

# Nested Variational Compression in Deep Gaussian Processes

James Hensman and Neil D. Lawrence

December 4, 2014

## Abstract

Deep Gaussian processes provide a flexible approach to probabilistic modeling of data using either supervised or unsupervised learning. For tractable inference approximations to the marginal likelihood of the model must be made. The original approach to approximate inference in these models used variational compression to allow for approximate variational marginalization of the hidden variables leading to a lower bound on the marginal likelihood of the model [Damianou and Lawrence, 2013]. In this paper we extend this idea with a *nested* variational compression. The resulting lower bound on the likelihood can be easily parallelised or adapted for stochastic variational inference.

## 1 Introduction

Gaussian process (GP) models provide flexible non-parametric probabilistic approaches to function estimation in an analytically tractable manner. However, their tractability comes at a price: they can only represent a restricted class of functions. Gaussian processes came to the attention of the machine learning community through the PhD thesis of Radford Neal, later published in book form [Neal, 1996]. At the time there was still a great deal of interest in the result that a neural network with a single layer and an infinite number of hidden units was a universal approximator [Hornik et al., 1989], but Neal was able to show that in such a limit the model became a Gaussian process with a particular covariance function (the form of the covariance function was later derived by Williams [1998]). Both Neal [1996] and MacKay [1998] pointed out some of the limitations of priors that ensure joint Gaussianity across observations and this has inspired work in moving beyond Gaussian processes [Wilson and Adams, 2013].

### 1.1 Process Composition

Deep Gaussian processes [Lawrence and Moore, 2007, Damianou et al., 2011, Lázaro-Gredilla, 2012, Damianou and Lawrence, 2013] are an attempt to address this limitation.

In a deep Gaussian process, rather than assuming that a data observation,  $\mathbf{y}$ , is a draw from a Gaussian process prior,

$$\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon},$$

where  $f(\mathbf{x})$  is a vector-valued function drawn from a Gaussian process and  $\boldsymbol{\epsilon}$  is a noise corruption, we make use of a functional composition,

$$\mathbf{y} = \mathbf{f}_\ell(\mathbf{f}_{\ell-1}(\dots \mathbf{f}_1(\mathbf{x}))) + \boldsymbol{\epsilon},$$

where we assume each function in the composition,  $\mathbf{f}_i(\cdot)$ , is itself a draw from a Gaussian process. In other words we develop our full probabilistic model through *process composition*.

Process composition has the appealing property of retaining the theoretical qualities of the underlying stochastic process (such as Kolmogorov consistency) whilst providing a richer class of process priors. For example, for deep Gaussian processes Duvenaud et al. [2014] have shown that, for particular assumptions of covariance function parameters, the derivatives of functions sampled from the process have a marginal distribution that is heavier tailed than a Gaussian. In contrast, it is known that in a standard Gaussian process the derivatives of functions sampled from the process are jointly Gaussian with the original function.

## 2 Deep Models

Process composition in Gaussian processes has become known as *deep Gaussian processes* due to the relationship between these models and deep neural network models. A single layer neural network has the following form,

$$\mathbf{g}(\mathbf{x}) = \mathbf{V}^\top \boldsymbol{\phi}(\mathbf{U}\mathbf{x})$$

where  $\boldsymbol{\phi}(\cdot)$  is a vector valued function of an adjustable basis, which is controlled by a parameter matrix  $\mathbf{W}$ , and  $\mathbf{V}$  is used to provide a linear weighted sum of the basis to give us the resulting vector valued function, in a similar way to generalised linear models. Deep neural networks then take the form of a functional composition for the basis functions,

$$\mathbf{g}(\mathbf{x}) = \mathbf{V}_\ell^\top \boldsymbol{\phi}_\ell(\mathbf{W}_{\ell-1} \boldsymbol{\phi}_{\ell-1}(\dots \mathbf{W}_2 \boldsymbol{\phi}(\mathbf{U}_1 \mathbf{x}))).$$

A serious challenge for deep networks, when trained in a feed-forward manner, is overfitting. As the number of layers increase, and the number of basis functions in each layer also goes up a very powerful representation that is highly parameterised is formed. The matrix mapping between each set of basis functions  $\mathbf{W}_i$  has size  $k_i \times k_{i+1}$ , where  $k_i$  is the number of basis functions in the  $k$ th layer. In practice networks containing sometimes thousands of basis functions can be used leading to a parameter explosion.

## 2.1 Weight Matrix Factorization

One approach to dealing with such matrices is to replace them with a lower rank form,

$$\mathbf{W}_i = \mathbf{U}_i \mathbf{V}_i^\top$$

where  $\mathbf{U}_i \in \mathbb{R}^{k_{i+1} \times q_i}$  and  $\mathbf{V}_i \in \mathbb{R}^{k_i \times q_i}$ , where  $q_i < k_i$  and  $q_i < k_{i+1}$ . Whilst this idea hasn't yet, to our knowledge, been yet pursued in the deep neural network community, Denil et al. [2013] have empirically shown that trained neural networks can have low rank matrices as evidenced by the ability to predict one set of part of the weight matrix given by another. The approach of 'dropout' [Srivastava et al., 2014] is also widely applied to control the complexity of the model implying the models are over parameterised.

Substituting the low rank form into the compositional structure for the deep network we have

$$\mathbf{g}(\mathbf{x}) = \mathbf{V}_\ell^\top \phi_\ell(\mathbf{U}_{\ell-1} \mathbf{V}_{\ell-1}^\top \phi_{\ell-1}(\dots(\mathbf{U}_2 \mathbf{V}_2 \phi(\mathbf{U}_1 \mathbf{x}))).$$

We can now identify the following form inside the functional decomposition,

$$f_i(\mathbf{z}) = \mathbf{V}_i^\top \phi_i(\mathbf{U}_{i-1} \mathbf{z})$$

and once again obtain a functional composition,

$$\mathbf{g}(\mathbf{x}) = \mathbf{f}_\ell(\mathbf{f}_{\ell-1}(\dots \mathbf{f}_1(\mathbf{x}))).$$

The standard deep network is recovered when we set  $q_i = \min(k_i, k_{i+1})$  and the deep Gaussian process is recovered by keeping  $q_i$  finite and allowing  $k_i \rightarrow \infty$  for all layers. Of course, the mappings in the Gaussian process are treated probabilistically and integrated out, rather than optimised, so despite the increase in layer size the number of parameters in the resulting model is many fewer than those in a standard deep neural network.

Deep Gaussian processes can also be used for unsupervised learning by replacing the input nodes with a white noise process. The resulting deep GPs can be seen as fully Bayesian generalizations of unsupervised deep neural networks with Gaussian nodes as proposed by Kingma and Welling [2013] and Rezende et al. [2014].

