Deep convolutional Gaussian processes

Kenneth Blomqvist, Samuel Kaski, and Markus Heinonen (✉)

Aalto University, Finland
Helsinki Institute for Information Technology HIIT, Finland
{kenneth.blomqvist,samuel.kaski,markus.o.heinonen}@aalto.fi

Abstract. We propose deep convolutional Gaussian processes, a deep Gaussian process architecture with convolutional structure. The model is a principled Bayesian framework for detecting hierarchical combinations of local features for image classification. We demonstrate greatly improved image classification performance compared to current convolutional Gaussian process approaches on the MNIST and CIFAR-10 datasets. In particular, we improve state-of-the-art CIFAR-10 accuracy by over 10 percentage points.

Keywords: Gaussian processes · Convolutions · Variational inference

1 Introduction

Gaussian processes (GPs) are a family of flexible function distributions defined by a kernel function [25]. The modeling capacity is determined by the chosen kernel. Standard stationary kernels lead to models that underperform in practice. Shallow – or single layer – Gaussian processes are often sub-optimal since flexible kernels that would account for non-stationary patterns and long-range interactions in the data are difficult to design and infer [35,26]. Deep Gaussian processes boost performance by modelling networks of GP nodes [8,30] or by mapping inputs through multiple Gaussian process 'layers' [5,27]. While more flexible and powerful than shallow GPs, deep Gaussian processes result in degenerate models if the individual GP layers are not invertible, which limits their potential [7].

Convolutional neural networks (CNN) are a celebrated approach for image recognition tasks with outstanding performance [21]. These models encode a hierarchical translation-invariance assumption into the structure of the model by applying convolutions to extract increasingly complex patterns through the layers.

While neural networks have achieved unparalleled results on many tasks, they have their shortcomings. Effective neural networks require large number of parameters that require careful optimisation to prevent overfitting. Neural networks can often leverage a large number of training data to counteract this problem. Developing methods that are better regularized and can incorporate prior knowledge would allow us to deploy machine learning methods in domains where massive amounts of data is not available. Conventional neural networks do

not provide reliable uncertainty estimates on predictions, which are important in many real world applications.

The deterministic CNN's have been extended into the probabilistic domain with weight uncertainties [3], while combinations of CNN's and Gaussian processes have been shown to improve calibration of the prediction uncertainty [31]. In deep kernel learning (DKL) a feature-extracting deep neural network is stacked with a Gaussian process predictor layer [38], learning the neural network weights by variational inference [37]. Neural networks are known to converge to Gaussian processes at the limit of infinite layer width [20,34,19], and similar correspondence have been shown between CNN's and Gaussian processes as well [10].

Recently van der Wilk et al. proposed the first convolution-based Gaussian process for images with promising performance [33]. They proposed a shallow weighted additive model where Gaussian process responses over image subpatches are aggregated for image classification. The convolutional Gaussian process is unable to model pattern combinations due to its restriction to a single layer. Very recently convolutional kernels have been applied in a deep Gaussian process, however with little improvement upon the shallow convolutional GP model [18]. The translation insensitive convolutional kernel adds increased flexibility by location-dependent convolutions for both shallow and deep models [6][1].

In this paper we propose a deep convolutional Gaussian process, which iteratively convolves several GP functions over an image. We learn multimodal probabilistic representations that encode combinations of increasingly complex pattern combinations as a function of depth. Our model is a fully Bayesian kernel method with no neural network component. On the CIFAR-10 dataset, deep convolutions increase the current state-of-the-art GP predictive accuracy from 65% to 76%. We show that our GP-based model performs better than a CNN model with similar depth, and provides better calibrated and more consistent uncertainty estimates on predictions.

2 Background

In this section we provide an overview of the main methods our work relies upon. We consider supervised image classification problems with N examples $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ each associated with a label $y_i \in \mathbb{Z}$. We assume images $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$ as 3D tensors of size $W \times H \times C$ over C channels, where RGB color images have $C = 3$ color channels.

2.1 Discrete convolutions

A convolution as used in convolutional neural networks takes a signal, two dimensional in the case of an image, and a tensor valued filter to produce a new

[1] We note that after placing our current manuscript in arXiv in October 2018, a subsequent arXiv manuscript has already extended the proposed deep convolution model by introducing location-dependent kernel [6].

signal [11]. The filter is moved across the signal and at each step taking a dot product with the corresponding section in the signal. The resulting signal will have a high value where the signal is similar to the filter, zero where it's orthogonal to the filter and a low value where it's very different from the filter. A convolution of a two dimensional image $\mathbf{x}$ and a convolutional filter $\mathbf{g}$ is defined:

$$(\mathbf{x} * \mathbf{g})[i,j] = \sum_{w=0}^{W-1} \sum_{h=0}^{H-1} \mathbf{x}[i+w, j+h]\mathbf{g}[w,h] \tag{1}$$

$\mathbf{x}[i,j] \in \mathbb{R}^3$ and $\mathbf{g}$ is in $\mathbb{R}^{H \times W \times 3}$. Here H and W define the size of the convolutional filter. Typical values could be $H = W = 5$ or $H = W = 3$. Typically multiple convolutional filters are used, each convolved over the input to produce several output signals which are stacked together.

By default the convolution is defined over every location of the image. Sometimes one might use only every other location. This is referred to as the *stride*. A stride of 2 means only every other location i, j is taken in the output.

