Input Dependent Sparse Gaussian Processes

Bahram Jafrasteh[*][1][2] **Carlos Villacampa-Calvo**[*][2] **Daniel Hernández-Lobato**[2]

Abstract

Gaussian Processes (GPs) are non-parametric models that provide accurate uncertainty estimates. Nevertheless, they have a cubic cost in the number of data instances N. To overcome this, sparse GP approximations are used, in which a set of $M \ll N$ inducing points is introduced. The location of the inducing points is learned by considering them parameters of an approximate posterior distribution q. Sparse GPs, combined with stochastic variational inference for inferring q have a cost per iteration in $\mathcal{O}(M^3)$. Critically, the inducing points determine the flexibility of the model and they are often located in regions where the latent function changes. A limitation is, however, that in some tasks a large number of inducing points may be required to obtain good results. To alleviate this, we propose here to amortize the computation of the inducing points locations, as well as the parameters of q. For this, we use a neural network that receives a data instance as an input and outputs the corresponding inducing points locations and the parameters of q. We evaluate our method in several experiments, showing that it performs similar or better than other state-of-the-art sparse variational GPs. However, in our method the number of inducing points is reduced drastically since they depend on the input data. This makes our method scale to larger datasets and have faster training and prediction times.

[*]Equal contribution [1]Biomedical Research and Innovation Institute of Cádiz (INiBICA) Research Unit, Puerta del Mar University, Cádiz, Spain [2]Computer Science Department, Universidad Autónoma de Madrid, Madrid, Spain. Correspondence to: Bahram Jafrasteh <jafrasteh.bahram@inibica.es;bahram.jafrasteh@uam.es>, Carlos Villacampa-Calvo <carlos.villacampa@uam.es>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

1. Introduction

Gaussian Processes (GPs) are non-parametric models that can be used to address regression and classification machine learning problems (Rasmussen & Williams, 2006). GPs become more expressive as the number of training instances N grows and, since they are Bayesian models, they provide a predictive distribution that estimates the uncertainty associated to the predictions made. This uncertainty estimation or ability to know what is not known is critical in many practical applications (Gal, 2016). Nevertheless, GPs suffer from poor scalability as their training cost is $\mathcal{O}(N^3)$ per iteration due to the need of computing the inverse of a $N \times N$ covariance matrix. Another limitation is that approximate inference is needed with non-Gaussian likelihoods.

Sparse approximations can improve the cost of GPs. The most popular ones introduce a set of $M \ll N$ inducing points. The inducing points and their associated posterior values completely specify the posterior process at test points. In Snelson & Ghahramani (2006), the computational gain is obtained by assuming independence among the process values at the training points given the inducing points and their values. This can also be seen as using an approximate GP prior (Quiñonero-Candela & Rasmussen, 2005). By contrast, in Titsias (2009) the computational gain is obtained by combining variational inference (VI) with a posterior approximation q that has a fixed part and a tunable part. In both methods the cost is $\mathcal{O}(NM^2)$ per iteration and the inducing points, considered as model's hyper-parameters, are found by maximizing an estimate of the marginal likelihood.

Importantly, the VI approach of Titsias (2009) maximizes a lower bound on the log-marginal likelihood as an indirect way of minimizing the KL-divergence between an approximate posterior distribution for the process values at the inducing points and the corresponding exact posterior. The advantage is that the objective is expressed as a sum over the training instances, allowing for mini-batch training and stochastic optimization techniques to be applied on the objective (Hensman et al., 2015b). This reduces the cost to $\mathcal{O}(M^3)$ per iteration, making GPs scalable to large datasets.

In sparse approximations one often observes in practice that after the optimization process the inducing points are located in regions of the input space in which the latent function changes (Snelson & Ghahramani, 2006; Titsias, 2009; Hensman et al., 2015a; Bauer et al., 2016). Therefore, the expressive power of the model depends on the number of inducing points M and their correct location on the input space. Some problems may require a large number of inducing points, in the order of thousands, to get good prediction results (Hensman et al., 2015b; Shi et al., 2020; Tran et al., 2021). Thus, using inducing point based sparse GPs in those problems becomes very expensive.

Some works tried to improve the training cost of sparse approximations. Cheng & Boots (2017) consider different sets of inducing points for the computation of the posterior mean and variance. Shi et al. (2020) use an orthogonal decomposition of the GP that allows to introduce an extra set of inducing points with less cost. Finally, Tran et al. (2021) consider a large set of inducing points, but restrict the computations for a particular data point to the nearest neighbors to that point from the set of inducing points.

Inspired by Tran et al. (2021), we propose here a new method to improve the cost of sparse GPs. Our method also tries to produce a set of inducing points (and associated VI approximation q) that are specific of each input data point. For that, we note that previous works have shown that one can learn a mapping from the inputs to the parameters of the approximate distribution q, instead of directly optimizing the parameters of q (Kingma & Welling, 2014; Shu et al., 2018). This approach, known as amortized variational inference, is a key contribution of variational auto-encoders (VAE) (Kingma & Welling, 2014), and has also been explored in the context of GPs to solve other problems such as multi-class classification with input noise (Villacampa-Calvo et al., 2021). Amortized inference has also been empirically shown to have useful regularization properties that improve generalization (Shu et al., 2018).

Here, we combine sparse GPs with a neural network (NN) architecture that computes, for each potential data point, the associated inducing points to be used for prediction. We also employ a NN to carry out amortized VI. The NN also computes the parameters of the variational distribution q approximating the posterior of the process values for the outputted inducing points. While the number of parameters that need optimization may increase with the use of a neural network, the extra parameters are parameters of the VI approximation, not the model. Importantly, this approach allows for a big reduction in the total number of inducing points without losing expressive power. In particular, it enables different sets of inducing points associated to each input location. The inducing points are simply given by a mapping from the inputs provided by a neural network. We show on several experiments that the proposed method is able to perform similar or better than standard sparse GPs and related methods for improving the cost of sparse GPs (Tran et al., 2021; Shi et al., 2020). However, the training and prediction times of our method are much better.

2. Gaussian Processes

A Gaussian Process (GP) is a stochastic process for which any finite set of variables has a Gaussian distribution (Rasmussen & Williams, 2006). Consider a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where we assume that $y_i = f(\mathbf{x_i}) + \epsilon_i$, with $f(\cdot)$ a latent function and ϵ_i Gaussian noise with variance σ^2, *i.e.*, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. A GP can be used as a prior for $f(\cdot)$. Then, Bayes' rule is used for making predictions by computing a posterior for $f(\cdot)$ given $\mathcal{D}$. The GP prior for $f(\cdot)$ is specified by a mean function $m(\mathbf{x})$ (often set to zero) and a covariance function $k(\mathbf{x}, \mathbf{x}')$ such that $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$. Covariance functions often have parameters $\boldsymbol{\theta}$. Given $\mathcal{D}$, the posterior of f at a new point $\mathbf{x}^\star$ is Gaussian with mean and variance

$$\mu(\mathbf{x}^\star) = \mathbf{k}(\mathbf{x}^\star)^\text{T}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}\,, \tag{1}$$

$$\sigma^2(\mathbf{x}^\star) = k^\star - \mathbf{k}(\mathbf{x}^\star)^\text{T}(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}^\star)\,, \tag{2}$$

respectively. $k^\star = k(\mathbf{x}^\star, \mathbf{x}^\star)$ and $\mathbf{k}(\mathbf{x}^\star)$, is a vector with the covariances between $f(\mathbf{x}^\star)$ and each $f(\mathbf{x}_i)$. Similarly, $\mathbf{K}$ has the covariances between $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ for $i, j = 1, \ldots, N$. Finally, $\mathbf{I}$ stands for the identity matrix. Popular covariances functions $k(\cdot, \cdot)$ are the squared exponential and the Matérn (Rasmussen & Williams, 2006). Their parameters, $\boldsymbol{\theta}$, and σ^2 can be found by maximizing $p(\mathbf{y})$ (Rasmussen & Williams, 2006). The cost of this approach is $O(N^3)$ since it needs the inversion of $\mathbf{K}$, a $N \times N$ matrix. This makes GPs unsuitable for large data sets.

2.1. Sparse Variational Gaussian Processes

Sparse approximations improve the cost of GPs. The most popular methods introduce, in the same input space as the original data, a new set of $M \ll N$ points , called the inducing points, denoted by $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_M)^\text{T}$ (Snelson & Ghahramani, 2006; Titsias, 2009). Let the corresponding latent function values be $\mathbf{u} = (f(\mathbf{z}_1), \ldots, f(\mathbf{z}_M))^\text{T}$. The inducing points are not restricted to be part of the observed data and their location can be learned during training. A GP prior is placed on $\mathbf{u}$. Namely, $p(\mathbf{u}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K_Z})$, where $\mathbf{K_Z}$ is a matrix with the covariances associated to each pair of points from $\mathbf{Z}$. The idea is that the posterior for f can be approximated in terms of the posterior for $\mathbf{u}$.

In this work we focus on a widely used variational inference (VI) approach to approximate the posterior for f (Titsias, 2009). Let $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\text{T}$. In VI, the goal is to find an approximate posterior for $\mathbf{f}$ and $\mathbf{u}$, $q(\mathbf{f}, \mathbf{u})$, that resembles as much as possible the true posterior $p(\mathbf{f}, \mathbf{u}|\mathbf{y})$. Critically, q is constrained to be $q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$, with $p(\mathbf{f}|\mathbf{u})$ fixed and $q(\mathbf{u})$ a tunable multi-variate Gaussian. To find $q(\mathbf{u})$ a lower bound of the marginal likelihood is maximized. The evidence lower bound (or ELBO) is obtained via Jensen's inequality, leading to (after some simplifications):

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

where $p(y_i|f_i)$ is the model's likelihood for the i-th point and $\text{KL}[\cdot|\cdot]$ is the Kullback-Leibler divergence between probability distributions. In Titsias (2009), they optimize $q(\mathbf{u})$ in closed-form. The resulting expression is then maximized to estimate $\mathbf{Z}$, $\boldsymbol{\theta}$ and σ^2. This leads to a complexity of $\mathcal{O}(NM^2)$. However, if the variational posterior $q(\mathbf{u})$ is optimized alongside with $\mathbf{Z}$, $\boldsymbol{\theta}$ and σ^2, as proposed in Hensman et al. (2013), the ELBO can be expressed as a sum over training instances, which allows for mini-batch training and stochastic optimization. Stochastic variational inference (SVI) reduces the training cost to $\mathcal{O}(M^3)$ per iteration (Hensman et al., 2013). Importantly, the expectation in (3) has a closed-form solution in the case of Gaussian likelihoods. It needs to be approximated in other cases, *e.g.*, binary classification. The second term in (3) is the KL-divergence between the q and the prior, which can be computed analytically, since both are Gaussian.

The expressive power of the sparse GP heavily depends on the number of inducing points M and on their correct placement in the input space via optimizing (3) (Titsias, 2009; Hensman et al., 2015a; Bauer et al., 2016). Critically, in some problems several thousands of inducing points may be required to get good results (Hensman et al., 2015b; Shi et al., 2020; Tran et al., 2021). This makes difficult and expensive using sparse GPs in those problems. In the next section we describe how to alleviate this using our method.

3. Input Dependent Sparse GPs

We develop a new formulation of sparse GPs which for every given input computes the corresponding inducing points to be used for prediction, and also the associated parameters of the approximate distribution q. To achieve this, we consider a meta-point $\tilde{\mathbf{x}}$ that is used to determine the inducing points $\mathbf{Z}$ and the corresponding $\mathbf{u}$. Namely, now $\mathbf{u}$ depends on $\tilde{\mathbf{x}}$, *i.e.*, $\mathbf{u} \sim p(\mathbf{u}|\tilde{\mathbf{x}})$. In particular, we set $p(\mathbf{u}|\tilde{\mathbf{x}}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{Z}(\tilde{\mathbf{x}})})$ where the inducing points $\mathbf{Z}$ depend non-linearly, *e.g.*, via a deep neural network, on $\tilde{\mathbf{x}}$. The joint distribution of $\mathbf{u}$ and $\tilde{\mathbf{x}}$ is then given by $p(\mathbf{u}, \tilde{\mathbf{x}}) = p(\mathbf{u}|\tilde{\mathbf{x}})p(\tilde{\mathbf{x}})$ for some prior distribution $p(\tilde{\mathbf{x}})$. Following Tran et al. (2021), we can consider an implicit distribution $p(\tilde{\mathbf{x}})$. That is, its analytical form is unknown, but we can draw samples from it. Later on, we fully specify $p(\tilde{\mathbf{x}})$.

