Development of New Machine Learning Models Based on Gaussian Processes. Applications to Remote Sensing and Astrophysics.

Presented to obtain the degree of

Doctor of Philosophy

by

Pablo Morales-Álvarez

supervised by

Rafael Molina Soriano and Aggelos K. Katsaggelos

UNIVERSIDAD DE GRANADA

Programa de Doctorado en Física y Matemáticas

Editor: Universidad de Granada. Tesis Doctorales
Autor: Pablo Morales Álvarez
ISBN: 978-84-1306-666-0
URI: http://hdl.handle.net/10481/63966

Nos convencemos a nosotros mismos de que la vida será mejor después de casarnos, después de tener un hijo y entonces después de tener otro. Entonces nos sentimos frustrados porque los hijos no son lo suficientemente grandes y pensamos que seremos más felices cuando lo sean. Después de eso nos frustramos porque son adolescentes (difíciles de tratar). Ciertamente seremos más felices cuando salgan de esta etapa. Nos decimos que nuestra vida estará completa cuando a nuestro esposo/a le vaya mejor, cuando tengamos un mejor carro o una mejor casa, cuando nos podamos ir de vacaciones, cuando estemos retirados.

La verdad es que no hay mejor momento para ser felices que ahora. Si no es ahora, ¿cuándo? Tu vida estará siempre llena de retos. Es mejor admitirlo y decidir ser felices de todas formas. Una de mis frases: "Por largo tiempo me parecía que la vida estaba a punto de comenzar. La vida de verdad. Pero siempre había algún obstáculo en el camino, algo que resolver primero, algún asunto sin terminar, tiempo por pasar, una deuda que pagar. Sólo entonces la vida comenzaría. Hasta que me di cuenta que esos obstáculos eran mi vida". Esta perspectiva me ha ayudado a ver que no hay un camino a la felicidad.

La felicidad "es" el camino; así que atesora cada momento que tienes y atesóralo más cuando lo compartiste con alguien especial, lo suficientemente especial para compartir tu tiempo y recuerda que el tiempo no espera por nadie... así que deja de esperar hasta que bajes cinco kilos, hasta que te cases, hasta que te divorcies, hasta el viernes por la noche, hasta el domingo por la mañana, hasta la primavera, el verano, el otoño o el invierno o hasta que te mueras, para decidir que no hay mejor momento que éste para ser feliz... la felicidad es un trayecto, no un destino.

Eduardo Galeano (1940-2015)

AGRADECIMIENTOS (ACKNOWLEDGEMENTS)

Quiero comenzar estas líneas dando las gracias a mi familia. Hay muchos factores que contribuyen al desarrollo de una persona, pero sin duda el ambiente en el hogar es un aspecto fundamental. Sabéis de sobra lo que significáis para mí. Gracias por estar ahí siempre, en los buenos y en los malos momentos.

También quiero expresar mi gratitud hacia mi director de tesis, Rafael Molina. Nuestras discusiones científicas y técnicas han motivado buena parte de los resultados aquí presentes. Pero, además, nuestras discusiones vitales, emocionales y filosóficas también forman parte de mí. Gracias por tu tiempo. Espero que la defensa de esta tesis doctoral sea solo un punto y seguido.

I would also like to thank my co-supervisor, Prof. Katsaggelos, for your generous support, your invaluable experience, and your welcome at Northwestern University. Among others, I will not forget the 2018 World Cup and the thunderstorm after the fireworks on the USA independence day.

Gracias también a todos los compañeros del grupo de investigación, a todos los colaboradores de esta tesis doctoral, y a todos los amigos que he conocido durante estos años. Y, por supuesto, gracias a los amigos de siempre.

Finalmente, gracias a la Fundación La Caixa por creer en este proyecto y brindarme la oportunidad de conocer a compañeros maravillosos en la promoción de 2017.

TABLE OF CONTENTS

CHAPTER 1

Introduction

Gaussian Processes (GPs) are non-parametric machine learning methods widely used in supervised learning Williams and Rasmussen (2006). One way to understand a (real-valued single-output) GP on a set X is as a probability distribution over the space of functions from X to $\mathbb{R}$. Specifically, a function f follows a GP distribution with mean function $\mu : X \to \mathbb{R}$ and covariance function $\kappa : X \times X \to \mathbb{R}$, which is denoted as $f \sim \mathrm{GP}(\mu, \kappa)$, if $(f(x_1), \ldots, f(x_N))$ follows a normal distribution $\mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ with $\boldsymbol{\mu} = (\mu(x_1), \ldots, \mu(x_N))$ and $\mathbf{K}$ the $N \times N$ matrix $(\kappa(x_i, x_j))_{i,j=1,\ldots,N}$ for any $x_1, \ldots, x_N$ in X. Notice that a GP covariance function must be positive semidefinite, i.e. the aforementioned $\mathbf{K}$ matrix must be positive semidefinite for every $x_1, \ldots, x_N$ in X (Bishop 2006, Chapter 6).