2.2 Primer on Gaussian processes

Gaussian processes are a family of Bayesian models that characterize distributions of functions [24]. A zero-mean Gaussian process prior on latent function $f(\mathbf{x}) \in \mathbb{R}$,

$$f(\mathbf{x}) \sim \mathcal{GP}(0, K(\mathbf{x}, \mathbf{x}')) \tag{2}$$

defines a *prior* distribution over function values $f(\mathbf{x})$ with mean and covariance:

$$\mathbb{E}[f(\mathbf{x})] = 0 \tag{3}$$

$$\mathbf{cov}[f(\mathbf{x}), f(\mathbf{x}')] = K(\mathbf{x}, \mathbf{x}') \tag{4}$$

A GP prior defines that for any collection of n inputs $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$, the corresponding function values

$$\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^T \in \mathbb{R}^n$$

follow a multivariate Normal distribution

$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \tag{5}$$

$\mathbf{K} = (K(\mathbf{x}_i, \mathbf{x}_j))_{i,j=1}^n \in \mathbb{R}^{n \times n}$ is the kernel matrix encoding the function covariances. A key property of GPs is that output predictions $f(\mathbf{x})$ and $f(\mathbf{x}')$ correlate according to the similarity of the inputs $\mathbf{x}$ and $\mathbf{x}'$ as defined by the kernel $K(\mathbf{x}, \mathbf{x}') \in \mathbb{R}$.

Low-rank Gaussian process functions are constructed by *augmenting* the Gaussian process with a small number M of inducing variables $u_j = f(\mathbf{z}_j)$, $u_j \in \mathbb{R}$ and $\mathbf{z}_j = \mathbb{R}^d$ to obtain the Gaussian function posterior

$$\mathbf{f}|\mathbf{u}, \mathbf{Z} \sim \mathcal{N}(\underbrace{\mathbf{K_{XZ}}\mathbf{K_{ZZ}^{-1}}\mathbf{u}}_{\text{predictive mean}}, \underbrace{\mathbf{K_{XX}} - \mathbf{K_{XZ}}\mathbf{K_{ZZ}^{-1}}\mathbf{K_{ZX}}}_{\text{predictive covariance}}) \tag{6}$$

where $\mathbf{K_{XX}} \in \mathbb{R}^{n \times n}$ is the kernel between observed image pairs $\mathbf{X}$, the kernel $\mathbf{K_{XZ}} \in \mathbb{R}^{n \times M}$ is between observed images $\mathbf{X}$ and inducing images $\mathbf{Z}$, and kernel $\mathbf{K_{ZZ}} \in \mathbb{R}^{m \times m}$ is between inducing images $\mathbf{Z}$. [28]

2.3 Variational inference

Exact inference in a GP entails optimizing the *evidence* $p(\mathbf{y}) = \mathbb{E}_{p(\mathbf{f})}[p(\mathbf{y}|\mathbf{f})]$ which has a limiting cubic complexity $O(n^3)$ and is in general intractable. We tackle this restriction by applying stochastic variational inference (SVI) [13].

We define a variational approximation

$$q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S}) \tag{7}$$

$$\begin{aligned} q(\mathbf{f}) &= \int p(\mathbf{f}|\mathbf{u}) q(\mathbf{u}) d\mathbf{u} \\ &= \mathcal{N}(\mathbf{f}|\mathbf{A}\mathbf{m}, \mathbf{K}_{ff} + \mathbf{A}(\mathbf{S} - \mathbf{K}_{zz})\mathbf{A}^T) \\ \mathbf{A} &= \mathbf{K}_{fz}\mathbf{K}_{zz}^{-1} \end{aligned} \tag{8}$$

with free variational parameters $\mathbf{m} \in \mathbb{R}^m$ and a matrix $\mathbf{S} \succeq 0 \in \mathbb{R}^{m \times m}$ to be optimised. It can be shown that minimizing the Kullback-Leibler divergence $\mathrm{KL}[q(\mathbf{u})||p(\mathbf{u}|\mathbf{y})]$ between the approximative posterior $q(\mathbf{u})$ and the true posterior $p(\mathbf{u}|\mathbf{y})$ is equivalent to maximizing the evidence lower bound (ELBO) [2]

$$\mathcal{L} = \sum_{i=1}^{n} \mathbb{E}_{q(f_i)}[\log p(y_i|f_i)] - \mathrm{KL}[q(\mathbf{u})||p(\mathbf{u})] \tag{9}$$

The variational expected likelihood in $\mathcal{L}$ can be computed using numerical quadrature approaches [13].

3 Deep convolutional Gaussian process

In this section we introduce the deep convolution Gaussian process. We stack multiple convolutional GP layers followed by a GP classifier with a convolutional kernel.

3.1 Convolutional GP layers

We assume an image representation $\mathbf{f}_c^\ell \in \mathbb{R}^{W_\ell \times H_\ell}$ of width W_ℓ and height H_ℓ pixels at layer ℓ. We collect C_ℓ *channels* into a 3D tensor $\mathbf{f}^\ell = (\mathbf{f}_1^\ell, \ldots, \mathbf{f}_C^\ell) \in \mathbb{R}^{H_\ell \times W_\ell \times C_\ell}$, where the channels are along the depth axis. The input image $\mathbf{f}^0 = \mathbf{x}$ is the $W_0 \times H_0 \times C_0$ sized representation of the original image with C color channels. For instance MNIST images are of size $W = H = 28$ pixels and have a single $C = 1$ grayscale channel.

We decompose the 3D tensor $\mathbf{f}^\ell$ into *patches* $\mathbf{f}^\ell[p] \in \mathbb{R}^{w_\ell \times h_\ell \times C_\ell}$ containing all depth channel. h_ℓ and w_ℓ are the height and width of the image patch at

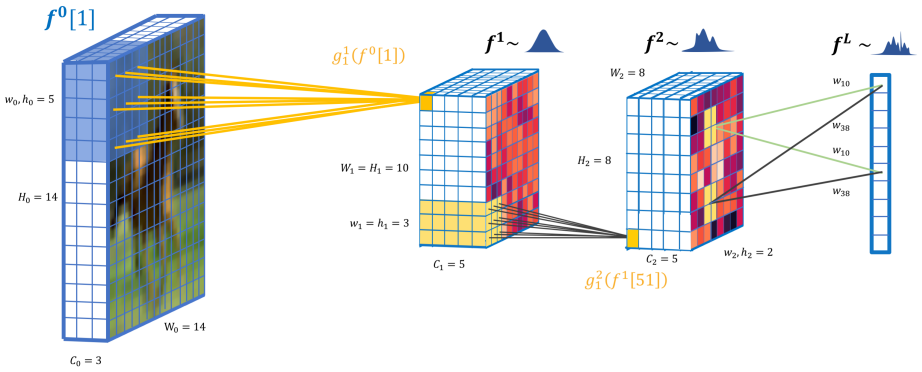


Fig. 1: A three layer deep convolutional gaussian process. First we construct an intermediate probabilistic representation of size $W_1 \times H_1 \times C_1$. We map this probabilistic representation through another convolutional GP layer yielding a representation of size $W_2 \times H_2 \times C_2$. Finally, we classify using a GP with a convolutional kernel by summing over patches of the intermediate representation.

