Convolutional Deep Gaussian Processes

Vaibhav Singh

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Master of Technology

Department of Computer Science And Engineering

June 2018

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

[illegible]

(Signature)

(Vaibhav Singh)

CS16MTECH11017

(Roll No.)

Approval Sheet

This Thesis entitled Convolutional Deep Gaussian Processes by Vaibhav Singh is approved for the degree of Master of Technology from IIT Hyderabad

[illegible]

(Maunendra S. Desarkar) Examiner
Department of Computer Science And Engineering
IITH

[illegible]

(Manish Singh) Examiner
Department of Computer Science And Engineering
IITH

[illegible]

(Dr. Srijith P. K.) Adviser
Department of Computer Science And Engineering
IITH

[illegible]

(Dr Vineeth N B) Chairman
Department of Computer Science And Engineering
IITH

Acknowledgements

I would like to thank my adviser, Dr. Srijith P K, for all the help, and providing me the guidance needed to complete this thesis. He was very patient with me during my early days where my learning pace was quite slow. We started with the very basics. He trusted me with his ideas and gave me liberty to come up with new ones too. Overall, he has inspired me both in academics and as a person too and, therefore, I feel very privileged to have worked under his supervision.

I would also like to thank whole Computer Science and Engineering department, IIT Hyderabad, for all the constructive feedbacks given during the intermediate evaluations. I would like to thank Vinayak Kumar, Project Assistant, IIT Hyderabad, for helpful discussions and Andreas Damianou, Amazon Research, Cambridge, for mentoring.

I am grateful to Ministry of Human Resource Development(MHRD) for the fellowship given to me during this period of time. I am grateful to all the funding bodies which helped me with necessary equipments required for the experimentations.

Last but not least, I am very tankful to my family for constantly supporting me with every decision in my life. Sending their continuous love for 2 years even after being far away.

Abstract

Deep Gaussian processes (DGPs) provide a Bayesian non-parametric alternative to standard parametric deep learning models. A DGP is formed by stacking multiple GPs resulting in a well-regularized composition of functions. The Bayesian framework that equips the model with attractive properties, such as implicit capacity control and predictive uncertainty, makes it at the same time challenging to combine with a convolutional structure. This has hindered the application of DGPs in computer vision tasks, an area where deep parametric models (i.e. CNNs) have made breakthroughs. Standard kernels used in DGPs such as radial basis functions (RBFs) are insufficient for handling pixel variability in raw images. In this paper, we build on the recent convolutional GP to develop Convolutional DGP (CDGP) models which effectively capture image level features through the use of convolution kernels, therefore opening up the way for applying DGPs to computer vision tasks. Our model learns local spatial influence and outperforms strong GP based baselines on multi-class image classification. We also consider various constructions of convolution kernel over the image patches, analyze the computational trade-offs and provide an efficient framework for convolutional DGP models. The experimental results on image data such as MNIST, rectangles-image, CIFAR10, Convex-sets and Caltech101 demonstrate the effectiveness of the proposed approaches. We also propose a method to reduce the computational complexity of the model. We sub-sample the number of patches and show the efficiency of the approach on caltech101 dataset.

Contents

Chapter 1

Introduction

The field of machine learning is growing very rapidly in current times. With all the computation resources available its use has become even more feasible. It has been used in many discipline of science where some sort of prediction is required. Some of the discipline where it is very commonly used are robotics, statistics, securities and computational science. With current advancements in machine learning literature such as deep learning which works too well when huge amount of data is available, development of autonomous security systems is boosted. It exploits the development of Deep learning in computer vision applications. In statistics, machine learning is being used from quite some time. Its being used in predicting the trends in data, such as fitting a curve(regression), time series analysis and various other applications. With the help of kernel methods and non parametric approaches machine learning works good even when the labeled data is quite small such as in the field of medical imagining.

1.1 Bayesian Perspective of machine learning

The work done in this thesis mainly focuses on bayesian perspective of machine learning. The main benefit of doing things in bayesian way is that along with the class labels(in case of supervised learning) it gives the uncertainty estimate too. The main rule around which whole bayesian machine learning lies is Bayes rule. Lets look into the Bayes theorem of probability theory with machine learning perspective:

$$\mathrm{P}(Y \mid X) = \frac{\mathrm{P}(X \mid Y)\mathrm{P}(Y)}{\mathrm{P}(X)}$$

where P$(Y \mid X)$ is posterior probability of Y given X, P$(X \mid Y)$ is likelihood term, P(Y) is prior probability of Y, and the denominator P(X) represents the marginal likelihood. In typical machine learning scenario Y is used for representing labels and X is used for input vector. For the sake of uniformity we also keep the notation same until/unless specified. So in typical machine learning cases we try to learn this posterior so we can generalize it for unseen data(X^*) too.

In bayesian machine learning we start with a belief, that is prior over the parameters. Then we observe the data that is likelihood, and then we use the likelihood to update our prior belief. The updated belief is known as the posterior. In frequentists way of doing thing we just try to maximize the likelihood with respect to model parameters (maximum likelihood approach) but in bayesian approach of doing things we try to maximize the posterior(maximum a posteriori approach). This method is also not the full bayesian approach. In full bayesian approach we marginalize out all the parameters of the model and maximize the marginal likelihood with respect model hyperparameters. Often the calculation of the posterior is not easy because of the normalization term (marginal) in the denominator in which integration is involved. We thus require various approximate approaches like sampling, variational inference and expectation maximization.

So be it a regression problem or a classification one, supervised or unsupervised, the most important task to do in machine learning is a learning an underlying function that maps the input to output most accurately. As mentioned in [1] there are prominently two main ways doing such kind of function learning. Firstly, we can either the select some fixed set of functions like linear, quadratic or etc. This approach has quite obvious problem the function, we need to have a quite strong domain knowledge to make such assumptions. If we don't have strong domain knowledge we might end up learning a function which doesn't represent the data well enough and hence the predictions too won't be good. And also we take too smooth a function then there is chances of under-fitting and if we give too much flexibility to the underlying function then there is chances of over-fitting. The other approach is consider all the functions in the space and put a distribution over these functions. Here the function which has higher chances of reproducing the underlying mapping will have higher probability. To follow this considering all the functions of certain types in space, gaussian processes (GPs) is the way. A more detailed discussion on GPs will be there in the upcoming chapters. The second approach discussed here is oftenly termed as bayesian non parametric way of machine learning. The name comes from the fact that in this approach we don't consider any prefixed parametric form of the underlying function.

1.2 Outline of the Thesis

The main contribution of this thesis is to bring a new modeling approach for making existing framework of GPs on image data work better. The model is developed on top of DGPs [2] and convolutional gaussian processes(CGPs) [3]. The work present in this thesis requires some mathematical background in Linear Algebra and Probability. Since this model is developed on top of other models it has some prerequisites. All the prerequisites required to understand the work done is described in thesis. Detailed description of the following chapters is given below:

- Chapter 2 discuss about basics of GPs and how it used for regression and classification task. We further go on to discuss some common problem associated with scaling of GPs. The work done in [4] was a breakthrough in GP literature, as it helps in scaling it up for large datasets too. The text in these chapters is highly inspired by [1] and [4].

- Chapter 3 talks about the recent development of convolutional kernel used as covariance function of GPs. This chapter also talks about various variants of convolutional kernel as discussed in [3].

- Chapter 4 discusses the details about how DGPs are formed and some properties of GPs. This chapter involves the lower bound estimation of GPs. This also talks in detail about the first proposed method of inference for DGPs as done in [5] and the method used in this work as done in [2].

- In Chapter 5 convolutional deep gaussian processes is being discussed. Here we discuss, how the convolutional kernel is incorporated in DGPs framework and effect of using it in image datasets. Comparison are being drawn on the performance with already existing state of art methods for GPs.

1.3 Associated Publication and Software

The work submitted in this thesis has been arxived with the title **Deep Gaussian Processes with convolutional kernel** [6]. The work is also under review for workshop Uncertainty in Deep Learning, at Uncertainty in Artificial Intelligence(UAI),2018.

The code for developing the model and reproducing the results as mentioned in the article are present in author's Github page.

Chapter 2

Gaussian Processes

A GP is defined as a collection of random variables such that any finite subset of which is Gaussian distributed [1]. It allows one to specify a prior distribution over real valued functions f, represented as $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ where $m(\mathbf{x})$ is the mean function and $k(\mathbf{x}, \mathbf{x}')$ provides the covariance across the function values at two data points $\mathbf{x}$ and $\mathbf{x}'$. The mean function $m(\mathbf{x})$ is essentially nothing but the average value of all the sampled functions from given GP at particular given value of $\mathbf{x}$. So mean function sometimes gives the bias effect to the learned functions. Mostly people typically uses a zero(which is a constant) mean function in GP literature. Rarely a linear function $\mathrm{m}\mathbf{x} + \mathbf{b}$ is used which helps in accounting for variability in data.
Covariance functions play a very important role in gaussian processes as it determines the class of functions to be learned. As it holds such an importance in the literature it is discussed separately in the following section.

2.1 Covariance Functions

By definition of general kernel function, any function that maps two vector or scalar (inputs) $\mathbf{x} \sim \mathbf{X}$ and $\mathbf{x}^{'} \sim \mathbf{X}$ to a scalar $\mathbb{R}$ is kernel. Not all such functions can be used as a covariance function for GPs. All the kernels functions in which the resulting covariance matrix is positive semi definite (PSD) can be used as a covariance function. The parameters of covariance function and mean function are termed as model hyper-parameters. The kernel function determines various properties of the function such as stationarity, smoothness etc. We will discuss certain types of covariance functions commonly used in GPs.