Unfortunately, exact inference in deep Gaussian process models is intractable. Damianou and Lawrence [2013] applied variational compression [Titsias, 2009] to perform approximate inference. In this paper we revisit that bound and apply a nested variational compression to obtain a new bound on the deep GP. The new bound factorizes across data points allowing parallel computation [Gal et al., 2014] or optimization via stochastic gradient descent [Hensman et al., 2013].

In the rest of the paper, we first review variational compression in the context of Gaussian processes. We then introduce deep Gaussian processes and show how the compression can be applied in a nested way to perform inference through an arbitrary number of model layers. We end with experiments on some standard data sets.

### 3 Variational Compression in Gaussian Processes

Let's assume that we are given a series of input-output pairs  $\{y_i, \mathbf{x}_i\}_{i=1}^n$ , which we stack into target vector and design matrix  $\mathbf{y}, \mathbf{X}$ . The data are modelled as noisy observations of a function  $f$ , so that

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \tag{1}$$

with  $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ . In a Gaussian process model we assume that the function  $f(\mathbf{x})$  is drawn from a Gaussian process,  $f(\mathbf{x}) \sim \mathcal{GP}(\mu_f(\mathbf{x}), k_{ff}(\mathbf{x}, \mathbf{x}'))$ , by which we mean that any finite set of values of the function will be jointly Gaussian distributed with a mean given by computing  $\mu(\cdot)$  at the relevant points and a covariance given by computing  $k_{ff}(\cdot, \cdot)$  at the relevant points.

The consistency property of the Gaussian process enables us to consider the values of the function on where the data are present, effectively marginalising the remaining values. If we assume that the mean function is zero, then we can write

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}_{\mathbf{ff}}). \tag{2}$$

The beauty of Gaussian processes lies in their tractability. Using properties of multivariate Gaussians it is straightforward to compute the marginal likelihood (in  $\mathcal{O}(n^3)$  complexity), as well as the posterior of the latent function values  $p(\mathbf{f}|\mathbf{y}, \mathbf{X})$  [see for example Rasmussen and Williams, 2006]. Note that the covariance function,  $k_{ff}(\cdot, \cdot)$ , or kernel, is also dependent on parameters that control properties (such as lengthscale, smoothness etc). We omit this dependence in our notation, and we assume that such parameters might be determined as an outer loop on our algorithm such as maximum (approximate) likelihood or an appropriate approximate Bayesian procedure.

#### 3.1 Inducing Points Representations

Gaussian processes are flexible non-parametric models for functions. However, whilst inference is analytically tractable, a challenge for these models lies in their computational complexity and storage which are worst case  $\mathcal{O}(n^3)$  and  $\mathcal{O}(n^2)$  respectively.

To deal with this, in the machine learning community, there has been a lot of focus on augmenting Gaussian process models by introducing an extra set of variables,  $\mathbf{u}$ , and their corresponding inputs,  $\mathbf{Z}$  [see e.g. Csató and Opper, 2002, Snelson and Ghahramani, 2006, Quiñonero Candela and Rasmussen, 2005]. Augmentation occurs by assuming that there is an additional set of variables  $\mathbf{u}$  which are jointly Gaussian with our original function  $\mathbf{f}$  allowing us to write

$$p(\mathbf{f}, \mathbf{u}|\mathbf{Z}, \mathbf{X}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} \mathbf{K}_{\mathbf{ff}} & \mathbf{K}_{\mathbf{fu}} \\ \mathbf{K}_{\mathbf{uf}} & \mathbf{K}_{\mathbf{uu}} \end{bmatrix}\right) \tag{3}$$

The multivariate joint Gaussian density has the convenient property that it is trivial to decompose it into associated conditional and marginal distributions

for each variable. We can decompose the joint distribution as follows

$$p(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})p(\mathbf{u}) \tag{4}$$

$$= \mathcal{N}(\mathbf{f}|\mathbf{K}_{\mathbf{f}\mathbf{u}}\mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1}\mathbf{u}, \Sigma) \mathcal{N}(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u}\mathbf{u}}) \tag{5}$$

where  $\Sigma = \mathbf{K}_{\mathbf{f}\mathbf{f}} - \mathbf{K}_{\mathbf{f}\mathbf{u}}\mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u}\mathbf{f}}$  is the conditional covariance of  $\mathbf{f}$  given  $\mathbf{u}$ . The model is now augmented with a set of additional latent variables  $\mathbf{u}$ . Of course, we can immediately marginalise these variables exactly and return to equation (2), but through their introduction we will be able to apply a variational approach to inference termed *variational compression*.

Reintroducing independent Gaussian noise we can write the joint density for the data and the augmented latent variables as

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u}), \tag{6}$$

where we have ignored the conditioning on the input locations  $\mathbf{X}$ . The technique of variational compression now proceeds by considering the conditional density

$$p(\mathbf{y}|\mathbf{u}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u}) d\mathbf{f} \tag{7}$$

which we choose to lower bound through assuming an approximation to the posterior of the form  $q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$  obtaining,

$$\log p(\mathbf{y}|\mathbf{u}, \mathbf{Z}, \mathbf{X}) \geq \mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{f}\mathbf{u}}\mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1}\mathbf{u}, \sigma^2\mathbf{I}) - \frac{1}{2\sigma^2}\text{tr}(\Sigma) \tag{8}$$

This conditional bound now has two parts, a part that looks like a likelihood conditioned on the inducing variables and a part that acts as a correction to the lower bound.

### 3.2 Low Rank Gaussian Process Approximations

Since the conditional bound (8) is conjugate to the prior  $p(\mathbf{u})$ , we can integrate out the remaining variables  $\mathbf{u}$ . The result is

$$\log p(\mathbf{y}|\mathbf{X}, \mathbf{Z}) \geq \log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{\mathbf{f}\mathbf{u}}\mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u}\mathbf{f}} + \sigma^2\mathbf{I}) - \frac{1}{2\sigma^2}\text{tr}(\Sigma), \tag{9}$$

Note the similarity here between the approximation and a traditional Bayesian parametric model. In a parametric model we normally have a likelihood conditioned on some parameters and we integrate over the parameters using a prior. Our inducing variables appear somewhat analogous to the parameters in that case. However, before marginalization the likelihood is independent across the data points. This observation inspires stochastic variational approaches [Hoffman et al., 2012] to allow GP inference for very large data sets Hensman et al. [2013].