layer ℓ. We index patches by $p \in \mathbb{Z} < H_\ell W_\ell$. H_ℓ and W_ℓ denotes the height and width of the output of layer ℓ. We compose a sequence of layers $\mathbf{f}^\ell$ that map the input image $\mathbf{x}_i$ to the label $\mathbf{y}_i$:

$$\underbrace{\mathbf{x}_i = \mathbf{f}^0}_{W_0 \times H_0 \times 3} \xrightarrow{g^1} \underbrace{\mathbf{f}^1}_{W_1 \times H_1 \times C_1} \cdots \xrightarrow{g^L} \underbrace{\mathbf{f}^L}_{C_y} \approx \underbrace{\mathbf{y}_i}_{\{0,1\}^{C_y}} . \tag{10}$$

Layers $\mathbf{f}^\ell$ with $\ell \geq 1$ are random variables with probability densities $p(\mathbf{f}^\ell)$.

We construct the layers by applying *convolutions* of *patch response* functions $\mathbf{g}_c^\ell : \mathbb{R}^{w_{\ell-1} \times h_{\ell-1} \times C_{\ell-1}} \to \mathbb{R}$ over the input one patch at a time producing the next layer representation:

$$\mathbf{f}^\ell[p] = \begin{bmatrix} g_1^\ell(\mathbf{f}^{\ell-1}[p]) \\ \vdots \\ g_C^\ell(\mathbf{f}^{\ell-1}[p]) \end{bmatrix} \in \mathbb{R}^C \tag{11}$$

Each individual patch response $g^\ell(\mathbf{f}^{\ell-1}[p])$ is a $1 \times 1 \times C$ pixel stack. By repeating the patch responses over the $P_{\ell-1} = W_\ell \times H_\ell$ patches we form a new $W_\ell \times H_\ell \times C_\ell$ representation $\mathbf{f}^\ell = (\mathbf{f}^\ell[1], \ldots, \mathbf{f}^\ell[P_{\ell-1}])$ (See Figure 1).

We model the C patch responses at each of the first $L-1$ layers as independent GPs with shared prior

$$g_c^\ell(\mathbf{f}^{\ell-1}[p]) \sim \mathcal{GP}\big(0, k(\mathbf{f}^{\ell-1}[p], \mathbf{f}'^{\ell-1}[p'])\big) \tag{12}$$

for $c = 1, \ldots, C$. The kernel $k(\cdot, \cdot)$ measures the similarity of two image patches. The standard property of Gaussian processes implies that the functions g_c^ℓ output similar responses for similar patches.

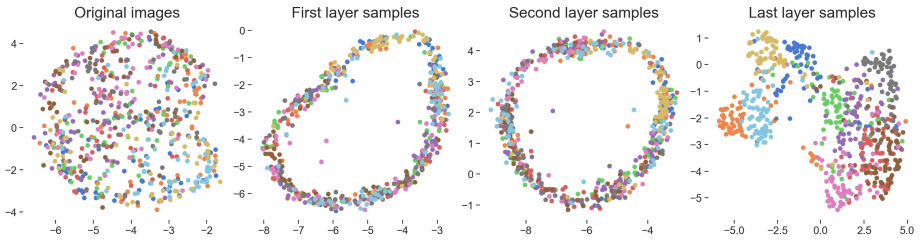


Fig. 2: UMAP embeddings [23] of the CIFAR-10 images and representations after each layer of the deep convolutional GP model. The colors correspond to different classes in the classification problem.

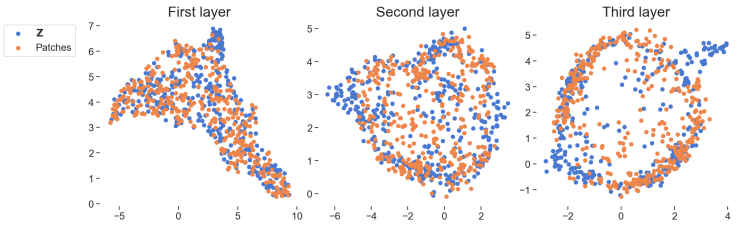


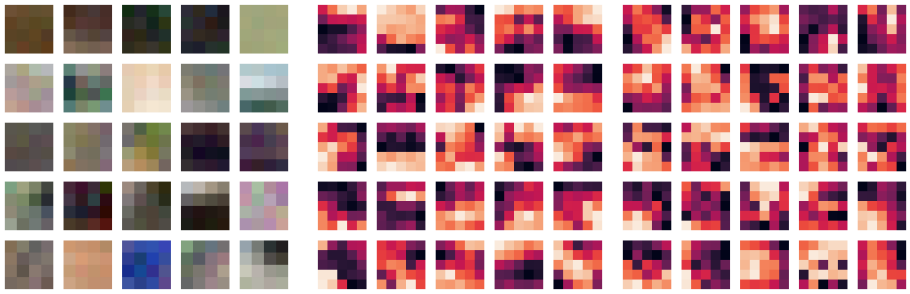
Fig. 3: UMAP embeddings of randomly selected patches of the input to the layer and learned inducing points of the fitted three layer model on CIFAR-10.

For example, on MNIST where images have size $28 \times 28 \times 1$ using patches of size $5 \times 5 \times 1$, a stride of 1 and $C = 10$ patch response functions, we obtain a representation of size $24 \times 24 \times 10$ after the first layer (height and width $W_1 = H_1 = (28 - 5)/1 + 1$). This is passed on to the next layer which produces an output of size $20 \times 20 \times 10$.

