Deep Gaussian Processes with Convolutional Kernels

Vinayak Kumar[*,1], **Vaibhav Singh**[*,1], **P.K. Srijith**[1], **and Andreas Damianou**[†,2]

[1]Department of Computer Science and Engineering,
Indian Institute of Technology, Hyderabad, India
[2]Amazon Research, Cambridge, United Kingdom
{vinayakk,cs16mtech11017,srijith}@iith.ac.in, damianou@amazon.com

Abstract

Deep Gaussian processes (DGPs) provide a Bayesian non-parametric alternative to standard parametric deep learning models by stacking multiple Gaussian Processes. DGP models typically use radial basis function (RBF) kernel which are insufficient for handling pixel variability in raw images. In this paper, we propose 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. The experimental results on image classification data such as MNIST, rectangles-image, Convex-nonconvex sets and Caltech101 demonstrate the effectiveness of the proposed approaches in obtaining a better generalization performance.

1 Introduction

Deep Gaussian processes (DGPs) constitute a deep Bayesian non-parametric approach using Gaussian processes (GPs) and provides implicit capacity control and predictive uncertainty [Damianou and Lawrence, 2013]. Many variants of DGP models have emerged in recent years mainly differing in the employed inference procedure [Hensman and Lawrence, 2014; Dai et al., 2016; Bui et al., 2016]. Attempts have been made to address scalability issues in DGP. Cutajar et al. [2017] used random features expansion while Salimbeni and Deisenroth [2017] considered a DGP model with a variational posterior maintaining the exact model structure and a doubly stochastic variational inference approach.

DGP models typically 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) [Goodfellow et al., 2016] 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.

The proposed model, Convolutional deep Gaussian processes (CDGP), extends the convolutional GP [van der Wilk et al., 2017] to a deep learning setting and performs hierarchical feature learning. We consider various DGP architectures obtained by stacking together GPs with 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 for high dimensional image data. We propose an approach to reduce the computational cost 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, Convex-nonconvex sets, and Caltech-101.

The proposed approach is different from recent works which combine CNNs and GPs in hybrid mode, such as Wilson et al. [2016]; Tran et al. [2018]; Bradshaw et al. [2017]. In particular, Tran et al. [2018] replaces the fully connected layers of a CNN with GPs, aiming at obtaining well-calibrated probabilities. While, in deep kernel learning [Wilson et al., 2016], the kernel in GPs are computed using deep neural networks. In contrast,

[*]Equal Contribution.
[†]A. Damianou contributed to this work prior to joining Amazon.

our approach brings the convolutional structure inside the deep GP model, through kernels, and remains fully non-parametric.

2 Convolutional Deep Gaussian Processes

Deep Gaussian processes (DGPs) [Damianou, 2015; Damianou and Lawrence, 2013] learn complex functions by stacking GPs one over the other resulting in a deep architecture of GPs. Deep GPs do not typically overfit on small data due to Bayesian model averaging, and is devoid of large number of parameters but only a few kernel hyper-parameters and variational parameters.

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 $f^l = \{f_j^l\}_{j=1}^{D^l}$ mapping representations in layer $l-1$ (F^{l-1}) to obtain D^l representation for layer l (F^l). Independent GP priors are placed over the function f_j^l, $f_j^l(\cdot) \sim \mathcal{GP}(0, k^l(\cdot,\cdot))$. Inference is intractable in DGP model, and we use the doubly stochastic variational inference techniques and the variational sparse approximation used in Salimbeni and Deisenroth [2017].

Existing DGP models used radial basis function kernels which fails to consider the convolutional structure in the images. We introduce convolutional kernels [van der Wilk et al., 2017] with deep Gaussian processes in order to obtain the convolutional deep Gaussian process (CDGP). A CDGP can capture salient features which are invariant to variations in the image through the convolutional structures and is simultaneously performing strong function learning through out its depth. This results in a powerful well-calibrated model for tasks like image classification.

Our starting point is the recently introduced convolutional Gaussian processes (CGP) [van der Wilk et al., 2017] 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 $f(\mathbf{x}) = \sum_{p=1}^{P} g(\mathbf{x}^{[p]})$. 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,

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k_f(\mathbf{x}_i, \mathbf{x}_j)),, \text{where}$$

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

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.

In CGP, 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

$$k_f(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p'=1}^{P} k_g(\mathbf{x}_i^{[1]}, \mathbf{x}_j^{[p']}) + \ldots + \sum_{p'=1}^{P} k_g(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p']}) + \ldots + \sum_{p'=1}^{P} k_g(\mathbf{x}_i^{[P]}, \mathbf{x}_j^{[p']}) \tag{2}$$

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. The signal value at any point p' is obtained by computing the dot product between the filter $\mathbf{x}_i^{[p]}$ and patch $\mathbf{x}_j^{[p']}$ through base kernel.