GPs have become popular due to their flexibility, the possibility of incorporating prior knowledge and their uncertainty quantification Williams and Rasmussen (2006). GPs have been used in many different domains, such as renewable energies Manobel et al. (2018), medicine Alaa and van der Schaar (2017), biology Swain et al. (2016) and chemistry Dai and Krems (2020). The mean and covariance functions allow for specifying the types of mappings to be modelled. For instance, in the popular RBF kernel Bishop (2006), the lengthscale parameter regulates the smoothness of the functions, whereas the amplitude parameter accounts for the range of the output value. The uncertainty estimation is also a very attractive property of GPs. Thanks to its fully Bayesian approach, GPs provide calibrated error bars for their predictions. This is essential in many high-risk real-world applications such as medicine Filos et al. (2019), Mobiny et al. (2019) and self-driving cars Kendall and Gal (2017), Gal (2016).

When used for regression tasks, GP predictions have an elegant closed-form solution. Namely, assume a (regression) training dataset $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N \subset \mathbb{R}^D \times \mathbb{R}$, a GP-based model with zero mean and covariance function κ with latent variables $\{f_n\}_{n=1}^N \subset \mathbb{R}$, and a Gaussian likelihood function $\mathrm{p}(y|f) = \mathcal{N}(y|f, \sigma^2_{\text{lik}})$. In this case, due to the conjugacy between the GP model and the Gaussian likelihood, the marginal likelihood can be obtained in closed-form. The predictive distribution for a new $\mathbf{x}_* \in \mathbb{R}^D$ is a Gaussian with mean

$$\mu_* = \kappa(\mathbf{x}_*, \mathbf{X})(\kappa(\mathbf{X}) + \sigma^2_{\text{lik}} \cdot \mathbf{I})^{-1}\mathbf{y} \tag{1.1}$$

and variance

$$\sigma^2_* = \kappa(\mathbf{x}_*) - \kappa(\mathbf{x}_*, \mathbf{X})(\kappa(\mathbf{X}) + \sigma^2_{\text{lik}} \cdot \mathbf{I})^{-1}\kappa(\mathbf{X}, \mathbf{x}_*). \tag{1.2}$$

As usual in GP literature, here we are writing $\mathbf{X}$ for the $N \times D$ matrix whose rows are given by $\{\mathbf{x}_n\}$ and $\mathbf{y}$ for the $N \times 1$ vector of $\{y_n\}$ (Williams and Rasmussen 2006, Chapter 2). These closed-form expressions have an intuitive interpretation. The predictive mean at $\mathbf{x}_*$ depends on the output values at $\mathbf{X}$ (i.e. $\mathbf{y}$), and the dependence is regulated by how related are $\mathbf{x}_*$ and $\mathbf{X}$ (which is mathematically codified within the covariance function κ). The predictive covariance at $\mathbf{x}_*$ is always less or equal to the prior covariance (notice that the substracted term is greater or equal to zero since κ is positive semidefinite), and the reduction in uncertainty is also governed by how related are $\mathbf{x}_*$ and $\mathbf{X}$ (i.e. through $\kappa(\mathbf{x}_*, \mathbf{X})$).

When used for classification, the (non-Gaussian) likelihood cannot be integrated out analytically, and therefore a closed-form expression is not available (Williams and Rasmussen 2006, Chapter 3). This has been typically addressed with local variational methods Bishop (2006), in which the likelihood function is lower bounded by an expression that can be integrated out analytically. This approach will be followed in Chapters 2-5 for the logistic likelihood, i.e. $\mathrm{p}(y|f) = \sigma(f)^y(1-\sigma(f))^{(1-y)}$, where $\sigma(f) = 1/(1+e^{-f})$ is the sigmoid/logistic function and $y \in \{0, 1\}$ (i.e. we are dealing with a binary classification problem). An alternative approach is to maintain the original non-tractable likelihood and estimate the expectations through Monte Carlo approximation. This approach will be followed in Chapters 6-8.

An important shortcoming of GPs has to do with scalability. In order to make predictions, the inverse of the $N \times N$ matrix $\kappa(\mathbf{X}) + \sigma^2_{\text{lik}} \cdot \mathbf{I}$ must be computed, recall eqs. (1.1) and (1.2). This implies a training computational complexity of $\mathcal{O}(N^3)$. In practice, this means that GPs cannot be applied to datasets with more than a few thousands instances (typically $N = 10^4$ is considered the practical limit). In order to address this problem, a plethora of methods have been proposed in the last years. In this thesis we will focus on three families of approaches.

First, Fourier features have been used to approximate stationary kernels through linear ones Rahimi and Recht (2008), Lázaro-Gredilla et al. (2010). Then, inference can be performed in the space of features, which yields a computational cost of $\mathcal{O}(ND^2)$, where $D \ll N$ is the amount of Fourier features used (the more, the better is the kernel approximation). Notice that the complexity on the training set size N is linear here (instead of cubic as in standard GPs). The main drawbacks of Fourier features based approaches are: 1) it can only be used for stationary kernels and 2) there is no rule of thumb for the choice of D (the more the better approximation, but also the less efficient). Fourier features will be used in Chapters 2, 3 and 5.