We follow the sparse GP approach of [13] and augment each patch response function by a set of M inducing patches $\mathbf{z}^\ell$ in the patch space $\mathbb{R}^{h_{\ell-1} \times w_{\ell-1} \times C_{\ell-1}}$ with corresponding responses u_c^ℓ. Each layer contains M_ℓ inducing patches $\mathbf{Z}^\ell = (\mathbf{z}_1^\ell, \ldots, \mathbf{z}_M^\ell)$ which are shared among the C patch response functions within that layer. Each patch response function has separate inducing responses $\mathbf{u}_c^\ell = (u_{c1}^\ell, \ldots, u_{cM}^\ell)$ which associate outputs to each inducing patch. We collect these into a matrix $\mathbf{U}^\ell$.

The conditional patch responses are

$$
\begin{aligned}
g_c^\ell | \mathbf{f}^{\ell-1}, \mathbf{u}_c^\ell, \mathbf{Z}^\ell &\sim \mathcal{N}(\boldsymbol{\mu}, \Sigma) \\
\boldsymbol{\mu} &= \mathbf{K}_{\mathbf{f}^{\ell-1}\mathbf{Z}^\ell} \mathbf{K}_{\mathbf{Z}^\ell \mathbf{Z}^\ell}^{-1} \mathbf{u}_c^\ell \\
\Sigma &= \mathbf{K}_{\mathbf{f}^{\ell-1}\mathbf{f}^{\ell-1}} - \mathbf{K}_{\mathbf{f}^{\ell-1}\mathbf{Z}^\ell} \mathbf{K}_{\mathbf{Z}^\ell \mathbf{Z}^\ell}^{-1} \mathbf{K}_{\mathbf{Z}^\ell \mathbf{f}^{\ell-1}},
\end{aligned}
\tag{13}
$$



(a) Layer 1 (b) Layer 2 (c) Layer 3

Fig. 4: Example inducing points $\mathbf{Z}$ pictured from all three layers from the CIFAR-10 experiment. The first layer inducing points channels correspond to color channels and are thus in color. For layers 2 and 3 only a single channel is visualized.

where the covariance between the input and the inducing variables are

$$K(\mathbf{f}^{\ell-1}, \mathbf{Z}^{\ell}) = \begin{bmatrix} k(\mathbf{f}^{\ell-1}[1], \mathbf{z}_1^{\ell}) & \cdots & k(\mathbf{f}^{\ell-1}[1], \mathbf{z}_M^{\ell}) \\ \vdots & \ddots & \vdots \\ k(\mathbf{f}^{\ell-1}[P], \mathbf{z}_1^{\ell}) & \cdots & k(\mathbf{f}^{\ell-1}[P], \mathbf{z}_M^{\ell}) \end{bmatrix}$$

a matrix of size $P_\ell \times M_\ell$ that measures the similarity of all patches against all filters $\mathbf{z}^\ell$. We set the base kernel k to be the RBF kernel. For each of the C patch response functions we obtain one output image channel.

The conditional for each layer can be evaluated in $O(P^\ell \cdot N \cdot (M^\ell)^2)$, where N is the data points being evaluated, P^ℓ the amount of patches ℓ and M^ℓ the amount of inducing points at layer ℓ.

In contrast to neural networks, the Gaussian process convolutions induce probabilistic layer representations. The first layer $p(\mathbf{f}^1|\mathbf{f}^0, \mathbf{U}^1, \mathbf{Z}^1)$ is a Gaussian directly from (13), while the following layers follow non-Gaussian distributions $p(\mathbf{f}^{\ell+1}|\mathbf{U}^{\ell+1}, \mathbf{Z}^{\ell+1})$ since we map all realisations of the random input $\mathbf{f}^\ell$ into Gaussian outputs $\mathbf{f}^{\ell+1}$.

3.2 Final classification layer

As the last layer of our model we aggregate the output of the convolutional layers using a GP with a weighted convolutional kernel as presented by [33]. We set a GP prior on the last layer patch response function

$$g^L\left(\mathbf{f}^{L-1}[p]\right) \sim \mathcal{GP}(0, K(\mathbf{f}^{L-1}[p], \mathbf{f}'^{L-1}[p'])). \tag{14}$$

with weights for each patch response. We get an additive GP

$$\begin{aligned}\mathbf{f}^L = g^L(\mathbf{f}^{L-1}) &= \sum_{p=1}^{P} w_p g^L(\mathbf{f}^{L-1}[p]) \\ &\sim \mathcal{GP}\Bigg(0, \underbrace{\sum_{p=1}^{P}\sum_{p'=1}^{P} w_p w_{p'} k(\mathbf{f}^{L-1}[p], \mathbf{f}'^{L-1}[p'])}_{K(\mathbf{x},\mathbf{x}')}\Bigg),\end{aligned}$$

where the kernel $K(\mathbf{f}^{L-1}, \mathbf{f}'^{L-1}) = \mathbf{w}^T\mathbf{K}\mathbf{w}$ is the weighted average patch similarity of the final tensor representation $\mathbf{f}^{L-1}$. $\mathbf{w} \in \mathbb{R}^P$. The matrix $\mathbf{K}$ collects all patch similarities $K(\mathbf{f}^{L-1}[p], \mathbf{f}'^{L-1}[p'])$. The last layer has one response GP per output class c.

As with the convolutional layers the inducing points live in the patch space of instead of in the image space. The inter-domain kernel is

$$\begin{aligned}K(\mathbf{f}^{L-1}, \mathbf{z}^L) &= \sum_{p=1}^{P} w_p K(\mathbf{x}[p], \mathbf{z}^L) &(15)\\ &= \mathbf{w}^T\mathbf{k}(\mathbf{f}^{L-1}, \mathbf{z}^L). &(16)\end{aligned}$$

The kernel $\mathbf{k}(\mathbf{f}^{L-1}, \mathbf{z}^L) \in \mathbb{R}^P$ collects all patch similarities of a single image $\mathbf{f}^{L-1}$ compared against inducing points $\mathbf{z}^L$. The covariance between inducing points is simply $K(\mathbf{z}^L, \mathbf{z}'^L)$. We have now defined all kernels necessary to evaluate and optimize the variational bound (9).

