Deep convolutional Gaussian processes

Kenneth Blomqvist  Samuel Kaski  Markus Heinonen
Department of Computer Science, Aalto University
Helsinki Institute for Information Technology HIIT

October 9, 2018

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 Gaussian process approaches on the MNIST and CIFAR-10 datasets. In particular, we improve CIFAR-10 accuracy by over 10 percentage points.

1 Introduction

Gaussian processes (GPs) are a family of flexible function distributions defined by a kernel function (Rasmussen and Williams, 2006). 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 (Wilson et al., 2013; Remes et al., 2017). Deep Gaussian processes boost performance by modelling networks of GP nodes (Duvenaud et al., 2011; Sun et al., 2018) or by mapping inputs through multiple Gaussian process ‘layers’ (Damianou and Lawrence, 2013; Salimbeni and Deisenroth, 2017). 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 (Duvenaud et al., 2014).

Convolutional neural networks (CNN) are a celebrated approach for image recognition tasks with superior performance (Mallat, 2016). 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 (Blundell et al., 2015; Gal and Ghahramani, 2016) explored the Bayesian connections of the dropout technique. Neural networks are known to converge to Gaussian processes at the limit of infinite layer width (MacKay, 1992; Williams, 1997; Lee et al., 2017). Garriga-Alonso et al. (2018) derive a kernel which is equivalent to residual CNNs with a certain prior over the weights. Wilson et al. (2016a) proposed a hybrid deep kernel learning approach, where a feature-extractor deep neural network is stacked with a Gaussian process predictor layer, learning the neural network weights by variational inference (Wilson et al., 2016a).

Recently Van der Wilk et al. (2017) proposed the first convolution-based Gaussian process for images with promising performance. They proposed a 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 Kumar et al. (2018) applied convolutional kernels in a deep Gaussian process, however they were unable to significantly improve upon the shallow convolutional GP model.

In this paper we propose a deep convolutional Gaussian process, which iteratively convolves several GP functions over the 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%. Our model demonstrates how a purely GP based approach can reach the performance of hybrid neural network...
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 signal (Goodfellow et al., 2016). 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 \( x \) and a convolutional filter \( g \) is defined:

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

where \( x[i, j] \in \mathbb{R}^3 \) and \( g \) 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 (Rasmussen, 2004). A zero-mean Gaussian process prior on latent function \( f(x) \in \mathbb{R} \),

\[
f(x) \sim \mathcal{GP}(0, K(x, x'))
\]

defines a prior distribution over function values \( f(x) \) with mean and covariance:

\[
\mathbb{E}[f(x)] = 0
\]

\[
\text{cov}(f(x), f(x')) = K(x, x')
\]

A GP prior defines that for any collection of \( n \) inputs \( X = (x_1, \ldots, x_n)^T \), the corresponding function values

\[
f = (f(x_1), \ldots, f(x_n))^T \in \mathbb{R}^n
\]

follow a multivariate Normal distribution

\[
f \sim \mathcal{N}(0, K)
\]

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

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

\[
f|u, Z \sim \mathcal{N}(K_{XX}(K_{ZZ} + K_{XZ}K_{ZZ}^{-1}K_{ZX}), K_{XX} - K_{XZ}K_{ZZ}^{-1}K_{ZX})
\]

where \( K_{XX} \in \mathbb{R}^{n \times n} \) is the kernel between observed image pairs \( X \), the kernel \( K_{XZ} \in \mathbb{R}^{n \times M} \) is between observed images \( X \) and inducing images \( Z \), and kernel \( K_{ZZ} \in \mathbb{R}^{M \times M} \) is between inducing images \( Z \) (Snelson and Ghahramani, 2006).

### 2.3 Variational inference

Exact inference in a GP entails optimizing the evidence \( p(y) = \mathbb{E}_{p(f)}[p(y|f)] \) which has a limiting cubic complexity \( O(n^3) \) and is in general intractable. We tackle this restriction by applying stochastic variational inference (Hensman et al., 2015a).