### 3.3 Stochastic Variational Inference

Direct marginalization of the inducing variables,  $\mathbf{u}$ , is tempting as it is analytically tractable. However it results in an expression which does not factor in  $n$ , and so does not lend itself to stochastic optimization. Now if we apply traditional parametric variational Bayes to this portion (treating  $\mathbf{u}$  as the ‘parameters’ of the model) we obtain

$$\begin{aligned} \log p(\mathbf{y}|\mathbf{X}, \mathbf{Z}) &\geq \log \mathcal{N}(\mathbf{y}|\mathbf{K}_{\mathbf{f}\mathbf{u}}\mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1}\mathbf{m}, \sigma^2\mathbf{I}) \\ &\quad - \frac{1}{2\sigma^2} \text{tr}(\mathbf{S}\mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u}\mathbf{f}}\mathbf{K}_{\mathbf{f}\mathbf{u}}\mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1}) \\ &\quad - \text{KL}(q(\mathbf{u}) \parallel p(\mathbf{u})) - \frac{1}{2\sigma^2} \text{tr}(\mathbf{\Sigma}) \end{aligned} \quad (10)$$

where the variational distribution is  $q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S})$ . This method was used by Hensman et al. [2013] to fit GPs to large datasets through stochastic variational optimization [Hoffman et al., 2012].

### 3.4 Mutual Information and the Total Conditional Variance

There are two conditions under which the variational compression will provide a tight lower bound. First, if the data  $\mathbf{y}$  are less informative about the latent function variables  $\mathbf{f}$ , and second where the inducing variables  $\mathbf{u}$  are highly informative about  $\mathbf{f}$ . The former depends on the noise variance,  $\sigma^2$ , and the latter on the conditional entropy of the latent variables given the inducing variables. The conditional entropy for the GP system is given by

$$H(\mathbf{f}|\mathbf{u}) = \frac{1}{2} \log |\mathbf{\Sigma}|.$$

From an information theoretic perspective, the conditional entropy represents the amount of additional information we need to specify the distribution of  $\mathbf{f}$  given that we already have the inducing vector,  $\mathbf{u}$ . This required additional information can be manipulated by changing the relationship between  $\mathbf{u}$  and  $\mathbf{f}$ . The inducing points themselves are parameterised by  $\mathbf{K}_{\mathbf{f}\mathbf{u}}$  and  $\mathbf{K}_{\mathbf{u}\mathbf{u}}$  which introduce a new set of *variational* parameters to the model. Those parameters typically include the inducing input locations and any parameters of the covariance function itself. We can be quite creative in how we specify these relationships, for example, placing the inducing variables in separate domain was suggested by Álvarez et al. [2010]. By minimizing this term we improve the quality of the bound.

Computing this entropy requires  $\mathcal{O}(n^3)$  computations due to the log determinant, but the log determinant of the matrix is upper bounded by its trace, this is trivially true because log determinant is the sum of the *log* eigenvalues of  $\mathbf{\Sigma}$  whereas the trace is the sum of the eigenvalues directly. The logarithm is upper bounded by the linear function. So it is a sufficient condition for  $\text{tr}(\mathbf{\Sigma})$  to be small, to minimize the conditional entropy of the inducing point relation.

The trace of a covariance is sometimes referred to as the *total variance* of the distribution. We therefore call this bound on the conditional entropy the total conditional variance (TCV).

The TCV of the density,  $p(\mathbf{f}|\mathbf{u})$ ,  $\text{tr}(\boldsymbol{\Sigma})$ , appears in (8) and remains throughout (9) and (10). It controls conditional entropy and the information in the data and upper bounds the additional information that is contained in the data that is not represented by  $\mathbf{u}$ . When this term is small, we expect the approximation to work well. Indeed, the two extrema of the variational compression behaviour are given when the TCV term is zero: either the noise variance is very large scaling out the term or the TCV is zero because the conditional entropy  $H(\mathbf{f}|\mathbf{u})$  is zero.

The TCV ensures that the inducing variables take up appropriate positions when manipulating  $\mathbf{Z}$  to tighten the variational bound. It also plays a key role in the formulation for of the variational bound we now present for deep GPs.

## 4 Variational Compression in Deep GPs

Deep Gaussian processes are models of the form

$$\mathbf{y} = \mathbf{h}_\ell(\mathbf{h}_{\ell-1}(\dots\mathbf{h}_1(\mathbf{x}))) + \boldsymbol{\epsilon}, \quad (11)$$

where we then make an assumption that each of the hidden (potentially multi-variate) functions  $\mathbf{h}_i$  is given by a Gaussian process. For a fixed set of inputs  $\mathbf{X}$  and a series of observed responses  $\mathbf{y}$ , and taking the vector  $\mathbf{h}_i$  to contain the vector of variables representing function values in the  $i^{\text{th}}$  layer, the joint probability can be written

$$p(\mathbf{y}, \{\mathbf{h}_i\}_{i=1}^\ell | \mathbf{X}) = p(\mathbf{y} | \mathbf{h}_{\ell-1}) \prod_{i=2}^{\ell} p(\mathbf{h}_i | \mathbf{h}_{i-1}) p(\mathbf{h}_1 | \mathbf{X}) \quad (12)$$

with

$$\begin{aligned} \mathbf{h}_1 | \mathbf{x} &\sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{h}_1 \mathbf{h}_1} + \sigma_1^2 \mathbf{I}), \\ \mathbf{h}_i | \mathbf{h}_{i-1} &\sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{h}_i \mathbf{h}_i} + \sigma_i^2 \mathbf{I}), \\ \mathbf{y} | \mathbf{h}_{\ell-1} &\sim \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{f}_\ell \mathbf{f}_\ell} + \sigma_\ell^2 \mathbf{I}). \end{aligned}$$

Inference over the latent variables  $\mathbf{h}$  is very challenging. Not only will computations scale with the usual  $\mathcal{O}(n^3)$  rule for GPs, but the hidden layer variables are also dependent *between* layers. We turn to approximate Gaussian processes based on variational compression. Then to deal with the dependence between layers we will To deal with the dependence between layers we use the VC trick again. The result is a tractable bound on the marginal likelihood of a deep GP which is scalable and interpretable.

## 4.1 Augmenting Each Layer

As per the original formulation [Damianou and Lawrence, 2013], we assume that the function at each layer includes some independent Gaussian noise with variance  $\sigma_i^2$ , and we augment that layer with a set of inducing variables  $\mathbf{u}_i$  in the same way as a for a single-layer GP model. Within each layer, then, we can apply variational compression to achieve a bound on the conditional probability, as per equation (8). Substituting this result in to the structure (13) results in

$$\begin{aligned}
 p(\mathbf{y}, \{\mathbf{h}_i\}_{i=1}^{\ell-1} | \{\mathbf{u}_i\}_{i=1}^{\ell}, \mathbf{X}) &\geq \tilde{p}(\mathbf{y} | \mathbf{u}_\ell, \mathbf{h}_{\ell-1}) \\
 &\times \prod_{i=2}^{\ell-1} \tilde{p}(\mathbf{h}_i | \mathbf{u}_i, \mathbf{h}_{i-1}) \tilde{p}(\mathbf{h}_1 | \mathbf{u}_1, \mathbf{X}) \\
 &\times \exp\left(\sum_{i=1}^{\ell} -\frac{1}{2\sigma_i^2} \text{tr}(\boldsymbol{\Sigma}_i)\right), \tag{13}
 \end{aligned}$$

where we have omitted dependence on  $\mathbf{Z}$  variables for clarity, and have defined

$$\tilde{p}(\mathbf{h}_i | \mathbf{u}_i, \mathbf{h}_{i-1}) = \mathcal{N}(\mathbf{h}_i | \mathbf{K}_{\mathbf{h}_i \mathbf{u}_i} \mathbf{K}_{\mathbf{u}_i \mathbf{u}_i}^{-1} \mathbf{u}_i, \sigma_i^2 \mathbf{I}).$$