- **Stationary Covariance Function:** These covariance functions are some function of $x - x^{'}$. This form of kernel makes it invariant to transnational transformations. One popular kernel in such category is square exponential kernel. The functional form of which is given below:

$$k(x, x^{'}) = \sigma_f^2 \exp(-\frac{1}{2\kappa}||\mathbf{x} - \mathbf{x}'||^2)$$

- **Isotropic:** These are more stricter version of stationary covariance function. These are only function of $x - x^{'}$. Squared exponential discussed above comes to this category too.
- **Dot Product Covariance function:** In this class of covariance function, resulting scalar is related to x and $x^{'}$ only through a product relation. Example of this is:

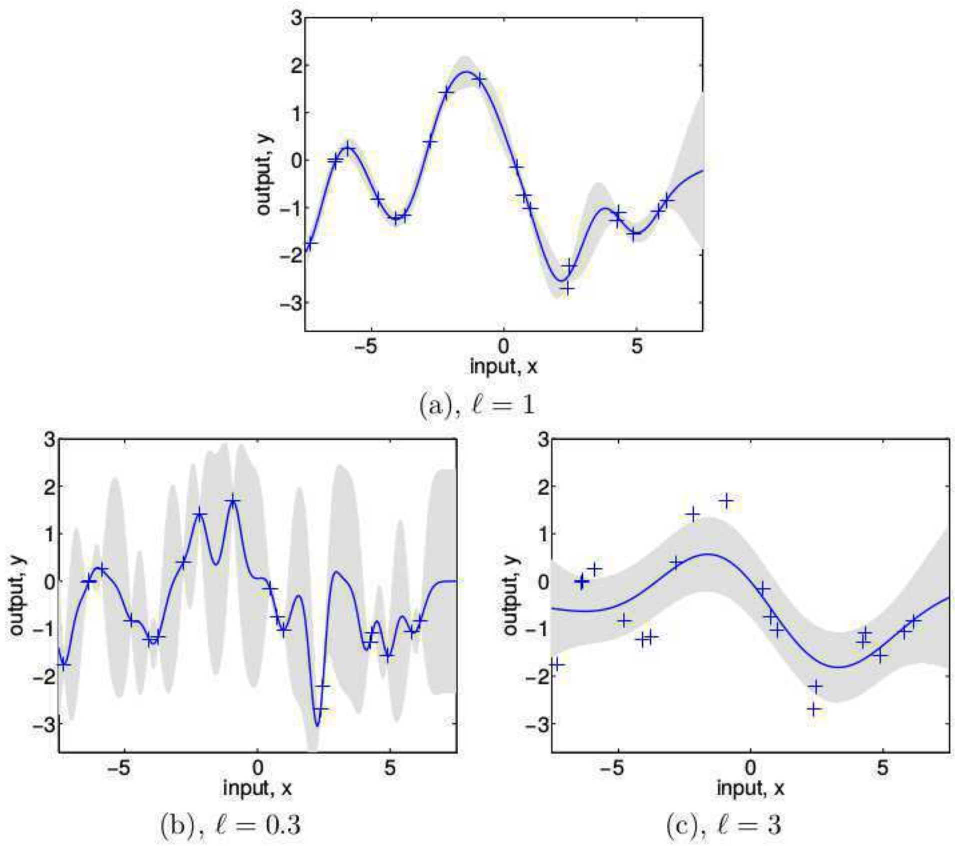
$$k(x, x^{'}) = \sigma_\circ^2 + x \cdot x^{'}$$

One of the many important covariance function that is used in literature in radial basis function (RBF) (squared exponential kernel), as it can model any smooth function. We will discuss a bit more about it in detail here as this kernel is used many times in this work. The term σ_f^2 is the noise variance term in the kernel which essentially acts as scaling factor in the kernel. Term κ determines the variations in function values across the inputs.
One popular variant of RBF kernel is Automatic Relevance Determination enabled RBF kernel. In this variant of RBF we take different lengthscales for each dimensions of inputs. This enables the model to have flexibility of assigning different relevance to different dimensions. The functional form this is given below:

$$k(x, x^{'}) = \sigma_{ard}^2 \exp\big(-\frac{1}{2}\sum_{q=1}^{Q} w_q(x_q - x_q^{'})^2\big)$$

In the above equation w_q represents the inverse of lengthscale corresponding to the q$^{\text{th}}$ dimension. The use of this kernel is a bit restricted as this increase the number of hyperparameters in the model quite significantly. But when there is ample data to train a model this can be used because of its helpfulness in increasing flexibility of the model.

During optimization we learn model hyperparameters which plays a very important role in differentiating the functions we learn for a particular kernel. Lets look into the effect of hyperparameters on the learned functions. We will take an example of lengthscale value in case of squared exponential kernel.



(a), $\ell = 1$

(b), $\ell = 0.3$

(c), $\ell = 3$

Figure 2.1: Effect of lengthscale on functions sampled [1].

As we can see from Fig. 2.1 as the lengthscale is increasing the learned function becomes more and more less wiggly for RBF kernel. This is also quite evident from the functional form of the kernel, as the lengthscale increase lesser amount of weight is given (as it appears in denominator) to the factor of distance value of points.

2.2 GP regression

In GP regression we consider y to be having form $y = f(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ and $f \sim \mathcal{N}(0, K)$
Here K is the covariance matrix of gaussian process.

As depicted in Fig. 2.2 posterior $p(f|x, y) \propto p(y|f, x) \cdot p(f|x)$ will also be gaussian since both prior and likelihood is gaussian.

$$p(f|x, y) \sim \mathcal{N}\Big(K(K + \sigma^2 I)^{-1} y, \sigma^2 K(K + \sigma^2 I)^{-1}\Big)$$

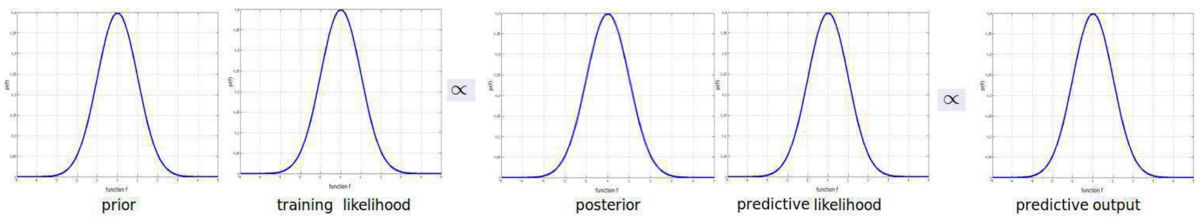


Figure 2.2: Computation of predictive distribution.

Now for consider the predictive distribution to be f_*. Then both f and f_* will be jointly gaussian

$$\begin{bmatrix} f \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

marginalizing f from the conditional distribution of f_*,

$$p(f_*|X,y,X_*) = \int p(f_*|f) \cdot p(f|X,y,X_*)df$$

We get the predictive distribution as follows,

$$f_*|X,y,X_* \sim \mathcal{N}(\bar{f}_*, cov(f_*)), \text{ where}$$

$$\bar{f}_* = K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1} y$$

$$cov(f_*) = k(X_*,X_*) - k(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1} K(X,X_*)$$

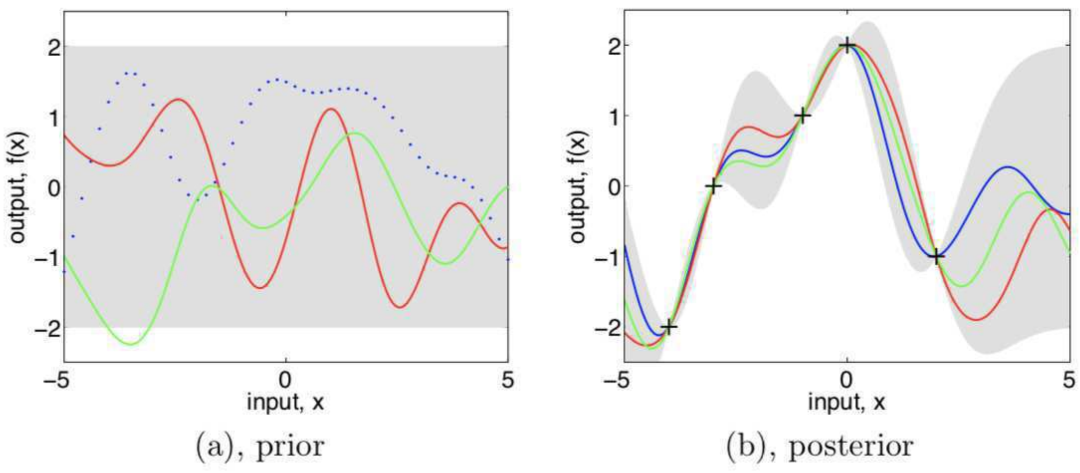


Figure 2.3: Sampled functions from prior and posterior [1].

The visual interpretation of above eqautions are described in Fig. 2.3(b). We can see that the variance near observed data point is nearly zero from equation too as at these points X_* will be equal to X. Puting this in equation fetches zero variance.