We define a variational approximation

\[
q(u) = \mathcal{N}(u|m, S)
\]

\[
q(f) = \int p(f|u)q(u)du
\]

\[
= \mathcal{N}(f|Am, K_{ff} - A(S - K_{zz})A^T)
\]

\[
A = K_{f^Tz}K_{zz}^{-1}
\]

with free variational parameters \( m \in \mathbb{R}^m \) and a matrix \( S \succ 0 \in \mathbb{R}^{m \times m} \) to be optimised. It can be shown that minimizing the Kullback-Leibler divergence \( \text{KL}[q(u)||p(u|y)] \) between the approximative posterior \( q(u) \) and the true posterior \( p(u|y) \) is equivalent to maximizing the evidence lower bound (ELBO) (Blei et al., 2017)

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

The variational expected likelihood in \( \mathcal{L} \) can be computed using numerical quadrature approaches (Hensman et al., 2015b).

### 3 Deep convolutional Gaussian process

In this section we introduce the deep convolutional 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 \( f^\ell \in \mathbb{R}^{H_\ell \times w_\ell} \) with \( H_\ell \) and height \( H_\ell \) pixels at layer \( \ell \). We collect \( C_\ell \) channels into a 3D tensor \( f^\ell = (f^\ell_1, \ldots, f^\ell_C) \in \mathbb{R}^{H_\ell \times w_\ell \times C_\ell} \), where the channels are along the depth axis. The input image \( f^0 = x \) 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 \( f^\ell \) into patches \( f^\ell[p] \in \mathbb{R}^{w_\ell \times h_\ell \times C_\ell} \) containing all depth channel, \( h_\ell \) and \( w_\ell \) are the height and width of the image patch at layer \( \ell \). We index patches by \( p \in \mathbb{Z} < H_\ell \times W_\ell \). \( H_\ell \) and \( W_\ell \) denote the height and width of the output of layer \( \ell \). We compose a sequence of layers \( f^\ell \) that map the input image \( x_i \) to the label \( y_i \):

\[
\begin{align*}
X_i & = f^0 \xrightarrow{g^0} f^1 \xrightarrow{g^1} \cdots \xrightarrow{g^{C-1}} f^C \approx y_i \quad (0,1)^C_y \quad (10)
\end{align*}
\]

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

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

\[
\begin{align*}
f^\ell[p] &= \begin{bmatrix}
g^\ell_1(f^{\ell-1}[p]) \\
\vdots \\
g^\ell_C(f^{\ell-1}[p])
\end{bmatrix} \in \mathbb{R}^C \quad (11)
\end{align*}
\]

Each individual patch response \( g^\ell(f^{\ell-1}[p]) \) is a \( 1 \times 1 \times C \) pixel stack. By repeating the patch responses over the \( P_{\ell-1} = H_\ell \times W_\ell \) patches we form a new \( W_\ell \times H_\ell \times C_\ell \) representation \( f^\ell = (f^\ell_1, \ldots, f^\ell_{C\ell}) \) (See Figure 1).

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

\[
g_c^\ell(f^{\ell-1}[p]) \sim GP(0, k(f^{\ell-1}[p], f^{\ell-1}[p'])) \quad (12)
\]

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 \( g^\ell_c \) output similar responses for similar patches.

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 Hensman et al. (2015a) and augment each patch response function by a set of \( M \) inducing patches \( z^\ell \) in the patch space \( \mathbb{R}^{h_{\ell-1} \times w_{\ell-1} \times C_{\ell-1}} \) with corresponding responses \( u^\ell_c \). Each layer contains \( M_\ell \) inducing patches \( Z^\ell = (z^\ell_1, \ldots, z^\ell_{M_\ell}) \) which are shared among the \( C \) patch response functions within that layer. Each patch response function has separate inducing responses \( u^\ell_c = (u^\ell_{c,1}, \ldots, u^\ell_{c,M_\ell}) \) which associate outputs to each inducing patch. We collect these into a matrix \( U^\ell \).

