Bayesian Image Classification with Deep Convolutional Gaussian Processes

Vincent Dutordoir **Mark van der Wilk** **Artem Artemev** **James Hensman**
PROWLER.io, Cambridge, United Kingdom

Abstract

In decision-making systems, it is important to have classifiers that have calibrated uncertainties, with an optimisation objective that can be used for automated model selection and training. Gaussian processes (GPs) provide uncertainty estimates and a marginal likelihood objective, but their weak inductive biases lead to inferior accuracy. This has limited their applicability in certain tasks (e.g. image classification). We propose a translation-insensitive convolutional kernel, which relaxes the translation invariance constraint imposed by previous convolutional GPs. We show how we can use the marginal likelihood to learn the degree of insensitivity. We also reformulate GP image-to-image convolutional mappings as multi-output GPs, leading to deep convolutional GPs. We show experimentally that our new kernel improves performance in both single-layer and deep models. We also demonstrate that our fully Bayesian approach improves on dropout-based Bayesian deep learning methods in terms of uncertainty and marginal likelihood estimates.

1 INTRODUCTION

To be useful in the real world, decision-making systems have to be able to represent uncertainty. This enables the system to gracefully deal with unseen or special cases and, for example, hand over control to a human operator when the uncertainty is high. It is also crucial to have an accurate measure of uncertainty when making automated decisions based on machine classification (e.g. medical diagnosing).

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108.

Recently, Bayesian deep learning methods based on dropout have been empirically successful in improving the robustness of Deep Neural Nets (DNN) predictions [Gal and Ghahramani, 2016], but it is unclear to what extent they accurately approximate the true posteriors [Hron et al., 2018]. They also do not deliver on an important promise of the Bayesian framework: automatic regularisation of model complexity which allows the training of hyperparameters [Rasmussen and Ghahramani, 2001]. Current marginal likelihood estimates are not usable for hyperparameter selection, and the strong relationship between their quality, and the quality of posterior approximations suggests that further improvements are possible with better Bayesian approximations.

We are interested in Gaussian processes (GPs) as an alternative building block for creating deep learning models with the benefits of Bayesian inference. Their practical application has been limited due to their large computational requirements for big datasets, and due to the limited inductive biases that they can encode. In recent years, however, advances in stochastic variational inference have enabled GPs to be scaled to large datasets for both regression and classification models [Hensman et al., 2013, 2015]. More sophisticated model structures that are common in the deep learning community, such as depth [Damianou and Lawrence, 2013] and convolutions [van der Wilk et al., 2017], have been incorporated as well. Notably, inference is still accurate enough to provide marginal likelihood estimates that can be used for hyperparameter selection (e.g. [van der Wilk et al., 2018]).

In this work, we focus on creating models for image inputs. While existing GP models with kernels like the Squared Exponential (SE) kernel have the capacity to learn any well-behaved function when given infinite data [Rasmussen and Williams, 2006, chapter 7], they are unlikely to work well for image tasks with realistic dataset sizes. Local kernels, like the SE, constrain only functions in the prior to be smooth, and allow the function to vary along any direction in the input space. This will allow these models to generalise only in neigh-

bourhoods near training data, with large uncertainties being predicted elsewhere. This excessive flexibility is a particular problem for images, which have high input dimensionality, while exhibiting a large amount of structure. When designing Bayesian models it is crucial to think about sensible inductive biases to incorporate into the model. For instance, convolutional structure has been widely used to address this issue [LeCun et al., 1989, Goodfellow et al., 2016]. Van der Wilk et al. [2017] introduced this structure into a single-layer GP, together with an efficient inference scheme, and showed that this improved performance on image classification tasks. Recently, Blomqvist et al. [2019] added convolutional structure to deep GPs, which led to deep convolutional Gaussian processes (DCGPs).

Contributions We start by re-formulating the hidden layers of a DCGP as a correlated multi-output GP. This is a convenient abstraction that enables us to code the convolutional layers in our efficient multi-output GP framework [van der Wilk et al., 2020]. We then identify that translational invariant properties of current convolutional models are too restrictive and limits performance. To remedy this, we introduce the Translation Insensitive Convolutional Kernel (TICK), which removes the restriction of requiring identical outputs for identical patch inputs. We compare our model to current convolutional GPs, and find improvement in performance in both accuracy and uncertainty quantification. Comparing our model to dropout-based Bayesian deep learning methods, we show how our model is competitive in terms of accuracy but also comes with the desirable properties of a truly Bayesian model: a marginal likelihood for model selection, automatically tuning of hyperparameters, and calibration.

2 BACKGROUND

Gaussian Process Models Gaussian processes (GPs) [Rasmussen and Williams, 2006] are non-parametric distributions over functions similar to Bayesian neural networks. The core difference is that neural networks represent distributions over functions through distributions on weights, while a Gaussian process specifies a distribution on function values at a collection of input locations. Using this representation allows us to use an infinite number of basis functions, while still enables Bayesian inference [Neal, 1996]. In a GP, the joint distribution of these function values is Gaussian and is fully determined by its mean $\mu(\cdot)$ and covariance (kernel) function $k(\cdot,\cdot)$. Taking the mean function to be zero without loss of generality, function values at inputs $X = \{\mathbf{x}_m\}_{m=1}^M$ are distributed as $f(X) \sim \mathcal{N}(f(X); \mu(X), \mathbf{K}_{\mathrm{XX}})$, where $[\mathbf{K}_{\mathrm{XX}}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. The Gaussianity, and the fact that we can manipulate function values at some finite points of interest without taking the behaviour at any other points into account (the marginalisation property) make GPs particularly convenient to manipulate and use as priors over functions in Bayesian models.