3.3 Doubly stochastic variational inference

The deep convolutional Gaussian process is an instance of a deep Gaussian process with the convolutional kernels and patch filter inducing points. We follow the doubly stochastic variational inference approach [27] for model learning. The key idea of doubly stochastic inference is to draw samples from the Gaussian

$$\tilde{\mathbf{f}}_i^\ell \sim p(\mathbf{f}_i^\ell | \tilde{\mathbf{f}}_i^{\ell-1}, \mathbf{U}^\ell, \mathbf{Z}^\ell) \tag{17}$$

through the deep system for a single input image $\mathbf{x}_i$.

The inducing points of each layer are independent. We assume a factorised likelihood

$$p(\mathbf{Y}|\mathbf{F}^L) = \prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{f}_i^L) \tag{18}$$

and a true joint density

$$p(\{\mathbf{f}^\ell, \mathbf{U}^\ell\}_\ell) = \prod_{\ell=1}^{L} p(\mathbf{f}^\ell|\mathbf{f}^{\ell-1}, \mathbf{U}^\ell, \mathbf{Z}^\ell)p(\mathbf{U}^\ell) \tag{19}$$

$$p(\mathbf{U}^\ell) = \prod_{c=1}^{C} \mathcal{N}(\mathbf{u}_c^\ell|\mathbf{0}, \mathbf{K}_{\mathbf{Z}^\ell\mathbf{Z}^\ell}). \tag{20}$$

The evidence framework [20] considers optimizing the evidence,

$$p(\mathbf{Y}) = \mathbb{E}_{p(\mathbf{F})} p(\mathbf{Y}|\mathbf{F}). \tag{21}$$

Following the variational approach we assume a variational joint model

$$q(\mathbf{U}^\ell) = \prod_{c=1}^{C} \mathcal{N}(\mathbf{u}_c^\ell | \mathbf{m}_c^\ell, \mathbf{S}_c^\ell) \tag{22}$$

$$q\big(\{\mathbf{f}^\ell, \mathbf{U}^\ell\}_\ell\big) = \prod_{\ell=1}^{L} p(\mathbf{f}^\ell | \mathbf{f}^{\ell-1}, \mathbf{U}^\ell, \mathbf{Z}^\ell) q(\mathbf{U}^\ell). \tag{23}$$

The distribution of the layer predictions $\mathbf{f}^\ell$ depends on current layer inducing points $\mathbf{U}^\ell, \mathbf{Z}^\ell$ and representation $\mathbf{f}^{\ell-1}$ at the previous layer. By marginalising the variational approximation $q(\mathbf{U}^\ell)$ we arrive at the factorized variational posterior of the last layer for individual data point $\mathbf{x}_i$,

$$q(\mathbf{f}_i^L; \{\mathbf{m}^\ell, \mathbf{S}^\ell, \mathbf{Z}^\ell\}_\ell) = \prod_{\ell=1}^{L-1} \int q(\mathbf{f}_i^\ell | \mathbf{f}_i^{\ell-1}, \mathbf{m}^l, \mathbf{S}^\ell, \mathbf{Z}^\ell) d\mathbf{f}_i^\ell, \tag{24}$$

where we integrate all paths $(\mathbf{f}_i^1, \ldots, \mathbf{f}_i^L)$ through the layers defined by the filters $\mathbf{Z}^\ell$, and the parameters $\mathbf{m}^\ell, \mathbf{S}^\ell$. Finally, the doubly stochastic evidence lower bound (ELBO) is

$$\begin{aligned} \log p(\mathbf{Y}) \geq \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{f}_i^L; \{\mathbf{m}^\ell, \mathbf{S}^\ell, \mathbf{Z}^\ell\}_\ell)} [\log p(\mathbf{y}_i | \mathbf{f}_i^L)] \\ - \sum_{\ell=1}^{L} \mathrm{KL}[q(\mathbf{U}^\ell) || p(\mathbf{U}^\ell)]. \end{aligned} \tag{25}$$

The variational expected likelihood is computed using a Monte Carlo approximation yielding the first source of stochasticity. The whole lower bound is optimized using stochastic gradient descent yielding the second source of stochasticity.

The Figure 2 visualises representations of CIFAR-10 images over the deep convolutional GP model. Figure 3 visualises the patch and filter spaces of the three layers, indicating high overlap. Finally, Figure 4 shows example filters $\mathbf{z}$ learned on the CIFAR-10 dataset, which extract image features.

Optimization All parameters $\{\mathbf{m}_\ell\}_{\ell=1}^L$, $\{\mathbf{S}_\ell\}_{\ell=1}^L$, $\{\mathbf{Z}^l\}_{\ell=1}^L$, the base kernel RBF lengthscales and variances and the patch weights for the last layer are learned using stochastic gradient Adam optimizer [15] by maximizing the likelihood lower bound. We use one shared base kernel for each layer.

3.4 Stochastic Gradient Hamiltonian Monte Carlo

An alternative to the variational posterior approximations $q(\mathbf{u})$ is to use Markov Chain Monte Carlo (MCMC) sampling of the true posterior $p(\mathbf{u}|\mathbf{y})$, where we

denote with $\mathbf{u} = \{\mathbf{U}^\ell\}_{\ell=1}^L$ all inducing values of all layers. We use the Stochastic Gradient Hamiltonian Monte Carlo (SG-HMC) to produce samples from the true posterior [4]. The SG-HMC can reveal the possibly multimodal and non-Gaussian inducing distributions, while the variational approximation is usually limited to Gaussian approximations. We follow the SG-HMC approach introduced for deep Gaussian processes [12].