The conditional patch responses are

\[
g_c^\ell(f^{\ell-1}), u^\ell_c, Z^\ell \sim \mathcal{N}(\mu, \Sigma) \quad (13)
\]

\[
\mu = \mathbf{K}_{f^{\ell-1}, Z^\ell} \mathbf{K}_{Z^\ell, Z^\ell}^{-1} u^\ell_c
\]

\[
\Sigma = \mathbf{K}_{f^{\ell-1}, f^{\ell-1}} - \mathbf{K}_{f^{\ell-1}, Z^\ell} \mathbf{K}_{Z^\ell, Z^\ell}^{-1} \mathbf{K}_{Z^\ell, f^{\ell-1}},
\]

where the covariance between the input and the inducing
variables are

\[
K(f^{\ell-1}, Z^{\ell}) = \begin{bmatrix}
k(f^{\ell-1}[1], z_1^\ell) & \cdots & k(f^{\ell-1}[1], z_M^\ell) \\
\vdots & \ddots & \vdots \\
k(f^{\ell-1}[P], z_1^\ell) & \cdots & k(f^{\ell-1}[P], z_M^\ell)
\end{bmatrix}
\]

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

\[
f^L = g^L(f^{L-1}) = \sum_{p=1}^P w_p g^L(f^{L-1}[p])
\]

\[
\sim \mathcal{GP}\left(0, \sum_{p=1}^P \sum_{p'=1}^P w_p w_{p'} k(f^{L-1}[p], f^{L-1}[p']) \right)
\]

where the kernel \(K(f^{L-1}, f'^{L-1}) = w^T K w\) is the weighted average patch similarity of the final tensor representation \(f^{L-1}\). The matrix \(K\) collects all patch similarities \(K(f^{L-1}[p], f'^{L-1}[p'])\). 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 interdomain kernel is

\[
K(f^{L-1}, z^L) = \sum_{p=1}^P w_p K(x[p], z^L)
\]

\[
= w^T k(f^{L-1}, z^L).
\]

The kernel \(k(f^{L-1}, z^L) \in \mathbb{R}^P\) collects all patch similarities of a single image \(f^{L-1}\) compared against inducing points \(z^L\).
The covariance between inducing points is simply $K(z^L, z^L)$. 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 filters inducing points. We follow the doubly stochastic variational inference approach of Salimbeni and Deisenroth (2017) for model learning. The key idea of doubly stochastic inference is to draw samples from the Gaussian

$$
\tilde{f}_i^\ell \sim p(f_i^\ell | f_i^{\ell-1}, U_\ell, Z^\ell)
$$

through the deep system for a single input image $x_i$.

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

$$
p(Y | F^L) = \prod_{i=1}^N p(y_i | f_i^L)
$$

and a true joint density

$$
p(f^L, U^L) = \prod_{\ell=1}^L p(f^\ell | f^{\ell-1}, U^\ell, Z^\ell) p(U^\ell) = \prod_{c=1}^C \mathcal{N}(u_{i,c}^\ell | m_{i,c}^\ell, S_{i,c}^\ell).
$$

The evidence framework MacKay (1992) considers optimizing the evidence,

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

Following the variational approach we assume a variational joint model

$$
q(U^\ell) = \prod_{c=1}^C \mathcal{N}(u_{i,c}^\ell | m_{i,c}^\ell, S_{i,c}^\ell)
$$

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

$$
q(f_i^L; \{m^\ell, S^\ell, Z^\ell\}_\ell) = \prod_{\ell=1}^{L-1} \int q(f_i^\ell | f_i^{\ell-1}, m^\ell, S^\ell, Z^\ell) df_i^\ell,
$$

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

$$
\log p(Y) \geq \sum_{i=1}^N \mathbb{E}_{q(f_i^1, \ldots, f_i^L)} \log p(y_i | f_i^L) - \sum_{\ell=1}^L \text{KL}[q(U^\ell) || p(U^\ell)].
$$

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.