Convolutional Gaussian Processes Van der Wilk et al. [2017] construct the convolutional kernel for functions from images of size $D = W \times H$ to real-valued responses $f : \mathbb{R}^D \to \mathbb{R}$. Their starting point is a *patch response function* $g : \mathbb{R}^E \to \mathbb{R}$ operating on patches of the input image of size $E = w \times h$. The output for a particular image is found by taking a sum of the patch response function applied to all patches of the image. A vectorised image $\mathbf{x}$ of height H and width W contains $P = (H - h + 1) \times (W - w + 1)$ overlapping patches when we slide the window one pixel at a time (i.e. a vertical and horizontal stride of 1). We denote the p^{th} patch of an image as $\mathbf{x}^{[p]}$. Placing a GP prior on $g(\cdot) \sim \mathcal{GP}(0, k_g(\cdot,\cdot))$ implies:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{p=1}^{P} g\left(\mathbf{x}^{[p]}\right) \\ \implies f(\mathbf{x}) &\sim \mathcal{GP}\left(0, \sum_{p=1}^{P}\sum_{p'=1}^{P} k_g\left(\mathbf{x}^{[p]}, \mathbf{x}^{[p']}\right)\right). \end{aligned} \tag{1}$$

The convolution kernel places much stronger constraints on the functions in the prior, based on the idea that similar patches contribute similarly to the function's output, regardless of their position. This prior places more mass in functions that are sensible for images, and therefore allow the model to generalise stronger and with less uncertainty than, for example, the SE kernel. If these assumptions are appropriate for a given dataset, this leads to a model with a higher marginal likelihood and better generalisation on unseen test data.

Deep Gaussian Processes Convolutional structure is an example of how the kernel and its associated feature representation influence the performance of a model. Deep learning models partially automate this feature selection by learning feature hierarchies from the training data. The first layers usually identify edges, corners, and other local features, while combining them into more complicated silhouettes further into the hierarchy. Eventually a simple regressor solves the task.

Deep GPs (DGPs) share this compositional nature, by composing layers of GPs [Damianou and Lawrence, 2013]. They can be defined as $f(\cdot) = f_L(\ldots f_2(f_1(\cdot)))$, where each component is a GP, itself $f_\ell(\cdot) \sim \mathcal{GP}(0, k_\ell(\cdot,\cdot))$. DGPs enable us to specify priors on flexible functions with compositional structure, and open the door to non-parametric Bayesian feature

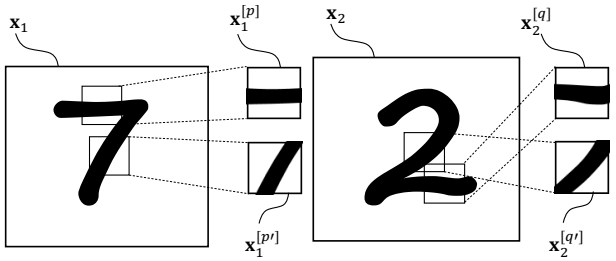


Figure 1: Illustration of why translation *invariance* may be an unrealistic modelling assumption. The highlighted patches are not useful for a translation invariant patch response function $g(\cdot)$, as used in the original convolutional GP [van der Wilk et al., 2017], because they appear in both images: only when their relative locations are taken into consideration are these patches useful for classification.

learning. Salimbeni and Deisenroth [2017] showed that this is crucial to achieve state-of-the-art performance on many datasets, and that DGP models never perform worse than single-layer GPs.

3 BAYESIAN MODELLING OF IMAGES

3.1 Limits of the Conv-GP Kernel

In this section we focus on analysing the behaviour of single-layer convolutional GPs (Conv-GPs), so we can develop improvements in a targeted way. The convolutional structure in eq. (1) introduces a form of translation invariance, because the same GP $g(\cdot)$ is used for all patches in the image, regardless of location. As stated by Liu et al. [2018], a strict form of invariance might or might not be beneficial for certain tasks. For example, in MNIST classification, a horizontal stroke near the top of the digit indicates a '7', while the same stroke near the bottom indicates a '2', as shown in fig. 1. The construction of eq. (1) will apply the same $g(\cdot)$ to each patch in the image, which is undesirable if we wish to distinguish between the two classes by summing $g(\cdot)$'s output. This also means that it is possible to conceive a complete rearrangement of the image, which appears very different to a human, but is indistinguishable from the original to the convolutional kernel.

Van der Wilk et al. [2017] circumvented the translation invariance problem of the Conv-GP by effectively adding a second, linear layer. By the introduction of weights it is possible to rescale the contribution of each patch, turning the uniform sum of eq. (1) into a weighted sum $f(\mathbf{x}) = \sum_p w_p\, g(\mathbf{x}^{[p]})$. This is a rudimentary approach which might be both too flexible, (in that it allows wildly varying weights for neighbouring pixels) and not flexible enough, (in that an image evaluation will always be a linear combination of evaluations of $g(\cdot)$ at the input patches).

We illustrate the problem of the original Conv-GP being too constrained in fig. 2. We trained a model to classify MNIST 2 vs 7 only, and display the deviations from the mean of samples from the posterior of the patch response function $g(\cdot)$. On the left (a) we show posterior samples for the original Conv-GP; we show on the right (b) samples from our TICK-GP. Note that all samples in (a) and (b) are plotted using the same colour range. We immediately notice that the samples in (a) are less vibrant than in (b), indicating the smaller variance of the Conv-GP. The small variance is the result of the Conv-GP being too constrained, which leads to a collapsed posterior that cannot to accommodate for patches that can be both positive and negative (i.e. those that belong to both classes). We also notice that all background pixels within an image have the exact same value. We discuss the behaviour of the TICK-GP samples in the next section.