Note that the marginalized prior $p(\mathbf{u})$ is no longer Gaussian. However, we can show that this formulation does not impact on the prior over $\mathbf{f}$. For an arbitrary selected meta-point $\tilde{\mathbf{x}}$

$$p(\mathbf{f}, \mathbf{u}|\tilde{\mathbf{x}}) = \mathcal{N}\left(\mathbf{0}, \left[\begin{array}{cc} \mathbf{K} & \mathbf{K}_{\mathbf{X},\mathbf{Z}(\tilde{\mathbf{x}})} \\ \mathbf{K}_{\mathbf{Z}(\tilde{\mathbf{x}}),\mathbf{X}} & \mathbf{K}_{\mathbf{Z}(\tilde{\mathbf{x}})} \end{array}\right]\right) \,, \quad (4)$$

where $\mathbf{K}_{\mathbf{X},\mathbf{Z}(\tilde{\mathbf{x}})}$ are the cross-covariances between $\mathbf{f}$ and $\mathbf{u}$. Therefore, if $\mathbf{u}$ is marginalized out in (4), the prior for $\mathbf{f}$ is the standard GP prior and does not depend on $\tilde{\mathbf{x}}$. Hence, $p(\mathbf{f}|\tilde{\mathbf{x}}) = p(\mathbf{f})$. Thus, $p(\mathbf{f}, \mathbf{u}) = \int p(\mathbf{f}, \mathbf{u}|\tilde{\mathbf{x}})p(\tilde{\mathbf{x}})d\tilde{\mathbf{x}}$ is a mixture of Gaussian densities, where the marginal over $\mathbf{f}$ is the same for every component of the mixture. In the standard sparse GP, the inducing points also have an impact on the variational approximation q via the fixed conditional $p(\mathbf{f}|\mathbf{u})$ (Titsias, 2009). Therefore, we also incorporate in the next section the input dependence on $\tilde{\mathbf{x}}$ in q.

3.1. Lower Bound on the Log-Marginal Likelihood

We follow Tran et al. (2021) to derive a lower bound on the log-marginal likelihood of the extended model described above. Consider a posterior approximation of the form $q(\mathbf{f}, \mathbf{u}, \tilde{\mathbf{x}}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u}|\tilde{\mathbf{x}})p(\tilde{\mathbf{x}})$, where only $q(\mathbf{u}|\tilde{\mathbf{x}})$ can be adjusted and the other factors are fixed. Using this posterior's factorization and Jensen's inequality we obtain the lower bound after some simplifications:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_q \left[\log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u}|\tilde{\mathbf{x}})p(\tilde{\mathbf{x}})}{q(\mathbf{f}, \mathbf{u}, \tilde{\mathbf{x}})}\right] \\ &= \mathbb{E}_q \left[\log \frac{p(\mathbf{y}|\mathbf{f})\cancel{p(\mathbf{f}|\mathbf{u})}p(\mathbf{u}|\tilde{\mathbf{x}})\cancel{p(\tilde{\mathbf{x}})}}{\cancel{p(\mathbf{f}|\mathbf{u})}q(\mathbf{u}|\tilde{\mathbf{x}})\cancel{p(\tilde{\mathbf{x}})}}\right] \\ &= \sum_{i=1}^{N} \int p(\tilde{\mathbf{x}}) \left[p(f_i|\mathbf{u})q(\mathbf{u}|\tilde{\mathbf{x}}) \log p(y_i|f_i) d\mathbf{f} d\mathbf{u} \right. \\ &\quad \left. - \tfrac{1}{N}\text{KL}[q(\mathbf{u}|\tilde{\mathbf{x}})|p(\mathbf{u}|\tilde{\mathbf{x}})]\right] d\tilde{\mathbf{x}} \,. \end{aligned} \quad (5)$$

Assume that $p(\tilde{\mathbf{x}})$ is an implicit distribution. We can draw samples from it and approximate the expectation w.r.t $p(\tilde{\mathbf{x}})$. Thus, for a sample $\tilde{\mathbf{x}}_s$ from $p(\tilde{\mathbf{x}})$, (5) is approximated as

$$\begin{aligned}\mathcal{L} \approx \sum_{i=1}^{N} &\left[\mathbb{E}_{p(f_i|\mathbf{u})q(\mathbf{u}|\tilde{\mathbf{x}}_s)}[\log p(y_i|f_i)] \right. \\ &\left. - \tfrac{1}{N}\text{KL}[q(\mathbf{u}|\tilde{\mathbf{x}}_s)|p(\mathbf{u}|\tilde{\mathbf{x}}_s)]\right] \,. \end{aligned} \quad (6)$$

We can evaluate (6) and its gradients to maximize the original objective in (5) using stochastic optimization techniques. This is valid for any implicit distribution $p(\tilde{\mathbf{x}})$. Consider now that we use mini-batch-based training for optimization, and we set $\tilde{\mathbf{x}}_s = \mathbf{x}_i$. In this case, the value of $\tilde{\mathbf{x}}$ remains random, as it depends on the points $(\mathbf{x}_i, y_i)$ that are selected in the random mini-batch. This results in a method that computes different inducing points for each input location. In practice, we use the same sample to approximate the expectation w.r.t. $p(\tilde{\mathbf{x}})$ and the sum across the data in (6). This could introduce a bias in the objective. However, such a reusing of the samples is done in Tran et al. (2021) with good empirical results. Moreover, our experiments in Section 5 also validate this approximation.

3.2. Amortized Variational Inference and Deep Neural Networks

Maximizing the lower bound finds the optimal approximate distribution q. A problem, however, is that we have a potential large number of parameters to fix, corresponding to each $q(\mathbf{u}|\mathbf{x}_i)$. In particular, if we set $q(\mathbf{u}|\mathbf{x}_i)$ to be Gaussian, we will have to infer different means and covariance matrices for each different $\mathbf{x}_i$. This is expected to be memory inefficient and to make optimization more difficult. To reduce the number of parameters of our method we use amortized variational inference and specify a function that can generate these parameters for each $\mathbf{x}_i$ (Shu et al., 2018). More precisely, we set the mean and covariance matrix of $q(\mathbf{u}|\mathbf{x}_i)$ to be $\mathbf{m}(\mathbf{x}_i)$ and $\mathbf{S}(\mathbf{x}_i)$, for some non-linear functions.

Deep neural networks (DNN) are flexible models that can describe complicated functions. In DNNs, the inputs go through several layers of non-linear transformations. We use these models to compute the non-linearities that generate from $\mathbf{x}_i$ the inducing points, $\mathbf{Z}(\mathbf{x}_i)$, and the means and covariances of $q(\mathbf{u}|\mathbf{x}_i)$, *i.e.*, $\mathbf{m}(\mathbf{x}_i)$ and $\mathbf{S}(\mathbf{x}_i)$. The architecture employed is displayed in Figure 1. At the output of the DNN we obtain $\mathbf{Z}$, a mean vector $\mathbf{m}$ and the Cholesky factor of the covariance matrix $\mathbf{S} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$. The maximization of the lower bound in (5) when using DNNs for the non-linearities is shown in Algorithm 1. In binary classification, the required expectations are computed using 1-dimensional quadrature, as in Hensman et al. (2015a).

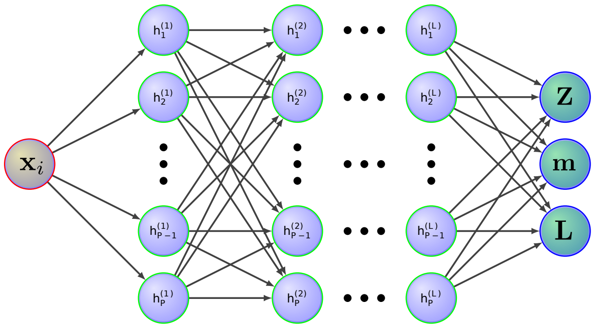

Figure 1. The network's inputs is $\tilde{\mathbf{x}}$. The outputs are the inducing points $\mathbf{Z}$, the mean $\mathbf{m}$ and the Cholesky factor, $\mathbf{L}$, of $q(\mathbf{u}|\mathbf{x}_i)$.

3.3. Predictions and Training Cost

As in Tran et al. (2021), at test time, instances are not randomly chosen. We set $p(\tilde{\mathbf{x}})$ to be a deterministic distribution placed on the candidate point $\mathbf{x}^\star$. The DNN is used to obtain $\mathbf{Z}$, and the parameters of $q(\mathbf{u}|\mathbf{x}^\star)$, $\mathbf{m}$ and $\mathbf{S}$. The predictive distribution for $f(\mathbf{x}^\star)$ is Gaussian with mean and variance:

$$m^\star = \mathbf{k}_{\mathbf{x}^\star,\mathbf{Z}}\mathbf{K}_{\mathbf{Z}}^{-1}\mathbf{m}\,, \tag{7}$$

$$s^\star = k^\star + \mathbf{k}_{\mathbf{x}^\star,\mathbf{Z}}\mathbf{K}_{\mathbf{Z}}^{-1}\left(\mathbf{S}-\mathbf{K}_{\mathbf{Z}}\right)\mathbf{K}_{\mathbf{Z}}^{-1}\mathbf{k}_{\mathbf{x}^\star,\mathbf{Z}}^{\mathrm{T}}\,. \tag{8}$$

Given this distribution for $f(\mathbf{x}^\star)$, the probability distribution for $y^\star$ can be computed in closed-form in regression problems and with 1-dimensional quadrature in binary classification. Initially, we only consider predictions at individual test points, as in Tran et al. (2021). This is enough in most learning applications. However, if test covariances are needed, a multi-variate Gaussian can also be computed using (7) and (8) (Titsias, 2009). In this case, however, there will be as many different predictive covariances as candidate test points since each test point will generate different $\mathbf{Z}$, $\mathbf{m}$ and $\mathbf{S}$. We simply average all the covariances corresponding to each $\mathbf{Z}$, $\mathbf{m}$ and $\mathbf{S}$. The mean is set equal to the mean of individual predictions. Empirically, this gives good results.

Algorithm 1 Training input dependent sparse GPs

```
Require: $\mathcal{D}$, $M$, neural network NNet with $L$ hidden layers
    and $P$ hidden units
Ensure: Optimal parameters of the model
  initialize NN's weights and kernel's parameters $\boldsymbol{\theta}$
  while stopping criteria is False do
    $\mathrm{Log}_{\mathrm{lk}} = 0$, $\mathrm{KL}_{\mathrm{div}} = 0$
    gather mini-batch $\mathbf{Mb}$ of size $n$ from $\mathcal{D}$
    for $(\mathbf{x_i}, \mathbf{y_i})$ in $\mathbf{Mb}$ do
      $(\mathbf{Z}_{\mathbf{x}_i}, \mathbf{m}_{\mathbf{x}_i}, \mathbf{L}_{\mathbf{x}_i})$ = NNet($\mathbf{x_i}$)
      $\mathrm{Log}_{\mathrm{lk}}$ += $\mathbb{E}_{q(f_i,\mathbf{u})}[\log p(y_i|f_i)]$
      $\mathrm{KL}_{\mathrm{div}}$ += $\mathrm{KL}[q(\mathbf{u}|\mathbf{x}_i)|p(\mathbf{u}|\mathbf{x}_i)]$
    end for
    ELBO $\leftarrow \frac{N}{n}\times \mathrm{Log}_{\mathrm{lk}} - \frac{1}{n}\times \mathrm{KL}_{\mathrm{div}}$
    Update parameters of the model using the gradient of
    ELBO
  end while
```

The cost of our method is smaller than the cost of a standard sparse GP if a smaller number of inducing points M is used. Given a mini-batch of size n, the cost of a DNN with L layers, P hidden units, d_i dimension of the input data, and output dimension d_o is $\mathcal{O}(nd_iP+nP^2L+nPd_o+n(L+1))$. The cost of the sparse GPs is $\mathcal{O}(nM^3)$. Therefore, the cost of our method per iteration is $\mathcal{O}(nd_iP + nP^2L + nPd_o + n(L+1) + nM^3)$. Since in our method the inducing points are input dependent, we expect to obtain good prediction results even for M values that are fairly small.

4. Related Work

Early works on sparse GPs simply chose a subset of the training data for inference based on an information criterion (Csató & Opper, 2002; Lawrence et al., 2003; Seeger et al., 2003; Henao & Winther, 2012). This approach is limited in practice and more advanced methods in which the inducing points need not be equal to the training points are believed to be superior. In the literature there are several works analyzing and studying sparse GP approximations based on inducing points. Some of these works include Quiñonero-Candela & Rasmussen (2005); Snelson & Ghahramani (2006); Naish-Guzman & Holden (2007); Titsias (2009); Bauer et al. (2016); Hernández-Lobato & Hernández-Lobato (2016). We focus here on a variational approach (Titsias, 2009) which allows for stochastic optimization (Hensman et al., 2013; 2015a). This enables learning in very large datasets with a cost of $\mathcal{O}(M^3)$ per iteration, with M the number of inducing points.