In Hamiltonian Monte Carlo an auxiliary variable $\mathbf{v}$ is introduced and we sample from the augmented posterior

$$p(\mathbf{u}, \mathbf{v}|\mathbf{y}) \propto \exp\left(-U(\mathbf{u}) - \frac{1}{2}\mathbf{v}M^{-1}\mathbf{v}\right) \tag{26}$$

$$U(\mathbf{u}) = -\log p(\mathbf{u}|\mathbf{y}), \tag{27}$$

which corresponds to a Hamiltonian with U representing potential energy and $\mathbf{v}$ representing kinetic energy. HMC requires computation of the gradient $\nabla U(\mathbf{u})$, which is prohibitive for large datasets. In Stochastic Gradient HMC the gradients can be computed over minibatches of data, resulting in update equations

$$\Delta\mathbf{u} = \epsilon M^{-1}\mathbf{v} \tag{28}$$

$$\Delta\mathbf{v} = -\epsilon\nabla U(\mathbf{u}) - \epsilon C M^{-1}\mathbf{v} + \mathcal{N}(0, 2\epsilon(C - \hat{B})), \tag{29}$$

where C is the friction term, ϵ is the stepsize, M is the mass matrix, and $\hat{B}$ is the Fisher information matrix. We use an auto-tuning approach of [29] to select these parameters, following [12]. To compute $\nabla U(\mathbf{u})$ we use stochastic samples $\mathbf{f}_{(s)}^L \sim p(\mathbf{f}^L)$ to approximate the final layer predictive distribution $p(\mathbf{f}^L)$. Finally, to also optimize the hyperparameters, we use the Monte Carlo Expectation Maximization (MCEM) technique [32], following [12].

4 Experiments

We compare our approach on the standard image classification benchmarks of MNIST and CIFAR-10 [17], which have standard training and test folds to facilitate direct performance comparisons. MNIST contains 60,000 training examples of 28×28 sized grayscale images of 10 hand-drawn digits, with a separate 10,000 validation set. CIFAR-10 contains 50,000 training examples of RGB colour images of size 32×32 from 10 classes, with 5,000 images per class. The images represents objects such as airplanes, cats or horses. There is a separate validation set of 10,000 images. We preprocess the images for zero mean and unit variance along the color channel.

We compare our model primarily against the original shallow convolutional Gaussian process [33], which is currently the only convolutional Gaussian process based image classifier. We also consider the performance of the hybrid neural network GP approach [37]. For completeness we report the performance of a state-of-the-art CNN method DenseNet [14].

Table 1: Performance on MNIST and CIFAR-10. Our method, the deep convolutional Gaussian process, is denoted DeepCGP. Asterisk $^{(*)}$ indicates results taken from the respective publications, which are directly comparable due to standard data folds. Other results are run using our implementation. The neural network based results are listed for completeness. The four layer CNN has two 5x5 convolutional layers (64 filters, strides 2 and 1), two fully connected layers and ReLu activations.

		Inducing	Test accuracy		
Gaussian process models	Layers	points	**MNIST**	**CIFAR-10**	**Reference**
RBF AutoGP	1	200	$98.29^{(*)}$	$55.05^{(*)}$	[16]
Multi-channel conv GP	1	1000	$98.83^{(*)}$	$64.6^{(*)}$	[33]
DeepCGP	1	384	98.38	58.65	current work
DeepCGP	2	2×384	99.24	73.85	”
DeepCGP	3	3×384	**99.44**	**75.89**	”
Neural network models	Layers	# params			
Four layer CNN	4	1.7M	98.53	63.54	
Deep kernel learning	5	2.3M .. 4.6M	$99.2^{(*)}$	$77.0^{(*)}$	[37]
DenseNet	250	15.3M	N/A	$94.81^{(*)}$	[14]

Implementation. Our TensorFlow [1] implementation is compatible with the GPflow framework [22] and freely available online[2]. We leverage GPU accelerated computation, 64bit floating point precision, and employ a minibatch size of 32. We start the Adam learning rate at 0.01 and multiply it by 0.1 every 100,000 optimization steps until the learning rate reaches 1e-5. We use $M = 384$ inducing points at each layer. We set a stride of 2 for the first layer and 1 for all other layers. The convolutional filter size is 5x5 on all layers except for the first layer on CIFAR-10 where it is 4x4. This is to make use of all the image pixels using a stride of 2.

Parameter initialization. Inducing points $\mathbf{Z}$ are initialized by running k-means with M clusters on image patches from the training set. The variational means $\mathbf{m}$ are initialised to zero. $\mathbf{S}$ are initialised to a tiny variance kernel prior $10^{-5} \cdot K_{\mathbf{ZZ}}$ following [27], except for the last layer where we use $K_{\mathbf{ZZ}}$. For models deeper than two layers, we employ iterative optimisation where the first $L-2$ layers and layer L are initialised to the learned values of an $L-1$ model, while the one additional layer added before the classification layer is initialised to default values.

[2] https://github.com/kekeblom/DeepCGP

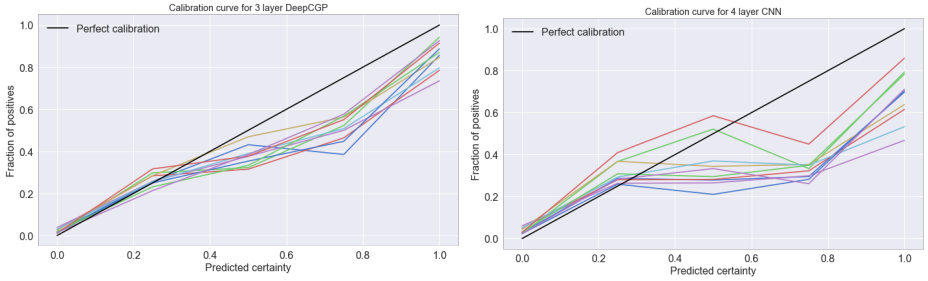


Fig. 5: Reliability curves with 5 bins for the 3 layer DeepCGP model and the 4 layer CNN on the CIFAR-10 test set. For the DeepCGP model, we average the probabilities over 25 samples of the output. We cast the classification problem as a binary one-vs-rest classification problem by summing the probabilities of the negative classes obtaining one calibration curve for each image class.