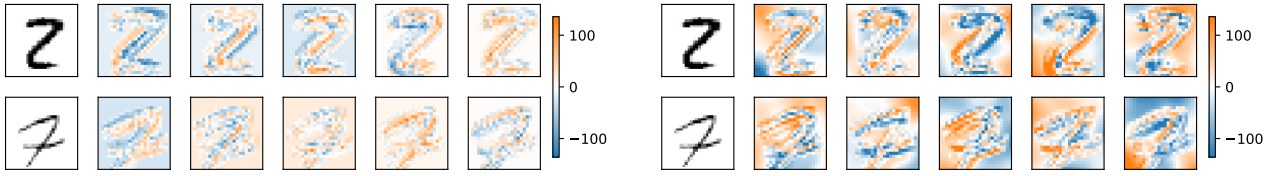
3.2 Translation Insensitive Convolutional Kernel (TICK)

A better modelling assumption would be to relax the "same patch, same output" constraint and have a patch response function $g(\cdot)$ that can vary its output depending on both the patch input and the patch location. We call this property translation *in*sensitivity, and propose a product kernel between the patches and their locations:

$$k_g\big((\mathbf{x}^{[p]}, p), (\mathbf{x}^{[p']}, p')\big) = k_{\text{patch}}\Big(\mathbf{x}^{[p]}, \mathbf{x}^{[p']}\Big) \times k_{\text{loc}}(\ell(p), \ell(p')), \quad (2)$$

where $\ell(p)$ returns the location of the upper-left corner of the patch in the image, and k_{patch} and k_{loc} are the kernels we use over the patches and patch locations, respectively. We refer to this kernel as the Translation Insensitive Convolutional Kernel (TICK). The term "insensitive" was used by Van der Wilk et al. [2018] as a relaxation of invariance. We use the term to indicate that the output is slightly sensitive to translations.

Similar approaches have been suggested in the CNN literature [Ghafoorian et al., 2017], but have not been adopted in popular, recent architectures (e.g. Inception [Szegedy et al., 2017] and DenseNet [Huang et al., 2017]). A explanation for this is that this parametric approach in neural nets adds a lot more parameters, in the order of $\mathcal{O}(w\, h\, c_{\text{in}}\, c_{\text{out}})$, leading to models that are prone to overfit in the absence of large datasets.



(a) Conv-GP (b) TICK-GP

Figure 2: We show five samples from the patch response function $g(\cdot)$ after training on MNIST 2 vs 7. The two black-and-white images (left) are the inputs. They were incorrectly classified by the Conv-GP (a), but correctly classified by the TICK-GP (b). The samples show that the posterior of the Conv-GP is overconstrained, noticeable by the paler colours and the even background (see text).

In TICK we introduce a single hyperparameter, the lengthscale of k_{loc}, to control only the degree of insensitivity (i.e. the degree to which the output of $g(\cdot)$ depends on the location of the input patch). We will learn this lengthscale and other hyperparameters automatically, using the marginal likelihood. We use [Adler et al., 1981, Theorem 4.1.1.] to get an intuition in how this parameter effects $g(\cdot)$ for the same patch input depending on its location. If we assume N_u to be the number of times a GP-draw from a stationary kernel k crosses the level u in the unit interval, then

$$\mathbb{E}_{g(\cdot)}[N_u] = \frac{1}{2\pi}\sqrt{\frac{-k''(0)}{k(0)}}\exp\left(\frac{-u^2}{2\ k(0)}\right).$$

A Squared Exponential (SE) kernel for $k_{\text{loc}}(r) = \sigma^2\exp(-r^2/\ell^2)$ gives an expected number of zero-crossings $\mathbb{E}[N_0] = (\pi\ell)^{-1}$.

You can observe that property most easily in fig. 2 (b), where the lengthscale of the SE in the trained TICK-GP approximated $(\pi/2)^{-1}$, corresponding to ≈ 2 zero-crossings in the image. Inspecting the identical background patches away from the digit, we see that $g(\cdot)$ varies smoothly, and changes sign (i.e. predicts a different class) depending on where background patches are appearing. The mapping of similar patches also varies smoothly across the stroke: the response of horizontal and vertical lines in the image gives only locally similar responses. We also notice that the samples from the TICK-GP have much larger deviations from the mean, showing that the patch-response function is less constrained and can represent epistemic uncertainty for observing certain patches at certain locations.

3.3 Deep Convolutional Gaussian Processes

With the ideas of improved convolutional kernels and deep Gaussian processes in place, it is straightforward to conceive of a model that does both: a deep GP with convolutional kernels at each layer. To do this we need to make these convolutional layers map from images to images, which we do using a multi-output kernel.

We propose a reformulation to the convolutional kernel of eq. (1): instead of summing over the patches, we apply $g(\cdot)$ to all patches in the input image. As a result, we obtain a vector-valued function $f : \mathbb{R}^D \to \mathbb{R}^P$ defined as

$$\mathbf{f}(\mathbf{x}) = \{f_p(\mathbf{x})\}_{p=1}^{P} = \left\{g(\mathbf{x}^{[p]})\right\}_{p=1}^{P}, \tag{3}$$

where $f_p(\cdot)$ indicates the p^{th} output of $f(\cdot)$. Because the *same* $g(\cdot)$ is applied to the different patches, there will be correlations between outputs. For this reason, we consider the mapping $\mathbf{f}(\cdot)$ a multi-output GP (MOGP), and name it the Multi-Output Convolutional Kernel (MOCK). Multi-output GPs [Alvarez et al., 2012] can be characterised by their covariance between the different outputs f_p and f_q of different inputs $\mathbf{x}$ and $\mathbf{x}'$, giving in our case

$$\text{Cov}[f_p(\mathbf{x}), f_q(\mathbf{x}')] = k_g\left(\mathbf{x}^{[p]}, \mathbf{x}'^{[q]}\right). \tag{4}$$

In this setting, if we are dealing with N images of P patches, the corresponding covariance matrix has a size of $N\times N\times P\times P$, which makes its calculation and inversion infeasible for most datasets.