To learn hyperparameters we need to maximize marginal likelihood $p(\mathbf{y}) = \int p(y|F)$ $p(F)dF$. The complexity of this model is $O(n^3)$ since we need to invert an $N \times N$ matrix for the computation of covariance matrix of posterior and predictive distribution. Such a high complexity restricts the scaling of the model. We will discuss the sparse GP approach [4] in the next section which helps in tackling the scaling problem of GPs.

2.3 Sparse GP Approximation

We are going to take an example of multiclass classification for discussing Sparse GP as it will be helpful throughout this thesis because experiments done in thesis are on classifications task too. It will also help in giving an insight how classification tasks differs from regression in GP literature.

For multi-class classification problems, we associate a separate function f_c with each class c. An independent GP prior is placed over each of these functions, $f_c(\mathbf{x}) \sim \mathcal{GP}(m_c(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. Let $\mathbf{f_c} = [f_c(\mathbf{x}_1), f_c(\mathbf{x}_2), \cdots, f_c(\mathbf{x}_N)]$ be a column vector indicating function values at the input data points for a class c. Further, let F be the matrix formed by stacking all column vectors $\{\mathbf{f_c}\}_{\mathbf{c=1}}^{\mathbf{C}}$, with $F_{n,c}$ representing the latent function value of n^{th} sample belonging to class c and F_n representing the vector of latent function values over classes for the n^{th} sample. The GP prior over F takes the following form : $p(F) = \prod_{c=1}^{C} \mathcal{N}(\mathbf{f}_c; m_c(X), K_{XX})$, where K_{XX} is the $N \times N$ covariance matrix formed by evaluating kernel over all pairs of training data points. For a data point n, the likelihood of it belonging to class c, $p(y_n = c|F_n)$, is obtained by considering a soft-max link function. The posterior distribution over F is obtained by combining the prior and the likelihood using Bayes theorem:

$$p(F|\mathbf{y}) = \frac{\prod_{n=1}^{N} p(y_n|F_n)p(F)}{p(\mathbf{y})}.$$

In GP multi-class classification, the posterior distribution cannot be computed in closed form due to the non-conjugacy between likelihood and prior. Learning in GPs involves learning the kernel hyper-parameters by maximizing the evidence

$p(\mathbf{y}) = \int \prod_{n=1}^{N} p(y_n|F_n)p(F)dF$, which also cannot be computed in closed form. The posterior distribution can be approximated as a Gaussian using approximate inference techniques such as Laplace approximation [7] and variational inference [8, 9, 10]. The Gaussian approximated posterior is then used to make predictions on the test data points. Variational inference has received a lot interest recently as it does not suffer from convergence problems unlike Markov chain Monte Carlo techniques and it provides a posterior approximation quickly by solving an optimization problem. It is scalable to large data sets and amenable to distributed processing. It also provides a lower bound on the marginal likelihood which can be used to perform model selection. The variational inference approach learns an approximate posterior distribution $q(F)$ by minimizing the KL divergence between $q(F)$ and $p(F|\mathbf{y})$. Choosing a mean field family of variational distributions, $q(F)$ factorizes across dimensions(or columns), i.e $q(F) = \prod q(\mathbf{f}_c)$. Each variational factor $q(\mathbf{f}_c)$ is assumed to be a Gaussian with variational parameters, mean vector $\mu_{\mathbf{c}}$ and covariance Σ_c. In the variational inference framework, minimizing the KL divergence with respect to the variational parameters is equivalent to maximizing the so-called variational **E**vidence **L**ower **BO**und**(ELBO)** which is given by

$$L(\{\mu_{\mathbf{c}}, \Sigma_c\}_{c=1}^{C}) = \mathbb{E}_{q(F)}[\log \prod_{n=1}^{N} p(y_n|F_n)] - \sum_{c=1}^{C} \mathrm{KL}(q(\mathbf{f}_c) \parallel p(\mathbf{f}_c)). \tag{2.1}$$

The variational parameters $\{\mu_{\mathbf{c}}, \Sigma_c\}_{c=1}^{C}$ and the kernel hyperparameters $\{\sigma_f^2, l\}$ are learnt by jointly maximizing the variational lower bound in eq. (2.1) using any gradient based approach.

The KL divergence term in eq. (2.1) involves inversion of the covariance matrix K_{XX} which scales as $\mathcal{O}(N^3)$ computationally. Therefore, we opt for the variational sparse Gaussian process approximation [4, 11] which reduces the computational complexity to $\mathcal{O}(NM^2)$, where $M \ll N$ represents the number of inducing points. Specifically, the variational sparse approximation expands the latent function space with M inducing variables $\mathbf{u} \in \mathcal{R}^M$ which are latent function values at inducing points $Z = \{\mathbf{z}_i\}_{i=1}^{M}$. Within the context of GP multi-class classification, we additionally have the inducing variable outputs $\mathbf{u}_c$ for each class c which are stacked together to form the matrix $U \in \mathcal{R}^{M \times C}$. The joint GP prior over $\{f, u\}$ is then

$$\begin{bmatrix} \mathbf{f}_c \\ \mathbf{u}_c \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \mathbf{f}_c \\ \mathbf{u}_c \end{bmatrix}; \begin{bmatrix} m_c(X) \\ m_c(Z) \end{bmatrix}, \begin{bmatrix} K_{XX} & K_{XZ} \\ K_{XZ}^{\top} & K_{ZZ} \end{bmatrix}), \tag{2.2}$$

where K_{XZ} is the $N \times M$ covariance matrix over training inputs X and inducing inputs Z and K_{ZZ} is the $M \times M$ covariance matrix over inducing points Z. The conditional distribution of $\mathbf{f}_c$ given $\mathbf{u}_c$ is given by

$$p(\mathbf{f}_c|\mathbf{u}_c, X, Z) = \mathcal{N}(\mathbf{f}_c; m_c(X) + K_{XZ}K_{ZZ}^{-1}(\mathbf{u}_c - m_c(Z)), K_{XX} - K_{XZ}K_{ZZ}^{-1}K_{XZ}^{\top})$$

and the marginal distribution over $\mathbf{u}_c$ is $p(\mathbf{u}_c) = \mathcal{N}(\mathbf{u}_c; m_c(Z), K_{ZZ})$. The variational sparse approximation of [11] considers a joint variational posterior over $\{\mathbf{f}_c, \mathbf{u}_c\}$ in factorized form and is written as $q(\mathbf{f}_c, \mathbf{u}_c) = p(\mathbf{f}_c|\mathbf{u}_c, X)q(\mathbf{u}_c)$. Assuming Gaussian variational factors for inducing points $q(\mathbf{u}_c) = \mathcal{N}(\mathbf{u}_c; \mathbf{m_c}, S_c)$, the variational lower bound (ELBO) can be derived as

$$L(\{\mathbf{m_c}, S_c\}_{c=1}^{C}) = \mathbb{E}_{q(F)}[\log \prod_{i=1}^{N} p(\mathbf{y}_n|F_n)] - \sum_{c=1}^{C} \mathrm{KL}(p(\mathbf{u}_c)||q(\mathbf{u}_c)). \tag{2.3}$$

Following [9], the variational posterior $q(F) = \prod_{c=1}^{C} q(\mathbf{f}_c)$ and $q(\mathbf{f}_c)$ is obtained by integrating out $\mathbf{u}_c$ from $p(\mathbf{f}_c|\mathbf{u}_c)q(\mathbf{u}_c)$ and is given by $\mathcal{N}(\mathbf{f}_c; \tilde{\mathbf{m}}_\mathbf{c}, \tilde{V}_c)$, where $\tilde{\mathbf{m}}_\mathbf{c} = m_c(X) + K_{XZ}K_{ZZ}^{-1}(\mathbf{m_c} - m_c(Z))$ and $\tilde{V}_c = K_{XX} - k_{XZ}K_{ZZ}^{-1}(K_{ZZ} - S_c)K_{ZZ}^{-1}K_{ZX}$. The Expected Log likelihood term above is intractable due to non-conjugate likelihood (softmax in this case). One could apply a quadrature [9] or reparameterization-based [12] monte carlo sampling scheme to approximate this.

Chapter 3

Convolution Gaussian Processes

In this chapter we will discuss about recently introduced convolutional Gaussian processes (CGP) [3] where the function evaluation on an image is considered as sum of functions over the patches of the input image. Assuming there are P patches in $\mathbf{x}$ with each patch $\mathbf{x}^{[p]}$ to be $w \times h$ dimensional, CGP considers $f(\mathbf{x}) = \sum\limits_{p=1}^{P} g(\mathbf{x}^{[p]})$. Placing a zero mean $\mathcal{GP}$ prior over the function $g(\mathbf{x}^{[p]})$, $g(\mathbf{x}^{[p]}) \sim \mathcal{GP}(0, k_g(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p]}))$, induces a zero mean $\mathcal{GP}$ prior over the function $f(\mathbf{x})$ with a convolutional kernel (Conv kernel) k_f,

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k_f(\mathbf{x}_i, \mathbf{x}_j)), \qquad k_f(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p=1}^{P} \sum_{p'=1}^{P} k_g(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p']}). \tag{3.1}$$

We refer to k_g as the base kernel. Considering a convolutional kernel in computing the similarities between the images is useful in capturing non-local similarities among the images. The convolutional kernel compares one region in the image $\mathbf{x}_i$ with another region in the image $\mathbf{x}_j$, and could provide a high similarity even under transformations in the image. The kernel computation over patches $(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p']})$ considers similarity in a spatial neighborhood, whereas with other kernels (such as RBF kernel) only global similarity across images can be computed and fails to capture similarity in images due to transformations.