4.1 MNIST and CIFAR-10 results

Table 1 shows the classification accuracy on MNIST and CIFAR-10. Adding a convolutional layer to the weighted convolutional kernel GP improves performance on CIFAR-10 from 58.65% to 73.85%. Adding another convolutional layer further improves the accuracy to 75.9%. On MNIST the performance increases from 1.42% error to 0.56% error with the three-layer deep convolutional GP.

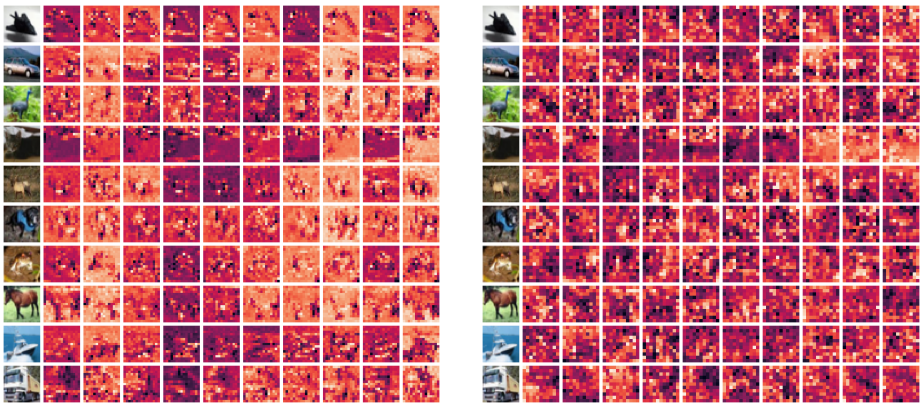
The deep kernel learning method uses a fully connected five-layer DNN instead of a CNN, and performs similarly to our model, but with much more parameters.

Figure 6 shows a single sample for 10 image class examples (rows) over the 10 patch response channels (columns) for the first layer (panel a) and second layer (panel b). The first layer indicates various edge detectors, while the second layer samples show the complexity of pattern extraction. The row object classes map to different kinds of representations, as expected.

Figure 2 shows UMAP embedding [23] visualisations of the image space of CIFAR-10 along with the structure of the layer representations $\mathbf{f}_i^\ell$ for three layers. The original images do not naturally cluster into the 10 classes (a). The DeepCGP model projects the images to circle shape with some class coherence in the intermediate layers, while the last layer shows the classification boundaries. An accompanying Figure 4 shows the learned inducing filters and layer patches on CIFAR-10. Some regions of the patch space are not covered by filters, indicating uninformative representations.

Figure 7 shows the effect of different channel numbers on a two layer model. The ELBO increases up to $C = 16$ response channels, while starts to decrease with $C = 32$ channels. A model with approximately $C = 10$ channels indicates best performance.

Figure 5 shows that the deep convolutional GP model has better calibration than a neural CNN model. CNN model results in badly calibrated class prob-



(a) Samples from the first layer. (b) Samples from the second layer.

Fig. 6: (a) and (b) show samples the first two layers of the three layer model. Rows corresponds to different test inputs and columns correspond to different patch response functions, which are realisations of the layer GPs. The first column shows the input image. The first layer seems to learn to detect edges, while the second layer appears to learn more abstract correlations of features and the representation produced no longer resembles the input image, indicating high-level feature extraction.

abilities especially between 0.2 and 0.8 prediction probability. The GP based model has more consistent calibration over the probability range.

5 Conclusions

We present a new type of deep Gaussian process with convolutional structure. The convolutional GP layers gradually linearize the data using multiple filters with nonlinear kernel functions. Our model greatly improves test results on the compared classification benchmarks compared to other GP-based approaches, and approaches the performance of hybrid neural-GP methods. The performance of our model seems to improve as more layers are added. We leave experimenting with deeper models for future work.

Convolutional neural networks have been shown to provide unreliable uncertainty estimates [31]. We showed that our model provides more accurate class probability estimates than an equivalent deep convolutional neural network.

Deep Gaussian process models lead to degenerate covariances, where each layer in the composition reduces the rank or degrees of freedom of the system [7]. In practise the rank reduces via successive layers mapping inputs to identical values, effectively merging inputs and resulting in rank-reducing covariance matrix with repeated rows and columns. To counter this pathology rank-preserving deep model was proposed by pseudo-monotonic layer mappings with GP priors

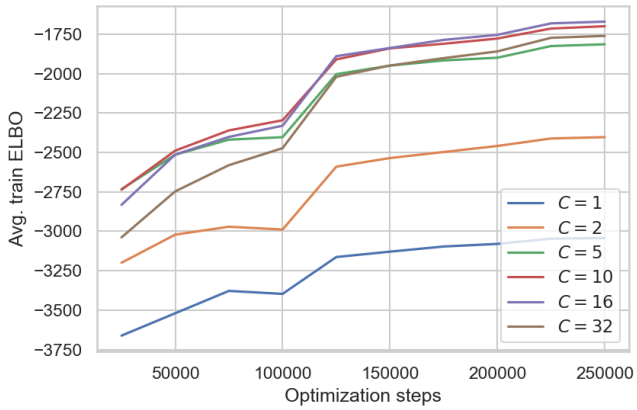


Fig. 7: Expected evidence lower bound computed on the training set using a two layer model for different amounts of patch response functions. The models with 10 and 16 patch response functions seem to perform the best. Models with one or two patch response functions struggle to explain the data even though they have the same amount of inducing points.

$f(\mathbf{x}) \sim \mathcal{GP}(\mathbf{x}, k)$ with identity means $\mathbb{E}[f(\mathbf{x})] = \mathbf{x}$ [27]. In contrast we employ zero-mean patch response functions. Remarkably we do not experience rank degeneracy, possibly due to the multiple channel mappings and the convolution structure.

The convolutional Gaussian process is still limited by the computationally expensive inference. The SG-HMC improves over variational inference, while an another avenue for improvement lies in kernel interpolation techniques [36,9], which would make inference and prediction faster. We leave further exploration of these directions as future work.