Convolutional deep Gaussian processes consider functions to be sampled from a GP prior with convolutional kernels to form the representations of the image in the intermediate layers. The function corresponding to o^{th} representation for layer 1 is obtained by considering $f_o^1(\mathbf{x}) = \sum_{p=1}^{P} g_o^1(\mathbf{x}^{[p]})$, where $g_o^1(\mathbf{x}^{[p]}) \sim \mathcal{GP}(0, k_g^1(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p]}))$. This results in

$$f_o^1(\mathbf{x}) \sim \mathcal{GP}(0, k_f^1(\mathbf{x}_i, \mathbf{x}_j)) \text{ , where}$$

$$k_f^1(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p=1}^{P} \sum_{p'=1}^{P} k_g^1(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p']}) \tag{3}$$

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

$$f_o^l(F^{l-1}) \sim \mathcal{GP}(0, k_f^l(F_i^{l-1}, F_j^{l-1})), \text{where}$$

$$k_f^l(F_i^{l-1}, F_j^{l-1}) = \sum_{p=1}^{P} \sum_{p'=1}^{P} k_g^l(F_i^{l-1[p]}, F_j^{l-1[p']}) \tag{4}$$

We also consider a variant of the convolutional kernel, weighted convolutional kernel (Wconv kernels) [van der Wilk et al., 2017]. It associates a weight with each patch which allows the kernel to provide differential weightage to the patches.

$$k_f(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p=1}^{P} \sum_{p'=1}^{P} w_p w_{p'} k_g(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p']}) \quad (5)$$

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 by treating the inducing points to be in the patch space rather than in the original image space [van der Wilk et al., 2017], *i.e.* $Z_j^l \in \mathcal{R}^{w \times h}$ rather than in the input space $\mathcal{R}^{W \times H}$. In this case, kernel evaluations between representations and inducing points can be performed in $\mathcal{O}(P)$ time since $k_f^l(F_i^{l-1}, Z_j^l) = \sum_{p=1}^{P} k_g^l(F_i^{l-1[p]}, Z_j^l)$, and kernel evaluations among inducing points $k_f^l(Z_i^l, Z_j^l)$ can be performed in constant time [van der Wilk et al., 2017]. However, kernel evaluations among representations $k_f^l(F_i^{l-1}, F_j^{l-1})$ still requires $\mathcal{O}(P^2)$ computations, making it a costly operation especially for the first layer, where $F_i^0 = \mathbf{x}_i$. 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 $k_f^l(F_i^{l-1}, F_j^{l-1})$ and $k_f^l(F_i^{l-1}, Z_j^l)$. Let $S, S' \subset \{1, 2, \ldots, P\}$ represent the random subsets. For the o^{th} representation of layer 1 ($F^0 = \mathbf{x}$), we consider the covariance functions to be as

$$f_o^1(\mathbf{x}) = \sum_{p \in S} g_o^1(\mathbf{x}^{[p]}) \implies$$
$$k_f^1(\mathbf{x}_i, \mathbf{x}_j) = \sum_{p \in S} \sum_{p' \in S'} k_g^1(\mathbf{x}_i^{[p]}, \mathbf{x}_j^{[p']}) \text{ , and} \quad (6)$$

$$k_f^1(\mathbf{x}_i, Z_j^1) = \mathbb{E}_g[f_o^1(\mathbf{x}_i) g_o^1(Z_j^1)] \quad (7)$$
$$= \mathbb{E}_g[\sum_{p \in S} g_o^1(\mathbf{x}^{[p]}) g_o^1(Z_j^1)] = \sum_{p \in S} k_g^1(\mathbf{x}^{[p]}, Z_j^1)$$

This reduces the cost of computing $k_f^l(F_i^{l-1}, F_j^{l-1})$ to $\mathcal{O}(|S||S'|)$ where the size of the subsets $|S|, |S'| \ll P$. Computational speedup achieved through random subsampling of patches is testified in our experiments on Caltech101.

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, Convex-Nonconvex sets and Caltech101. We consider different kernel architectures of the proposed CDGP model and compare it with deep GP (DGP) models using RBF kernel and with convolutional GPs (CGP) [1]. Dimensionality of each latent layer 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), and the number of inducing points is taken to be 100. 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 for the rest of the data sets. We use ADAM optimizer with 0.01 step size and a mini-batch of 1000 for experimentations on MNIST, Rectangle-Image and Convex-Nonconvex sets datasets. 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.

MNIST: We performed experiments with MNIST dataset having 10 classes representing the digits $0 - 9$. We consider the standard train/test split with 60K training and 10K test images. The best performance was obtained with a 2 layer CDGP which gave an accuracy of 98.66% and NLPP value of 0.046. We can also observe from Table. 1 that all the CDGP models performed better than the DGP and CGP models for the MNIST dataset in terms of both accuracy and NLPP. We also conducted experiments with a combination of two RBF kernels as the base kernel with length scales initialized to 2 different values 0.01 and 10. 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.