Convolutional Neural Networks(CNNs) convolve image with multiple kernels (filters), apply a non-linear operation and then feature pooling (average, max) multiple times to learn discriminative features useful for the object detection task. Similar to CNN, the function $f(\mathbf{x})$ could be seen to perform average pooling of the non-linear feature maps produced by the patch response functions $g(\mathbf{x}^{[p]})$. This pooling operation results in convolution operation in kernel space. The convolutional kernel computation between two images $\mathbf{x}_i$ and $\mathbf{x}_j$ is expanded as

$$k_f(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p'=1}^{P} k_g(\mathbf{x}_i^{[1]}, \mathbf{x}_j^{[p']}) + \ldots + \sum_{p'=1}^{P} k_g(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p']}) + \ldots + \sum_{p'=1}^{P} k_g(\mathbf{x}_i^{[P]}, \mathbf{x}_j^{[p']}).$$

The convolution operation between p^{th} patch of image $\mathbf{x}_i$ (which now acts as a filter) and the image $\mathbf{x}_j$ results in a convolution signal, where signal value at any point p' is obtained by computing the dot product between the filter $\mathbf{x}_i^{[p]}$ and patch $\mathbf{x}_j^{[p']}$. This dot product is performed by the base kernel which transforms these patches into feature vectors in a high dimensional space and computes the dot product between them in that space. Any p^{th} summand is the sum of the convolution signal values obtained at all the points.

3.1 Inducing Patch Space

The motivation behind introducing inducing patch space in [3] is computational cost. Apart from usual N^3 cost of k_f(since N^2 is needed to be inverted), there is computational cost of for each element of k_f which is P^2 evaluations, where P is the patch size. Now if the sparse GP approximation is brought into the image space, cost will be reduced from N^3 to M^2N but P^2 evaluations required to compute each element of k_f will be still there.

In [3] a novel method to bring inducing variable to patch space is being proposed. The motivation of this work is being drawn from inter-domain GPs. Now having inducing variable being in a patch space means each z is a patch not an image. The formulation of both k_{uu} and K_{fu} is changed. The expression to compute elements of each is given below:

$$k_{uu}(z, z^{'}) = \mathbb{E}_g[g(z)g(z^{'}] = k_g(z, z^{'}) \tag{3.2}$$

$$k_{fu}(x, z) = \mathbb{E}_g[f(x)g(z)] = \mathbb{E}_g\Big[\sum_p g(x^{[p]})g(z)\Big] = \sum_p k_g(x^{[p]}, z) \tag{3.3}$$

In above equations $x^{[p]}$ represents the p$^{\text{th}}$ patch of image x and **z is a patch not image.**

3.2 Variants of Convolutional Kernel

In [3] some variants of convolutional kernel is being presented which result in better performance of the model. We here will discuss some of those variants which is being used in the work presented in this thesis.

3.2.1 Weighted Convolutional Kernel

In this variant of convolutional kernel we assign weight corresponding to each patch of the image. This gives different importance to different parts of the image. This increases the number of hyperparameters in kernel by the factor of p where p is the number of patches in the image. The corresponding value of k_f and k_{fu} is defined as follows:

$$k_f(x, x^{'}) = \sum_{p} \sum_{p^{'}} w_p w_{p'} k_g(x^{[p]}, x^{'[p^{'}]}) \tag{3.4}$$

$$k_{fu}(x, z) = \sum_{p} w_p k_g(x^{[p]}, z) \tag{3.5}$$

3.2.2 Convolutional kernels for Colored images

Colored images have a much more lager dimensionality as compared to the one in gray scale. The challenge here is the increased dimensionality and how various channels interacts with each other. One obvious way of doing things for colored images is considering all the patches from all the channels. This will increase the number of patches by a factor of C where C is the number of channels. One other way around is to consider all the channels for a particular patch i.e. making the patch size to be $W \times H \times C$ where W, H and C are width, height and channels respectively. The problem with the above two approaches is that it does not consider different non-linearity for different channel. This kind of assumption restricts the model representability. As proposed in [3] as multi-channel convolutional kernel, capture different non-linearity for different channels.

Chapter 4

Deep Gaussian Processes

Deep Gaussian processes (DGPs) [5, 13, 2] learn complex functions by stacking GPs one over the other resulting in a deep architecture of GPs. The function mapping one hidden layer to the next in DGPs is more expressive and data dependent compared to the pre-fixed sigmoid non-linear function used in standard parametric deep learning approaches. In addition, it is devoid of large number of parameters but only a few kernel hyper-parameters and few variational parameters (few due to sparse GP approach). Deep GPs do not typically overfit on small data due to Bayesian model averaging, and the stochasticity inherent in GPs naturally allows them to handle uncertainty in the data. Furthermore, by using a specific kernel which enables automatic relevance determination, one can automatically learn the dimensionality of hidden layers (number of neurons) [13]. This overcomes the model selection problem in deep learning to a great extent.

DGPs consider the function mapping input to output to be represented as a composition of functions, $f(\mathbf{x}) = f^L \circ (f^{L-1} \ldots \circ (f^1(\mathbf{x})))$, assuming there are L layers. The l^{th} layer consists of D^l functions $f^l = \{f^l_j\}_{j=1}^{D^l}$ mapping representations in layer $l-1$ to obtain D^l representation for layer l. Independent GP priors are placed over the function f^l_j producing j^{th} representation in layer l, $f^l_j(\cdot) \sim \mathcal{GP}(m^l_j(\cdot), k^l(\cdot,\cdot))$. The j^{th} function in layer l, f^1_j, acts on the input data point $\mathbf{x}_i$ to produce the representation $F^1_{i,j} = f^1_j(\mathbf{x}_i)$. In general, the j^{th} function in layer l, $f^l_j(\cdot)$ acts on the representation of the data point $\mathbf{x}_i$ at layer $l-1$, F^{l-1}_i to produce the representation $F^l_{i,j} = f^l_j(F^{l-1}_i)$. Let $\mathbf{f}^l_j$ denote the j^{th} representation at layer l computed over all inputs. The final layer L will have C functions corresponding to the classes and these functions values are squashed through a soft-max function to produce the class probabilities.

We follow the DGP variant presented in [2] where the noise between layers is absorbed into the kernel. The kernel function associated with a GP in layer l is defined as $k^l(F_i^l, F_j^l) = {\sigma_f^l}^2 \exp(\frac{-1}{2\kappa^l}||F_i^l - F_j^l||^2) + {\sigma_n^l}^2 \delta_{ij}$. Following the variational sparse Gaussian process approximation as explained in the section **??**, each layer l is associated with inducing variables $\{U^l\}$ which are function values over M inducing points Z^l associated with layer l, $Z^l = \{\mathbf{z}_i^l\}_{i=1}^M$. Let $\mathbf{u}_j^l$ represent the inducing variables associated with the j^{th} representation at layer l. The number of inducing points are kept fixed for all layers (only for convenience) as M and a joint GP prior is considered over latent function values and inducing points. The joint distribution $p(\mathbf{y}, F, U)$ is given by

$$\underbrace{\prod_{n=1}^{N} P(y_n|F_n^L)}_{\text{Likelihood}} \underbrace{\prod_{l=1}^{L} \prod_{j=1}^{D^l} p(\mathbf{f}_j^l|\mathbf{u}_j^l, F^{l-1}, Z^l) p(\mathbf{u}_j^l|Z^l),}_{\text{Deep GP Prior}} \tag{4.1}$$

where a deep GP prior is put recursively over the entire latent space with $F^0 = X$ and a soft-max likelihood is used for classification. The conditional above is:

$$\begin{aligned} &p(\mathbf{f}_j^l|\mathbf{u}_j^l, F^{l-1}, Z^l) = \mathcal{N}(\mathbf{f}_j^l; mean(\mathbf{f}_j^l), cov(\mathbf{f}_j^l)) \quad \text{where} \\ &mean(\mathbf{f}_j^l) = m_j^l(F^{l-1}) + K_{F^{l-1}Z^l}^l (K_{Z^l Z^l}^l)^{-1} (\mathbf{u}_j^l - m_j^l(Z^l)) \\ &cov(\mathbf{f}_j^l) = K_{F^{l-1}F^{l-1}}^l - K_{F^{l-1}Z^l}^l (K_{Z^l Z^l}^l)^{-1} (K_{F^{l-1}Z^l}^l)^\top \end{aligned} \tag{4.2}$$