Figure 4: Example inducing points $Z$ 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.
We preprocess the images for zero mean and unit variance. We compare our approach on the standard image classification benchmarks of MNIST and CIFAR-10, 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 represent 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 (Van der Wilk et al., 2017), which is currently the only convolutional Gaussian process based image classifier. We also consider the performance of the hybrid neural network GP approach of Wilson et al. (2016b). For completeness we report the performance of a state-of-the-art CNN method DenseNet (Huang et al., 2017).

### 4 Experiments

We compare our approach on the standard image classification benchmarks of MNIST and CIFAR-10 (Krizhevsky and Hinton, 2009), 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 represent 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 (Van der Wilk et al., 2017), which is currently the only convolutional Gaussian process based image classifier. We also consider the performance of the hybrid neural network GP approach of Wilson et al. (2016b). For completeness we report the performance of a state-of-the-art CNN method DenseNet (Huang et al., 2017).

### Implementation

Our TensorFlow (Abadi et al., 2016) implementation is compatible with the GPflow framework (Matthews et al., 2017) and freely available online. 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 $Z$ are initialized by running $k$-means with $M$ clusters on image patches from the training set. The variational means $m$ are initialised to zero. $S$ are initialised to a tiny variance kernel prior $10^{-5}$ · $K_{zz}$ following Salimbeni and Deisenroth (2017), except for the last layer where we use $K_{zz}$. For models deeper than two layers, we employ iterative optimisation where the first $L − 2$ layers and layer $L$ are initialised to the learnt values of an $L − 1$ model, while the one additional layer added before the classification layer is initialised to default values.

### 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.

| Model                      | Layers | # params | MNIST | CIFAR-10 | Reference          |
|----------------------------|--------|----------|-------|----------|--------------------|
| RBF AutoGP                 | 1      | 200      | 98.29(*) | 55.05(*) | Krauth et al. (2017) |
| Multi-channel conv GP      | 1      | 1000     | 98.83(*) | 64.6(*)  | Van der Wilk et al. (2017) |
| 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    | "                  |

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 Figure visualises representations of CIFAR-10 images over the deep convolutional GP model. Figure visualises the patch and filter spaces of the three layers, indicating high overlap. Finally, Figure shows example filters learned on the CIFAR-10 dataset, which extract image features.

Optimization

All parameters $\{m_\ell\}_{\ell=1}^L$, $\{S_\ell\}_{\ell=1}^L$, $\{Z_\ell\}_{\ell=1}^L$, the base kernel RBF lengthscales and variances and the patch weights for the last layer are learned using stochastic gradient Adam optimizer (Kingma and Ba, 2014) by maximizing the likelihood lower bound. We use one shared base kernel for each layer.

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

| Gaussian process models | Layers | Inducing points | Test accuracy | Reference          |
|-------------------------|--------|----------------|--------------|--------------------|
| RBF AutoGP              | 1      | 200            | 98.29(*)     | 55.05(*)          | Krauth et al. (2017) |
| Multi-channel conv GP   | 1      | 1000           | 98.83(*)     | 64.6(*)           | Van der Wilk et al. (2017) |
| 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 | Test accuracy |
|-------------------------|--------|----------|---------------|
| Deep kernel learning    | 5      | 2.3M .. 4.6M | 99.2(*)     | 77.0(*)           | Wilson et al. (2016b) |
| DenseNet                | 250    | 15.3M    | N/A           | 94.81(*)          | Huang et al. (2017)   |
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 5 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 [McInnes and Healy (2018)] visualisations of the image space of CIFAR-10 along with the structure of the layer representations $f^i$ for three layers. The original images do not naturally cluster into the 10 classes (a). The DCGP 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 6 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.

5 Conclusions

We presented 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 did not experiment with using a stride of 1 at the first layer. Neither did we try models with 4 or more layers. The added complexity comes with an increased computational cost and we were thus limited from experimenting with these improvements. We believe that both of these enhancements would increase performance.