In some problems, however, several thousands of inducing points must be considered to get good prediction results (Hensman et al., 2015b; Shi et al., 2020; Tran et al., 2021). There is hence a need to improve the cost of sparse GPs, without losing expressive power. One work addressing this task is that of Cheng & Boots (2017). In that work it is proposed to decouple the process of inferring the posterior mean and variance, allowing to consider a different number of inducing points for each one. Importantly, the computation of the mean has a linear complexity, which allows to have more expressive posterior means at a lower cost. A disadvantage is that such an approach suffers from optimization difficulties. An alternative decoupled parameterization adopts an orthogonal basis in the mean Salimbeni et al. (2018a). Such a method can be considered as a specific case of Shi et al. (2020). There, the authors introduce a new interpretation of sparse variational approximations for GP using inducing points. For this, the GP is decomposed as a sum of two independent processes. This leads to tighter lower bounds on the marginal likelihood and new inference algorithms considering two different sets of inducing points. This enables using more inducing points at a linear cost.

Our work is closer to that of Tran et al. (2021). They also describe a mechanism to consider input dependent inducing points in the context of sparse GP. However, the difference is significant. In particular, in Tran et al. (2021) a very large set of inducing points M is considered initially. Then, for each input point, a subset of these inducing points is considered. This subset is obtained by finding the $H \ll M$ nearest inducing points to the current data instance $\mathbf{x}_i$. IDSGP does not require the large initial set, and uses a NN to output $\mathbf{Z}(\mathbf{x}_i)$. The approach suggested by Tran et al. (2021) significantly reduces the cost of the standard sparse GP described in Titsias (2009). However, it suffers from the difficulty of having to find the H nearest neighbors for each point in a mini-batch, which is expensive. Therefore, the final cost is higher than what would be thought initially. Our method is expected to be better because of the extra flexibility of the non-linear relation between $\mathbf{x}_i$ and $\mathbf{Z}$ given by the DNN. Furthermore, the DNN is expected to make a better use of GPU acceleration. Our proposed method, IDSGP, also amortizes the parameters of q. The approach of Tran et al. (2021) does not. Amortization has been shown to improve performance (Shu et al., 2018).

Wu et al. (2022) also consider a sparse approximation to GPs based on nearest neighbors. That work is different from IDGSP in that we do not change the prior for $\mathbf{f}$, which is the standard GP prior. In the work of Wu et al. (2022) they do approximate the GP prior. Moreover, they approximate $p(\mathbf{f}|\mathbf{u})$ using a factorizing distribution across the training instances. IDSGP does not need this approximation in $p(\mathbf{f}|\mathbf{u})$.

Another method to improve the training cost of GP is described by Wilson & Nickisch (2015); Evans & Nair (2018); Gardner et al. (2018). It consists in placing the inducing points on a grid. This allows to perform a fast computation exploiting the inducing points structure. One can easily consider values for M that are even larger than N. However, to get such benefits the inducing points need to be fixed due to the structure constraints. This may be detrimental in high-dimensional problems.

Instead of using inducing points, there are some works that scale GPs by approximating the posterior GP process using an inference network (Shi et al., 2019; Sun et al., 2019). An inference network receives some random noise and outputs function values for each input. Particular examples include among others Bayesian DNNs. Inference networks are expected to lead to flexible stochastic processes. However, it is difficult to enforce that the approximate posterior process looks similar to the prior GP in regions where there is no data. For this, approximate inference is carried out on a finite subset of points chosen at random from the input space. This is expected to lead to poor results in high-dimensional spaces. Moreover, another problem of using an inference network is that tuning the prior GP hyper-parameters is challenging and has often to be done in a separate step.

Amortized variational inference (Shu et al., 2018) has also been explored in the context of GPs in Villacampa-Calvo et al. (2021). There, input noise is considered in a multi-class problem and the performance of the final GP model is improved by amortizing the variational parameters of the posterior approximation for the noiseless inputs. A DNN that receives both $\mathbf{x}_i$ and y_i as an input is used for this task.

Other sparse GPs in the literature do not fully rely on inducing points, *e.g.*, (Tresp, 2000; Snelson, 2007; Gramacy & Apley, 2015). These techniques, however, cannot use, in general, stochastic optimization and do not scale to very large problems. Finally, sparse GPs, and our method, can benefit from natural-gradients (Salimbeni et al., 2018b). They could result in an orthogonal improvement.

5. Experiments

We evaluate the performance of our method, to which we refer to as Input Dependent Sparse GP (IDSGP). We consider regression and binary classification with a probit likelihood. In this later case, we approximate the expectation in the ELBO using 1-dimensional quadrature Hensman et al. (2015a). The code of IDSGP in Tensorflow 2.0 (Abadi et al., 2015) is given in the supplementary material. We compare results with the standard variational sparse GP (VSGP) (Hensman et al., 2013) and with two methods described in

Section 4. Namely, the sparse within sparse GP (SWSGP) proposed by Tran et al. (2021), and the sparse GP based on an orthogonal decomposition that considers two different sets of inducing points (Shi et al., 2020). We refer to this last method as SOLVE. All methods use a Matérn $3/2$ covariance function (Rasmussen & Williams, 2006) and estimate all hyper-parameters by maximizing the ELBO. The DNN architecture used in IDSGP is detailed in Appendix B.

5.1. Toy Problems

We show the posterior mean and standard deviation of each method on the 1-dimensional regression problem from (Snelson & Ghahramani, 2006). We compare results with a full GP. Figure 2 shows the results obtained, including the learned locations of the inducing points. In the case of IDSGP we show the locations of the inducing points for the point represented with a star at $x = 3.9$. The number of inducing points, for each method, are indicated in the figure's caption. We consider a small number of inducing points to study the benefits of having input dependent inducing points. IDSGP uses smaller number of inducing points than the other methods. The figure shows that, in regions with observed data, IDSGP's predictions look closer to those of the full GP. This is a consequence of the extra flexibility given by the NN. In regions with no data, the uncertainty increases and the predictions of IDSGP become similar to those of the GP prior, although the uncertainty is a bit smaller. This underestimation also happens, to some extent, in the other sparse GP methods. See the results of, *e.g.* SWSGP. Appendix F.1 has results for an increasing number of inducing points M. They show that as M increases, IDSGP becomes more and more similar to the full GP, also in regions with no observed data, as expected. Appendix F.2 shows results for the VSGP method when q is not optimized, as in Titsias (2009). They are very similar to the ones in Appendix F.1 for VSGP. Predicting the prior in regions with no data is not guaranteed in IDSGP. The predictions will depend on the particular output of the NN. However, Figure 2 shows that in IDSGP, as we move away from the observed data, the mean becomes closer to zero and the variance increases, which is the expected behavior. This could be due to the term $\text{KL}(q(\mathbf{u}|\tilde{\mathbf{x}})|p(\mathbf{u}|\tilde{\mathbf{x}}))$, which enforces that the NN output is similar to the prior.

Finally, Figure 3 shows the decision boundary of each method on the banana classification dataset (Hensman et al., 2013). We observe that IDSGP produces the most accurate boundaries. This is so, even though it uses a smaller number of inducing points M that the other methods. See the caption of the figure.

5.2. Experiments on UCI Datasets

We consider several regression and binary classification datasets extracted from the UCI repository (Dua & Graff,

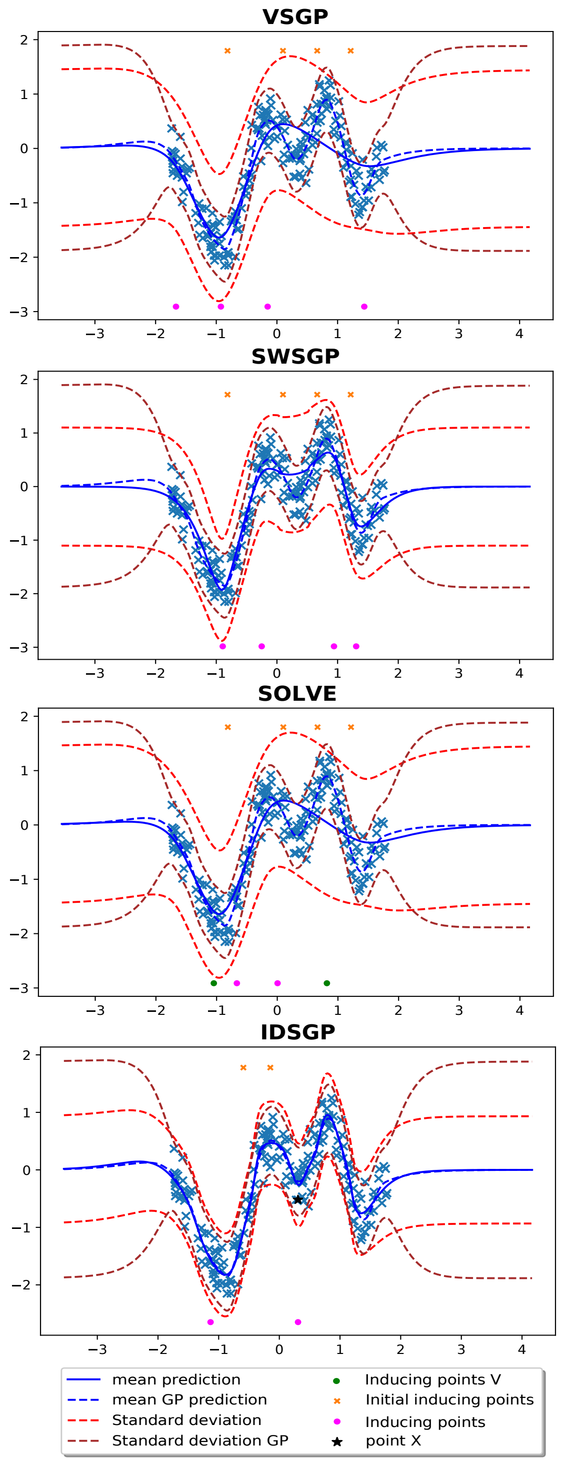

Figure 2. Initial and final inducing points are shown on the top and bottom of each figure. In IDSGP, the inducing points correspond to the point drawn with a star. The mean and std. deviation of full GP (each method) are shown with blue and brown dashed lines (solid blue and dashed red), respectively. In VSGP $M = 4$. In IDSGP $M = 2$ and the NN has 2 layers with 50 units. In SWSGP $M = 4$ and $H = 2$ neighbors. In SOLVE $M_1 = M_2 = 2$.

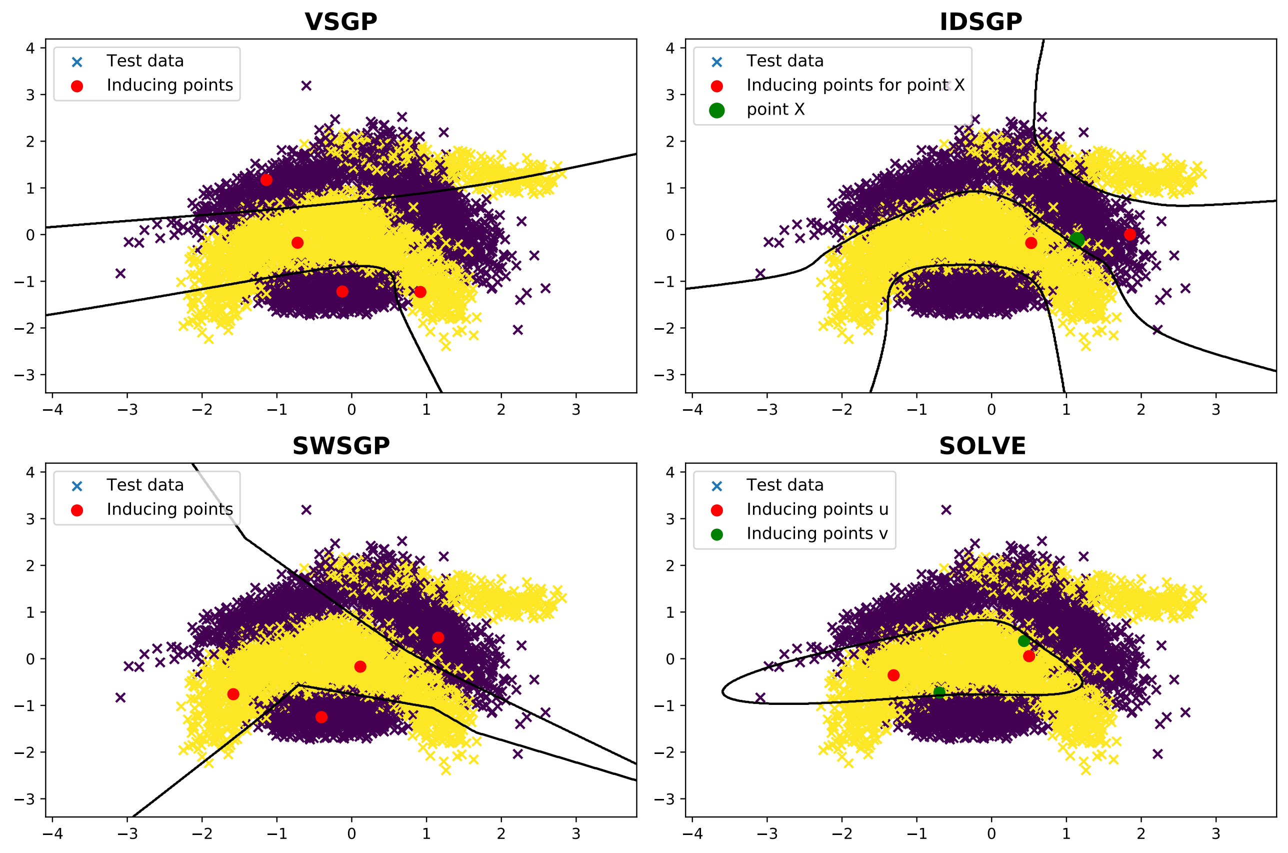

Figure 3. Banana classification data set with $N = 5300$ points. The final location of inducing points are shown inside the figures. For IDSGP, we show the location of inducing points related to the green colored point. VSGP with $M = 4$. IDSGP with $M = 2$ and a neural network with 2 hidden layers, each with 50 hidden units. SWSGP with $M = 4$ and 2 neighbors. SOLVE with $M_1 = M_2 = 2$.