Acknowledgements We thank Michael Riis Andersen for his invaluable comments and helpful suggestions.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: a system for large-scale machine learning. In: OSDI. vol. 16, pp. 265–283 (2016)
2. Blei, D.M., Kucukelbir, A., McAuliffe, J.D.: Variational inference: A review for statisticians. Journal of the American Statistical Association **112**(518), 859–877 (2017)
3. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural networks. In: International Conference on Machine Learning. pp. 1613–1622 (2015)

4. Chen, T., Fox, E., Guestrin, C.: Stochastic gradient hamiltonian monte carlo. In: International Conference on Machine Learning. pp. 1683–1691 (2014)
5. Damianou, A., Lawrence, N.: Deep gaussian processes. In: AISTATS. PMLR, vol. 31, pp. 207–215 (2013)
6. Dutordoir, V., van der Wilk, M., Artemev, A., Tomczak, M., Hensman, J.: Translation insensitivity for deep convolutional gaussian processes. arXiv:1902.05888 (2019)
7. Duvenaud, D., Rippel, O., Adams, R., Ghahramani, Z.: Avoiding pathologies in very deep networks. In: AISTATS. PMLR, vol. 33, pp. 202–210 (2014)
8. Duvenaud, D.K., Nickisch, H., Rasmussen, C.E.: Additive gaussian processes. In: Advances in neural information processing systems. pp. 226–234 (2011)
9. Evans, T.W., Nair, P.B.: Scalable gaussian processes with grid-structured eigenfunctions (GP-GRIEF). In: International Conference on Machine Learning (2018)
10. Garriga-Alonso, A., Aitchison, L., Rasmussen, C.E.: Deep convolutional networks as shallow gaussian processes. In: ICLR (2019)
11. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning, vol. 1. MIT press Cambridge (2016)
12. Havasi, M., Lobato, J.M.H., Fuentes, J.J.M.: Inference in Deep Gaussian Processes using Stochastic Gradient Hamiltonian Monte Carlo. NIPS (2018)
13. Hensman, J., Matthews, A., Ghahramani, Z.: Scalable variational gaussian process classification. In: AISTATS. PMLR, vol. 38, pp. 351–360 (2015)
14. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR. vol. 1, p. 3 (2017)
15. Kingma, D.P., Ba, J.L.: Adam: A method for stochastic optimization. In: ICLR (2014)
16. Krauth, K., Bonilla, E.V., Cutajar, K., Filippone, M.: AutoGP: Exploring the capabilities and limitations of Gaussian process models. In: Uncertainty in Artificial Intelligence (2017)
17. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
18. Kumar, V., Singh, V., Srijith, P., Damianou, A.: Deep gaussian processes with convolutional kernels. arXiv preprint arXiv:1806.01655 (2018)
19. Lee, J., Bahri, Y., Novak, R., Schoenholz, S.S., Pennington, J., Sohl-Dickstein, J.: Deep neural networks as gaussian processes. In: ICLR (2018)
20. MacKay, D.J.: A practical bayesian framework for backpropagation networks. Neural computation **4**, 448–472 (1992)
21. Mallat, S.: Understanding deep convolutional networks. Phil. Trans. R. Soc. A **374**(2065), 20150203 (2016)
22. Matthews, A.G.d.G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., Hensman, J.: GPflow: A Gaussian process library using TensorFlow. Journal of Machine Learning Research **18**, 1–6 (2017)
23. McInnes, L., Healy, J.: UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. ArXiv e-prints (Feb 2018)
24. Rasmussen, C.E.: Gaussian processes in machine learning. In: Advanced lectures on machine learning, pp. 63–71. Springer (2004)
25. Rasmussen, C.E., Williams, C.K.: Gaussian process for machine learning. MIT press (2006)
26. Remes, S., Heinonen, M., Kaski, S.: Non-stationary spectral kernels. In: Advances in Neural Information Processing Systems. pp. 4642–4651 (2017)

27. Salimbeni, H., Deisenroth, M.: Doubly stochastic variational inference for deep gaussian processes. In: Advances in Neural Information Processing Systems. pp. 4588–4599 (2017)
28. Snelson, E., Ghahramani, Z.: Sparse gaussian processes using pseudo-inputs. In: Advances in neural information processing systems. pp. 1257–1264 (2006)
29. Springerberg, J., Klein, A., Falkner, S., Hutter, F.: Bayesian optimization with robust bayesian neural networks. In: Advances in Neural Information Processing Systems. pp. 4134–4142 (2016)
30. Sun, S., Zhang, G., Wang, C., Zeng, W., Li, J., Grosse, R.: Differentiable compositional kernel learning for gaussian processes. In: ICML. PMLR, vol. 80 (2018)
31. Tran, G.L., Bonilla, E.V., Cunningham, J.P., Michiardi, P., Filippone, M.: Calibrating deep convolutional gaussian processes. In: AISTATS. PMLR, vol. 89, pp. 1554–1563 (2019)
32. Wei, G., Tanner, M.: A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. Journal of the American statistical Association **85**, 699–704 (1990)
33. Van der Wilk, M., Rasmussen, C.E., Hensman, J.: Convolutional gaussian processes. In: Advances in Neural Information Processing Systems. pp. 2849–2858 (2017)
34. Williams, C.K.: Computing with infinite networks. In: Advances in Neural Information Processing Systems. pp. 295–301 (1997)
35. Wilson, A., Gilboa, E., Nehorai, A., Cunningham, J.: Fast multidimensional pattern extrapolation with gaussian processes. In: AISTATS. PMLR, vol. 31 (2013)
36. Wilson, A., Nickisch, H.: Kernel interpolation for scalable structured gaussian processes (KISS-GP). In: International Conference on Machine Learning. PMLR, vol. 37, pp. 1775–1784 (2015)
37. Wilson, A.G., Hu, Z., Salakhutdinov, R.R., Xing, E.P.: Stochastic variational deep kernel learning. In: Advances in Neural Information Processing Systems. pp. 2586–2594 (2016)
38. Wilson, A.G., Hu, Z., Salakhutdinov, R., Xing, E.P.: Deep kernel learning. In: AISTATS. PMLR, vol. 51, pp. 370–378 (2016)