Second, inducing points have become one of the most popular approaches for sparse GPs Snelson and Ghahramani (2006), Quiñonero-Candela and Rasmussen (2005), Titsias (2009), Hensman et al. (2013, 2015), Burt et al. (2019). The idea is to summarize the information from the training data in a reduced set of $M \ll N$ inducing points. The location of these inducing points is typically estimated to maximize the marginal likelihood of the observed data. Similarly to Fourier features based approaches, the computational cost is reduced down to $\mathcal{O}(NM^2)$, i.e. linear on N. Again, the approximation becomes better for larger values of M, but the efficiency is reduced too. Therefore, the choice of M is also a tricky aspect of inducing points based methods, see e.g. Burt et al. (2019). However, in this case there is no restriction on the type of kernels that can be used. This approach will be followed in Chapters 6, 7 and 8.

Third, inference networks have been recently proposed as an alternative to inducing points Shi et al. (2019). The goal is to leverage more expressive predictive distributions that are not limited by the amount of inducing points. In order to do so, variational inference is applied in the space of functions Cheng and Boots (2016), Sun et al. (2019), and a flexible parametric variational family of predictive distributions is considered through a functional mirror descent algorithm Dai et al. (2016). In each iteration, a measurement set of size M is used to match the variational family with the real approximation at that step. The computational cost is $\mathcal{O}(M^3)$. Interestingly, the expressiveness of the predictive distribution is not limited by M, and therefore it can be typically kept smaller than in inducing point based approaches. This method will be used in Chapter 6.

A very relevant setting for this PhD thesis is that of *crowdsourcing*. The term crowdsourcing was coined in 2006 by J. Howe to describe "the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined generally large group of people in the form of an open call" Howe (2006). In our case, we focus on the task of labelling a training dataset. The proliferation of web services such as Amazon Mechanical Turk (`www.mturk.com`) and Figure-Eight (`www.figure-eight.com`, formerly Crowdflower) allows for outsourcing this process to a distributed workforce that can collaborate virtually, sharing the effort among a huge number of annotators Snow et al. (2008), Buhrmester et al. (2011). This approach is rapidly growing in popularity, and is being applied to many different fields such as medical imaging Albarqouni et al. (2016), genetics Sáez-Rodríguez et al. (2016), remote sensing Fritz et al. (2017), topic modelling Rodrigues et al. (2017), and object segmentation Heim et al. (2017). Very recently, it has been applied to study the effect of the Covid-19 pandemic in oncology patients Desai et al. (2020).

Due to the great number of potential annotators, large datasets can be labeled in a very short time. However, this approach introduces new challenging problems: combining the unknown expertise of annotators, dealing with disagreements on the annotated samples, or detecting the existence of spammer and adversarial annotators Zhang et al. (2016). GPs have proved successful to model this scenario, which requires an excellent uncertainty quantification Rodrigues et al. (2014, 2017). Yet, GPs present the aforementioned limitations of scalability and non-conjugacy for non-Gaussian likelihoods. Both drawbacks are inherited by the crowdsourcing scenario.

In this PhD thesis we have developed different machine learning models based on Gaussian Processes. Different settings (regression, classification and crowdsourcing) are considered, and various application fields (specially remote sensing and astrophysics, but also threat detection and sentiment analysis) are targeted. Three main blocks can be distinguished in this thesis:

- **Fourier features for Gaussian Process classification.** This includes Chapters 2–3. The idea here is to approximate a stationary kernel by a linear one through Bochner theorem Rudin (1962), Rahimi and Recht (2008). The algorithms are applied to different areas such as remote sensing (cloud detection) and security (detection of objects concealed under clothes).

- **Gaussian Processes for crowdsourcing.** This includes Chapters 4-7. Here we extend some of the most popular GP approaches to the crowdsourcing scenario, addressing the main limitations of GPs. For example, we use variational inference for GP based crowdsourcing, Fourier features, inducing points and inference networks. The range of applications is also extensive: astrophysics (glitch identification in the search of gravitational waves), social media (evaluation of movies and music) and healthcare (movement prediction in residential environments).

- **Activation-level uncertainty in deep neural networks.** This includes Chapter 8. Here we carry out more fundamental research in machine learning. Specifically, we propose a new approach to estimate uncertainty in Deep Neural Networks, which is referred to as activation-level uncertainty. Interestingly, the activation functions are modelled with simple 1-dimensional GPs. Compared to weight-level stochasticity, our method yields better calibrated predictions. Moreover, the connections between our approach and deep GPs are analyzed too.

Next, we provide a general overview for each chapter. The main contributions will be highlighted at the beginning of the corresponding chapter, and the main joint conclusions will be drawn at Chapter 9. Notice how the different developments share a common element: the underlying GP-based modelling.

Chapter 2: In this work we propose a first method to sparsify GPs for classification based on Fourier features. Importantly, the Fourier features are sampled from the beginning and fixed (as opposed to a second method that is proposed in the next chapter). The proposed approach is evaluated on a Passive Millimeter Wave Images dataset to detect threats concealed under clothes, a technology that is being increasingly used at airports worldwide.