The posterior distribution $p(F, U|\mathbf{y})$ and marginal likelihood $p(\mathbf{y})$ cannot be computed in closed form due to the intractability in obtaining the marginal prior over $\{F^l\}_{l=2}^L$. This involves integrating out the previous layer, which is present in a non linear manner inside the covariance matrices $(K_{F^{l-1}F^{l-1}}^l)$ appearing in (4.2). Along with non-conjugate likelihood, this brings in additional difficulty to the DGP model. Multiple approaches have been suggested in the literature for achieving tractability in DGPs, such as variational inference [5, 14, 2], amortized inference [15], expectation propagation [16] and random Fourier features [17]. Here we follow the variational inference approach, and we assume the variational posterior to be having form $q(F, U) = \prod_{l=1}^{L} \prod_{j=1}^{D^l} p(\mathbf{f}_j^l|\mathbf{u}_j^l, F^{l-1}, Z^l) q(\mathbf{u}_j^l)$, where $q(\mathbf{u}_j^l) = \mathcal{N}(\mathbf{u}_j^l; \mathbf{m}_j^l, S_j^l)$ [11, 5, 4]. Let $\mathbf{m}^l$ be a vector formed by concatenating the vectors $\mathbf{m}_j^l$ and S^l be the block diagonal covariance matrix formed from S_j^l. We can formulate the **ELBO** by extending the methodology described in Section **??** to multiple layers [5, 2] as follows:

$$L(\{\mathbf{m}^l, S^l\}_{l=1}^L) = \sum_{n=1}^{N} \mathbb{E}_{q(F_n^L)}[\log p(y_n|F_n^L)]_- \sum_{l=1}^{L} KL[q(U^l)||p(U^l)] \tag{4.3}$$

where, the marginal distribution of the functions values for layer L over all the data points is obtained as

$$q(F^L|\{Z^l, \mathbf{m}^l, S^l\}_{l=1}^L) = \int\limits_{F^1, F^2, \dots F^{L-1}} \prod_{l=1}^{L} q(F^l|F^{l-1}, Z^l, \mathbf{m}^l, S^l) dF^1 \dots dF^{L-1} \quad (4.4)$$

and the conditional distribution in (4.4) is computed as

$$q(F^l|F^{l-1}, Z^l, \mathbf{m}^l, S^l) = \prod_{j=1}^{D^l} \int\limits_{\mathbf{u}_j^l} p(\mathbf{f}_j^l|\mathbf{u}_j^l, F^{l-1}, Z^l) q(\mathbf{u}_j^l) d\mathbf{u}_j^l = \prod_{j=1}^{D^l} \mathcal{N}(\mathbf{f}_j^l; \tilde{\mathbf{m}}_j^l, \tilde{V}_j^l) \quad (4.5)$$

$$\text{where } \tilde{\mathbf{m}}_j^l = m_j^l(F^{l-1}) + K_{F^{l-1}Z^L}^l (K_{Z^lZ^l}^l)^{-l} (\mathbf{m}_j^l - m_j^l(Z^l)) \text{ and} \quad (4.6)$$

$$\tilde{V}_j^l = K_{F^{l-1}F^{l-1}}^l - K_{F^{l-1}Z^l}^l (K_{Z^lZ^l}^l)^{-l} (K_{Z^lZ^l}^l - S_j^l)(K_{Z^lZ^l}^l)^{-l} (K_{F^{l-1}Z^l}^l)^\top. \quad (4.7)$$

The marginal distribution in (4.4) is intractable, due to presence of stochastic term $\{F^{l-1}\}_{l=2}^L$ inside the conditional distributions $\{q(F^l|F^{l-1}, Z^l, \mathbf{m}^l, S^l)\}_{l=2}^{L-1}$ in a non-linear manner. This intractability results in the expected log likelihood in (4.3) to be intractable even for Gaussian likelihood. We approximate it via Monte Carlo sampling as done in [2].

As has been shown in [2], the marginal variational posterior over function values in the final layer for n^{th} data point, i.e $q(F_n^L)$ depends only on the n^{th} marginals of all the previous layers. Each F_n^l is sampled from $q(F_n^l|F_n^{l-1}, Z^l, \mathbf{m}^l, S^l) = \mathcal{N}(F_n^l; \tilde{\mathbf{m}}^l[n], \tilde{V}^l[n])$, where $\tilde{\mathbf{m}}^l[n]$ (D^l dimensional vector) and $\tilde{V}^l[n]$ ($D^l \times D^l$ diagonal matrix) are respectively the mean and covariance of the n^{th} data point over representations in layer l and depends on F_n^{l-1}. Applying the "reparametarization trick" the sampling can be written as:

$$F_n^l = \tilde{\mathbf{m}}^l[n] + \boldsymbol{\epsilon}^l \odot \tilde{V}^l[n]^{\frac{1}{2}}; \quad \boldsymbol{\epsilon}^l \sim \mathcal{N}(\boldsymbol{\epsilon}^l; 0, \mathbb{I}_{D^l}).$$

The lower bound can be written as sum over data points and the parameters can be updated based gradients computed on a mini-batch of data. This enables one to use stochastic gradient techniques for maximizing the variational lower bound. This stochasticity in gradient computation combined with the stochasticity introduced by the Monte Carlo sampling in variational lower bound computation results in the doubly stochastic variational inference method for deep GPs.

Chapter 5

Deep Convolutional Gaussian Processes

5.1 Introduction

Deep learning models have made tremendous progress in computer vision problems through their ability to learn complex functions and representations [18]. They learn complex functions mapping some input x to output y through composition of linear and non-linear functions. However, popular deep learning models based on convolutional and recurrent neural networks have significant limitations. The parametric form of the functions lead them to have millions of parameters to estimate which is less suitable for problems where the data are scarce. Deep learning models though probabilistic in nature, do not provide any uncertainty estimates on its predictions. Knowledge of uncertainty helps in better decision making and is crucial in high risk applications such as disease diagnosis and autonomous driving [19]. Another major limitation with the existing deep learning networks is model selection. Developing an appropriate deep learning model to solve a problem is time consuming and computationally expensive. Deep Gaussian processes (DGPs) [5] constitute a deep Bayesian non-parametric approach based on Gaussian processes (GPs) and have the potential to overcome the aforementioned limitations.

The original DGP model was introduced by [5, 13] inspired by the hierarchical GP-LVM structure [20] and variations have emerged in recent years, mainly differing in the employed inference procedure. While [5] employs a mean field variational posterior over the latent layers, [15] extends this formulation with amortized inference, [14] considers a nested variational inference approach, [16] uses an approximate

Expectation Propagation procedure. Further, [17] achieves scalability through random Fourier features while the approach of [2] considers the variational posterior to be conditioned over the previous layer, preserving correlations across the layers, and uses a doubly stochastic variational inference approach.

All the DGP models use kernels such as radial basis function (RBF) which is inadequate for problems in computer vision, such as object detection. They fail to capture wide variability of objects in images due to pose, illumination and complex backgrounds. RBF captures similarity between images on a global scale and is not invariant to unwanted variations in the image. On the other hand, convolutional neural networks (CNN) [18] learn image representations from raw pixel data which are invariant to such perturbations in the image. They learn features important for the object detection task by successively convolving the representations by filters, applying non-linearity and performing feature pooling. We propose to use convolutional kernels [21] in DGPs to learn salient features from the images which are invariant to transformations. This is different from recent works which combine CNNs and GPs in hybrid mode, such as [22, 23, 24]. In particular, [23] replaces the fully connected layers of a CNN with GPs, aiming at obtaining well-calibrated probabilities. While, in deep kernel learning [22], the kernel in GPs are computed using deep neural networks. In contrast, our approach brings the convolutional structure inside the deep GP model, through kernels, and remains fully non-parametric.

Convolutional kernels could effectively learn rich representations of the data. The similarity between structured objects such as images are computed by considering the similarity of the sub-structures in the object which makes them invariant to transformations in the image. They have been used to compute similarities between structured objects such as graphs and trees [25, 26]. Recently, they were used as a covariance function in GPs and were found to be very effective for object recognition tasks [3]. Here, the kernel computations between images are done by summing the base kernel acting over different patches of the images.

We introduce convolutional kernels in the DGP framework in order to extract discriminative features from images for object classification. Our work builds on the convolutional GP [3] and extends it for the deep learning case, allowing the resulting model to additionally perform hierarchical feature learning. We consider various DGP architectures obtained by stacking together convolutional and RBF kernels in various combinations. Further, we consider variants of the convolutional kernel such as weighted convolutional kernels which provide more discriminative features, and combination of RBF kernels as the base kernel. Convolutional kernels are computationally expensive as they require performing summation over all patches of the image. We propose an approach to improve the computational efficiency by random sub-sampling of the patches. We demonstrate the effectiveness of the proposed approaches for image classification on benchmark data sets such as MNIST, Rectangles-image, CIFAR10, Convex sets and Caltech101. The experiments show that DGP models typically achieve better generalization performance by using convolutional kernels compared to state-of-the-art shallow GP models.

5.2 Model

Convolutional DGP considers multiple functions from a GP prior with convolutional kernels to form a representation of the image in the first layer. The function corresponding to o^{th} representation for layer 1 is obtained as

$$f_o^1(\mathbf{x}) = \sum_{p=1}^{P} g_o^1(\mathbf{x}^{[p]}) \quad ; \quad g_o^1(\mathbf{x}^{[p]}) \sim \mathcal{GP}(m_o^1(\mathbf{x}^{[p]}), k_g^1(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p]})) \tag{5.1}$$

$$f_o^1(\mathbf{x}) \sim \mathcal{GP}(m_o^1(\mathbf{x}), k_f^1(\mathbf{x}_i, \mathbf{x}_j)) \quad ; \quad k_f^1(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p=1}^{P}\sum_{p'=1}^{P} k_g^1(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p']}). \tag{5.2}$$

Each output in layer 1 captures different features of the image. The feature representations of the image obtained in the first layer are then mapped using a GP with convolutional or RBF kernel to obtain further representations. In general, the function corresponding to o^{th} representation for layer l is considered as

$$\begin{aligned} f_o^l(F^{l-1}) &\sim \mathcal{GP}(m_o^l(F^{l-1}), k_f^l(F_i^{l-1}, F_j^{l-1})) \\ k_f^l(F_i^{l-1}, F_j^{l-1}) &= \sum_{p=1}^{P}\sum_{p'=1}^{P} k_g^l(F_i^{l-1[p]}, F_j^{l-1[p']}). \end{aligned} \tag{5.3}$$

The kernel matrices involved in the computation of the conditional distribution in eq. (4.5) such as $K^l_{F^{l-1}F^{l-1}}$, $K^l_{F^{l-1}Z^l}$ and $K^l_{Z^lZ^l}$ use the convolutional kernel defined in (5.3). As before, Z^l represents the inducing points associated with layer l and has the same dimension as F^{l-1}. The variational lower bound expression and "reparameterization trick" remains the same as has been derived for deep GPs in Section ??.