Efficient inference for MOGPs relies strongly on choosing useful inducing variables. We developed a framework for generic MOGPs that allows for the flexible specification of both multi-output priors and inducing variables. This means that we can take computational advantage of independence properties of the prior. Given our framework, which puts the right mathematical and software abstractions in place, the implementation of a complex MOGP, such as a DCGP, is not much more difficult than that of a single-output GP [van der Wilk et al., 2020].

4 VARIATIONAL INFERENCE WITH SPARSE GAUSSIAN PROCESSES

Consider a training dataset $\{(\mathbf{x}_n, y_n)\}_{n=1}^N \subset \mathbb{R}^D \times \mathbb{R}$, consisting of N images $\mathbf{x}_n \in \mathbb{R}^D$ and class labels y_n. We set up a deep convolutional GP as $f(\cdot) := f_L(\dots f_2(f_1(\cdot)))$, where each

$$f_\ell(\cdot) \sim \mathcal{GP}\big(0, k_\ell(\cdot,\cdot)\big) \text{ and } y_n \,|\, f, \mathbf{x}_n \sim p\big(y_n \,|\, f(\mathbf{x}_n)\big).$$

We refer to the latent function-evaluation of a hidden GP as $\mathbf{h}_{n,\ell} = f_\ell(\mathbf{h}_{n,\ell-1})$ and, for convenience, we define $\mathbf{h}_{n,0} := \mathbf{x}_n$. We assume that each function is a MOGP with P_ℓ (correlated) outputs.

Given this setup, we are interested in both the posterior $p(f(\cdot)\,|\,\mathbf{y})$ for making subsequent predictions and the marginal likelihood (evidence) $p(\mathbf{y})$ to optimise the model's hyperparameters. Exact inference is not possible in this setting given our non-conjugate likelihood $p(y_n \,|\, \mathbf{h}_{n,L})$ and the $\mathcal{O}(N^3)$ cost of operations on covariance matrices, limiting the size of the datasets.

We use sparse variational GPs to address these issues, following Titsias [2009], Hensman et al. [2013], and Matthews et al. [2016]. The framework conditions the prior on *inducing variables* $\mathbf{u}_\ell$, and then specifies a free Gaussian density $q(\mathbf{u}_\ell) = \mathcal{N}(\mathbf{m}_\ell, \mathbf{S}_\ell)$. This gives the approximation $q(f_\ell(\cdot)) = \int p(f_\ell(\cdot)\,|\,\mathbf{u}_\ell)\, q(\mathbf{u}_\ell)\, \mathrm{d}\mathbf{u}_\ell$ for each layer. The original framework chose the inducing outputs $\mathbf{u}_\ell$ to be observations of the GP to some inducing inputs $\mathbf{Z}_\ell = \{\mathbf{z}_{\ell,m}\}_{m=1}^M$, i.e. $\mathbf{u}_\ell = f_\ell(\mathbf{Z}_\ell)$. Even though we are representing the GP at a finite set of points, the posterior is still a full-rank GP. It predicts using an infinite number of basis functions thanks to the use of the prior conditional. The overall approximate posterior has the form $q(f_\ell(\cdot)) = \mathcal{GP}(\mu_\ell(\cdot), \nu_\ell(\cdot))$ with

$$\begin{aligned}\mu_\ell(\cdot) &= \mathbf{k}_{\mathbf{u}_\ell}^\top(\cdot)\mathbf{K}_{\mathbf{u}_\ell\mathbf{u}_\ell}^{-1}\mathbf{m}_\ell \qquad (5)\\ \nu_\ell(\cdot) &= k_\ell(\cdot,\cdot) + \mathbf{k}_{\mathbf{u}_\ell}^\top(\cdot)\mathbf{K}_{\mathbf{u}\mathbf{u}_\ell}^{-1}(\mathbf{S}_\ell - \mathbf{K}_{\mathbf{u}\mathbf{u}_\ell})\mathbf{K}_{\mathbf{u}\mathbf{u}_\ell}^{-1}\mathbf{k}_{\mathbf{u}_\ell}(\cdot),\end{aligned}$$

where $\mathbf{m}_\ell \in \mathbb{R}^M$, and $\mathbf{S}_\ell \in \mathbb{R}^{M\times M}$ are variational parameters to be learned by optimisation. When we predict for a single point, the size of $\mathbf{k}_{\mathbf{u}_\ell}(\cdot)$ is $P_\ell \times M$ (that is, the number of outputs by the number of inducing variables), while $k_\ell(\cdot,\cdot)$ returns the $P_\ell \times P_\ell$ covariance matrix for all outputs. Crucially, because we are dealing with MOGPs, our posterior mean $\mu_\ell(\cdot)$ has size $\mathbb{R}^{P_\ell}$ and $\nu_\ell(\cdot) \in \mathbb{R}^{P_\ell \times P_\ell}$ grows quadratically in the number of outputs, roughly corresponding to the number of input pixels.

