \mathbf{K}_{fb}\mathbf{K}_{bb}^{-1}\mathbf{b}, \mathbf{Q}_f) \log \mathcal{N}(\mathbf{y}; \mathbf{f}, \sigma^2 \mathbf{I}) \\ &= \log \mathcal{N}(\mathbf{y}; \mathbf{K}_{fb}\mathbf{K}_{bb}^{-1}\mathbf{b}, \sigma^2 \mathbf{I}) - \frac{1}{2\sigma^2} \text{tr}(\mathbf{K}_{ff} - \mathbf{Q}_{ff}), \\ 2\Delta_a &= -\log \frac{|\mathbf{S}_a|}{|\mathbf{K}'_{aa}| |\mathbf{D}_a|} + \mathbf{m}_a^\top \mathbf{S}_a^{-1} \mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a - \mathbf{m}_a^\top \mathbf{S}_a^{-1} \mathbf{m}_a + M_a \log(2\pi),\end{aligned}\quad (22)$$

where Δ_a is a constant term with respect to $\mathbf{b}$ and M_a is the dimension of $\mathbf{a}$. The optimal form of q_n is then given by:

$$q_{\text{opt}}(\mathbf{b}) \propto p(\mathbf{b}|\theta_n) \mathcal{N}(\hat{\mathbf{y}}, \mathbf{K}_{fb}\mathbf{K}_{bb}^{-1}\mathbf{b}, \Sigma_{\hat{\mathbf{y}}}) \quad (23)$$

where

$$\hat{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a \end{bmatrix}, \mathbf{K}_{fb} = \begin{bmatrix} \mathbf{K}_{fb} \\ \mathbf{K}_{ab} \end{bmatrix}, \Sigma_{\hat{\mathbf{y}}} = \begin{bmatrix} \sigma_y^2 \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_a \end{bmatrix}. \quad (24)$$

After normalisation, we have

$$q_{\text{opt}}(\mathbf{b}) = \frac{p(\mathbf{b}|\theta_n) \mathcal{N}(\hat{\mathbf{y}}, \mathbf{K}_{fb}\mathbf{K}_{bb}^{-1}\mathbf{b}, \Sigma_{\hat{\mathbf{y}}})}{\int d\mathbf{b} p(\mathbf{b}|\theta_n) \mathcal{N}(\hat{\mathbf{y}}, \mathbf{K}_{fb}\mathbf{K}_{bb}^{-1}\mathbf{b}, \Sigma_{\hat{\mathbf{y}}})} \quad (25)$$

Substituting the above results into the lower bound, we have

$$\hat{\mathcal{L}} = \log \mathcal{N}(\hat{\mathbf{y}}; \mathbf{0}, \mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}} + \Sigma_{\hat{\mathbf{y}}}) + \Delta_{\mathbf{a}} - \frac{1}{2} \text{tr} [\mathbf{D}_{\mathbf{a}}^{-1} (\mathbf{K}_{\mathbf{a}\mathbf{a}} - \mathbf{Q}_{\mathbf{a}\mathbf{a}})] - \frac{1}{2\sigma^2} \text{tr}(\mathbf{K}_{\mathbf{f}\mathbf{f}} - \mathbf{Q}_{\mathbf{f}\mathbf{f}}). \quad (26)$$

E.1. Derivation of $\mathcal{L}^*$

Recall that

$$\mathcal{L}^* = \log \mathcal{N}(\hat{\mathbf{y}}; \mathbf{0}, \mathbf{K}_{\hat{\mathbf{f}}\hat{\mathbf{f}}} + \Sigma_{\hat{\mathbf{y}}}) + \Delta_{\mathbf{a}}, \quad (27)$$

$$\text{with } \mathbf{K}_{\hat{\mathbf{f}}\hat{\mathbf{f}}} = \begin{bmatrix} \mathbf{K}_{\mathbf{f}\mathbf{f}} & \mathbf{K}_{\mathbf{f}\mathbf{a}} \\ \mathbf{K}_{\mathbf{a}\mathbf{f}} & \mathbf{K}_{\mathbf{a}\mathbf{a}} \end{bmatrix}.$$