We also consider variants of the convolutional kernel such as weighted convolutional kernels (Wconv kernels) [3]. It associates a weight with each patch which allows the kernel to provide differential weightage to the patches which is useful for object detection. The function $f(\mathbf{x})$ in general for any layer is considered as

$$f(\mathbf{x}) = \sum_{p=1}^{P} w_p g(\mathbf{x}^{[p]}) \quad ; \quad k_f(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p=1}^{P} \sum_{p'=1}^{P} w_p w_{p'} k_g(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p']}). \tag{5.4}$$

5.2.1 Reducing computational complexity through patch subsets

Convolutional kernels provide an effective way to capture the similarity across images, but are computationally expensive. Computing the similarity between two images involves $\mathcal{O}(P^2)$ computational cost, where P is the number of patches in the input image or the feature representation. For the input image of size $W \times H$, it is of the order of $\mathcal{O}(WH)$ when stride length and patch sizes are small. This is costly even for image data sets such as MNIST and rectangles which contain images of size (28×28). This makes the computations impractical on higher dimensional data such as Caltech101 (250×250). This can be addressed to some extent using the idea of treating the inducing points in the patch space [3], where $Z_j^l \in \mathcal{R}^{w \times h}$ rather than in the input space $\mathcal{R}^{W \times H}$. In this case, computation of the entries in the matrix $K_{F^{l-1}Z^l}^l$ can be performed in $\mathcal{O}(P)$ time, and that of $K_{Z^l Z^l}^l$ can be performed in constant time.

$$K_{F^{l-1}Z^l}^l[i,j] = k_f^l(F_i^{l-1}, Z_j^l) = \sum_{p=1}^{P} k_g^l(F_i^{l-1[p]}, Z_j^l) \tag{5.5}$$

$$K_{Z^l Z^l}^l[i,j] = k_g^l(Z_i^l, Z_j^l) \tag{5.6}$$

However, computation of the entries in the matrix $K_{F^{l-1}F^{l-1}}^l$ matrix which appears in the conditional distribution in (4.5) still requires $\mathcal{O}(P^2)$ computations for the first layer making it a costly operation especially. This makes the approach practically inapplicable to high dimensional data sets such as Caltech101 even with a reduced image size. Moreover in these images, a lot of information will be shared by overlapping patches and will be redundant for the computation of the similarity across images. We propose to use random sub-sampling of the patches in computing the convolutional kernel for the entries in the matrix $K_{F^{l-1}F^{l-1}}^l$ and $K_{F^{l-1}Z^l}^l$. Let $S, S' \subset \{1, 2, \ldots, P\}$ represent the random subsets. For the o^{th} representation of layer 1 ($F^0 = X$), we consider the covariance functions to be as follows

$$f_o^1(\mathbf{x}) = \sum_{p \in S} g_o^1(\mathbf{x}^{[p]}) \tag{5.7}$$

$$k_f^1(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p \in S} \sum_{p' \in S'} k_g^1(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p']}) \text{ and} \tag{5.8}$$

$$k_f^1(\mathbf{x}_i, Z_j^1) = \mathbb{E}_g[f_o^1(\mathbf{x}_i) g_o^1(Z_j^1)] = \mathbb{E}_g[\sum_{p \in S} g_o^1(\mathbf{x}^{[p]}) g_o^1(Z_j^1)] \tag{5.9}$$

$$= \sum_{p \in S} k_g^1(\mathbf{x}^{[p]}, Z_j^1) \tag{5.10}$$

This reduces the cost of computing the matrix $K^l_{F^{l-1}F^{l-1}}$ for layer 1 to $\mathcal{O}(|S||S'|)$ where the size of the subsets $|S|, |S'| \ll P$. Computational speedup achieved through random sub-sampling of patches is testified in our experiments on Caltech101.

5.3 Experiments

We evaluate the generalization performance of the proposed model, convolutional deep Gaussian processes (CDGP), on various image classification data sets, namely MNIST, Rectangle-Images, CIFAR10, Convex sets and Caltech101. We consider different kernel architectures of the proposed CDGP model and compare it with sparse GPs (SGP)[1], deep GP (DGP) models with RBF kernel and with convolutional GPs (CGP) red with different convolutional kernels. The convolutional deep GP uses the same inference procedure as in deep GP ("re-parameterization trick") and uses an a priori fixed inducing input points by considering centroids of the clustered images [2]. The inducing points and the linear mean function for each of the inner layers is obtained using the singular value decomposition approach red mentioned in [2]. The number of inducing points is taken to be 100. We follow the same approach for convolutional GPs also to maintain a fair playground. The kernel parameters are kept the same across various outputs in a layer while it is different across the layers. The number of outputs in the latent layers is taken to be 30 for MNIST and 50 for other datasets (except for the final layer which will be equal to the number of classes). For the models considering convolutional kernels, the patch size is taken to be 3×3 with a stride length of 1 for the rectangles data while a patch size of 5×5 is considered

[1]Results as reported in [2].

for the rest of the data sets. We consider the RBF kernel as the base kernel k_g for all our experiments. The approaches are compared in terms of their accuracy in making predictions on the test data and the negative log predictive probability (NLPP) on test data which considers uncertainty in predictions. The code has been developed on top of GPflow [27] framework with ADAM [28] optimizer to learn the kernel and variational parameters by maximizing the variational lower bound. The variational mean parameters are initialized to 0, variance parameters to $1e^{-5}$ and length-scales are initialized to 2 for MNIST and 10 for other datasets.

5.3.1 MNIST-10

We performed experiments with MNIST dataset with 10 classes corresponding to the digits $0-9$. We consider the standard train/test split with 60K training and 10K test images. We considered CDGP and DGP models with various architectures as described in Table 5.1. Parameters of the model are learned by running the ADAM optimizer for 400 epochs with 0.01 step size and a mini-batch of 1000. Experimental comparison indicates that the proposed CDGP models with 2 layers, first layer with a weighted convolutional kernel and the second layer with an RBF kernel gave the best performance, an accuracy of 98.66 and an NLPP score of 0.0463. Second best performance was given again by a CDGP model with 4 layers, 2 weighted convolutional kernels followed by 2 RBF layers. We could observe that all the CDGP models performed better than the DGP and CGP models in the MNIST data. We also conducted experiments with the combinations of two RBF kernels with length scales initialized to 2 different values 0.01 and 10, as the base kernel in a convolutional kernel. The approach gave an accuracy of 98.46. We found that the learned length scales are also quite far apart which shows that one RBF kernel is trying to capture long distance correlations while the other one captures short distant correlations. This did not result in better results as MNIST is quite simple dataset for which capturing such information might not be necessary.

5.3.2 Rectangles-Image

We consider the rectangles-image data set used in [2], where a rectangle of varying height and width is placed inside images. The patches in the border and inside of the rectangle and the background patches are sampled to make the rectangle hard to

Table 5.1: Comparison of SGP, DGP, CGP and CDGP approaches with different architectures on the MNIST data set along with the kernels used by GP in each layer.

Model	Layer 1	Layer 2	Layer 3	Layer 4	Accuracy%	NLPP
SGP	RBF	–	–	–	97.48	–
DGP1	RBF	RBF	–	–	97.94	0.073
DGP2	RBF	RBF	RBF	–	97.99	0.070
CGP1	Conv	–	–	–	95.59	0.170
CGP2	Wconv	–	–	–	97.54	0.103
CDGP1	Wconv	RBF	–	–	**98.66**	**0.046**
CDGP2	Conv	RBF	–	–	98.53	0.536
CDGP3	Conv	RBF	RBF	–	98.40	0.055
CDGP4	Conv	RBF	RBF	RBF	98.41	0.051
CDGP5	Wconv	Wconv	RBF	–	98.44	0.048
CDGP6	Wconv	Wconv	RBF	RBF	98.60	0.046

detect [2]. The task is to classify if a rectangle in an image has a larger height or width. The data set consists of 12K training images and 50K test images, and is known to require deep architectures for correct classification. We consider two different architectures of CDGP, and compare it against sparse GPs, deep GPs with 2 and 3 layers and convolutional GPs. Parameters of the models are learnt by running the ADAM optimizer for 200 epochs with 0.01 step size and a mini-batch of 1000. Experimental comparison across different approaches is provided in Table 5.2. We could observe that the proposed CDGP model with 2 layers, first layer using a weighted convolutional kernel and the second layer using an RBF, provided the best performance beating DGP, CGP and SGP models by a large margin. To the best of our knowledge, this is the highest accuracy reported by a GP model on the rectangles-image data. This indicates the usefulness of the representation learning capability of CDGP model for complex image classification.