Following the standard variational [Hensman et al., 2013, Hoffman et al., 2013] approach, we construct a lower bound to the marginal likelihood (known as the Evidence Lower BOund, or ELBO) which we then optimise to find the optimal approximate posterior and the model's hyperparameters. To derive the ELBO, we start with the joint density for the generative model

$$p(\{y_n\}_n, \{\mathbf{h}_{n,\ell}\}_{n,\ell}, \{f_\ell(\cdot)\}_\ell) = \prod_n p(y_n \,|\, \mathbf{h}_{n,L}) \prod_\ell p(\mathbf{h}_{n,\ell} \,|\, \mathbf{h}_{n,\ell-1}, f_\ell(\cdot))\; p(f_\ell(\cdot)),$$

and an approximate variational posterior $q(\{\mathbf{h}_{n,\ell}\}_{n,\ell}, \{f_\ell(\cdot)\}_\ell)$ which we give the form $\prod_{n=1}^N \prod_{\ell=1}^L p(\mathbf{h}_{n,\ell} \,|\, \mathbf{h}_{n,\ell-1}, f_\ell(\cdot))\, q(f_\ell(\cdot))$. The repetition of $p(\mathbf{h}_{n,\ell} \,|\, \mathbf{h}_{n,\ell-1}, f_\ell(\cdot))$ in both the prior and posterior leads to their cancellation in the lower bound

$$\log p(\mathbf{y}) \geq \sum\nolimits_n \mathbb{E}_{q(\mathbf{h}_{n,L})}[\log p(y_n \,|\, \mathbf{h}_{n,L})] - \sum\nolimits_\ell \mathrm{KL}[q(\mathbf{u}_\ell)||p(\mathbf{u}_\ell)]. \qquad (6)$$

The form of $p(\mathbf{h}_{n,\ell} \,|\, \mathbf{h}_{n,\ell-1}, f_\ell(\cdot))$ leads to different DGP models. Damianou and Lawrence [2013] used a Gaussian distribution $\mathcal{N}(\mathbf{h}_{n,\ell} \,|\, f_\ell(\mathbf{h}_{n,\ell-1}), \sigma_\ell^2)$, which requires an additional approximate posterior over the $\mathbf{h}_\ell$'s in the bound. We follow Salimbeni and Deisenroth [2017] and use a deterministic map between $\mathbf{h}_{n,\ell}$ and $\mathbf{h}_{n,\ell-1}$ given $f_\ell(\cdot)$, corresponding to $\delta\{\mathbf{h}_{n,\ell} = f_\ell(\mathbf{h}_{n,\ell-1})\}$.

We can obtain an unbiased estimate of eq. (6) by only considering a random subset of the training data to cheaply estimate the first term and by rescaling the KL term appropriately. The expectation over $q(\{\mathbf{h}_{n,\ell}\}_{n,\ell}, \{f_\ell(\cdot)\}_\ell)$ is evaluated using Monte-Carlo, see Salimbeni and Deisenroth [2017] for details.

4.1 Computational Complexity of Dealing with Correlated Convolutional Layers

To evaluate the expectation in the ELBO as described above, we need to generate samples of $q(f_\ell(\cdot))$ with the covariance $\nu_\ell(\mathbf{h}_{n,\ell-1})$. This requires taking a Cholesky of this covariance, of which we have one for each datapoint in the minibatch. This presents a significant computational problem, because its size is $P_\ell \times P_\ell$, with P_ℓ being roughly the same as the number of patches in the input image. Compared to a non-convolutional deep GP, where we only need a single Cholesky for each layer of $\mathbf{K}_{\mathbf{u}_\ell\mathbf{u}_\ell}$, this adds a large computational cost. The deep convolutional GP model of Blomqvist et al. [2019] suffers from this problem as well. Their method avoids this computational cost by simply sampling from the P_ℓ marginals, ignoring the between-patch correlation. In the supplementary material (see fig. 10) we study the difference between both approaches and find that the lower computation cost, $\mathcal{O}(P_\ell)$, of sampling from the marginals drastically improves the number of

iterations per second and is worth the minor reduction in performance. The bias introduced to the gradient of the ELBO appears to have little effect.

4.2 Inter-Domain Inducing Patches

So far, we have set up the optimisation objective (eq. (6)) and defined the approximate posterior GP for each layer $q(f_\ell(\cdot))$. The final issues we need to address are (1) the impractically large double sums over all patches for computing entries of the $\mathbf{K}_{\mathbf{u}_\ell\mathbf{u}_\ell}$ and (2) the organisational complexity of dealing with inducing variables multi-output $\mathbf{u}_\ell$ in MOGP.

Using inter-domain inducing variables Lázaro-Gredilla and Figueiras-Vidal, 2009 solves the mathematical, organisational, and software problems of both issues. We follow Van der Wilk et al. 2017 to define for each layer $\mathbf{u}_\ell$ as evaluations of the patch response function $g_\ell(\cdot)$, and we place the inducing inputs in the patch space $\mathbb{R}^{wh}$, rather than image space $\mathbb{R}^{P_{\ell-1}}$. The GP inter-domain and multi-output software framework, which we codeveloped with this work, enables us to implement this in an efficient and modular way (van der Wilk et al., 2020).

To apply this approximation in (5) and implement this in the framework, we need to find $\mathbf{k}_{\mathbf{u}_\ell}(\cdot)$ and $\mathbf{K}_{\mathbf{u}_\ell\mathbf{u}_\ell}$:

$$\mathbf{k}_{\mathbf{u}_\ell}(\mathbf{h}_{n,\ell-1}) = \mathbb{E}[g_\ell(\mathbf{Z}_\ell) f_\ell(\mathbf{h}_{n,\ell-1})] = \left[k_{g_\ell}(\mathbf{Z}_\ell, \mathbf{h}_{n,\ell-1}^{[p]})\right]_{p=1}^{P_\ell}$$

$$\mathbf{K}_{\mathbf{u}_\ell\mathbf{u}_\ell} = k_g(\mathbf{Z}_\ell, \mathbf{Z}_\ell).$$

Choosing the inducing variables in this way greatly reduces the computational cost of the method, because we now require covariances only between the patches of the input image and the inducing patches. More precisely, $\mathbf{K}_{\mathbf{u}_\ell\mathbf{u}_\ell}$ and $\mathbf{k}_{\mathbf{u}_\ell}(\mathbf{h}_{n,\ell-1})$ become $M \times M$ and $M \times P_\ell$ sized tensors.