2017). Namely, 8 regression datasets and 8 binary classification datasets (see Appendix D for the datasets' details). These datasets have large number of instances N so we use a large number of inducing points M for the non-input dependent methods, as in Shi et al. (2020). In SOLVE we use $M_1 = 1024$ and $M_2 = 1024$ inducing points. In VSGP we set $M = 1024$. In SWSGP we set $M = 1024$ and $H = 50$ neighbors. The number of inducing points of IDSGP is set to $M = 15$, which is significantly smaller. All the methods are trained using ADAM (Kingma & Ba, 2015) with a mini-batch size of 100 and a learning rate of 0.01. In the classification setting we use the same setup, but the number of inducing points of IDSGP is set even smaller. Namely, $M = 3$. All methods are trained on a Tesla P100 GPU with 16GB of memory. On each dataset we use 80% of the data for training and the rest for testing. We report results across 5 splits of the data since the datasets are already quite big.

The average negative test log-likelihood of each method on each dataset is displayed in Table 1, for the regression datasets, and in Table 2, for the classification datasets, respectively. The average rank of each method is also displayed at the last row of each table. The RMSE and prediction accuracy results are similar to those displayed here. They can be found in Appendix F.3 and F.4. We observe that in the regression datasets, the proposed method, IDSGP, obtains best results in 6 out of the 8 datasets. IDSGP also obtains the best average rank (closer to always performing best on each train / test data split). This is remarkable given that IDSGP uses a much smaller number of inducing points (*e.g.*, $M = 15$ for IDSGP vs. $M = 1024$ for VSGP). This due to the extra flexibility that the input dependent inducing points provide. In classification, however, all the methods seem to perform similar to each other and the differences between them are smaller. Again, IDSGP uses here a smaller number of $M = 3$ inducing points. We tried increasing M in IDSGP, but it does not improve the results obtained.

In these experiments we also measure the average training time per epoch, for each method. The results corresponding to the UCI regression datasets are displayed in Table 3. The results for the UCI classification datasets are found in

Table 1. Avg. neg. test log-likelihood values for the UCI regression datasets. The numbers in parentheses are standard errors. Best mean values are highlighted in bold face.

	VSGP	SOLVE	SWSGP	IDSGP
Kin40k	-0.047 (0.003)	-0.093 (0.005)	-0.110 (0.008)	**-1.461 (0.018)**
Protein	2.848 (0.002)	2.847 (0.002)	2.835 (0.003)	**2.775 (0.007)**
KeggDirected	-1.955 (0.013)	-2.031 (0.015)	-2.256 (0.013)	**-2.410 (0.014)**
KEGGU	-2.344 (0.013)	-2.421 (0.007)	-2.396 (0.007)	**-2.908 (0.050)**
3dRoad	3.691 (0.006)	3.681 (0.004)	3.879 (0.023)	**3.399 (0.010)**
Song	**3.613 (0.002)**	3.617 (0.002)	3.618 (0.003)	3.637 (0.003)
Buzz	6.272 (0.012)	6.284 (0.018)	**6.137 (0.007)**	6.317 (0.060)
HouseElectric	-1.737 (0.005)	-1.739 (0.004)	-1.711 (0.010)	**-1.774 (0.004)**
Avg. Ranks	2.925 (0.158)	2.525 (0.206)	2.700 (0.114)	**1.850 (0.177)**
# Ind. points	1024	1024 / 1024	(H=50) / 1024	15

Table 2. Avg. test neg. log-likelihood values for the UCI classification datasets. The numbers in parentheses are standard errors. Best mean values are highlighted in bold face.

	VSGP	SOLVE	SWSGP	IDSGP
MagicGamma	0.308 (0.004)	**0.307 (0.004)**	0.371 (0.005)	0.311 (0.002)
DefaultOrCredit	0.000 (0.000)	0.000 (0.000)	**0.000 (0.000)**	0.000 (0.000)
NOMAO	0.113 (0.004)	**0.112 (0.004)**	0.134 (0.004)	0.121 (0.004)
BankMarketing	0.206 (0.001)	**0.205 (0.001)**	0.304 (0.023)	0.209 (0.002)
Miniboone	0.151 (0.001)	**0.149 (0.001)**	0.180 (0.008)	0.153 (0.001)
Skin	0.005 (0.000)	0.004 (0.000)	0.006 (0.001)	**0.003 (0.000)**
Crop	0.003 (0.000)	0.003 (0.000)	**0.002 (0.000)**	0.003 (0.000)
HTSensor	0.003 (0.001)	**0.001 (0.000)**	0.030 (0.009)	0.005 (0.001)
Avg. Ranks	2.525 (0.129)	**1.750 (0.147)**	3.075 (0.204)	2.650 (0.158)
# Ind. points	1024	1024 / 1024	(H=50) / 1024	3

Appendix F.4. They look very similar to ones reported here. We observe that the fastest method is IDSGP. The speed-up obtained is very high even though there is some overhead of having to compute the output of the DNN and update its parameters. IDSGP also results in fastest prediction times than VSGP, SOLVE or SWSGP. See Appendix F.3 and F.4.

5.3. Large Scale Datasets

We consider two very large datasets. The first dataset is the Airlines Delay binary classification dataset, as described in Hernández-Lobato & Hernández-Lobato (2016), with $N = 2,127,068$ data instances and $d = 8$ attributes. The second dataset is the regression Yellow taxi dataset, as described in Salimbeni & Deisenroth (2017), with $N = 1$ billion data-points and $d = 9$ attributes. In each dataset we use a test set of $10,000$ instances chosen at random. The number of inducing points in IDSGP is set to $M = 50$. In the other methods, we use the same number of inducing points as in the previous section. The mini-batch size is set to 100. Training is also performed on the same GPU as in the previous section. The ADAM learning rate is set to 0.001.

The average negative test log-likelihood of each method is displayed in Figure 4, for each dataset. We report performance in terms of the training time, in a $\log_{10}$ scale. The results corresponding to the RMSE are very similar to the ones displayed here. They can be found in Appendix F.5. We observe that the proposed method IDSGP performs best on each dataset. In particular, it obtains a better performance in a smaller computational time. We believe this due to using a smaller number of inducing points, and also because of the extra flexibility of the NN that can specify an input-dependent location of the inducing points.

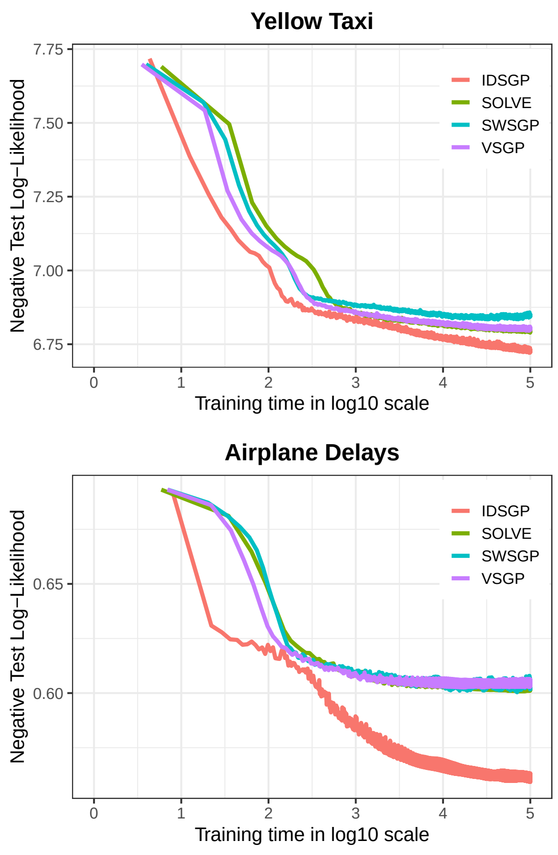

Figure 4. Negative log-likelihood on the test set for each method as a function of the training time in seconds, in $\log_{10}$ scale, for the Yellow taxi and the Airline delays datasets. Best seen in color.

5.4. Joint Predictive Covariances

We evaluate each method when computing a joint predictive distribution. In IDSGP, given a test mini-batch on which to compute the joint predictive distribution, we get the covariances obtained when each point from the mini-batch is an input to the NN. Recall from Section 3.3 that there are different predictive covariances for each test point. We get the final covariances by averaging them across all test points in the mini-batch. The mean for each test point is equal to mean computed for individual predictions. We also compare results with SWSGP, using the approach described in Tran et al. (2021). That is, we train the model using the union of nearest neighbors for the points within a mini-batch, so that the effective number of inducing points is at most 128, and with a total number of inducing points M of 512. We compute the average multivariate neg. test log-likelihood across test mini-batches of the same size as in training. We consider the *Kin40k* dataset and report average results across 5 training/test splits, as in Section 5.2. We also evaluate here SOLVE with $M_1 = 512$ and $M_2 = 512$.

Table 3. Average training time per epoch across the 5 splits for the UCI regression datasets. The numbers in parentheses are standard errors. Best mean values are highlighted.

	Kin40k	Protein	KeggDirected	KEGGU	3dRoad	Song	Buzz	HouseElectric
VSGP	591.7 (0.58)	737.2 (1.16)	932.7 (2.56)	1128.1 (3.78)	7880.9 (66.79)	9777.7 (42.84)	9901.0 (146.07)	32784.2 (190.18)
SOLVE	1739.3 (0.45)	2015.9 (0.66)	2357.3 (1.70)	2909.1 (1.19)	19567.1 (10.34)	23196.6 (98.35)	25769.5 (20.12)	92214.9 (452.18)
SWSGP	875.7 (0.68)	1023.5 (0.35)	1220.6 (1.89)	1458.0 (5.57)	10203.4 (12.03)	12241.7 (62.01)	13371.5 (12.34)	46163.3 (427.23)
IDSGP	**190.3 (0.75)**	**371.5 (1.25)**	**533.0 (1.73)**	**693.7 (5.77)**	**4070.1 (201.09)**	**4296.5 (25.03)**	**3640.4 (33.36)**	**16352.2 (90.15)**

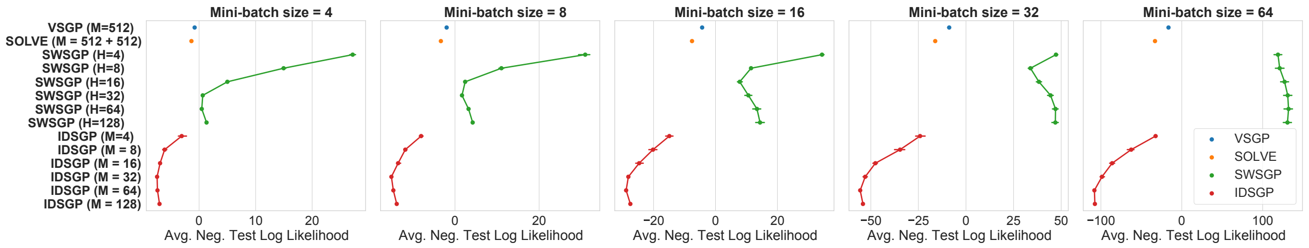

Figure 5. Average joint predictive negative log-likelihood on the test set for SVGP, IDSGP, SOLVE and SWSGP for an increasing mini-batch size (for training and testing). We report the average the multivariate NLL across test mini-batches and a 95% CI.

Figure 5 shows the results obtained for an increasing size of the mini-batch used for training and testing. Moreover, in the case of SWSGP and IDSGP, we show results for an increasing number of neighbors H and inducing points M. Horizontal lines represent a 95% confidence interval. We observe that IDSGP gives good results, specially for batch sizes bigger than 4. Also, the performance is better as we increase M. SWSGP gives worse results than VSGP, probably because we are using at most 128 inducing points at each training step. Finally, SOLVE performs better than VSGP. These results confirm that IDSGP can provide accurate joint predictive distributions.