5.3.3 CIFAR-10

The CIFAR-10 dataset [29] consists of total 60K images out of which 50K are used as training images while the rest 10K images are being used for testing. The dataset contains colored images of objects like airplane, automobile, etc. There are 10 classes

[2]Rectangles-image data is different from the simpler rectangles data used in [3], where a random size rectangle is placed in black background with the pixels corresponding to the border of the rectangle in white, while that of inside in black.

Table 5.2: Comparison of SGP, DGP, CGP and CDGP approaches with different architectures on the Rectangles-Image data set along with the kernels used in by GP in each layer.

Model	Layer 1	Layer 2	Layer 3	Accuracy%	NLPP
SGP	RBF	–	–	76.1	0.493
DGP1	RBF	RBF	–	76.93	0.478
DGP2	RBF	RBF	RBF	76.98	0.476
CGP	Wconv	–	–	71.06	0.602
CDGP1	Wconv	RBF	–	**79.74**	**0.422**
CDGP2	Wconv	RBF	RBF	77.95	0.449

in total having 6K images per class. The dimensionality of each image is $32 \times 32 \times 3$ (3 is for channels). We compare the performance of CDGP, DGP and CGP models in Table 5.3. Parameters of the models are learned by running the ADAM optimizer for 200 epochs with a mini-batch size of 40[3].

We observe that DGP models gave a relatively low performance on the CIFAR10 datasets. Equipping DGP models with convolutional kernels have boosted the performance by 10% showing the effectiveness of convolutional kernels for image classification. However, CDGP models were not able to obtain a performance close to CGP. This could be an indication that, for this particular dataset, the properties of a single-layer CDGP i.e, CGP is enough to learn a good classifier. In fact, the previous experiments have shown that 2-layer CDGPs typically result in the best accuracy (in comparison with deeper models), implying that a CGP has already very large capacity for classification and therefore the addition of one layer is usually enough to improve on the results.

Table 5.3: Comparison of DGP, CGP and CDGP approaches with different architectures on the CIFAR10 data set along with the kernels used by GP in each layer.

Model	Layer 1	Layer 2	Layer 3	Accuracy%	NLPP
DGP1	RBF	RBF	–	42.20	3.2579
DGP2	RBF	RBF	RBF	40.13	3.5785
CGP	Wconv	–	–	**55**	–
CDGP1	Wconv	RBF	–	51.74	2.4893
CDGP2	Wconv	RBF	RBF	51.59	2.4607

[3]Learning took around 11 hours on Nvidia GTX 1080 Ti GPU, while the best results reported in [3] is obtained after running the optimization for 40 hours.

5.3.4 Convex sets

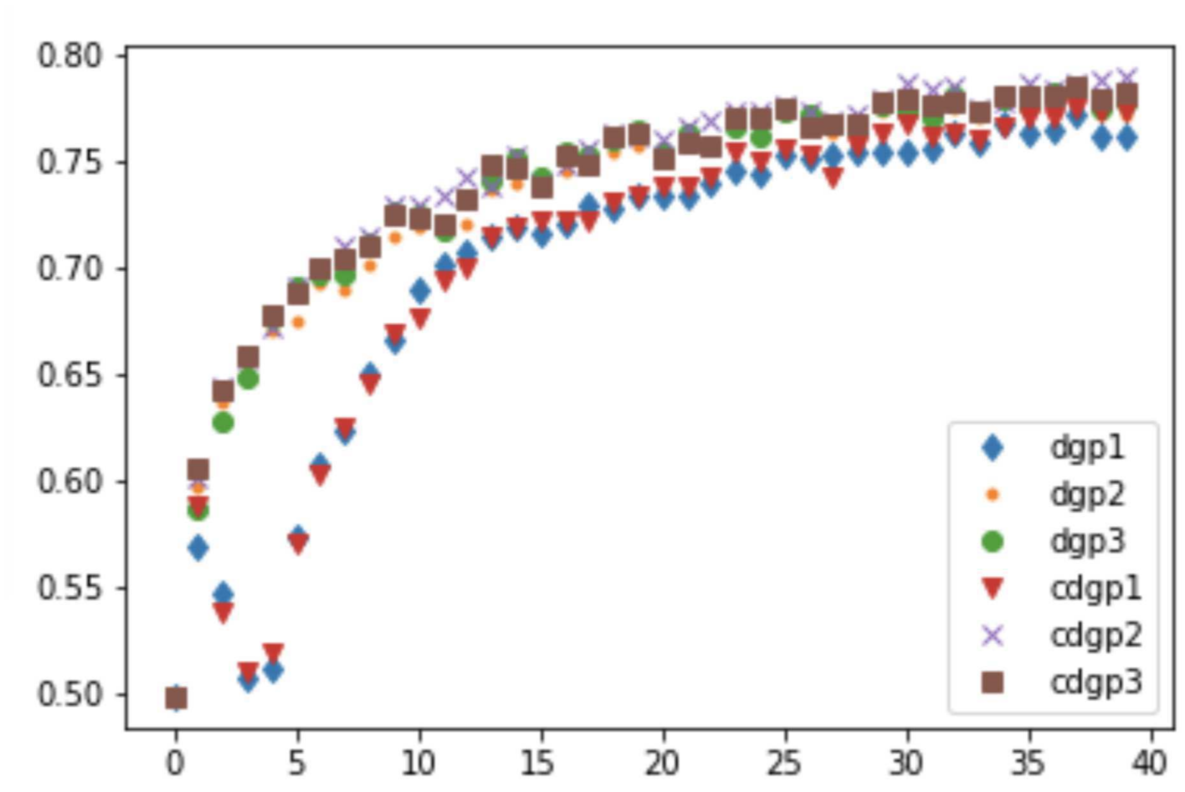


Figure 5.1: Plot representing the training accuracy of after every 5 epochs of training.

Convex sets is a dataset for binary classification of images. The images are greyscaled with value of pixel being 0 at the dark part and having maximum pixel value of 255 at its bright region. The task here is to identify whether the bright region in the image is a convex one or non-convex. The image size of this dataset is 28×28. The difficult part about this dataset is smaller training set. It has only 8000 images in training set and have 50000 images in the test set. We considered CDGP and DGP models with various architectures as described in Table 5.4. Parameters of the model are learned by running the ADAM optimizer for 200 epochs with 0.01 step size and a mini-batch of 1000. This dataset too is a artificial dataset like rectangle-image which performs better if the network architecture is deep. It clear from the results mentioned in Table. 5.4 as we go deeper the test nlpp is keep on decreasing implying that deeper models have better representation capabilities. The accuracy and nlpp both are better if the 1st layer of the model have a convolutional kernel instead of a RBF (for corresponding depth of the network). The training accuracy for all the 200 epochs (sampled at every 5 epochs) has been plotted in Fig. 5.1. It

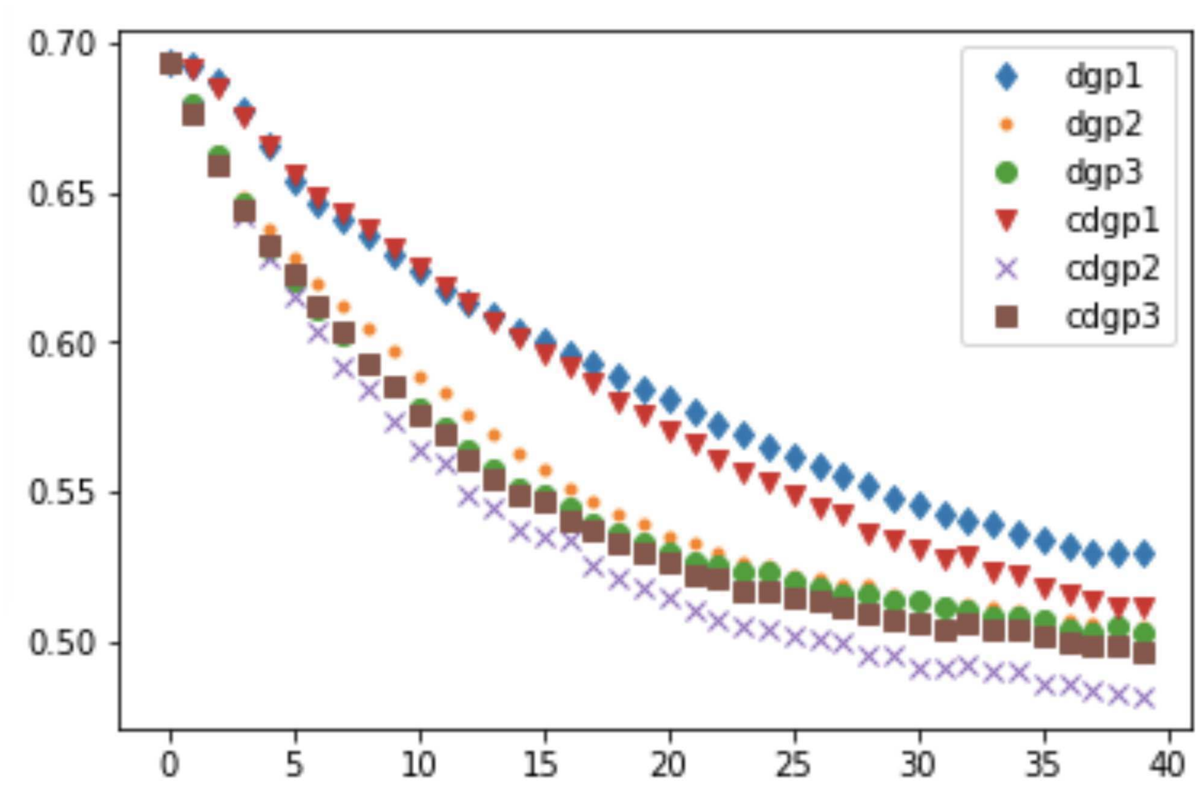


Figure 5.2: Plot representing the training nlpp(essentially the loss function value) of after every 5 epochs of training.

is clear from the Fig. 5.1 that finally cdgp3 outperforms the cdgp4 with cdgp4 being very close throughout. Figure. 5.2 shows the value of loss function sampled after every 5 epochs. It is clearly visible from the Fig.. 5.2 that the cdgp2 fits the data more adequately and hence which results in better accuracy too.