5 EXPERIMENTS

We present results using our TICK in deep and shallow GP models. First, we show that TICK-GP improves over Conv-GP and achieves the highest reported classification result for GPs on standard classification tasks in terms of accuracy and calibration. Secondly, we compare our method with Bayesian CNNs and find that TICK-GP's uncertainty estimates are superior, and that the ELBO can be used for model selection and automated training. We show that the CNN is confidently wrong on some ambiguous cases, while TICK-GP provides calibrated uncertainty. In Appendix D, we demonstrate that this effect is even more pronounced in a transfer learning task. In section 5.3, we also show the benefits of translation insensitivity in deep GPs.

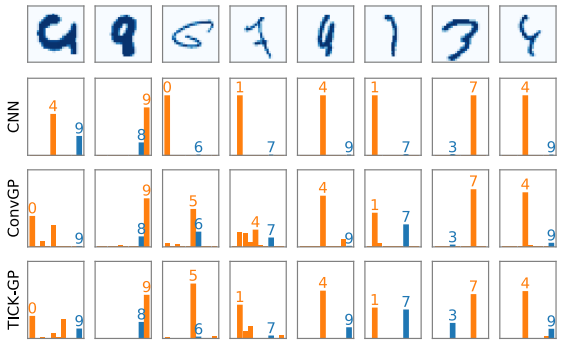


Figure 3: Posterior prediction probabilities for a (random) subset of *misclassified* images (top row) from MNIST. The barplot shows the probabilities for each of the classes, 0 to 9. The largest (orange) bar is the model's prediction. The blue bar is the true class label. The CNN predicts the wrong classes with high certainty, while the GP quantifies uncertainty better.

5.1 Comparison to Conv-GP and Bayesian Neural Nets on Image Classification

We evaluate TICK-GP on three image benchmarks (MNIST, FASHION-MNIST, and grey-scale CIFAR-10) and compare its performance to a SE-GP, Conv-GP van der Wilk et al., 2017 and a Bayesian CNN (B-CNN) based on dropout Gal and Ghahramani, 2016. All GP models in this experiment are single-layered and trained following the method outlined in section 4. Their exact setup (kernel, number of inducing points, learning rate schedule, etc.) is detailed in Appendix A. We compare the GP models to a Bayesian CNN architecture, with two convolutional layers followed by two full dense layers. We use dropout at train and test time, following Gal and Ghahramani, 2016, with 50% keep-probability, which we found by running a grid search and selecting the best model based on its NLL on a 10% validation set. We further detail the CNN configuration in Appendix B. The SE-GP is a vanilla Sparse Variational GP (SVGP) Hensman et al., 2013 using a SE kernel.

Table 1 reports the top-k error rate and the Negative Log-Likelihood (NLL). We use NLL as our main metric for calibration because it is a proper scoring rule Gneiting and Raftery, 2007 and has a useful relationship to returns obtained from bets on the future based on the predicted belief Roulston and Smith, 2002. The top-k error rate is the percentage of test images for whom the true class label is not within the highest k predictive probabilities. We see that TICK-GP outperforms previous GP models and dropout-based CNNs in terms of NLL, both on the complete test set and on the misclassified images. The single-layer TICK-GP sets the new records of classification with GP models on the listed

Table 1: Results of classification experiments with Bayesian CNN (B-CNN) and shallow GP models. We report the top-k error rate, and Negative Log-Likelihood (NLL) on the full test set and on the misclassified images of the test set. The TICK-GP outperforms the Conv-GP on every dataset in both accuracy and NLL, illustrating the clear benefits of translation insensitivity. The single-layer TICK-GP models get a similar accuracy to the CNNs but have more calibrated predictive probabilities (lower NLL).

	MNIST				FASHION-MNIST				GREY CIFAR-10			
metric	SE	Conv	TICK	B-CNN	SE	Conv	TICK	B-CNN	SE	Conv	TICK	B-CNN
top-1 error (%)	2.31	1.70	**0.83**	0.91	12.15	11.06	10.01	**8.31**	58.24	41.65	37.82	**37.44**
top-2 error (%)	0.69	0.49	**0.11**	0.22	3.67	3.18	2.69	**2.17**	38.91	24.09	**20.52**	21.48
top-3 error (%)	0.35	0.19	**0.05**	**0.05**	1.21	1.11	0.92	**0.75**	27.18	14.93	**12.21**	12.91
NLL full (×10)	0.60	0.57	**0.29**	0.35	2.68	2.52	**2.28**	2.45	15.56	11.68	**10.56**	11.16
NLL misses	1.86	1.97	**1.70**	2.58	1.90	1.90	**1.89**	2.10	2.20	2.12	**2.10**	2.23

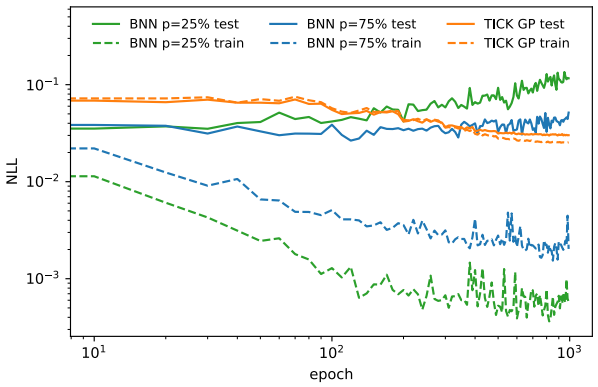


Figure 4: B-CNN marginal likelihood estimates are not usable for hyperparameter selection. The plot shows the train and test NLL of two B-CNNs w.r.t. epochs on MNIST. We notice that both models overfit, shown by the gap between train and test performance, and the deteriorating test performance. A higher dropout rate ($p = 75\%$ vs. 25%) postpones this effect, but does not prevent it. For the TICK-GP model we see that the train NLL is a good proxy for the test NLL, and that it outperforms the B-CNNs over time.

datasets, showing the importance of encoding the right inductive biases into a GP model. Most importantly, the model is comparable with the B-CNN in terms of error rate, but has better-calibrated predictive probabilities, which enables ELBO-based model selection, as shown in the next section.