5.5. Comparison with a Simple Neural Network

Consider a Neural Network (NN) such as the one used in IDSGP. Assume it is trained via maximum likelihood to compute the parameters of a Gaussian predictive distribution. Appendix F.6 shows that such an approach will have high confidence in regions with no data, unlike IDSGP.

6. Conclusions

Gaussian processes (GPs) are flexible models for regression and classification. However, they have a cost of $\mathcal{O}(N^3)$ per iteration with N the number of training points. Sparse approximations based on $M \ll N$ inducing points reduce such a cost to $\mathcal{O}(M^3)$. A problem, however, is that in some situations a large number of inducing points have to be used, since they determine the flexibility of the resulting approximation. There is hence a need to reduce their training cost.

We have proposed a method that can improve the training time and the flexibility of sparse GP approximations. Namely, input dependent sparse GP (IDSGP). IDSGP uses a deep neural network (DNN) to output specific inducing points for each point at which the predictive distribution of the GP needs to be computed. The DNN also outputs the parameters of the corresponding variational approximation on the inducing values associated to the inducing points. IDSGP can be obtained under a formulation that considers an implicit distribution for the input instance to the DNN. Importantly, such a formulation keeps intact the GP prior on the latent function values associated to the training points.

The extra flexibility provided by the DNN allows to significantly reduce the number M of inducing points used in IDSGP. Such a model provides similar or better results than other sparse GP approximations from the literature, at a smaller training cost. IDSGP has been evaluated on several regression and binary classification problems. The results obtained show that it improves the quality of the predictive distribution and reduces the training cost. Better results are most of the times obtained in regression problems. In classification problems, however, the performances obtained are similar to those of the state-of-the-art. Nevertheless, the training and prediction times are always shorter. The scalability of IDSGP is also illustrated on massive datasets of up to 1 billion points. There, IDSGP also obtains better results than alternative sparse GP approximations at a smaller training cost. IDSGP also provides joint predictive distributions that are better in terms of the test log-likelihood than those of the other state-of-the-art sparse GP approximations.

Acknowledgments

The authors acknowledge the use of the facilities of Centro de Computacion Cientifica (CCC) at Universidad Autónoma de Madrid. They also acknowledge financial support from Spanish Plan Nacional I+D+i, PID2019-106827GB-I00.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Bauer, M., van der Wilk, M., and Rasmussen, C. E. Understanding probabilistic sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems 29*, pp. 1533–1541. 2016.

Cheng, C.-A. and Boots, B. Variational inference for Gaussian process models with linear complexity. In *Advances in Neural Information Processing Systems*, pp. 51845194, 2017.

Csató, L. and Opper, M. Sparse on-line Gaussian processes. *Neural Computation*, 14:641–668, 2002.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Evans, T. and Nair, P. Scalable Gaussian processes with grid-structured eigenfunctions (GP-GRIEF). In *International Conference on Machine Learning*, pp. 1417–1426, 2018.

Foong, A., Li, Y., Hernández-Lobato, J., and Turner, R. 'In-between' uncertainty in Bayesian neural networks. In *ICML 2019 Workshop on Uncertainty and Robustness in Deep Learning*, 2019.

Gal, Y. *Uncertainty in deep learning*. PhD thesis, University of Cambridge, 2016.

Gardner, J., Pleiss, G., Wu, R., Weinberger, K., and Wilson, A. G. Product kernel interpolation for scalable Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 1407–1416, 2018.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Gramacy, R. B. and Apley, D. W. Local Gaussian process approximation for large computer experiments. *Journal of Computational and Graphical Statistics*, 24:561–578, 2015.

Henao, R. and Winther, O. Predictive active set selection methods for Gaussian processes. *Neurocomputing*, 80: 10–18, 2012.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence*, pp. 282290, 2013.

Hensman, J., Matthews, A., and Ghahramani, Z. Scalable variational Gaussian process classification. In *International Conference on Artificial Intelligence and Statistics*, pp. 351–360, 2015a.

Hensman, J., Matthews, A. G., Filippone, M., and Ghahramani, Z. MCMC for variationally sparse Gaussian processes. In *Advances in Neural Information Processing Systems 28*, pp. 1648–1656. 2015b.

Hernández-Lobato, D. and Hernández-Lobato, J. M. Scalable Gaussian process classification via expectation propagation. In *Artificial Intelligence and Statistics*, pp. 168–176, 2016.

Kingma, D. P. and Ba, J. ADAM: a method for stochastic optimization. In *Inrernational Conference on Learning Representations*, pp. 1–15, 2015.

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

Lawrence, N., Seeger, M., and Herbrich, R. Fast sparse Gaussian process methods: The informative vector machine. In *Neural Information Processing Systems*, pp. 609–616, 2003.

Morales-Alvarez, P., Hernández-Lobato, D., Molina, R., and Hernández-Lobato, J. M. Activation-level uncertainty in deep neural networks. In *International Conference on Learning Representations*, 2021.

Naish-Guzman, A. and Holden, S. The generalized FITC approximation. *Advances in neural information processing systems*, 20:1057–1064, 2007.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

Salimbeni, H. and Deisenroth, M. Doubly stochastic variational inference for deep Gaussian processes. In *Advances in Neural Information Processing Systems*, pp. 4588–4599, 2017.

Salimbeni, H., Cheng, C.-A., Boots, B., and Deisenroth, M. Orthogonally decoupled variational Gaussian processes. In *Neural Information Processing Systems*, pp. 8725–8734, 2018a.

Salimbeni, H., Eleftheriadis, S., and Hensman, J. Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. In *International Conference on Artificial Intelligence and Statistics*, pp. 689–697, 2018b.

Seeger, M. W., Williams, C. K. I., and Lawrence, N. D. Fast forward selection to speed up sparse Gaussian process regression. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, pp. 254–261, 2003.

Shi, J., Khan, M. E., and Zhu, J. Scalable training of inference networks for Gaussian-process models. In *International Conference on Machine Learning*, pp. 5758–5768. PMLR, 2019.

Shi, J., Titsias, M., and Mnih, A. Sparse orthogonal variational inference for Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 1932–1942, 2020.

Shu, R., Bui, H. H., Zhao, S., Kochenderfer, M. J., and Ermon, S. Amortized inference regularization. In *Advances in Neural Information Processing Systems*, pp. 4393–4402, 2018.

Snelson, E. and Ghahramani, Z. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pp. 1257–1264, 2006.

Snelson, E.and Ghahramani, Z. Local and global sparse Gaussian process approximations. In *International Conference on Artificial Intelligence and Statistics*, pp. 524–531, 2007.

Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational Bayesian neural networks. In *International Conference on Learning Representations*, 2019.

Titsias, M. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pp. 567–574, 2009.

Tran, G., Milios, D., Michiardi, P., and Filippone, M. Sparse within sparse Gaussian processes using neighbor information. In *International Conference on Machine Learning*, pp. 10369–10378, 2021.

Tresp, V. A Bayesian committee machine. *Neural Computation*, 12:2719–2741, 2000.

Villacampa-Calvo, C., Zaldívar, B., Garrido-Merchán, E. C., and Hernández-Lobato, D. Multi-class Gaussian process classification with noisy inputs. *Journal of Machine Learning Research*, 22:1–52, 2021.

Wilson, A. G. and Nickisch, H. Kernel interpolation for scalable structured Gaussian processes (KISS-GP). In *International Conference on International Conference on Machine Learning*, pp. 1775–1784, 2015.

Wu, L., Pleiss, G., and Cunningham, J. Variational nearest neighbor Gaussian processes. In *International Conference on Machine Learning*, 2022.

A. Datasets Pre-Processing

All the datasets are publicly available. The UCI repository datasets can be downloaded from the repository (Dua & Graff, 2017). Yellow taxi dataset was preprocessed following Salimbeni & Deisenroth (2017) and downloaded from `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`, where we have used data records from year 2015. Similarly, the Airlines Delay dataset was preprocessed following Hernández-Lobato & Hernández-Lobato (2016) and was downloaded from `https://community.amstat.org/jointscsg-section/dataexpo/dataexpo2009`, keeping only the records from January 2008 to April 2008. All regression datasets have been standardized using scikit-learn's built-int StandardScaler class (Pedregosa et al., 2011), which removes the mean and scales to unit variance. Classification datasets have been standardized using RobustScaler, on the other hand.

B. Neural Network Architecture

About the choice of architecture for the DNN we have tried to keep it small in order to take more advantage of the computational gain of the amortized scheme. In particular, we used a 2 hidden-layer with 50 hidden units network for the toy problems in Section 5.1, a 1 layer with 50 hidden units network for the UCI datasets in Section 5.2 and a 2 layer with 25 hidden units for the large scale datasets in Section 5.3. We used sigmoid activation functions. Keeping the network small reduces the number of parameters to optimize making the optimization process easier. In all problems we are using fully-connected layers with batch normalization and no skip layers. Regarding the initialization of the weights, all were initialized using the Glorot initialization (Glorot & Bengio, 2010). In our experiments we did not exhaustively explore the DNN architecture. This choice of architecture and initialization was based on some preliminary tests done before running the experiments. This does not mean that this is the best possible configuration. We did not optimize the architecture of the neural network. In practical applications, we suggest to run some preliminary tests in order to choose a configuration that performs well. The main suggestion, however, is to keep the network small as the input dependence will make the model expressive enough to still get very good results.

C. Choosing the Number of Inducing Points

We have observed that our proposed method IDSGP performs well in general with a fairly small number of inducing points, much smaller than the number of inducing points used in the other methods. Namely, SOLVE, VSGP and SWSGP. This is probably related to the extra flexibility of having input-dependent inducing points in IDSGP. In very large datasets we recommend using around $M = 50$ inducing points. In medium-size regression datasets $M = 15$ inducing points seem enough. In medium-size binary classification datasets, however, a smaller number of inducing points is enough $M = 3$. We believe the reason is that binary classification problems require less complicated latent functions. We did not optimize the number of inducing points. In practical applications, we suggest to run some preliminary tests in order to choose a number of inducing points that performs well.

D. Details of the UCI Datasets

Tables 4 and 5 show the characteristics of the regression and binary classification datasets considered from the UCI repository in the main document. These tables show, for each problem, the number of instances N and the number of attributes d.

Table 4. Characteristics of the UCI regression datasets.

Dataset	**N**	**d**
Kin40k	32,000	8
Protein	36,584	9
KeggDirected	42,730	19
KEGGU	51,686	26
3dRoad	347,899	3
Song	412,276	90
Buzz	466,600	77
HouseElectric	1,639,424	6

Table 5. Characteristics of the UCI binary classification datasets.

Dataset	**N**	**d**
MagicGamma	15,216	10
DefaultOrCredit	24,000	30
NOMAO	27,572	174
BankMarketing	36,169	51
Miniboone	104,051	50
Skin	196,046	3
Crop	260,667	174
HTSensor	743,193	11

E. KL-Divergence Minimization

In this section we show that maximizing (5) effectively minimizes the KL-divergence between $q(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u})$ and $p(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u}|\mathbf{y})$. In particular,

$$\begin{aligned}
\text{KL}(q(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u})|p(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u}|\mathbf{y})) &= -\int q(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u})}{q(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u})} d\mathbf{x} d\mathbf{f} d\mathbf{u} + \text{const.} \\
&= -\int q(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u}|\tilde{\mathbf{x}})p(\tilde{\mathbf{x}})}{p(\mathbf{f}|\mathbf{u})q(\mathbf{u}|\tilde{\mathbf{x}})p(\tilde{\mathbf{x}})} d\mathbf{x} d\mathbf{f} d\mathbf{u} + \text{const.} \\
&= -\int q(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{u}|\tilde{\mathbf{x}})}{q(\mathbf{u}|\tilde{\mathbf{x}})} d\mathbf{x} d\mathbf{f} d\mathbf{u} + \text{const.} \\
&= -\int q(\mathbf{f}, \tilde{\mathbf{x}}, \mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} d\mathbf{x} d\mathbf{u} \\
&\quad + \int q(\mathbf{u}, \tilde{\mathbf{x}}) \log \frac{p(\mathbf{u}|\tilde{\mathbf{x}})}{q(\mathbf{u}|\tilde{\mathbf{x}})} d\mathbf{x} d\mathbf{u} + \text{const.} \\
&= -\mathbb{E}_q[\log p(\mathbf{y}|\mathbf{f})] + \mathbb{E}_{p(\tilde{\mathbf{x}})}[\text{KL}(q(\mathbf{u}|\tilde{\mathbf{x}})|p(\mathbf{u}|\tilde{\mathbf{x}}))] \\
&= -\mathcal{L} + \text{const.}\,,
\end{aligned} \tag{9}$$

where we have used that the posterior is equal to the joint $p(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u}, \mathbf{y})$ divided by a normalization constant, *i.e.*, the marginal likelihood. Moreover, $\mathcal{L}$ is simply the lower bound defined in (5). Therefore, maximizing $\mathcal{L}$ effectively leads to the minimization of the KL-divergence between $q(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u})$ and $p(\tilde{\mathbf{x}}, \mathbf{f}, \mathbf{u}|\mathbf{y})$.