Rectangle-Image: The task is to classify if a rectangle in

[1] We report the best result among different variants of the convolutional kernel used in CGP.

Table 1: Comparison of DGP, CGP and CDGP approaches with different architectures (kernel used in each layer) on the MNIST, Rectangle-Image and Convex-Nonconvex dataset.

Model	Architecture	MNIST		Rectangle Image		Convex-Nonconvex	
		Accuracy %	NLPP	Accuracy %	NLPP	Accuracy %	NLPP
DGP1	RBF-RBF	97.94	0.073	76.93	0.478	73.57	0.547
DGP2	RBF-RBF-RBF	97.99	0.070	76.98	0.476	74.27	0.529
CGP	Wconv	97.54	0.103	71.06	0.602	70.08	0.602
CDGP1	Wconv-RBF	**98.66**	**0.046**	**79.74**	**0.422**	73.87	0.541
CDGP2	Wconv-RBF-RBF	98.55	**0.046**	77.95	0.449	**75.19**	**0.523**

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 [Larochelle et al., 2007]. We can observe from Table. 1 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. It gave an accuracy of 79.74% and an NLPP value of 0.422 beating DGP and CGP 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.

Convex-Nonconvex sets: The task in this dataset is to identify whether the bright region in a grayscaled image is convex or nonconvex [Larochelle et al., 2007]. The image size is 28×28 with 8K images in training set and 50K images in test set. From Table.1, it is clearly visible that adding a convolutional kernel in DGP structure gives an increment in the generalization performance of the model. The best results obtained in this dataset is with a 3 layered CDGP model which gave an accuracy of 75.19% and an NLPP value of 0.523. It used a weighted convolutional kernel in the first layer and RBF kernels in the second and third layers.

Experiments with Random Sub-sampling of Patches on Caltech-101 Dataset: Computation of convolutional kernels becomes prohibitive on data sets such as Caltech101 [Fei-Fei et al., 2004] 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 300×200 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 [2]. The test accuracy obtained with 2 layer CDGP (Weighted convolutional kernel in first layer and RBF in second) is 20.39% and time taken for training is 11hrs 18min. On the other hand, for the same model considering random subset of image patches, training time drops to 1hr 15min with an accuracy drop of only 0.39% making it around 10 times faster. Similar observation were found for a 3 layer CDGP (weighted convolutional kernel in first layer, RBF in the second and third layer) considering random subset of patches, where training time improved from 12hrs 2min to 1hr 19min with an accuracy drop of just 0.69% from 19.51 to 18.82, providing a speedup of around 10 hours. The classification accuracies of the models are low due to resizing of the original image to size 50×50 which facilitates running standard CDGP in our GPU but results in losing some information content.

4 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. We proposed DGP models which captures convolutional structure in image data using convolutional kernels. Our model extends the convolutional GPs with the ability to learn hierarchical latent representations. We demonstrated their usefulness for image classification in benchmark data sets such as MNIST, Rectangles-Image, Convex-Nonconvex sets, and Caltech-101. 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.

[2] The experiments are run on Nvidia GTX 1080 Ti GPU.

References

Bradshaw, J., Matthews, A. G. d. G., and Ghahramani, Z. (2017). Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*.

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

Cutajar, K., Bonilla, E. V., Michiardi, P., and Filippone, M. (2017). Random feature expansions for deep Gaussian processes. In *International Conference on Machine Learning*, volume 70, pages 884–893.

Dai, Z., Damianou, A., González, J., and Lawrence, N. (2016). Variational auto-encoded deep Gaussian processes. *International Conference on Learning Representations (ICLR)*.

Damianou, A. (2015). Deep Gaussian processes and variational propagation of uncertainty. *PhD Thesis, University of Sheffield*.

Damianou, A. and Lawrence, N. (2013). Deep Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 207–215.

Fei-Fei, L., Fergus, R., and Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Conference on Computer Vision and Pattern Recognition Workshop*, pages 178–178.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. In *MIT Press*.

Hensman, J. and Lawrence, N. D. (2014). Nested variational compression in deep Gaussian processes. *arXiv preprint arXiv:1412.1370*.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning*, pages 473–480.

Salimbeni, H. and Deisenroth, M. (2017). Doubly stochastic variational inference for deep gaussian processes. In *Advances in Neural Information Processing Systems 30*, pages 4588–4599.

Tran, G.-L., Bonilla, E. V., Cunningham, J. P., Michiardi, P., and Filippone, M. (2018). Calibrating Deep Convolutional Gaussian Processes. *ArXiv e-prints arXiv:1805.10522*.

van der Wilk, M., Rasmussen, C. E., and Hensman, J. (2017). Convolutional Gaussian processes. In *Advances in Neural Information Processing Systems 30*, pages 2849–2858.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics*, volume 51, pages 370–378.