The conditional covariance matrices have been indexed by layer, so that

$$\boldsymbol{\Sigma}_i = \mathbf{K}_{\mathbf{h}_i \mathbf{h}_i} - \mathbf{K}_{\mathbf{h}_i \mathbf{u}_i} \mathbf{K}_{\mathbf{u}_i \mathbf{u}_i}^{-1} \mathbf{K}_{\mathbf{u}_i \mathbf{h}_i}.$$

If we ignore the TCV terms, then the remaining factors in (13) describe a joint probability density over  $\mathbf{y}$  and  $\mathbf{f}$  conditioned on  $\mathbf{u}$ . Figure 1 shows the resulting graphical model associated with this distribution. The plate notation indicates that the terms factorize across data points, this property will be preserved when we apply variational compression through the layers.

Here, our work diverges from that of Damianou and Lawrence [2013]. In their work, the inducing variables  $\mathbf{u}$  are marginalised (similarly to equation (9)), and a variational distribution  $q(\mathbf{h})$  is introduced for the hidden variables. This has two consequences: marginalising  $\mathbf{u}$  re-introduces dependencies between the hidden variables within a layer, resulting in an objective function which cannot be written as a sum of independent data terms. Secondly, the size of the optimization problem grows with  $n$ , as distributions representing latent variables have to be introduced, corresponding to each datum in each of the hidden layers. In this alternative formulation we will seek to retain the factorization across the data points allowing us to consider stochastic variational inference approaches to the model.

## 4.2 The First Layer

We now apply the variational compression approach to marginalise the variables associated with the first Gaussian process. Although to ensure the cancellation we now assume that

$$q(\mathbf{h}_1 | \mathbf{u}_1) = \tilde{p}(\mathbf{h}_1 | \mathbf{u}_1, \mathbf{X}), \tag{14}$$



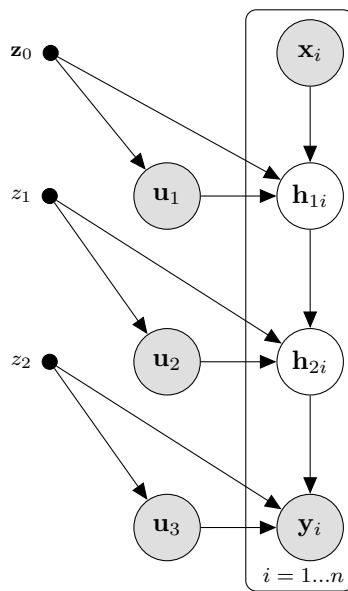


Figure 1: A graphical model representation of the deep GP structure with inducing variables. Nodes arranged horizontally form part of the same layer. we've added the index over  $i$  to highlight the fact that conditioned on  $\mathbf{u}_i$  we obtain independence over the data points.

and form a bound on the conditional distribution. Taking the relevant terms from equation (13), we obtain

$$\begin{aligned} \log p(\mathbf{h}_2|\mathbf{u}_1, \mathbf{u}_2) &\geq \tilde{p}(\mathbf{h}_1|\mathbf{u}_1) \langle \log \tilde{p}(\mathbf{h}_2|\mathbf{h}_1, \mathbf{u}_2) \rangle \\ &\quad - \left\langle \frac{1}{2\sigma_2^2} \text{tr}(\boldsymbol{\Sigma}_1) \right\rangle - \frac{1}{2\sigma_1^2} \text{tr}(\boldsymbol{\Sigma}_0). \end{aligned} \quad (15)$$

The second stage of variational compression is to marginalize  $\mathbf{u}_1$ . Since the above expression is not conjugate to the Gaussian prior  $p(\mathbf{u}_1)$ , we must introduce variational parameters  $q(\mathbf{u}_1) = \mathcal{N}(\mathbf{u}_1|\mathbf{m}_1, \mathbf{S}_1)$ . Along with (14), we now have:

$$q(\mathbf{h}_1) = \int \tilde{p}(\mathbf{h}_1|\mathbf{u}_1) q(\mathbf{u}_1) d\mathbf{u}_1, \quad (16)$$

which is a straightforward Gaussian integral.

Although our approximation to the latent space is *not* factorized across the data dimension  $n$ , we are still able to construct an algorithm that depends on the data points independently; in practice, we need only ever compute the diagonal parts of the covariance in  $q(\mathbf{h})$  at each layer.

Using this definition of  $q(\mathbf{f}_1)$  with equation (15) allows us to variationally marginalize  $\mathbf{u}_1$ :

$$\begin{aligned} \log p(\mathbf{h}_2|\mathbf{X}, \mathbf{u}_2) &\geq \\ &\quad - \frac{1}{\sigma_1^2} \text{tr}(\boldsymbol{\Sigma}_0) - \frac{1}{\sigma_2^2} \langle \text{tr}(\boldsymbol{\Sigma}_1) \rangle_{q(\mathbf{h}_1)} \\ &\quad - \text{KL}(q(\mathbf{u}_1) \| p(\mathbf{u}_1)) + \\ &\quad \langle \log \mathcal{N}(\mathbf{h}_2|\mathbf{K}_{\mathbf{h}_2, \mathbf{u}_2} \mathbf{K}_{\mathbf{u}_2 \mathbf{u}_2}^{-1} \mathbf{u}_2, \sigma_2^2 \mathbf{I}) \rangle_{q(\mathbf{h}_1)}. \end{aligned} \quad (17)$$