The first term can be lower bounded using Jensen's inequality as,

$$\begin{aligned} & \log \mathcal{N}(\hat{\mathbf{y}}; \mathbf{0}, \mathbf{K}_{\hat{\mathbf{f}}\hat{\mathbf{f}}} + \Sigma_{\hat{\mathbf{y}}}) \\ & \geq \log \mathcal{N}(\hat{\mathbf{y}}; \mathbf{0}, \mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}} + \Sigma_{\hat{\mathbf{y}}}) - \frac{1}{2} \text{tr} \left(\Sigma_{\hat{\mathbf{y}}}^{-1} (\mathbf{K}_{\hat{\mathbf{f}}\hat{\mathbf{f}}} - \mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}}) \right) \end{aligned} \quad (28)$$

where $\mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}}$ is the Nyström approximation of $\mathbf{K}_{\hat{\mathbf{f}}\hat{\mathbf{f}}}$. The trace will be small when $\mathbf{b} = \{f(\mathbf{x}_n, \mathbf{a})\}$ and can be simplified as follows:

$$\begin{aligned} & \text{tr} \left(\Sigma_{\hat{\mathbf{y}}}^{-1} (\mathbf{K}_{\hat{\mathbf{f}}\hat{\mathbf{f}}} - \mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}}) \right) \\ & = \text{tr} \left(\begin{bmatrix} \sigma_y^{-2} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\mathbf{a}}^{-1} \end{bmatrix} \left(\begin{bmatrix} \mathbf{K}_{\mathbf{f}\mathbf{f}} & \mathbf{K}_{\mathbf{f}\mathbf{a}} \\ \mathbf{K}_{\mathbf{a}\mathbf{f}} & \mathbf{K}_{\mathbf{a}\mathbf{a}} \end{bmatrix} - \begin{bmatrix} \mathbf{K}_{\mathbf{f}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\mathbf{f}} & \mathbf{K}_{\mathbf{f}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\mathbf{a}} \\ \mathbf{K}_{\mathbf{a}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\mathbf{f}} & \mathbf{K}_{\mathbf{a}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\mathbf{a}} \end{bmatrix} \right) \right) \\ & = \text{tr} (\mathbf{D}_{\mathbf{a}}^{-1} (\mathbf{K}_{\mathbf{a}\mathbf{a}} - \mathbf{K}_{\mathbf{a}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\mathbf{a}})) + \frac{1}{\sigma^2} \text{tr}(\mathbf{K}_{\mathbf{f}\mathbf{f}} - \mathbf{K}_{\mathbf{f}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\mathbf{f}}) \end{aligned} \quad (29)$$

which recovers the expression for $\hat{\mathcal{L}} - \Delta_{\mathbf{a}}$.

F. Online Upper Bound Implementation

In this section, we provide efficient forms for practical implementation of the online upper bound $\hat{\mathcal{U}}$. As the second term is constant we focus on the first term,

$$\hat{\mathcal{U}}_2 = -\frac{(N + M_{\mathbf{a}})}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}} + \Sigma_{\hat{\mathbf{y}}}| - \frac{1}{2} \hat{\mathbf{y}}^T (\mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}} + t\mathbf{I} + \Sigma_{\hat{\mathbf{y}}})^{-1} \hat{\mathbf{y}}. \quad (30)$$

This term is an upper bound for the first term of $\mathcal{L}^* = \log \mathcal{N}(\hat{\mathbf{y}}; \mathbf{0}, \mathbf{K}_{\hat{\mathbf{f}}\hat{\mathbf{f}}} + \Sigma_{\hat{\mathbf{y}}}) + \Delta_{\mathbf{a}}$.

F.1. Determinant term

Letting $\mathbf{K}_{\mathbf{b}\mathbf{b}} = \mathbf{L}_{\mathbf{b}} \mathbf{L}_{\mathbf{b}}^T$ and using the matrix determinant lemma, we can rewrite the determinant term as

$$\begin{aligned} \log |\mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{K}_{\mathbf{b}\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}} + \Sigma_{\hat{\mathbf{y}}}| &= \log |\Sigma_{\hat{\mathbf{y}}}| + \log |\mathbf{I} + \mathbf{L}_{\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}} \Sigma_{\hat{\mathbf{y}}}^{-1} \mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{L}_{\mathbf{b}}^{-T}| \\ &= N \log \sigma_y^2 + \log |\mathbf{D}_{\mathbf{a}}| + \log |\mathbf{I} + \mathbf{L}_{\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}} \Sigma_{\hat{\mathbf{y}}}^{-1} \mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{L}_{\mathbf{b}}^{-T}| \end{aligned} \quad (31)$$