Chapter 3: In this work we propose a second method to sparsify GPs for classification based on Fourier features. The main difference is that Fourier features are estimated during the training step. This allows for learning the optimal ones, although the computational cost in practice is higher. In this case, the methods are applied to cloud detection from multispectral imagery and infrared sounding data, achieving excellent empirical results.

Chapter 4: In this paper we introduce the use of local variational methods for GPs in the crowdsourcing scenario, as an alternative to the Expectation Propagation (EP) algorithm that had been proposed in Rodrigues et al. (2014). We show that our method outperforms the EP-based one in terms of accuracy and efficiency, and reliably estimates the expertise of the involved annotators. The algorithm is applied in standard crowdsourcing benchmarks related to sentiment analysis and music genre estimation.

Chapter 5: In this paper we extend the Fourier features algorithms of Chapter 3 to the GP based formulation of the crowdsourcing problem. This allows for accurate and efficient learning from crowds. In addition to standard crowdsourcing benchmarks, the proposed algorithms are applied on the Sphere (Sensor Platform for HEalthcare in Residential Environment) dataset, a recognition task coordinated by the University of Bristol to improve the services provided in UK residences.

Chapter 6: In this work we explore alternative approaches to achieve scalability beyond that proposed in Chapter 5. The main motivation is that Fourier features approaches do not allow for mini-batch training, which hampers the application to massive datasets. Specifically, here we extend the formulation of inducing points and inference networks to the crowdsourcing setting. The algorithms are successfully applied to detect glitches in the search of gravitational waves within the LIGO project (Laser Interferometer Gravitational-waves Observatory).

Chapter 7: Motivated by the challenging LIGO problem, in this work we propose a methodology to integrate expert labels within the crowdsourcing annotations. This is precisely the scenario available in the LIGO project. We show that the proposed approach naturally integrates both types of information, surpassing the results obtained when only one of them is used.

Chapter 8: This work differs from all the previous ones in the fact that GPs are not the central core of the algorithm, but are used as a component to model the activation functions in Deep Neural Networks. This allows us to introduce the so-called “activation-level uncertainty”, as opposed to the classical weight-space stochasticity. We show that our approach achieves better calibrated estimates. We also establish that our proposal requires fewer inducing points and is better suited than deep GPs for deep architectures.

CHAPTER 2

Gaussian Processes and Fourier features for threat detection

2.1 Publication details

Authors: Pablo Morales-Álvarez, Adrián Pérez-Suay, Rafael Molina, Gustau Camps-Valls, Aggelos K Katsaggelos
Title: Passive millimeter wave image classification with large scale Gaussian processes
Reference: 2017 IEEE International Conference on Image Processing (ICIP), 370-374
Status: Published
Quality indices:

- GGS Rating: A-
- GGS Class: 2
- CORE: B

2.2 Main contributions

- We introduce *RFF-GPC* (Random Fourier Features for Gaussian Processes Classification). RFF-GPC utilizes Fourier features to approximate stationary kernels with linear ones. Local variational methods are applied for inference, and the Fourier features are sampled from the beginning and fixed.
- The proposed approach is successfully evaluated on a unique, large and real database of Passive Millimeter Wave Images to detect objects concealed under clothes.

PASSIVE MILLIMETER WAVE IMAGE CLASSIFICATION WITH LARGE SCALE GAUSSIAN PROCESSES

Pablo Morales-Álvarez
Dept. of Computer Science and AI
University of Granada, Spain

Adrián Pérez-Suay
Image Processing Laboratory
University of Valencia, Spain

Rafael Molina
Dept. of Computer Science and AI
University of Granada, Spain

Gustau Camps-Valls
Image Processing Laboratory
University of Valencia, Spain

Aggelos K. Katsaggelos
Dept. of Electrical Engineering and Computer Science
Northwestern University, USA

ABSTRACT

Passive Millimeter Wave Images (PMMWIs) are being increasingly used to identify and localize objects concealed under clothing. Taking into account the quality of these images and the unknown position, shape, and size of the hidden objects, large data sets are required to build successful classification/detection systems. Kernel methods, in particular Gaussian Processes (GPs), are sound, flexible, and popular techniques to address supervised learning problems. Unfortunately, their computational cost is known to be prohibitive for large scale applications. In this work, we present a novel approach to PMMWI classification based on the use of Gaussian Processes for large data sets. The proposed methodology relies on linear approximations to kernel functions through random Fourier features. Model hyperparameters are learned within a variational Bayes inference scheme. Our proposal is well suited for real-time applications, since its computational cost at training and test times is much lower than the original GP formulation. The proposed approach is tested on a unique, large, and real PMMWI database containing a broad variety of sizes, types, and locations of hidden objects.

1 Introduction

Millimeter wave images can be used to recognize and localize hidden objects under clothing [Sheen et al., 2001]. This type of images is becoming increasingly popular in threat detection systems located at warehouses and airports (such as international ones in Los Angeles or San Francisco). In contrast to *active* scanners, which direct millimeter wave radiation to the subject and then interpret the reflected energy, *passive* systems use only ambient radiation and that emitted from the human body or objects [Yujiri et al., 2003]. This means safer and less invasive systems, which are better suited for practical applications.