The expectations under  $q(\mathbf{h}_1)$  involve the covariance function in the same way as Titsias and Lawrence [2010], Damianou and Lawrence [2013]. The required quantities are

$$\begin{aligned} \psi_i &= \langle \text{tr}(\mathbf{K}_{\mathbf{h}_i \mathbf{h}_i}) \rangle_{q(\mathbf{h}_{i-1})} \\ \Psi_i &= \langle \mathbf{K}_{\mathbf{h}_i \mathbf{u}_i} \rangle_{q(\mathbf{h}_{i-1})} \\ \Phi_i &= \langle \mathbf{K}_{\mathbf{u}_i \mathbf{h}_i} \mathbf{K}_{\mathbf{h}_i \mathbf{u}_i} \rangle_{q(\mathbf{h}_{i-1})}. \end{aligned}$$

which can be computed analytically for some popular choices of covariance function including the exponentiated quadratic,

$$k(\mathbf{x}, \mathbf{x}') = \alpha \exp\left(\frac{1}{\ell^2} \|\mathbf{x} - \mathbf{x}'\|_2^2\right),$$

and linear forms,

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}.$$

With these definitions, it is possible to substitute into (17) and re-arrange to give

$$\begin{aligned}
\log p(\mathbf{h}_2|\mathbf{X}, \mathbf{u}_2) &\geq -\frac{1}{\sigma_1^2} \text{tr}(\boldsymbol{\Sigma}_0) - \frac{1}{\sigma_2^2} \langle \text{tr}(\boldsymbol{\Sigma}_1) \rangle_{q(\mathbf{h}_1)} \\
&\quad - \text{KL}(q(\mathbf{u}_1) \| p(\mathbf{u}_1)) \\
&\quad - \frac{1}{\sigma_2^2} \text{tr} \left( (\boldsymbol{\Phi}_2 - \boldsymbol{\Psi}_2^\top \boldsymbol{\Psi}_2) \mathbf{K}_{\mathbf{u}_2 \mathbf{u}_2}^{-1} \mathbf{u}_2 \mathbf{u}_2^\top \mathbf{K}_{\mathbf{u}_2 \mathbf{u}_2}^{-1} \right) \\
&\quad + \log \mathcal{N}(\mathbf{h}_2 | \boldsymbol{\Psi}_2 \mathbf{K}_{\mathbf{u}_2 \mathbf{u}_2}^{-1} \mathbf{u}_2, \sigma_2^2 \mathbf{I})
\end{aligned} \tag{18}$$

where we have completed the square to ensure that the  $\mathbf{h}_2$  appears in a normalised Gaussian distribution. We are left with a conditional expression for the second layer, where the information from the layer above has been propagated variationally, as well as a series of terms that ensure that the expression remains a variational bound.

### 4.3 Subsequent Layers

Having arrived at an expression for the second layer, with the first layer marginalized, we are in a position to apply variational compression again to marginalize layer 2. The procedure follows much the same pattern as for the first layer. The result is a bound on  $p(\mathbf{h}_3|\mathbf{u}_3)$ , with some additional terms, where the relation  $\mathbf{h}_3|\mathbf{u}_3$  is again a normal distribution.

Recursively applying this formulation through an arbitrarily deep network results in the following bound:

$$\begin{aligned}
\log p(\mathbf{y}|\mathbf{X}) &\geq -\frac{1}{\sigma_1^2} \text{tr}(\boldsymbol{\Sigma}_1) - \sum_{i=2}^{\ell} \frac{1}{2\sigma_i^2} (\psi_i - \text{tr}(\boldsymbol{\Phi}_i \mathbf{K}_{\mathbf{u}_i \mathbf{u}_i}^{-1})) \\
&\quad - \sum_{i=1}^{\ell} \text{KL}(q(\mathbf{u}_i) \| p(\mathbf{u}_i)) \\
&\quad - \sum_{i=2}^{\ell} \frac{1}{2\sigma_i^2} \text{tr} \left( (\boldsymbol{\Phi}_i - \boldsymbol{\Psi}_i^\top \boldsymbol{\Psi}_i) \mathbf{K}_{\mathbf{u}_i \mathbf{u}_i}^{-1} \langle \mathbf{u}_i \mathbf{u}_i^\top \rangle_{q(\mathbf{u}_i)} \mathbf{K}_{\mathbf{u}_i \mathbf{u}_i}^{-1} \right) \\
&\quad + \log \mathcal{N}(\mathbf{y} | \boldsymbol{\Psi}_\ell \mathbf{K}_{\mathbf{u}_\ell \mathbf{u}_\ell}^{-1} \mathbf{m}_\ell, \sigma_\ell^2 \mathbf{I})
\end{aligned} \tag{19}$$

This expression constitutes the main contribution of this work. We now have a bound on the marginal likelihood, parameterised by variational distributions  $q(\mathbf{u}_i)$  at each layer, along with inducing inputs for each layer  $\mathbf{Z}_i$  and covariance parameters of the kernel at each layer. Importantly, each part of the bound can be written as a sum of  $n$  terms, with each term depending on only one datum. This allows straightforward inference in the model using parallelized or stochastic methods, without having to deal with large numbers of latent variables,  $\mathbf{h}_i$ .

All the latent variables  $\{\mathbf{h}_i\}_{i=1}^\ell$  have now been marginalised using our variational compression scheme. From the previous discussion, we know that the approximation will be reasonable if they are highly correlated with the inducing points  $\mathbf{u}_i$ . These correlations are reduced if we inject a larger amount of noise at each hidden layer: if the variance variables  $\sigma_i^2$  are large, then our approximation will fail.

#### 4.4 Examining the Bound

Our bound on the marginal likelihood has a single ‘likelihood’ expression (equation (19), last line), preceded by a series of ‘regularizing’ terms. We first examine the likelihood part, which has a similar form to a neural network. The term is Gaussian with mean given by a linear combination of the columns of  $\Psi_\ell$ , with weights,  $\mathbf{K}_{\mathbf{u}_\ell \mathbf{u}_\ell} \mathbf{m}_\ell$ , which are given by the mean of the variational distribution  $q(\mathbf{u}_\ell)$  weighted by its prior covariance. Information from previous layers feeds in through the matrix,

$$\Psi_\ell = \langle \mathbf{K}_{\mathbf{f}_\ell \mathbf{u}_\ell} \rangle_{q(\mathbf{f}_{\ell-1})}.$$

We can examine the  $j$ th column of this matrix for the  $i$ th input in the case of the exponentiated quadratic covariance,

$$\psi_{i,j}^\ell = \left\langle \alpha \exp \left( -\frac{\|\mathbf{f}_{\ell-1} - \mathbf{z}_{\ell-1}\|_2^2}{2\ell^2} \right) \right\rangle_{q(\mathbf{f}_{\ell-1})}.$$

We can compare this with a radial basis function network. In deep versions of those models the  $i$ th layer the basis functions would be centred on particular values,  $\mathbf{c}_i$ . Those centres are functionally equivalent to our inducing inputs  $\mathbf{z}_{\ell-1}$ . However, in our model they are variational parameters, rather than model parameters. This is an important difference because we can increase the number of them at any time without risk of overfitting. If we *do* use more centres we can only reduce the conditional entropy  $H(\mathbf{f}_i | \mathbf{u}_i)$  thereby improving the quality of the variational bound. In each layer, we can adjust the weights  $\mathbf{K}_{\mathbf{u}_\ell \mathbf{u}_\ell} \mathbf{m}_\ell$  through varying the mean of the variational distribution over the inducing variables. If we were to use other covariance functions we would obtain similar neural network-like models but with different activation functions.