Table 5.4: Comparison of DGP, CGP and CDGP approaches with different architectures on the Convex sets data set along with the kernels used in by GP in each layer.

Model	Layer 1	Layer 2	Layer 3	Layer 4	Accuracy%	NLPP
DGP1	RBF	RBF	–	–	73.57	0.5472
DGP2	RBF	RBF	RBF	–	74.27	0.5297
DGP3	RBF	RBF	RBF	RBF	74.34	0.5274
CGP	Wconv	–	–	–	70.08	0.6028
CDGP1	Wconv	RBF	–	–	73.87	0.5411
CDGP2	Wconv	RBF	RBF	–	**75.19**	**0.5233**
CDGP3	Wconv	RBF	RBF	RBF	74.64	0.5233

5.3.5 Experiments with Random Sub-sampling of Patches on Caltech-101 Dataset

Table 5.5: Comparison of Training time required for different CDGP architectures with different number of patches on the Caltech-101 dataset.

Model	Layer 1	Layer 2	Layer 3	Training time	Accuracy%	NLPP
CDGP1(All patches)	Wconv	RBF	–	11 hrs 18 min	20.39	6.5811
CDGP2(All patches)	Wconv	RBF	RBF	12 hrs 2min	19.51	6.787
CDGP1(Random patches)	Wconv	RBF	–	1 hr 15 min	20	6.7009
CDGP2(Random patches)	Wconv	RBF	RBF	1hr 19 min	18.82	7.0473

Computation of convolutional kernels becomes prohibitive on data sets such as Caltech101 [30] with very high dimensionality. It consists of 101 classes with 20 images per class for training and 10 images per class for testing. The size varies slightly for each image in the actual dataset but is roughly around 300 $\times$ 200 pixels per channel. The images are colored so each image has 3 channels. The experiments are conducted on images resized to 50 $\times$ 50 $\times$ 3. Instead of taking all the patches of the image for computing the convolutional kernel, we randomly picked up one-tenth of the total number of image patches for computing the kernel. This resulted in a very significant speed-up in learning time without much loss in accuracy, as can be seen from table

5.5 [4]. The test accuracy obtained with CDGP1 is 20.39% and time taken for training is 11 hrs 18min. On the other hand, for the same model considering random subset of image patches, training time drops to only 1 hr 15 min with an accuracy drop of only 0.39% making it around 10 times faster. Similar phenomenon has been observed in case of CDGP2 considering random subset of patches, where training time improved from 12 hrs 2 min to 1 hrs 19 min with an accuracy drop of just 0.69%, providing a speedup of around 10 hours. The classification accuracies of the models presented in Table 5.5 are low due to resizing of the original image to size 50×50, resulting in loss of information. As a future work, we will conduct experiments by keeping the original image size, and study the effectiveness of random sub-sampling of patches and generalization performance of the proposed approach.CDGP2,3 doesn't run with stride 1 using 150x150 images but with random patches it runs, but giving a low accuracy.

[4]We ran the experiments on Nvidia GTX 1080 Ti GPU.

Chapter 6

Conclusion

Deep GP models provide a lot of advantages in terms of capacity control and predictive uncertainty, but they are less effective in computer vision tasks. Commonly used RBF kernels in the DGP models fail to capture variations in image data and are not invariant to translations. In this paper we proposed a DGP model which captures convolutional structure in image data using convolutional kernels. Our model extends the convolutional GPs with the ability to learn hierarchical latent representations making it a useful model for image classification. We incorporated different types of convolutional kernels [3] in the DGP models and demonstrated their usefulness for image classification in benchmark data sets such as MNIST, Rectangles-Image and CIFAR10. In the future, we plan to develop methods to further reduce the cost of convolutional kernel computation and memory requirements of the CDGP model for high dimensional datasets. This will allow us to consider a higher mini-batch size, leading to reduced stochastic gradient variance and faster convergence of the optimization routine. We found that increasing the number of layers in CDGP did not bring much improvements in performance contrary to what we expected. We provide automatic patch selection flexibility to the model which is not there in traditional CNNs. We tried optimizing variational parameters with natural gradients and other hyperparameters with ADAM optimizer. For our experimental setup it didn't result in faster convergence of the variational parameters.

We hope that our future research on faster and more effective variational inference techniques will address these limitations with convolutional DGPs. DGPs posses a huge representational power theoretically. Each GP regression is theoretically equivalent to a infinite dimensional multi-layer peceptron. The less effectiveness of the model on large scale computer vision datasets can be a result of high computational requirement for GPs and less polished approach of modeling and optimizations.

Bibliography

[1] C. E. Rasmussen and C. K. I. Williams. Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, 2005.

[2] H. Salimbeni and M. Deisenroth. Doubly Stochastic Variational Inference for Deep Gaussian Processes. In Advances in Neural Information Processing Systems 30, 4588–4599. 2017.

[3] M. van der Wilk, C. E. Rasmussen, and J. Hensman. Convolutional Gaussian Processes. In Advances in Neural Information Processing Systems 30. 2017 2849–2858.

[4] J. Hensman, N. Fusi, and N. D. Lawrence. Gaussian Processes for Big Data 2013.

[5] A. Damianou and N. Lawrence. Deep Gaussian Processes. In International Conference on Artificial Intelligence and Statistics. 2013 207–215.

[6] V. Kumar, V. Singh, P. K. Srijith, and A. Damianou. Deep Gaussian Processes with Convolutional Kernels. *ArXiv e-prints* .

[7] C. Williams and D. Barber. Bayesian Classification with Gaussian Processes. In IEEE Transactions on Pattern Analysis and Machine Intelligence. 1998 1342–1351.

[8] K. Chai. Variational Multinomial Logit Gaussian Process. In Journal of Machine Learning Research. 2012 .

[9] J. Hensman, A. G. d. G. Matthews, and Z. Ghahramani. Scalable Variational Gaussian Process Classification. In International Conference on Artificial Intelligence and Statistics. 2015 .

[10] P. K. Srijith, P. Balamurugan, and S. Shevade. Gaussian Process Pseudo-Likelihood Models for Sequence Labeling. In European Conference on Machine Learning and Knowledge Discovery in Databases. 2016 215–231.

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

[12] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *CoRR* abs/1312.6114.

[13] A. Damianou. Deep Gaussian Processes and Variational Propagation of Uncertainty. *PhD Thesis, University of Sheffield* .

[14] J. Hensman and N. D. Lawrence. Nested variational compression in deep Gaussian processes. *arXiv preprint arXiv:1412.1370* .

[15] Z. Dai, A. Damianou, J. González, and N. Lawrence. Variational Auto-encoded Deep Gaussian Processes. *International Conference on Learning Representations (ICLR)* .

[16] T. Bui, D. Hernández-Lobato, J. Hernandez-Lobato, Y. Li, and R. Turner. Deep Gaussian processes for regression using approximate expectation propagation. In International Conference on Machine Learning. 2016 1472–1481.

[17] K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone. Random Feature Expansions for Deep Gaussian Processes. In International Conference on Machine Learning, volume 70. 2017 884–893.

[18] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. In MIT Press. 2016 .

[19] Y. Gal. Uncertainty in Deep Learning. In University of Cambridge. 2016 .

[20] N. D. Lawrence and A. J. Moore. Hierarchical Gaussian process latent variable models. In Proceedings of the 24th international conference on Machine learning. ACM, 2007 481–488.

[21] D. Haussler. Convolution Kernels on Discrete Structures. Technical Report 1999.

[22] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing. Deep Kernel Learning. In International Conference on Artificial Intelligence and Statistics, volume 51. 2016 370–378.

[23] G.-L. Tran, E. V. Bonilla, J. P. Cunningham, P. Michiardi, and M. Filippone. Calibrating Deep Convolutional Gaussian Processes 2018.

[24] J. Bradshaw, A. G. d. G. Matthews, and Z. Ghahramani. Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks. *arXiv preprint arXiv:1707.02476* .

[25] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. . *Journal of Machine Learning Research* 11, (2010) 12011242.

[26] M. Collins and N. Duffy. Convolution kernels for natural language. In Advances in Neural Information Processing Systems. 2001 25–632.

[27] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research* 18, (2017) 1–6.

[28] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980.

[29] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical Report 2009.

[30] L. Fei-Fei, R. Fergus, and P. Perona. Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In 2004 Conference on Computer Vision and Pattern Recognition Workshop. 2004 178–178.