Sensor modelling and image processing techniques have been used on PMMWIs. The main concepts related to millimeter images are introduced in [Yujiri et al., 2003, Yujiri, 2006]. Compressive sensing and superresolution are explored in [Babacan et al., 2011, Saafin et al., 2016]. Denoising, deconvolution, and enhancement techniques have also been applied to this kind of images [Han et al., 2010, Mateos et al., 2016, Yang et al., 2010, Yu et al., 2011].

Unfortunately, the literature on classification using these images is still scarce. K-means is used to segment PMMWIs in [Haworth et al., 2004]. Gaussian mixture models are applied in [Haworth et al., 2006] to characterize people with and without threats. In [Martínez et al., 2010] the authors apply noise elimination and image segmentation using Local Binary Fitting (LBF). A highly efficient two-step algorithm, based on denoising and mathematical morphology, was proposed in [Maqueda et al., 2015]. It achieves an acceptable detection rate on noisy and low contrast images. A comprehensive comparison between classical methods (Logistic Regression, SVM, Random Forest, Boosting) is provided in [Tapia et al., 2016]. In that work, the large size of the data set imposed constraints on the kernel used for SVM, and prevented the application of GPs.

Kernel machines [Camps-Valls and Bruzzone, 2005, Chapelle et al., 1999] are among the most popular approaches for supervised learning. Due to its solid Bayesian treatment, GP [Rasmussen and Williams, 2006] is a current state-of-the-art kernel method, which has also been used for image classification [Bazi and Melgani, 2010, Ruiz et al., 2016]. For a problem with n training instances, kernel machines store and manipulate kernel matrices of size $n \times n$, which makes them scale as $\mathcal{O}(n^3)$ in training and $\mathcal{O}(n^2)$ for each test instance. These two orders hamper their applicability to large scale problems in terms of computational limitations and impossibility of real-time prediction respectively.

In this paper we develop a new method that allows for large scale PMMWI Gaussian Processes classification. Our approach relies on linear approximations to kernel matrices based on random Fourier features (RFF) [Rahimi and Recht, 2007]. The use of GPs with RFF for regression problems has been presented in [Lázaro-Gredilla et al., 2010, Hensman et al., 2016, Damianou and Lawrence, 2014]. To deal with the non-conjugate observation model typical of classification problems, we resort here to the variational inference approach [Bishop, 2006, Section 10.6]. Computational complexity at test turns out to be independent on the training set size. Cost at training is also much lower than original GP. These capabilities make our approach suitable for real-time applications.

The rest of the paper is organized as follows. The proposed method is derived from standard GP theory and random Fourier features in Section 2. Section 3 presents our PMMWI data set, its preprocessing, and the experimental setting. Section 4 shows competitive empirical results for our approach, which outperforms direct application of GPs and allows for real-time prediction. Section 5 concludes the paper with some remarks and future outlook.

2 Gaussian Processes and Random Fourier features

Gaussian Processes (GPs) [Rasmussen and Williams, 2006] are popular Bayesian models for supervised machine learning tasks, such as regression and classification. In the latter case, we are provided with an input-output data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$. Following a function-space view, GPs codify this relationship by means of latent variables $\{f_i = f(\mathbf{x}_i) \in \mathbb{R}\}_{i=1}^n$. These variables are jointly normally distributed as $\mathcal{N}(\mathbf{0}, (k(\mathbf{x}_i, \mathbf{x}_j))_{1\leq i,j\leq n})$, with $k : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ the kernel function. The outputs y_i depend on f_i by means of the sigmoid observation model $\mathrm{p}(y_i|f_i) = \psi(f_i) = (1 + \exp(-f_i))^{-1}$.

As any other kernel method, GP for classification (GPC) is computationally prohibitive when dealing with large scale problems. The training step scales as $\mathcal{O}(n^3)$, whereas the computational complexity of the prediction is $\mathcal{O}(n^2)$ for each test instance [Rasmussen and Williams, 2006]. In the case of a standard desktop computer, this places its computational feasibility limit around $n = 10^4$ training examples.

2.1 Random Fourier features

The work [Rahimi and Recht, 2007] presents a general methodology (based on Bochner's theorem [Rudin, 2011]) for theoretically approximating a positive-definite shift-invariant kernel k. This is achieved by explicitly projecting the original d-dimensional data onto $\mathcal{O}(D)$ random Fourier features, whose standard linear kernel approximates k. This linearity allows us to reduce $n \times n$ matrix inversions to $\mathcal{O}(D) \times \mathcal{O}(D)$ ones, which decreases the computational cost to $\mathcal{O}(nD^2)$ during training and to $\mathcal{O}(D^2)$ for each test instance. For large scale applications, D will be much lower than n, which implies a significant benefit.