A significant difference to a neural network approach is that we are propagating *distributions* through each layer of the network rather than just point values. In each layer we must compute  $q(\mathbf{h}_i) = \int \tilde{p}(\mathbf{h}_i | \mathbf{u}_i) q(\mathbf{u}_i) d\mathbf{u}_i$ , and propagate this to the next layer, where it is used to take expectations of the kernel function to pass into the subsequent layers. The form of our variational compression means that these distributions are Gaussian, so we are propagating both a mean and a variance through the model at every layer. On computing the derivative of any part of the approximation, we must do a feed-forward pass of these Gaussian messages, followed by a backpropagation step using the chain-rule.

Our bound contains the trace of the conditional matrix (now in expectation), which we have studied in section 3.4. It also contains the usual KL divergence

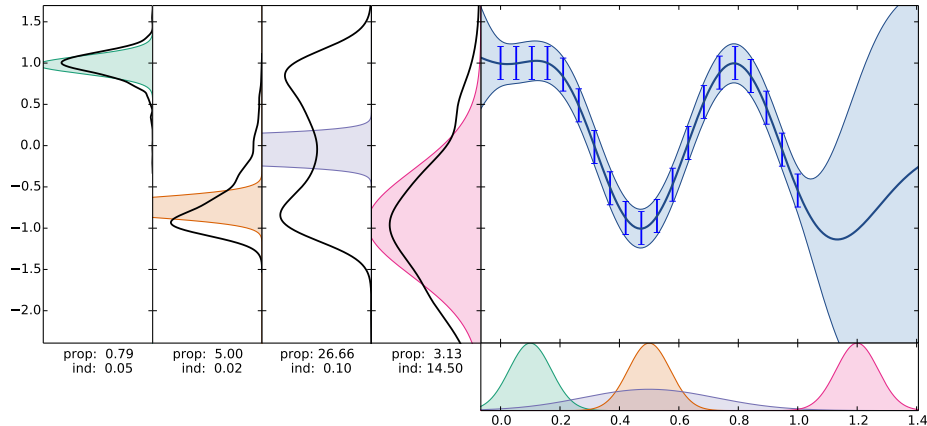


Figure 2: Forward propagation of Gaussian messages through a layer of a deep GP. Bottom right: four colour-coded Gaussian distributions to be passed through the GP layer. Top-right: a Gaussian process, represented by a finite series of inducing variables  $\mathbf{u}$ , depicted by vertical red bars. Left: the responses of the GP to the input distributions, with Gaussian approximation shown in colours matching bottom right, and the ground truth represented by a solid line, computed by Monte Carlo. Below each approximation we give the two penalty terms associated with propagating this distribution. The propagation term (top) is large when the function is locally highly nonlinear, and the compression term (below) is large when the input is far from the inducing variables.

term from the prior. The final ‘regularization’ term (third line of (19)) has interesting regularization properties: it contains the variance of the GP function at each layer, under our approximation. To see this, if we are given the values of a GP function  $\mathbf{u}$  at  $\mathbf{Z}$ , the mean of the prediction for the points  $\mathbf{f}$  at  $\mathbf{X}$  is given by  $\langle \mathbf{f} \rangle = \mathbf{K}_{\mathbf{f}\mathbf{u}} \mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1} \mathbf{u}$ . The sum of the variance of this vector is then  $\text{tr}(\mathbf{K}_{\mathbf{f}\mathbf{u}} \mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1} \mathbf{u} \mathbf{u}^{\top} \mathbf{K}_{\mathbf{u}\mathbf{u}}^{-1} \mathbf{K}_{\mathbf{u}\mathbf{f}})$ , and the ‘regularization’ term is the variance of this under  $q(\mathbf{h}_{i-1})$ , in expectation under  $q(\mathbf{u})$ . The term summarizes how the values of the function will change as the input to the function changes: a measure of the local ‘wiggleness’ of the function under the variational input distribution.

We refer to these two regularization terms as the ‘compression term’ and the ‘propagation term’. In figure 2, we illustrate the forward passing of the Gaussian distributions (according to our likelihood computation) for different scenarios, and the incurred penalty terms. In summary, the regularization terms encourage approximations that can be well represented by the limited representative power of  $q(\mathbf{u})$ .

## 5 Experiments

We present three applications of the variationally compressed deep GP method. In each case, we used simple gradient based optimization to maximize the bound

on the marginal likelihood with respect to the covariance function parameters, the inducing input positions  $Z$  in each layer, and the variational parameters of  $q(\mathbf{u})$  in each layer,  $m_i, \mathbf{S}_i$ . To maintain positive-definiteness of the covariance matrices, we employed a lower-triangular factorized representation,  $\mathbf{S}_i = \mathbf{L}_i \mathbf{L}_i^\top$ .

To exploit the factorised nature of the objective function, our python implementation uses MPI to parallelize computation across several cores of a desktop machine. Parallelism is across chunks of data, since equation ?? can be written as a sum of data dependent terms. Parallelization to larger systems is straightforward, and the objective lends itself to stochastic optimization.

## 5.1 Step Function Data

In Rasmussen and Williams [2006], a simple example is presented which is challenging for standard GP regression: data representing a noisy step function as in Figure 3. The GP with exponentiated quadratic covariance is unable to satisfactorily fit to the data, as the top plot in Figure 3 shows. Rasmussen and Williams [2006] propose using a neural-network based covariance function, which is able to model the data somewhat better. Calandra et al. [2014] propose a two layer model where the data are transformed by some deterministic function before being passed into a GP, which appears to perform somewhat better, though this is arguably still a kernel selection problem, with parameters of the deterministic function being incorporated into the kernel.

The idea of deep networks is to avoid this kind of kernel selection, using layers of representation to infer features from the data. Can deep GPs do away with the problem of having to select a kernel in a Bayesian fashion? Figure 3 suggests that this might be the case. Moving down the panels of the Figure, the depth of the model is increased from 1 (a GP model) to 3. The GP model is unable to cope with the non-smoothness of the function, provides an ‘overshot’ mean function and extrapolates poorly. The two layer deep GP has more capability to fit the sharp change in the function though it still overshoots a little, and the three layer deep model performs well, with a sharp mean function at the step. Extrapolation from the data has interesting behaviour, we note that despite the posterior appearing bimodal, any sample drawn from the posterior will be a continuous function, which will switch between the two modes with a lengthscale of around 0.2.

## 5.2 Robot Wireless Data

Our second data set consists of a robotics localization problem [Ferris et al., 2007]. We are presented with the signal strengths of thirty wireless access points located around a building, as detected by a robot which is tracing a path through the corridors. The wireless signal strengths fluctuate as a function of time as the robot moves nearer or further from each access point. The underlying structure of the signals must surely represent the position of the robot in the building.