Let $\mathbf{D} = \mathbf{I} + \mathbf{L}_{\mathbf{b}}^{-1} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}} \Sigma_{\hat{\mathbf{y}}}^{-1} \mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} \mathbf{L}_{\mathbf{b}}^{-T}$. Note that,

$$\begin{aligned} \mathbf{K}_{\mathbf{b}\hat{\mathbf{f}}} \Sigma_{\hat{\mathbf{y}}}^{-1} \mathbf{K}_{\hat{\mathbf{f}}\mathbf{b}} &= \begin{bmatrix} \mathbf{K}_{\mathbf{b}\mathbf{f}} & \mathbf{K}_{\mathbf{b}\mathbf{a}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_y^2} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{\mathbf{a}}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{\mathbf{f}\mathbf{b}} \\ \mathbf{K}_{\mathbf{a}\mathbf{b}} \end{bmatrix} \\ &= \frac{1}{\sigma_y^2} \mathbf{K}_{\mathbf{b}\mathbf{f}} \mathbf{K}_{\mathbf{f}\mathbf{b}} + \mathbf{K}_{\mathbf{b}\mathbf{a}} \mathbf{D}_{\mathbf{a}}^{-1} \mathbf{K}_{\mathbf{a}\mathbf{b}} \\ &= \frac{1}{\sigma_y^2} \mathbf{K}_{\mathbf{b}\mathbf{f}} \mathbf{K}_{\mathbf{f}\mathbf{b}} + \mathbf{K}_{\mathbf{b}\mathbf{a}} \mathbf{S}_{\mathbf{a}}^{-1} \mathbf{K}_{\mathbf{a}\mathbf{b}} - \mathbf{K}_{\mathbf{b}\mathbf{a}} \mathbf{K}_{\mathbf{a}\mathbf{a}}'^{-1} \mathbf{K}_{\mathbf{a}\mathbf{b}}. \end{aligned} \quad (32)$$

Therefore,

$$\mathbf{D} = \mathbf{I} + \frac{1}{\sigma_y^2} \mathbf{L}_b^{-1} \mathbf{K}_{bf} \mathbf{K}_{fb} \mathbf{L}_b^{-T} + \mathbf{L}_b^{-1} \mathbf{K}_{ba} \mathbf{S}_a^{-1} \mathbf{K}_{ab} \mathbf{L}_b^{-T} - \mathbf{L}_b^{-1} \mathbf{K}_{ba} \mathbf{K}_{aa}'^{-1} \mathbf{K}_{ab} \mathbf{L}_b^{-T}. \quad (33)$$

F.2. Quadratic term

Given the quadratic term,

$$-\frac{1}{2} \hat{\mathbf{y}}^T (\mathbf{K}_{fb} \mathbf{K}_{bb}^{-1} \mathbf{K}_{bf} + t\mathbf{I} + \Sigma_{\hat{\mathbf{y}}})^{-1} \hat{\mathbf{y}}.$$

Letting $\hat{\Sigma}_{\hat{\mathbf{y}}} = t\mathbf{I} + \Sigma_{\hat{\mathbf{y}}}$ and by Woodbury's formula, we obtain:

$$(\mathbf{K}_{fb} \mathbf{K}_{bb}^{-1} \mathbf{K}_{bf} + \hat{\Sigma}_{\hat{\mathbf{y}}})^{-1} = \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1} - \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1} \mathbf{K}_{fb} \mathbf{L}_b^{-T} \left(\mathbf{I} + \mathbf{L}_b^{-1} \mathbf{K}_{bf} \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1} \mathbf{K}_{fb} \mathbf{L}_b^{-T} \right)^{-1} \mathbf{L}_b^{-1} \mathbf{K}_{bf} \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1}.$$

We have,

$$\hat{\mathbf{y}}^T \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1} \hat{\mathbf{y}} = \frac{1}{\sigma_y^2 + t} \mathbf{y}^T \mathbf{y} + (\mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a)^T (\mathbf{D}_a + t\mathbf{I})^{-1} (\mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a). \quad (34)$$