In this work we use the well-known Gaussian kernel $k(\mathbf{x}, \mathbf{x}') = \gamma \cdot \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/(2\sigma^2))$, which can be linearly approximated as $k(\mathbf{x}, \mathbf{x}') \approx \mathbf{z}(\mathbf{x})^\intercal \mathbf{z}(\mathbf{x}')$, with

$$\mathbf{z}(\mathbf{x})^\intercal = \sqrt{\gamma D^{-1}} \left(\cos\left(\sigma^{-1}\mathbf{w}_1^\intercal \mathbf{x}\right), \sin\left(\sigma^{-1}\mathbf{w}_1^\intercal \mathbf{x}\right), \ldots, \cos\left(\sigma^{-1}\mathbf{w}_D^\intercal \mathbf{x}\right), \sin\left(\sigma^{-1}\mathbf{w}_D^\intercal \mathbf{x}\right)\right) \in \mathbb{R}^{2D}. \tag{1}$$

As indicated in [Rahimi and Recht, 2007], the error in this approximation exponentially decreases with the number D of Fourier frequencies $\mathbf{w}_i \in \mathbb{R}^d$, which must be independently sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. In matrix notation, we approximate the kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ with the explicitly mapped data $\mathbf{Z} = [\mathbf{z}_1 \cdots \mathbf{z}_n]^\intercal \in \mathbb{R}^{n\times D}$ as $\mathbf{K} \approx \mathbf{Z}\mathbf{Z}^\intercal$.

In order to approximate a GP classifier with Gaussian kernel, we will consider a Bayesian linear model working on these new features. Hyperparameters γ and σ in eq. (1) will be optimized within a variational approach in order to maximize the marginal likelihood of observed data. Notice that although we could include the estimation of $\mathbf{w}_1, \ldots, \mathbf{w}_D$ in the Bayesian framework, we concentrate here on the computing capabilities of the proposed approach and assume the frequencies are sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and fixed.

2.2 Modelling and Inference

We consider the standard binary-classification logistic regression model defined over the explicitly mapped Fourier features $\mathbf{z}_i$:

$$\mathrm{p}_{\boldsymbol{\theta}}(y_i = 1|\boldsymbol{\beta}) = (1 + \exp(-\boldsymbol{\beta}^\intercal \mathbf{z}_i))^{-1}, \tag{2}$$

where $\boldsymbol{\theta} = (\theta_1 = \sqrt{\gamma}, \theta_2 = \sigma)$ and $\mathbf{z}_i$ is given by eq. (1).

Weights $\boldsymbol{\beta} \in \mathbb{R}^{2D}$ are assigned a normal prior distribution $\mathrm{p}(\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \mathbf{I})$. Thus, the joint p.d.f. reads

$$\mathrm{p}_{\boldsymbol{\theta}}(\mathbf{y}, \boldsymbol{\beta}) = \mathrm{p}_{\boldsymbol{\theta}}(\mathbf{y}|\boldsymbol{\beta})\mathrm{p}(\boldsymbol{\beta}). \tag{3}$$

To obtain the maximum likelihood (ML) estimate of $\boldsymbol{\theta}$, $\overline{\boldsymbol{\theta}}$, we integrate the above joint distribution on $\boldsymbol{\beta}$ and maximize on $\boldsymbol{\theta}$ the marginal distribution $\mathrm{p}_{\boldsymbol{\theta}}(\mathbf{y})$. The posterior distribution $\mathrm{p}_{\overline{\boldsymbol{\theta}}}(\boldsymbol{\beta}|\mathbf{y})$ is then calculated. However, the sigmoid likelihood in eq. (2) makes these computations analytically intractable, and we resort to the *variational* inference approximation [Bishop, 2006, Section 10.6].

First, we use the variational bound [Bishop, 2006]

$$\log\left(1 + e^x\right) \leq \lambda(\xi)(x^2 - \xi^2) + \frac{x - \xi}{2} + \log\left(1 + e^{\xi}\right), \tag{4}$$

where $x, \xi \in \mathbb{R}$ and $\lambda(\xi) = (1/2\xi)\left((1+e^{-\xi})^{-1} - 1/2\right)$. For a fixed x, it is easy to check that this bound is minimum when $\xi^2 = x^2$. This produces the following lower bound for the joint probability distribution in eq. (3):

$$\log \mathrm{p}_{\boldsymbol{\theta}}(\mathbf{y}, \boldsymbol{\beta}) \geq -\frac{1}{2}\boldsymbol{\beta}^{\intercal}\left(\mathbf{Z}^{\intercal}(2\boldsymbol{\Lambda})\mathbf{Z} + \mathbf{I}\right)\boldsymbol{\beta} + \mathbf{v}^{\intercal}\mathbf{Z}\boldsymbol{\beta} + C(\xi_i), \tag{5}$$

where $\mathbf{v} = \mathbf{y} - (1/2)\cdot \mathbf{1}_{n\times 1}$ and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda(\xi_1), \ldots, \lambda(\xi_n))$.