Figure 3 shows the predictive probability for a few randomly selected misclassified images, demonstrating both the better-calibrated probabilities of GP-based models compared to the CNN models, and the improvements of the new TICK-GP over the Conv-GP. We clearly see how the CNN can be very confidently wrong. In Appendix C we show the complete set of misclassified images.

5.2 Comparison to Bayesian Neural Networks with different Dropout Rates

While Bayesian deep learning methods based on dropout (Gal and Ghahramani, 2016) have been empirically successful in improving the quality of uncertainty estimates, it is unclear to what extent they accurately approximate the true posteriors (Hron et al., 2018). Additionally, in this experiment we find that they do not provide a Bayesian objective that allows for the automated training of hyperparameters and model selection (Rasmussen and Ghahramani, 2001). In this experiment we have not considered other Bayesian deep learning approaches like (Lee et al., 2018; Osawa et al., 2019).

Figure 4 shows how powerful the marginal likelihood (ELBO) of our fully Bayesian model is. In the plot, we show the training traces of a TICK-GP and two B-CNNs with different dropout rates on MNIST. The models have the same setup as in section 5.1. We notice the gap between the train and test NLL of the B-CNNs, and a train NLL which keeps decreasing while the NLL of the test set starts to increase. Using larger dropout rates ($p_{\text{dropout}} = 0.75$ instead of 0.25) postpones this effect but does not prevent it. By contrast, the proper marginal likelihood objective of the TICK-GP model is reflected by the close similarity of the test NLL and the train NLL. This enables us to do automated model selection through higher marginal likelihood and ELBO-based hyperparameter learning (e.g. to learn the degree of translation insensitivity of a convolutional layer).

5.3 Translation Insensitivity in Deep Convolutional GPs (DCGPs)

Having shown how TICK-GP compares to vanilla Conv-GPs, we now consider deep architectures. In table 2 we list the performance of a deep Conv-GP (DCGP) (Blomqvist et al., 2019) and a deep TICK-GP (ours)

Table 2: DCGP [Blomqvist et al., 2019] (reproduced with our code) and Deep TICK-GP (our method) on MNIST and CIFAR-10.

		MNIST		CIFAR-10	
depth	metric	Conv	TICK	Conv	TICK
1	top-1 error (%)	1.87	**1.19**	41.06	**37.10**
	NLL full	0.06	**0.04**	1.17	**1.08**
	neg. ELBO ($\times 10^3$)	8.29	**5.83**	65.72	**63.51**
2	top-1 error (%)	0.96	**0.67**	28.60	**25.59**
	NLL full	0.04	**0.02**	0.84	**0.75**
	neg. ELBO ($\times 10^3$)	5.37	**4.25**	52.81	**48.31**
3	top-1 error (%)	0.93	**0.64**	25.33	**23.83**
	NLL full	0.03	**0.02**	0.74	**0.69**
	neg. ELBO ($\times 10^3$)	5.045	**4.19**	49.38	**47.53**

on MNIST and CIFAR-10. We configure the models as identically as possible: each layer uses 384 inducing 5x5 patches (initialised using random patches from the training images), an identity Conv2D mean function for the hidden layers, and a SE kernel for the patch response function (the complete setup can be found in Appendix A).

The deep TICK-GP, which can learn the degree of insensitivity, outperforms the plain DCGP in terms of accuracy and NLL for any depth. We see that both models improve with depth, and more importantly, that the ELBO is reflecting this. This can also be observed in the appendix (fig. 8), where deep TICK-GPs are consistently outperforming deep convolutional GPs. We also compare to a growing dropout-based B-CNN, which gives for the top-1 error and NLL: 1.93%, 0.07 (1 layer), 1.04%, 0.03 (2 layers) 0.86%, 0.04 (3 layers) on MNIST. As expected, the B-CNN's accuracy improves with depth, but the NLL (uncertainty quantification) gets worse. This is in contrast with our model which continues to improve NLL and accuracy with depth.

To further position the TICK-GP, we compare its performance against non-convolutional deep GPs. On MNIST we found that a deep GP [Salimbeni and Deisenroth, 2017] with SE kernels, 2 layers, and 384 inducing inputs per layer managed 98% accuracy. This is equal to a vanilla GP classifier ([Hensman et al., 2015] report 98% accuracy), illustrating that depth by itself does not always improve performance when the wrong inductive biases are encoded in the model. [Havasi et al., 2018] report similar conclusions in their work; their HMC approach delivers 98.0% accuracy. Our model beats all of these methods with 99.33% accuracy. Both non-convolutional deep GP papers do not report results for CIFAR. We ran the method of [Salimbeni and Deisenroth, 2017] and managed to get 47.26% accuracy. Our model outperforms this with 74.41% accuracy.

5.4 Implementation and Reproducibility

The main implementation difficulty for multi-output GPs is dealing with a large amount of special cases to ensure the most efficient code path is used. This makes it a challenge to implement modular and reusable code; the correct software abstractions should be used to keep the code readable, manageable, and extendable. We noticed, however, that most multi-output GPs [Alvarez et al., 2012] can be reformulated in terms of single-output inducing outputs, leaning towards inter-domain approximations. Based on this observation we developed a general multi-output GP framework [van der Wilk et al., 2020].

To implement our model in the framework we need to provide components specific to our model. In particular, we need to implement the multi-output and single-output convolutional kernels (eq. (1) and eq. (3)) and the corresponding single and multi-output approximate posterior GP $q(f(\cdot))$ (different flavours of eq. (5)). Finally, we also need to implement the bound in eq. (6).