Letting $\hat{\mathbf{D}} = \mathbf{I} + \mathbf{L}_b^{-1} \mathbf{K}_{bf} \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1} \mathbf{K}_{fb} \mathbf{L}_b^{-T}$ where,

$$\mathbf{K}_{bf} \Sigma_c^{-1} \mathbf{K}_{fb} = \begin{bmatrix} \mathbf{K}_{fb} \\ \mathbf{K}_{ab} \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_y^2 + t} \mathbf{I} & 0 \\ 0 & (\mathbf{D}_a + t\mathbf{I})^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{fb} \\ \mathbf{K}_{ab} \end{bmatrix} = \frac{1}{\sigma_y^2 + t} \mathbf{K}_{bf} \mathbf{K}_{fb} + \mathbf{K}_{ba} (\mathbf{D}_a + t\mathbf{I})^{-1} \mathbf{K}_{ab}$$

and letting $\hat{\mathbf{c}} = \mathbf{K}_{bf} \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1} \hat{\mathbf{y}} = \frac{1}{\sigma_y^2 + t} \mathbf{K}_{bf} \mathbf{y} + \mathbf{K}_{ba} (\mathbf{D}_a + t\mathbf{I})^{-1} (\mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a)$, we obtain,

$$\hat{\mathbf{y}}^T \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1} \mathbf{K}_{fb} \mathbf{L}_b^{-T} \left(\mathbf{I} + \mathbf{L}_b^{-1} \mathbf{K}_{bf} \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1} \mathbf{K}_{fb} \mathbf{L}_b^{-T} \right)^{-1} \mathbf{L}_b^{-1} \mathbf{K}_{bf} \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1} \hat{\mathbf{y}} = \hat{\mathbf{c}}^T \mathbf{L}_b^{-T} \hat{\mathbf{D}}^{-1} \mathbf{L}_b^{-1} \hat{\mathbf{c}}. \quad (35)$$

Putting this back into the upper bound:

$$\hat{\mathcal{U}}_2 = -\frac{(N + M_a)}{2} \log(2\pi) - \frac{1}{2} N \log \sigma_y^2 - \frac{1}{2} \log |\mathbf{D}_a| - \frac{1}{2} \log |\mathbf{D}| - \frac{1}{2} \hat{\mathbf{y}}^T \hat{\Sigma}_{\hat{\mathbf{y}}}^{-1} \hat{\mathbf{y}} + \frac{1}{2} \hat{\mathbf{c}}^T \mathbf{L}_b^{-T} \hat{\mathbf{D}}^{-1} \mathbf{L}_b^{-1} \hat{\mathbf{c}}. \quad (36)$$

The upper bound for $\mathcal{L}^*$ is therefore

$$\begin{aligned} \hat{\mathcal{U}} &= -\frac{(N + M_a)}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{K}_{fb} \mathbf{K}_{bb}^{-1} \mathbf{K}_{bf} + \Sigma_{\hat{\mathbf{y}}}| - \frac{1}{2} \hat{\mathbf{y}}^T (\mathbf{K}_{fb} \mathbf{K}_{bb}^{-1} \mathbf{K}_{bf} + t\mathbf{I} + \Sigma_{\hat{\mathbf{y}}})^{-1} \hat{\mathbf{y}} + \Delta_a \\ &= -\frac{N}{2} \log(2\pi \sigma_y^2) - \frac{1}{2} \log |\mathbf{D}| - \frac{1}{2} \frac{1}{\sigma_y^2 + t} \mathbf{y}^T \mathbf{y} - \frac{1}{2} (\mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a)^T (\mathbf{D}_a + t\mathbf{I})^{-1} (\mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a) \\ &\quad + \frac{1}{2} \hat{\mathbf{c}}^T \mathbf{L}_b^{-T} \hat{\mathbf{D}}^{-1} \mathbf{L}_b^{-1} \hat{\mathbf{c}} - \frac{1}{2} \log \frac{|\mathbf{S}_a|}{|\mathbf{K}_{aa}'|} - \frac{1}{2} \mathbf{m}_a^T \mathbf{S}_a^{-1} \mathbf{m}_a + \frac{1}{2} \mathbf{m}_a^T \mathbf{S}_a^{-1} \mathbf{D}_a \mathbf{S}_a^{-1} \mathbf{m}_a. \end{aligned} \quad (37)$$