F. Extra Experimental Results

In this section, we include some extra results that do not fit in the main manuscript. Namely, the RMSE in the test set results and prediction times for the UCI regression datasets, and the accuracy in the test set, training and prediction times for the UCI classification datasets. In both cases, the setup is the same as described in Section 5 and the results are similar that the ones obtained in terms of the negative test log likelihood and training times in that section. Finally, we include similar plots to those in Section 5.3 but in terms of the test RMSE for the Yellow Taxi dataset and in terms of the test classification error for the Airline Delays dataset.

F.1. Toy Regression Datasets

Our method looks more and more similar to the full GP as number of inducing points M increases. However, with a small number of inducing points, it gives similar results to those of the full GP and similar to the results obtained when more inducing points are considered, which does not happen in the other methods. This is probably due to the extra flexibility of the neural network. The figures below (Figures 7 to 9) show the results of each method on the toy regression problem as we increase the number of inducing points M. For $M = 128$ IDSGP gives almost the same results as VSGP.

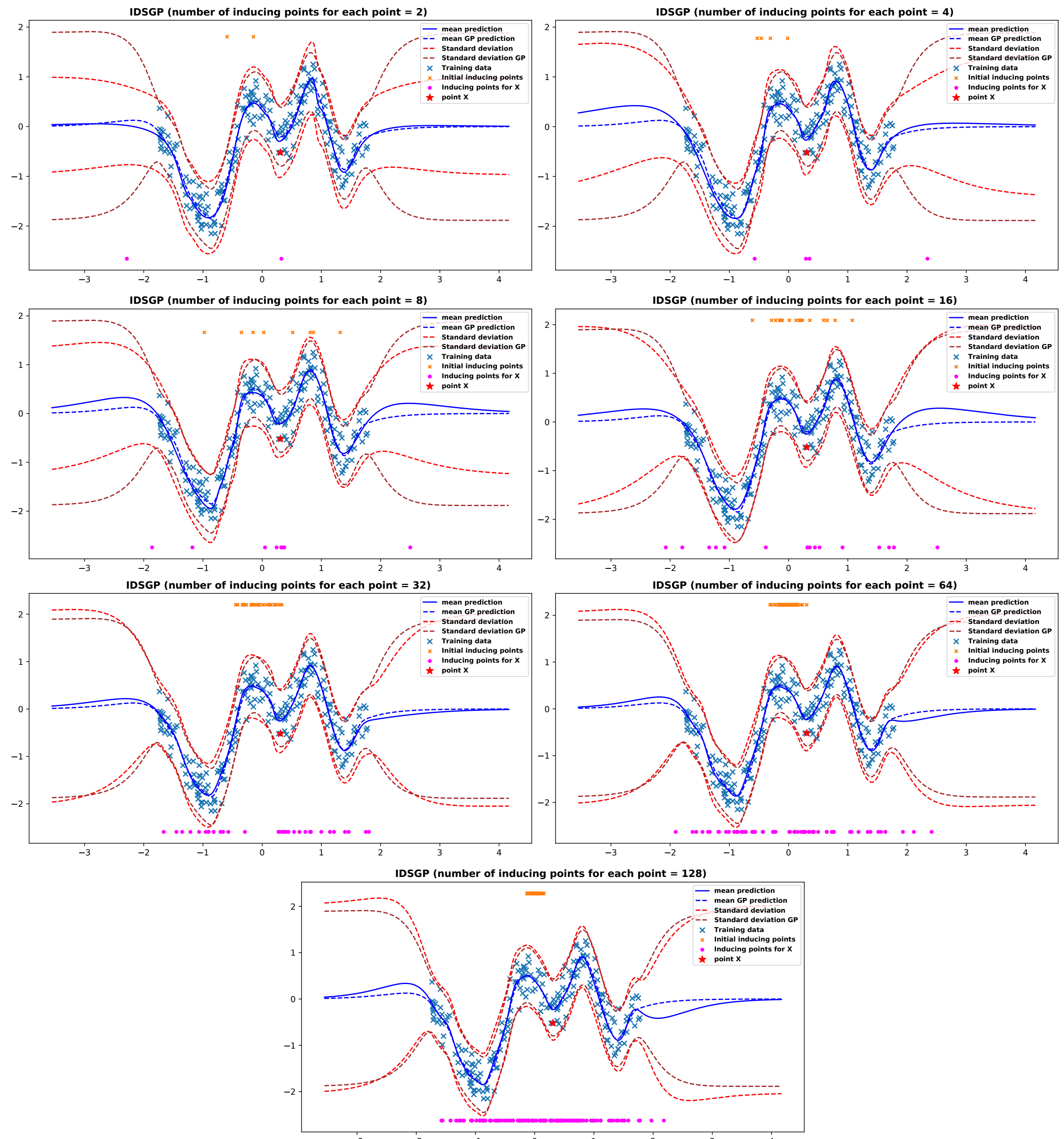

Figure 6. Toy regression example by varying number of inducing points $M_x = \{2, 4, 8, 16, 32, 64, 128\}$ with location of initial and final inducing points for an arbitrary selected point x from training sets. The mean and standard deviation of full GP prediction are shown with dashed blue and brown lines, respectively. The blue lines and the dashed red lines are the mean and standard deviation of IDSGP.

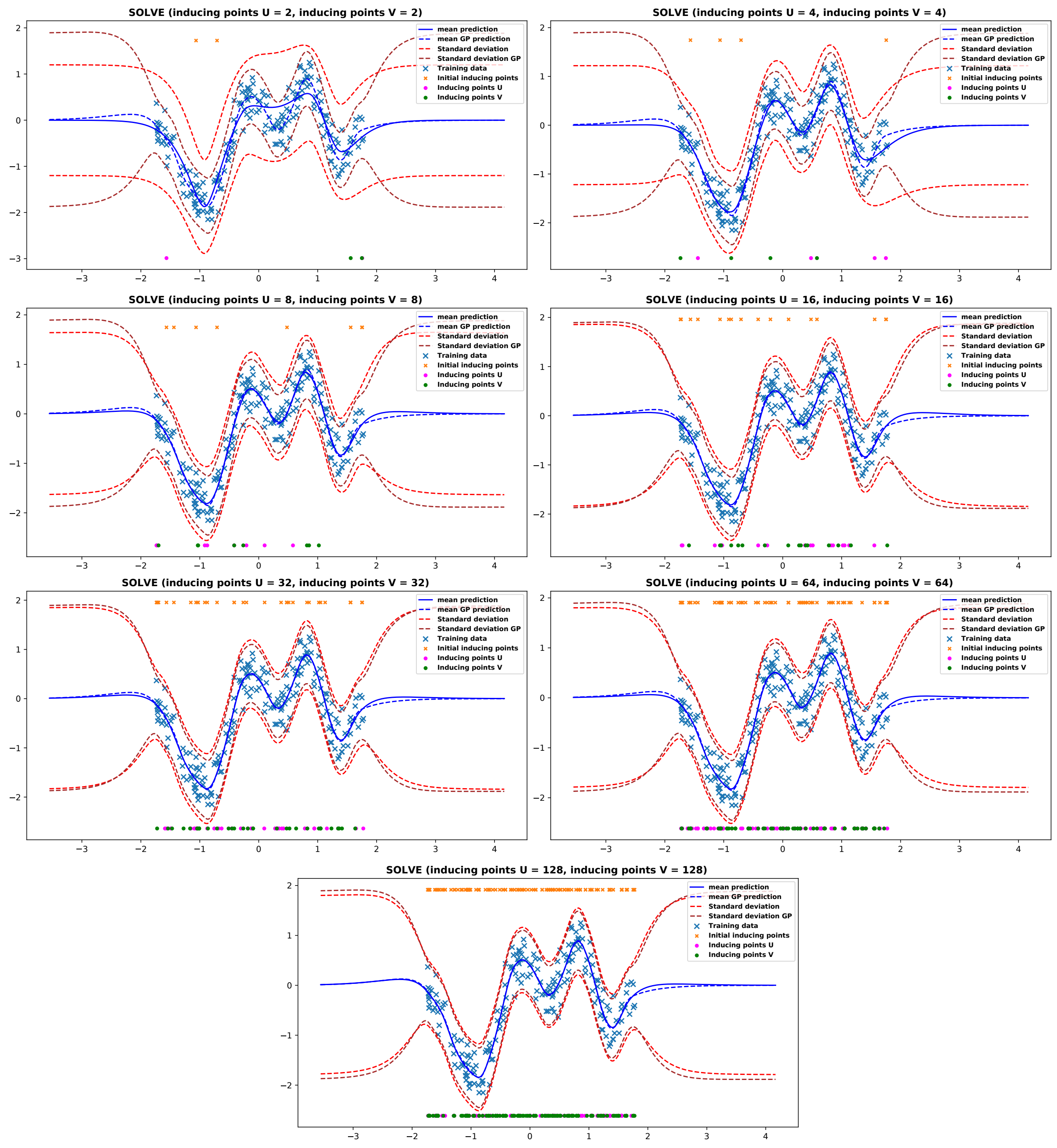

Figure 7. Toy regression example by varying number of inducing points $M_u, M_v = \{2, 4, 8, 16, 32, 64, 128\}$ with location of initial and final inducing points. The mean and standard deviation of full GP prediction are shown with dashed blue and brown lines, respectively. The blue lines and the dashed red lines are the mean and standard deviation of SOLVE.

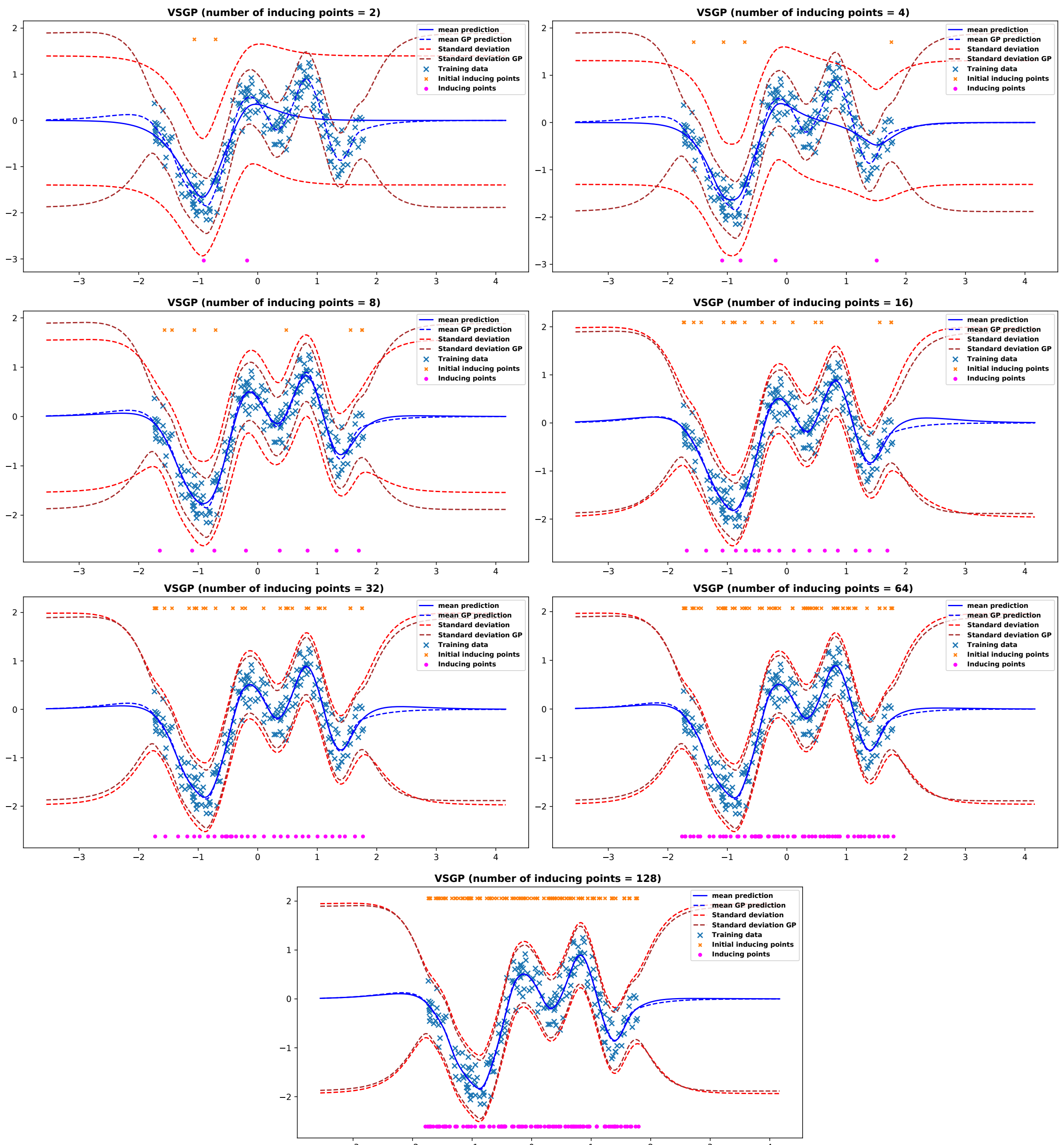

Figure 8. Toy regression example by varying number of inducing points $M = \{2, 4, 8, 16, 32, 64, 128\}$ with location of initial and final inducing points. The mean and standard deviation of full GP prediction are shown with dashed blue and brown lines, respectively. The blue lines and the dashed red lines are the mean and standard deviation of VSGP.