Notice that this lower bound is quadratic in $\boldsymbol{\beta}$, which enables us to analytically work with it. Namely, using the exponential function on eq. (5), integrating out $\boldsymbol{\beta}$, and maximizing on $\boldsymbol{\theta}$ (recall $\mathbf{Z} = \mathbf{Z}(\boldsymbol{\theta})$), we obtain the following approximation of $\overline{\boldsymbol{\theta}}$:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left(\mathbf{v}^{\intercal}\mathbf{Z}\left(\mathbf{Z}^{\intercal}(2\boldsymbol{\Lambda})\mathbf{Z} + \mathbf{I}\right)^{-1}\mathbf{Z}^{\intercal}\mathbf{v} - \log\left|\mathbf{Z}^{\intercal}(2\boldsymbol{\Lambda})\mathbf{Z} + \mathbf{I}\right|\right). \tag{6}$$

To obtain the posterior $\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{\beta}|\mathbf{y})$, we fix $\boldsymbol{\theta}$ in eq. (3) and find the same problem with the sigmoid function. We again resort to the quadratic variational bound given by eq. (5), which provides the following approximate posterior normal distribution for $\boldsymbol{\beta}$:

$$\begin{aligned} &\mathrm{p}_{\boldsymbol{\theta}}(\boldsymbol{\beta}|\mathbf{y}) \approx \mathrm{q}_{\boldsymbol{\theta}}(\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\beta}|\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}), \\ &\boldsymbol{\Sigma}_{\boldsymbol{\beta}} = \left(\mathbf{Z}^{\intercal}(2\boldsymbol{\Lambda})\mathbf{Z} + \mathbf{I}\right)^{-1}, \quad \boldsymbol{\mu}_{\boldsymbol{\beta}} = \boldsymbol{\Sigma}_{\boldsymbol{\beta}}\mathbf{Z}^{\intercal}\mathbf{v}. \end{aligned} \tag{7}$$

Finally, for a given $\boldsymbol{\theta}$, the bound in eq. (5) is optimal when

$$\xi_i^2 = \langle(\boldsymbol{\beta}^{\intercal}\mathbf{z}_i)^2\rangle_{\mathrm{q}_{\boldsymbol{\theta}}(\boldsymbol{\beta}|\mathbf{y})} \quad \forall i = 1, \ldots, n, \tag{8}$$

which yields

$$\xi_i = \sqrt{\left(\mathbf{z}_i^{\intercal}\boldsymbol{\Sigma}_{\boldsymbol{\beta}}\mathbf{Z}^{\intercal}\mathbf{v}\right)^2 + \mathbf{z}_i^{\intercal}\boldsymbol{\Sigma}_{\boldsymbol{\beta}}\mathbf{z}_i} \quad \forall i = 1, \ldots, n. \tag{9}$$

Our method is named RFF-GPC. At training time, it is fed with labelled data and runs iteratively to calculate $\hat{\boldsymbol{\theta}}$ and the approximate normal posterior $\mathrm{q}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\beta})$ (see Algorithm 1).

Algorithm 1 Training RFF-GPC

```
Input: Data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0,1\}$.
  Randomly sample Fourier frequencies $\mathbf{w}_i$ in eq. (1) from $\mathcal{N}(\mathbf{0}, \mathbf{I})$.
  Initialize $\mathrm{q}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\beta})$ to $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (that is, its prior distribution).
  Initialize $\hat{\boldsymbol{\theta}}$
  repeat
    Update each $\xi_i$ using eq. (9).
    Update $\hat{\boldsymbol{\theta}}$ using eq. (6).
    Update the posterior $\mathrm{q}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\beta})$ using eq. (7).
  until convergence
  return Optimal estimator $\hat{\boldsymbol{\theta}}$ and the posterior $\mathrm{q}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\beta})$.
```

At test time, the probability of class 1 for a previously unseen instance $\mathbf{x}_* \in \mathbb{R}^d$ is:

$$\mathrm{p}(y_* = 1) \approx \int \mathrm{p}_{\hat{\boldsymbol{\theta}}}(y_* = 1|\boldsymbol{\beta})\mathrm{q}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{\beta})\mathrm{d}\boldsymbol{\beta} \approx \psi\left(\mathbf{z}_*^{\intercal}\boldsymbol{\mu}_{\boldsymbol{\beta}} \cdot \left(1 + (\pi/8)\mathbf{z}_*^{\intercal}\boldsymbol{\Sigma}_{\boldsymbol{\beta}}\mathbf{z}_*\right)^{-1/2}\right),$$

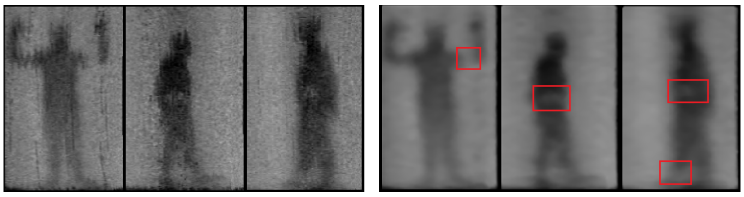


Figure 1: Examples of PMMWIs with hidden threats before preprocessing (left) and after it (right). Red boxes on the right highlight object locations, which correspond to lighter areas of the body.

with ψ the sigmoid function. The computations involved in these algorithms scale as $\mathcal{O}(nD^2)$ for training stage, and $\mathcal{O}(D^2)$ for prediction on each test point. Unlike standard GPC, the latter is independent on the number of training instances. This significant reduction makes our proposal suitable for large scale real-time applications.