6 CONCLUDING DISCUSSION

We have shown that the accuracy of Bayesian methods, and the quality of their posterior uncertainties, depends strongly on the suitability of the modelling assumptions made in the prior, and that Bayesian inference by itself is often not enough. This motivated us to develop the Translation Insensitive Convolutional Kernel (TICK), which leads to improved uncertainty estimates and accuracy on a range of different problems, and sets the new state-of-art results for GP models.

While we appreciate that our experiments are still on rudimentary image datasets (e.g. not of the calibre of ImageNet), they do show that when the accuracy of our method is on a par with that of a neural network, we outperform the neural network in terms of uncertainty estimation (section 5.1) and in the use of marginal likelihood approximations for hyperparameter learning (section 5.2). We believe that this suggests that the full benefits of the Bayesian framework are not currently realised by Bayesian deep learning.

We further presented deep convolutional GPs in a new and clear way: image-to-image layers modelled as correlated multi-output GPs (section 3.3). This enabled efficient implementation in our general-purpose open-sourced framework [van der Wilk et al., 2020]. We also highlighted a computational limitation of current convolutional layers (section 4.1) which had not been addressed in earlier work, and which future work should focus on.

Acknowledgements

We have greatly appreciated valuable discussions with Marc Deisenroth and Zhe Dong in the preparation of this work. We would like to thank Fergus Simpson, Hugh Salimbeni, ST John, Victor Picheny, and anonymous reviewers for helpful feedback on the manuscript.

References

Robert J. Adler, D. Firmin, and David George Kendall. A non-Gaussian model for random surfaces. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 303 (1479):433–462, 1981.

Mauricio A. Alvarez, Lorenzo Rosasco, and Neil D. Lawrence. Kernels for vector-valued functions: A review. *Foundations and Trends in Machine Learning*, 2012.

Kenneth Blomqvist, Samuel Kaski, and Markus Heinonen. Deep convolutional Gaussian Processes. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2019.

Andreas Damianou and Neil D. Lawrence. Deep Gaussian Processes. In *Artificial Intelligence and Statistics*, 2013.

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

Mohsen Ghafoorian, Nico Karssemeijer, Tom Heskes, Inge W. M. van Uden, Clara I Sanchez, Geert Litjens, Frank-Erik de Leeuw, Bram van Ginneken, Elena Marchiori, and Bram Platel. Location sensitive deep convolutional neural networks for segmentation of white matter hyperintensities. *Scientific Reports*, 7 (1):5110, 2017.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102 (477):359–378, 2007.

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 José Murillo-Fuentes. Inference In deep Gaussian Processes using stochastic gradient Hamiltonian Monte Carlo. In *Advances in Neural Information Processing Systems*, pages 7506–7516, 2018.

James Hensman, Nicolo Fusi, and Neil D. Lawrence. Gaussian Processes for Big Data. *Uncertainty in Artificial Intelligence*, 2013.

James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani. Scalable variational Gaussian Process Classification. In *Artificial Intelligence and Statistics*, 2015.

Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *Journal of Machine Learning Research*, 2013.

Jiri Hron, Alexander G. de G. Matthews, and Zoubin Ghahramani. Variational Bayesian dropout: pitfalls and fixes. In *International Conference on Machine Learning*, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *IEEE conference on computer vision and pattern recognition*, 2017.

Keras. Keras implementation of CNN for MNIST. Available from `https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.

Miguel Lázaro-Gredilla and Aníbal Figueiras-Vidal. Inter-domain Gaussian Processes for sparse inference using inducing features. In *Neural Information Processing Systems*, 2009.

Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as Gaussian Processes. In *International Conference on Learning Representations*, 2018.

Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Neural Information Processing Systems*, 2018.

Alexander G. de G. Matthews, James Hensman, Turner Richard, and Zoubin Ghahramani. On Sparse Variational Methods and the Kullback-Leibler Divergence between Stochastic Processes. *Artificial Intelligence and Statistics*, 2016.

Radford M. Neal. *Bayesian learning for neural networks*, volume 118. Springer, 1996.

Kazuki Osawa, Siddharth Swaroop, Mohammad Emtiyaz E Khan, Anirudh Jain, Runa Eschenhagen, Richard E Turner, and Rio Yokota. Practical deep learning with bayesian principles. In *Advances in*

Neural Information Processing Systems 32, pages 4287–4299. Curran Associates, Inc., 2019.

Carl E. Rasmussen and Zoubin Ghahramani. Occam's Razor. In *Neural Information Processing Systems*. 2001.

Carl E. Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Mark S. Roulston and Leonard A. Smith. Evaluating probabilistic forecasts using information theory. *Monthly Weather Review*, 130(6):1653–1660, 2002.

Hugh Salimbeni and Marc P. Deisenroth. Doubly Stochastic Variational Inference for Deep Gaussian Processes. *Neural Information Processing Systems*, 2017.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI Conference on Artificial Intelligence*, 2017.

Michalis Titsias. Variational Learning of Inducing Variables in Sparse Gaussian Processes. *Artificial Intelligence and Statistics*, 2009.

UCI. Semeion handwritten digit data set. Available from `https://archive.ics.uci.edu/ml/datasets/semeion+handwritten+digit`.

Mark van der Wilk, Carl E. Rasmussen, and James Hensman. Convolutional Gaussian Processes. In *Neural Information Processing Systems*, 2017.

Mark van der Wilk, Matthias Bauer, ST John, and James Hensman. Learning Invariances using the Marginal Likelihood. In *Neural Information Processing Systems*, 2018.

Mark van der Wilk, Vincent Dutordoir, ST John, Artem Artemev, Vincent Adam, and James Hensman. A Framework for Interdomain and Multioutput Gaussian Processes. *arXiv preprint*, 2020.