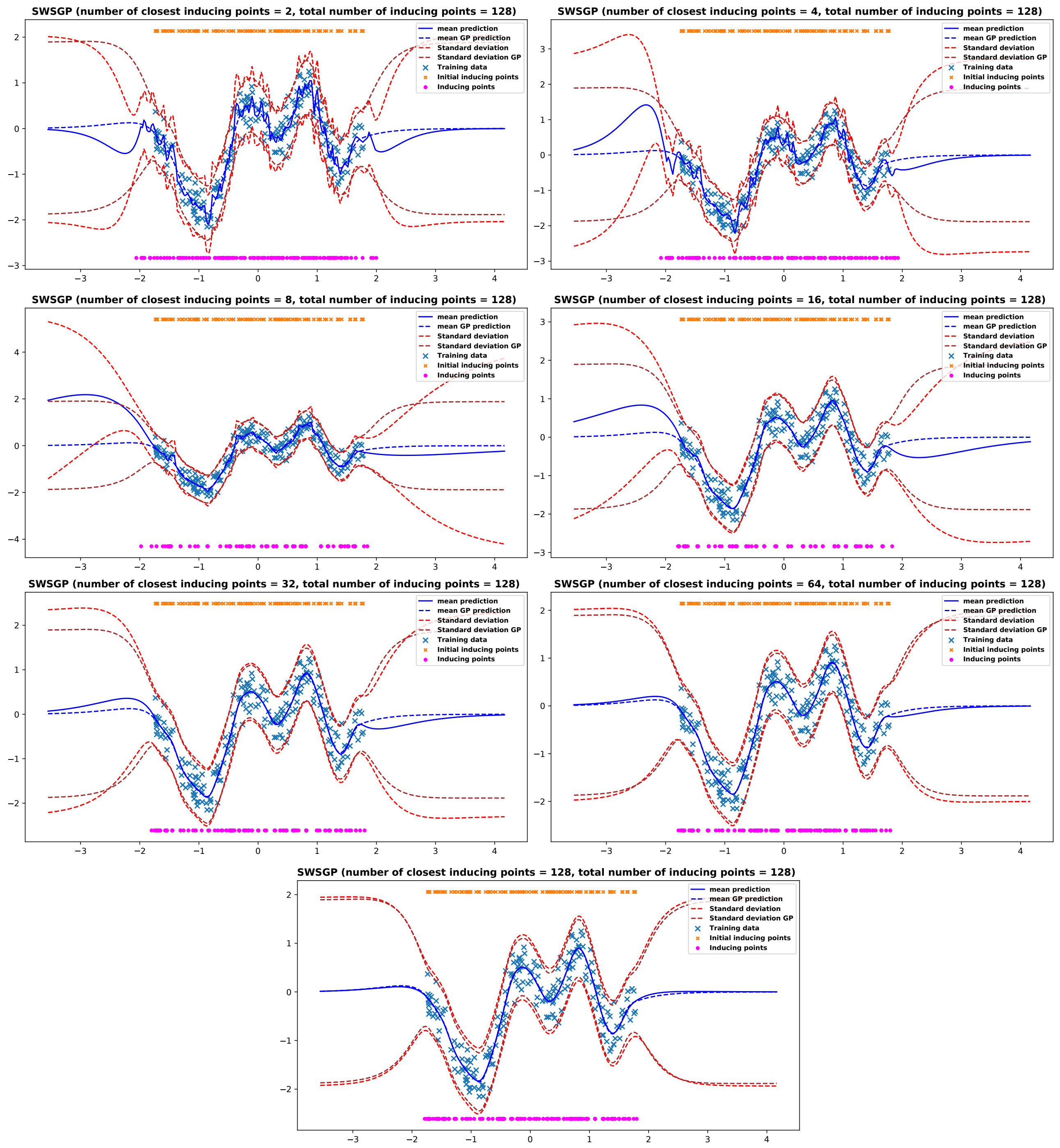

Figure 9. Toy regression example by varying number of the neighbor inducing points $M_c = \{2, 4, 8, 16, 32, 64, 128\}$ and total number of inducing points $M = 128$, with location of initial and final inducing points. The mean and standard deviation of full GP prediction are shown with dashed blue and brown lines, respectively. The blue lines and the dashed red lines are the mean and standard deviation of SWSGP.

F.2. Extra Results for the Toy Regression Experiment

Here we run the 1D toy regression experiment of Section 5.1 using the closed-form solution approach of Titsias (2009) for finding q. More precisely, this method is exactly the same as the SVGP method we compare results with, but where the approximate distribution q is not optimized at all. The reason for this is that it is possible to find a closed-form solution for q. However and importantly, the resulting method does not allow for mini-batch training. Since SVGP* does not allow for stochastic optimization, the batch size is set equal to the number of training points ($N = 200$). Figure 10 shows the fit obtained for an increasing number of inducing points M. The results are very similar to the ones of SVGP in Figure 8.

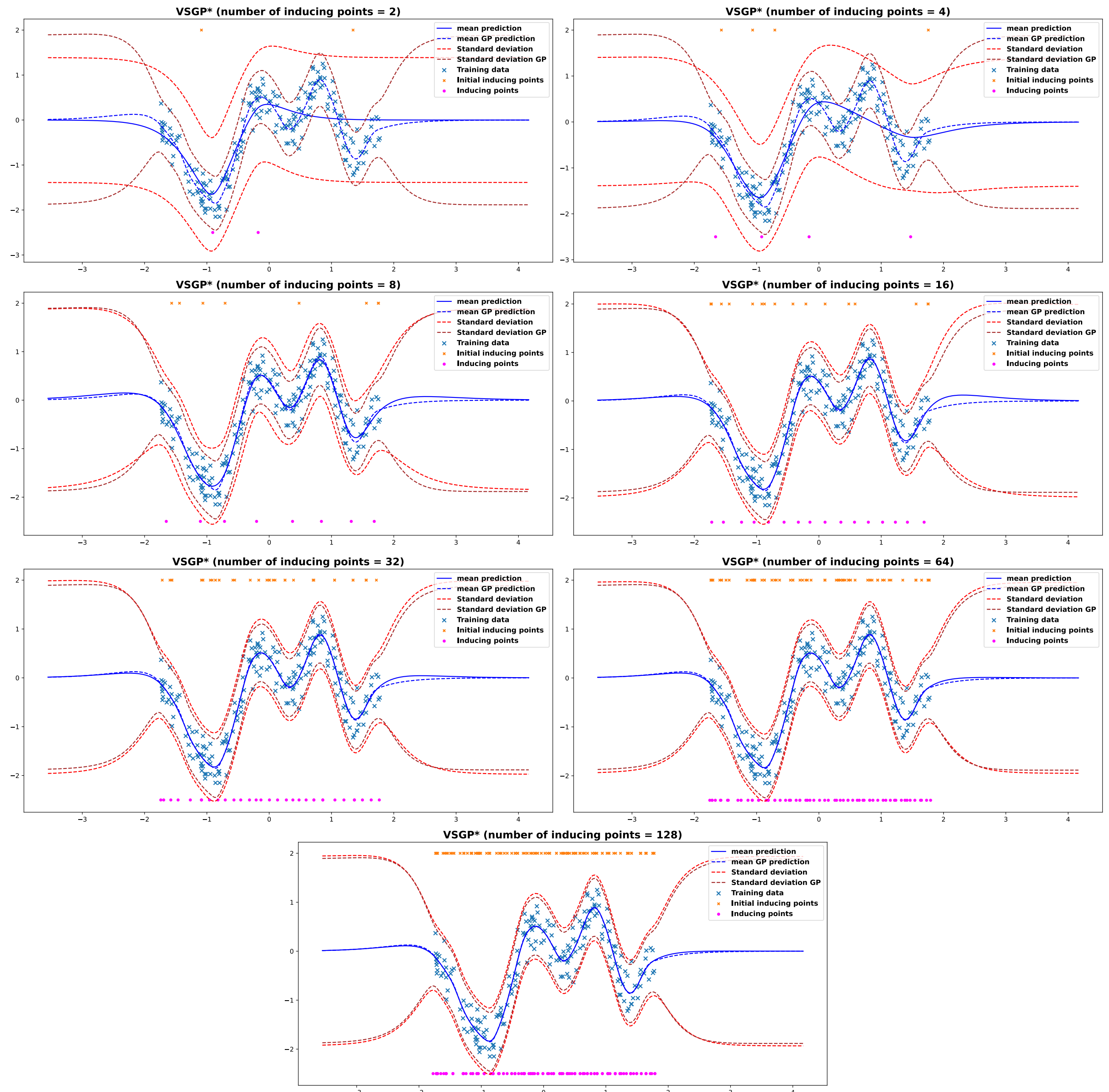

Figure 10. Toy regression example by varying number of inducing points $M = \{2, 4, 8, 16, 32, 64, 128\}$ with location of initial and final inducing points. The mean and standard deviation of full GP prediction are shown with dashed blue and brown lines, respectively. The blue lines and the dashed red lines are the mean and standard deviation of VSGP*.

F.3. UCI Regression Datasets

Table 6. Test Root Mean Squared Error (RMSE) values for the UCI regression datasets. The numbers in parentheses are standard errors. Best mean values are highlighted.

	N	d	VSGP	SOLVE	SWSGP	IDSGP
Kin40k	32,000	8	0.198 (0.002)	0.188 (0.002)	0.215 (0.002)	**0.050 (0.002)**
Protein	36,584	9	4.161 (0.010)	4.153 (0.007)	4.133 (0.007)	**3.756 (0.021)**
KeggDirected	42,730	19	0.032 (0.001)	0.030 (0.001)	0.024 (0.000)	**0.022 (0.001)**
KEGGU	51,686	26	0.024 (0.000)	0.022 (0.000)	0.022 (0.000)	**0.014 (0.000)**
3dRoad	347,899	3	9.641 (0.070)	9.559 (0.029)	11.726 (0.303)	**7.250 (0.072)**
Song	412,276	90	**8.966 (0.025)**	9.007 (0.018)	9.013 (0.026)	9.068 (0.011)
Buzz	466,600	77	175.076 (17.772)	177.757 (17.992)	**160.744 (14.046)**	166.784 (15.661)
HouseElectric	1,639,424	6	0.035 (0.000)	0.034 (0.000)	0.036 (0.001)	**0.032 (0.000)**
Avg. Ranks			2.800 (0.169)	2.525 (0.203)	2.850 (0.116)	**1.825 (0.168)**
# Inducing points			1024	1024 / 1024	(H=50) / 1024	15

Table 7. Average prediction time per epoch across the 5 splits for the UCI regression datasets. The numbers in parentheses are standard errors. Best mean values are highlighted.

	Kin40k	Protein	KeggDirected	KEGGU	3dRoad	Song	Buzz	HouseElectric
VSGP	0.9(0.00)	1.1(0.0)	1.4(0.01)	1.7(0.01)	11.6(0.07)	14.5(0.08)	18.0(0.08)	48.3(0.12)
SOLVE	2.4(0.00)	2.8(0.0)	3.2(0.00)	4.0(0.00)	27.0(0.04)	31.8(0.19)	37.0(0.03)	127.7(0.43)
SWSGP	1.3(0.00)	1.5(0.0)	1.8(0.01)	2.1(0.01)	14.8(0.03)	17.6(0.02)	21.3(0.10)	66.2(0.88)
IDSGP	**0.3(0.00)**	**0.5(0.0)**	**0.7(0.00)**	**1.0(0.02)**	**5.7(0.26)**	**5.6(0.05)**	**8.1(0.08)**	**22.1(0.14)**

F.4. UCI Classification Datasets

Table 8. Test Accuracy values for the UCI classification datasets. The numbers in parentheses are standard errors. Best mean values are highlighted.