Deep Gaussian process models lead to degenerate covariances, where each layer in the composition reduces the rank or degrees of freedom of the system [Duvenaud et al., 2014]. 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 [Salimbeni and Deisenroth, 2017] proposed rank-preserving deep model by pseudo-monotonic layer mappings with GP priors $f(x) \sim GP(x, k)$ with identity means $E[f(x)] = x$. 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.

There are several avenues for improved efficiency and modelling capacity. The Stochastic Gradient Hamiltonian Monte Carlo approach (Ma et al., 2015) has proven efficient in deep GPs (Havasi et al., 2018) and in GANs (Saati and Wilson, 2017). Another avenue for improvement lies in kernel interpolation techniques (Wilson and Nickisch, 2015; Evans and Nair, 2018) which would make inference and prediction faster. We leave these directions for future work.

### Acknowledgements

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

### References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In OSDI, volume 16, pages 265–283, 2016.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *International Conference on International Conference on Machine Learning*, pages 1613–1622, 2015.

Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *AISTATS*, pages 207–215, 2013.

David Duvenaud, Oren Rippel, Ryan Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. In *AISTATS*, pages 202–210, 2014.

David K Duvenaud, Hannes Nickisch, and Carl E Rasmussen. Additive gaussian processes. In *Advances in neural information processing systems*, pages 226–234, 2011.

Trefor W Evans and Prasanth B Nair. Scalable gaussian processes with grid-structured eigenfunctions (GP-GRIEF). In *International Conference on Machine Learning*, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

Adrià Garriga-Alonso, Laurence Aitchison, and Carl Edward Rasmussen. Deep convolutional networks as shallow gaussian processes. *arXiv preprint arXiv:1806.05490*, 2018.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Marton Havasi, José Miguel Hernández Lobato, and Juan Jose Murillo Fuentes. Inference in deep gaussian processes using stochastic gradient hamiltonian monte carlo. *arXiv preprint arXiv:1806.05490*, 2018.

James Hensman, Alexander G de G Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *AISTATS*, pages 351–360, 2015a.

James Hensman, Alexander G de G Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *AISTATS*, 2015b.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, volume 1, page 3, 2017.

Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Proc. 3rd Int. Conf. Learn. Representations*, 2014.

Karl Krauth, Edwin V Bonilla, Kurt Cutajar, and Maurizio Filippone. Autogp: Exploring the capabilities and limitations of Gaussian process models. In *Uncertainty in Artificial Intelligence*, Sydney, Australia, 08 2017. URL http://www.eurecom.fr/publication/5038.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.
Vinayak Kumar, Vaibhav Singh, PK Srijith, and Andreas Damianou. Deep gaussian processes with convolutional kernels. *arXiv preprint arXiv:1806.01655*, 2018.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

Yi-An Ma, Tianqi Chen, and Emily Fox. A complete recipe for stochastic gradient mcmc. In *Advances in Neural Information Processing Systems*, pages 2917–2925, 2015.

David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4:448–472, 1992.

Stéphane Mallat. Understanding deep convolutional networks. *Phil. Trans. R. Soc. A*, 374(2065):20150203, 2016.

Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo Leon-Villagrá, Zoubin Ghahramani, and James Hensman. GIPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, apr 2017. [URL](http://jmlr.org/papers/v18/16-537.html).

L. McInnes and J. Healy. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, February 2018.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

Carl Edward Rasmussen and Christopher KI Williams. Gaussian process for machine learning. MIT press, 2006.

Sami Remes, Markus Heinonen, and Samuel Kaski. Non-stationary spectral kernels. In *Advances in Neural Information Processing Systems*, pages 4642–4651, 2017.

Yunus Saatci and Andrew G Wilson. Bayesian gan. In *Advances in neural information processing systems*, pages 3622–3631, 2017.

Hugh Salimbeni and Marc Deisenroth. Doubly stochastic variational inference for deep gaussian processes. In *Advances in Neural Information Processing Systems*, pages 4588–4599, 2017.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2006.

S. Sun, G. Zhang, C. Wang, W. Zeng, J. Li, and R. Grosse. Differentiable compositional kernel learning for gaussian processes. In *ICML*, 2018.