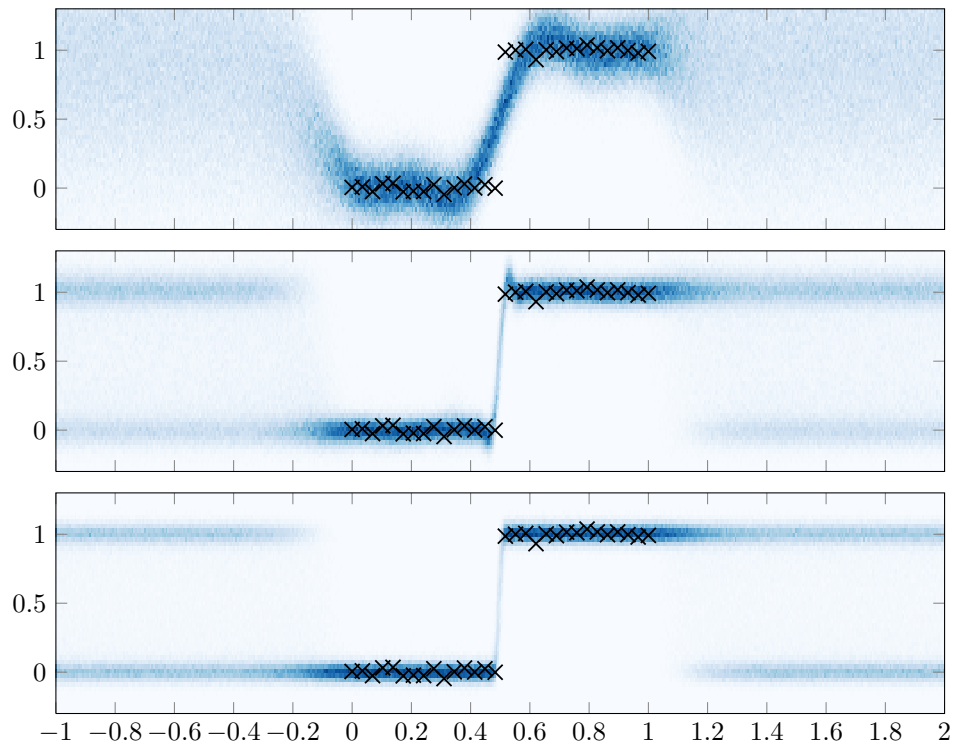


Figure 3: Regression on the noisy step-function data [Rasmussen and Williams, 2006]. Top: Gaussian process regression, middle: deep GP with one hidden layer, bottom: deep GP with two hidden layers. In each plot, the posterior predictive density is shown in blue.

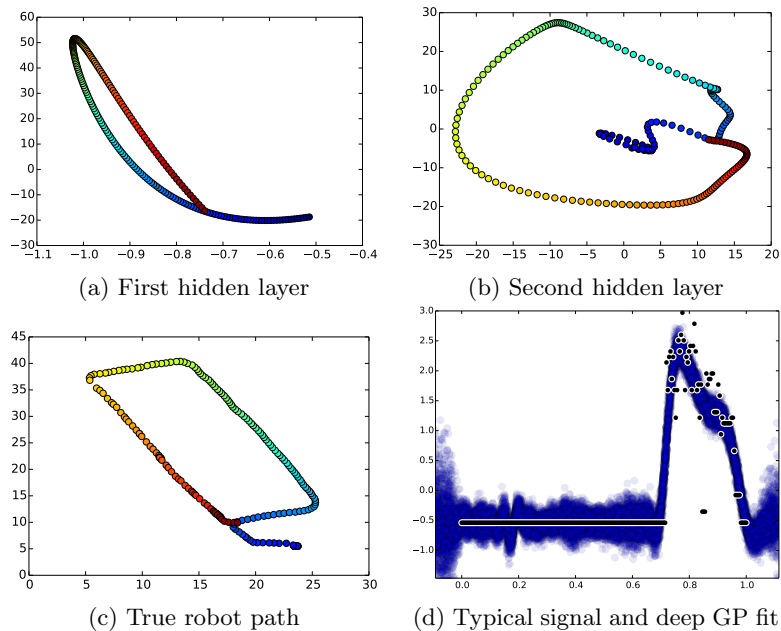


Figure 4: Latent representation of the robot wireless data. In each figure, time (the input to the deep model) is represented by color, and each circle represents one temporal data point. (a) the latent representation in the first layer. (b) The latent representation in the second layer. (c) The ground truth: the known path of the robot. The latent representations qualitatively represent the topology (a) and finer structure (b) of the ground truth. (d) Shows a typical signal, with data represented by black points and the posterior density for this signal in blue.

We build a 3 layer deep GP, in which the input layer was time, through two hidden layers with 3 dimensions each, and finally mapping to the 30 dimensional signal strengths. After optimization of the marginal-likelihood bound using L-BFGS-B [Zhu et al., 1997], the learned structure of the model is as in Figure 4.

Figure 4 (c) shows the true path of the robots, which completes a rectangular loop around the building, with a 'tail' of data at the start. In the first hidden layer (figure 4(a)), the model learns the topology of the structure: a loop closes at the correct point. In the subsequent layer, the representation contains 'corners', representing structure in the data where correlations in the signal strength must vary rapidly or slowly. This hierarchically structured learning of features is a key feature of the deep GP model which is not possible using a standard GP.

The final frame of figure 4 shows one of the thirty signals used. Two interesting features are prominent: the deep GP is insensitive to outliers (some occur at around  $t = 0.85$ ), and the deep GP predicts structure where none is present (around  $t = 0.2$ ). Both of these effects are a result of the strong structural prior imposed by the deep GP model: output variables are expected to covary in a



consistent way, and structure in some part of the data is imposed on other parts also.

In both these cases, the strong structural prior is reasonable. At around  $t = 0.2$  the WiFi drops out because the device only retains the largest signals it can measure. However, the true underlying signal is still likely to covary with the other measurements in a fashion as predicted by the model. The outlying data are also explained well by the deep GP.

The variational approximation to the posterior is unimodal, and the deep GP prior contains many possible modes which can be constructed by symmetrical and rotational arguments. Re-running the experiment shown in figure 4 will result in a different representation each time. Selection of the ‘best’ approximating posterior is possible by comparing the bounds on the marginal likelihood. We note that the majority of solutions found for this problems qualitatively represent that presented in figure 4. There would also be scope to combine the variational approximations through mixture distributions [Lawrence, 2000].

### 5.3 Autoencoders

Building auto-encoding deep GPs is straightforward: we simple use the same data as input and output to the model.