	N	d	VSGP	SOLVE	SWSGP	IDSGP
MagicGamma	15,216	10	0.876 (0.001)	0.876 (0.001)	0.867 (0.002)	**0.877 (0.002)**
DefaultOrCredit	24,000	30	**1.000 (0.000)**	**1.000 (0.000)**	**1.000 (0.000)**	1.000 (0.000)
NOMAO	27,572	174	0.956 (0.002)	0.957 (0.002)	**0.961 (0.001)**	0.955 (0.002)
BankMarketing	36,169	51	0.906 (0.001)	**0.907 (0.001)**	0.897 (0.001)	0.905 (0.001)
Miniboone	104,051	50	0.941 (0.001)	**0.941 (0.000)**	0.938 (0.000)	0.937 (0.001)
Skin	196,046	3	0.999 (0.000)	0.999 (0.000)	0.999 (0.000)	**0.999 (0.000)**
Crop	260,667	174	0.999 (0.000)	0.999 (0.000)	**0.999 (0.000)**	0.999 (0.000)
HTSensor	743,193	11	0.999 (0.000)	**1.000 (0.000)**	0.989 (0.003)	0.999 (0.000)
Avg. Ranks			2.513 (0.143)	2.938 (0.143)	**2.125 (0.188)**	2.425 (0.169)
# Inducing points			1024	1024 / 1024	(H=50) / 1024	3

Table 9. Average training time per epoch across the 5 splits for the UCI classification datasets. The numbers in parentheses are standard errors. Best mean values are highlighted.

	Magic	DefaultOrCredit	NOMAO	BankMarket	Miniboone	Skin	Crop	HTSensor
VSGP	3105(459)	4759(516)	4445(549)	6231(862)	18447(1279)	37835(7065)	49962(9292)	115463(17511)
SOLVE	5154(1061)	7554(1039)	6718(1028)	8949(1635)	37022(7902)	64606(13314)	88819(18864)	168709(21194)
SWSGP	1547(145)	2354(182)	2728(188)	3682(351)	10040(347)	20283(2796)	21770(3038)	67687(5880)
IDSGP	**1143(100)**	**1293(90)**	**2026(94)**	**2987(354)**	**7654(134)**	**15700(1918)**	**21378(2561)**	**53895(5652)**

Table 10. Average prediction time per epoch across the 5 splits for the UCI classification datasets. The numbers in parentheses are standard errors. Best mean values are highlighted.

	MagicGamma	DefaultOrCredit	NOMAO	BankMarketing	Miniboone	Skin	Crop	HTSensor
VSGP	3.6(0.56)	4.1(0.59)	4.4(0.74)	9.5(4.39)	17.9(1.84)	49.7(11.15)	58.7(12.82)	139.7(41.67)
SOLVE	4.0(0.85)	5.4(0.73)	3.4(0.76)	4.9(1.24)	49.2(17.83)	48.8(16.73)	86.8(36.26)	93.5(15.73)
SWSGP	3.0(0.43)	3.9(0.38)	4.3(0.51)	5.4(0.72)	16.4(1.82)	36.0(8.00)	**33.9(6.25)**	88.4(9.16)
IDSGP	**2.5(0.24)**	**2.5(0.21)**	**3.5(0.39)**	**4.8(0.54)**	**13.9(0.75)**	**26.1(4.92)**	37.5(4.96)	**83.4(8.23)**

F.5. Large Scale Datasets

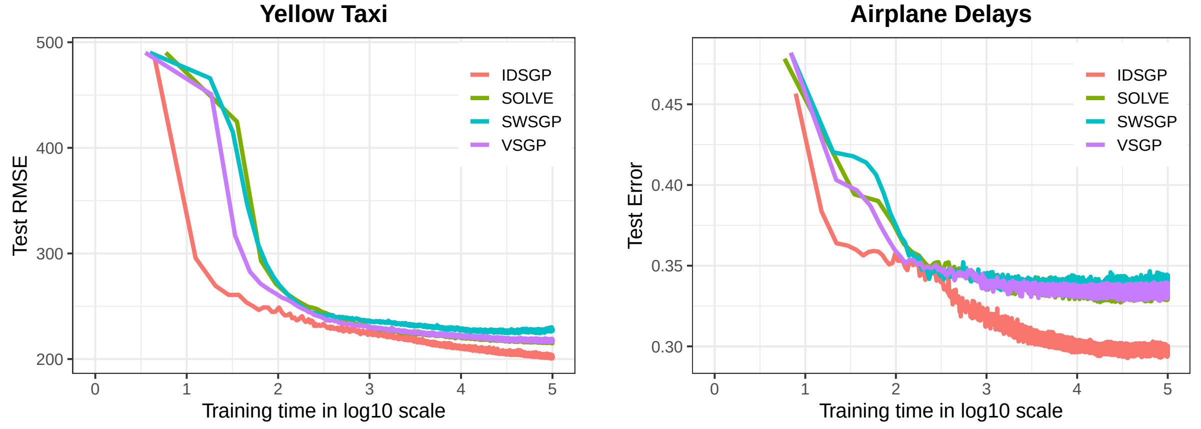

Figure 11. (left) Test RMSE for each method as a function of the training time in seconds, in $\log_{10}$ scale, for the Yellow taxi dataset. (right) Prediction error on the test set for each method as a function of the training time in seconds, in $\log_{10}$ scale, for the Airlines Delays dataset. Best seen in color

F.6. Neural Network Trained via Maximum Likelihood

We compare our method, IDSGP, with a neural network (NN) model trained via maximum likelihood. The architecture of the NN is the same as the one of the network used in the proposed method IDSGP in the UCI regression experiments. Training is done using ADAM with a learning rate of 0.001. The mini-batch size is the same than for the GP-based methods. We use the NN to predict the mean and variance of the Gaussian predictive distribution.

A NN approach is expected to be limited by the fact that it does not consider epistemic uncertainty. This results in that the NN can be very confident in regions with no data, which is contrary to what one should expect. On the other hand, GP-based approaches do not suffer from this problem since, in regions far from the observed data, the predictive distribution is expected to be similar to the prior. This behavior happens in GPs because the covariances are expected to be small in regions where observed data is far away.

To confirm this, we compare the quality of the predictive distribution of IDSGP and the neural network approach in in-between data (Morales-Alvarez et al., 2021; Foong et al., 2019). That is, in a region between two clusters of observed points. Standard train/test splits are not adequate to estimate the quality of the predictive distribution in regions with no data Morales-Alvarez et al. (2021); Foong et al. (2019). Gap splits are preferred. In gap splits one sorts the data across each dimension (there is one split per dimension) to then use for training the first $1/3$ of the instances and the last $1/3$ of the instances. The middle $1/3$ of the instances are left aside for testing.

We have evaluated the NN approach and IDSGP on the Energy dataset extracted from the UCI repository, using gap splits. It is well known that this dataset requires sensible in-between data uncertainty estimation to get good prediction results when using gap splits (Morales-Alvarez et al., 2021). The results are summarized in Table 11. We observe that the NN approach and IDSGP have similar RMSE test values when using gap splits. However, the negative test log-likelihood of the NN approach is much worse. Because the RMSE is similar, the explanation is that the NN approach is providing a less sensible estimation of the prediction uncertainty. This confirms that the NN approach underestimates the predictive variance in regions with no observed data.

Table 11. Avg. neg. test log-likelihood and RMSE on the Energy dataset with gap splits for IDSGP and the neural network. The numbers in parentheses are standard errors. Best results are highlighted in bold-face.

	IDSGP	NN
NLL	**3.568 (0.69)**	167.497 (105.80)
RMSE	5.117 (1.38)	**4.795 (1.22)**

F.7. Neural network architecture

We evaluate the robustness of the IDSGP by adding more layers and increasing the number of hidden units. Figure 12 shows the performance of IDSGP using two layered neural network and 25, 50, 100 and 200 hidden units. Figure 13 shows the performance of IDSGP where the number of hidden layers is fixed to be three and we change the number of hidden units to be 25, 50, 100, and 200. The number of inducing points are set to be 64 in all these experiments.

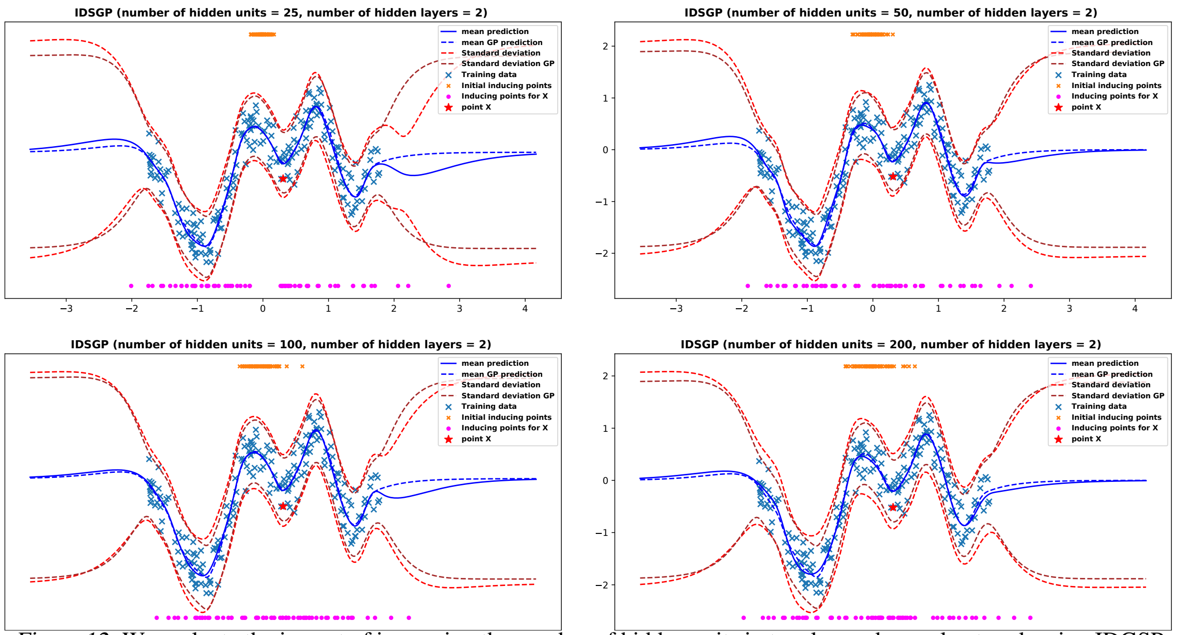

Figure 12. We evaluate the impact of increasing the number of hidden units in two layered neural network using IDGSP.

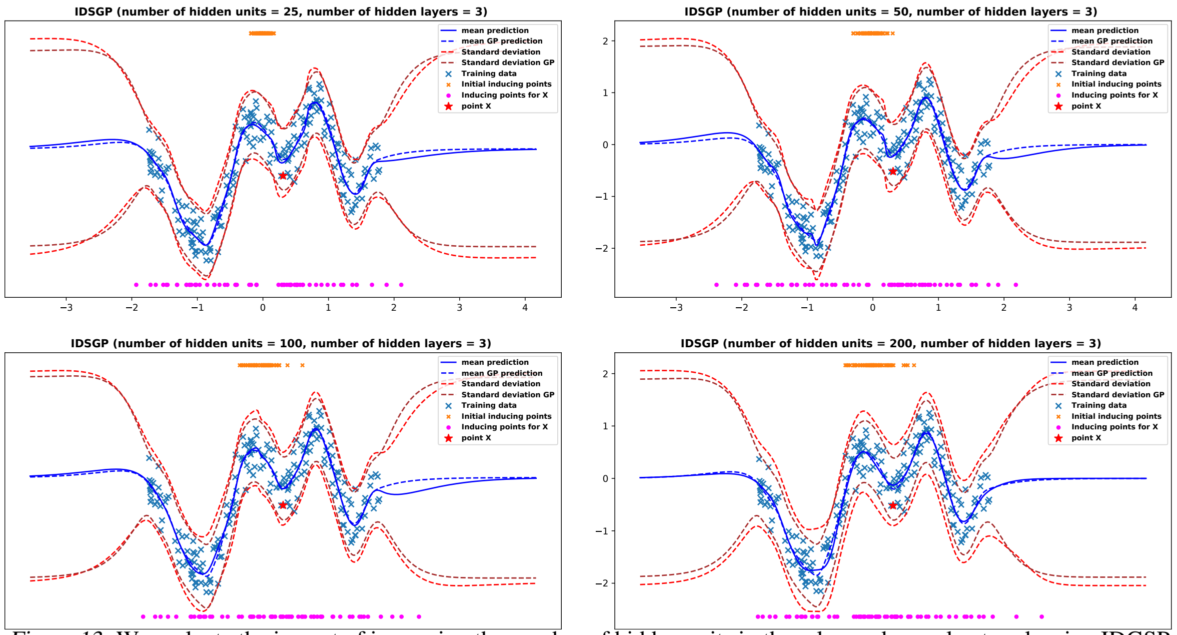

Figure 13. We evaluate the impact of increasing the number of hidden units in three layered neural network using IDGSP.