We build a two-layer deep GP, with the same data on the input and output. That is, a single hidden representation with an ‘encoding’ layer and a ‘decoding’ layer. Using the Frey faces dataset [Frey et al., 1998], which has 784 dimensions (pixels) and approximately 2,000 training points, we optimized the marginal likelihood bound using L-BFGS-B [Zhu et al., 1997].

## 6 Discussion

We have shown how variational compression bounds can be applied to inference in deep Gaussian process models. The resulting bounds are scalable due to their decompositional nature over the data points. In analysing the bound on the deep GP marginal likelihood, we have seen that the approximation appears as a neural network structure, but with Gaussian messages being fed forward. We have shown how natural penalty terms arise which decrease the bound when the propagation of the Gaussian messages performs poorly.

This work opens the possibility of application of Deep GPs to big datasets for the first time.

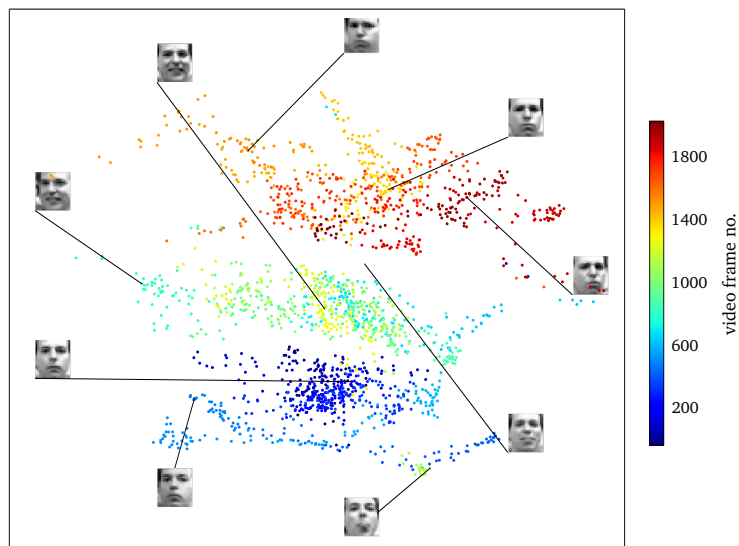


Figure 5: Latent space of the Frey faces data set using an autoencoding deep GP. We show illustrative latent points shown reconstructed. Each point is colored according to its position in the video, although the model is trained ignoring the temporal structure of the data.

## References

- M. A. Álvarez, D. Luengo, M. K. Titsias, and N. D. Lawrence. Efficient multi-output Gaussian processes through variational inducing kernels. In Teh and Titterton [2010], pages 25–32.
- R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth. Manifold Gaussian processes for regression. Technical report, 2014.
- L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002.
- A. Damianou and N. D. Lawrence. Deep Gaussian processes. In C. Carvalho and P. Ravikumar, editors, *Proceedings of the Sixteenth International Workshop on Artificial Intelligence and Statistics*, volume 31, AZ, USA, 2013. JMLR W&CP 31.
- A. Damianou, M. K. Titsias, and N. D. Lawrence. Variational Gaussian process dynamical systems. In P. Bartlett, F. Peirrer, C. Williams, and J. Lafferty, editors, *Advances in Neural Information Processing Systems*, volume 24, Cambridge, MA, 2011. MIT Press.
- M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. D. Freitas. Advances in neural information processing systems. In C. J. C. Burges, L. Bottou, Z. Ghahra-

- mani, M. Welling, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26, pages 2148–2156, Cambridge, MA, 2013.
- D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani. Avoiding pathologies in very deep networks. In S. Kaski and J. Corander, editors, *Proceedings of the Seventeenth International Workshop on Artificial Intelligence and Statistics*, volume 33, Iceland, 2014. JMLR W&CP 33.
- B. D. Ferris, D. Fox, and N. D. Lawrence. WiFi-SLAM using Gaussian process latent variable models. In M. M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 2480–2485, 2007.
- B. J. Frey, A. Colmenarez, and T. S. Huang. Mixtures of local linear subspaces for face recognition. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 32–37. IEEE, 1998.
- Y. Gal, M. van der Wilk, and C. E. Rasmussen. Distributed variational inference in sparse Gaussian process regression and latent variable models. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, Cambridge, MA, 2014.
- J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian processes for big data. In A. Nicholson and P. Smyth, editors, *Uncertainty in Artificial Intelligence*, volume 29. AUAI Press, 2013.
- M. Hoffman, D. M. Blei, C. Wang, and J. Paisley. Stochastic variational inference. Technical report, 2012.
- K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.
- D. P. Kingma and M. Welling. Auto-encoding variational Bayes. Technical report, 2013.
- N. D. Lawrence. *Variational Inference in Probabilistic Models*. PhD thesis, Computer Laboratory, University of Cambridge, New Museums Site, Pembroke Street, Cambridge, CB2 3QG, U.K., 2000. Available from <http://www.thelawrences.net/neil>.
- N. D. Lawrence and A. J. Moore. Hierarchical Gaussian process latent variable models. In Z. Ghahramani, editor, *Proceedings of the International Conference in Machine Learning*, volume 24, pages 481–488. Omnipress, 2007. ISBN 1-59593-793-3.

- M. Lázaro-Gredilla. Bayesian warped Gaussian processes. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, Cambridge, MA, 2012.
- D. J. C. MacKay. Introduction to Gaussian Processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, volume 168 of *Series F: Computer and Systems Sciences*, pages 133–166. Springer-Verlag, Berlin, 1998.
- R. M. Neal. *Bayesian Learning for Neural Networks*. Springer, 1996. Lecture Notes in Statistics 118.
- J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6: 1939–1959, 2005.
- C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006. ISBN 0-262-18253-X.
- D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic back-propagation and variational inference in deep latent Gaussian models. Technical report, 2014.
- E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18, Cambridge, MA, 2006. MIT Press.
- N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- Y. W. Teh and D. M. Titterton, editors. *Artificial Intelligence and Statistics*, volume 9, Chia Laguna Resort, Sardinia, Italy, 13-16 May 2010. JMLR W&CP 9.
- M. K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In D. van Dyk and M. Welling, editors, *Proceedings of the Twelfth International Workshop on Artificial Intelligence and Statistics*, volume 5, pages 567–574, Clearwater Beach, FL, 16-18 April 2009. JMLR W&CP 5.
- M. K. Titsias and N. D. Lawrence. Bayesian Gaussian process latent variable model. In Teh and Titterton [2010], pages 844–851.
- C. K. I. Williams. Computation with infinite neural networks. *Neural Computation*, 10(5):1203–1216, 1998.
- A. G. Wilson and R. P. Adams. Gaussian process kernels for pattern discovery and extrapolation. In S. Dasgupta and D. McAllester, editors, *ICML*, volume 28 of *JMLR Proceedings*, pages 1067–1075. JMLR.org, 2013.

C. Zhu, R. H. Byrd, and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.