3 Image preprocessing and experimental setting

3.1 General preprocessing

We have available a PMMWIs database of 3309 images of size 125×195. They show people in different positions who may hide (up to two) objects of varied sizes and shapes at different locations. Unfortunately, acquired images suffer from spatially variant noise, and their general quality is poor. These problems specially manifest at the contour of the individuals, which can be confused (even by a human thorough gaze) with hidden threats. This prevents classical models from achieving competitive detection results, see also [Tapia et al., 2016]. To better identify threats, the image signal to noise ratio must be increased, and the contrast enhanced. This is addressed by a combination of linear (local mean) and non-linear (local median) smoothing filters. Figure 1 shows examples of raw and processed images in the database (see [Tapia et al., 2016] for more images).

Given a new image, our goal is to automatically discern whether it contains some threat or not. We will not fit the machine learning algorithms directly over the global images, but extract relevant features from local patches. Namely, for every 2×2 non-overlapping block we consider three regions of areas 39×39, 19×19, and 9×9 (these areas have been selected taking into account the targeted threat sizes, neither too big nor too small) centered at one of its four points. We only consider *active* pixels, i.e. those whose 39×39 region is completely contained in the image. To each one of these blocks we assign a feature vector constructed by concatenating Haar traits [Papageorgiou et al., 1998] extracted from its three associated regions. On each region we use 115 Haar filters, which are appropriately chosen for the shapes of hidden objects. This yields a 345-dimensional feature vector for each block, which are considered positive (having threat) if their 39×39 region overlaps a hidden object by at least 50%, see [Tapia et al., 2016] for additional details. In the sequel, we will refer to these blocks (together with their feature vector) as learning *instances*. In total, we have 3476 such instances for each image.

3.2 Experimental setting

To test the proposed model we perform a five folds cross validation. We ensure that each partition contains the same number of images with none, one, and two threats. In the training data sets we only include a positive instance if it (more precisely, its 39×39 associated region) completely covers the threat. Moreover, since adjacent negative instances have equivalent feature vectors, we only keep one from every 2×2 group of them. Table 1 summarises the number of available training instances for the experiments.

Table 1: Number of negative and positive instances in the training data set for each fold.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Negative	1896222	1895130	1898030	1897209	1898825
Positive	299838	302417	299829	299644	298212
Total	2196060	2197547	2197859	2196853	2197037

Table 1 also reveals that the number of negative instances is approximately six times the number of positive ones in each training data set. This imbalance could bias the algorithms against the minority class. To avoid it we train six classifiers per fold, each one using all positive instances and a random sample (without replacement and of equivalent size) of the negative instances in that fold. This means a training data set with $n = 6 \cdot 10^5$ instances approximately.

Standard GPC cannot cope with this large scale setting. In order to comparatively evaluate their performance against RFF-GPC, we consider a (balanced) subsample of size $n = 10^4$ for each experiment, thus taking GPC until its computational limit. Once the six models are trained, predictions (i.e. probabilities of class 1) for test instances are collected and the mean is computed.

For GPC we make use of the full 345-dimensional feature vectors. However, in the case of RFF-GPC we will need to previously reduce this dimensionality.

3.3 Preprocessing for RFF-GPC

A key parameter in our method is the number D of random Fourier features used. It can be checked, both empirically and theoretically [Rahimi and Recht, 2007, Claim 1], that the approximation $\mathbf{K} \approx \mathbf{Z}\mathbf{Z}^\intercal$ exponentially improves with D. Likewise, it exhibits an exponential deterioration with the number d of original features in the data set [Rahimi and Recht, 2007].

In practice, for $d = 345$ like in our case, we would need a value of D greater than 10^5. However, D should not exceed $5 \cdot 10^3$ if we want the training $\mathcal{O}(nD^2)$ to be computationally feasible in a standard desktop computer. This would make RFF-GPC even more computationally prohibitive than standard Gaussian Processes.

In order to overcome this problem, we need to significantly reduce the dimensionality of the data set before applying RFF-GPC, but preserving as much information as possible. For this, we resort to Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) (see [Martínez and Kak, 2001] for an interesting comparison). The first one extracts projections that keep most of the variability of the original data [Bishop, 2006, Chapter 12], whereas the latter provides features that best linearly separate both classes [Bishop, 2006, Section 4.1.4]. We decided to include 15 principal components (a PCA analysis reveals that they already explain the 97.69% of original variance) and 5 linear discriminant directions. Therefore, we reduce the dimensionality to $d' = 20$. With this, we can consider more reasonable values for D, which will be fixed at $D = 500$ in our experiments. In Sections 4 and 5 we discuss future work related to this parameter.

4 Experimental results

First, we compare prediction time between RFF-GPC and GPC. Figure 2 shows elapsed time to predict one image, i.e. 3476 patches. We show the evolution in terms of training data set size. To do this, we extracted balanced subsamples from the largest training data set in each experiment. We observe that, whereas GPC clearly exhibits an increasing dependence on the training set size, RFF-GPC is not affected by this quantity. This is the expected behaviour from theoretical test orders $\mathcal{O}(n^2)$ and $\mathcal{O}(D^2)$ respectively. RFF-GPC outperforms GPC at any training size, being more than 100 times faster at the full models ($n = 6 \cdot 10^5$ and $n = 10^4$ respectively).

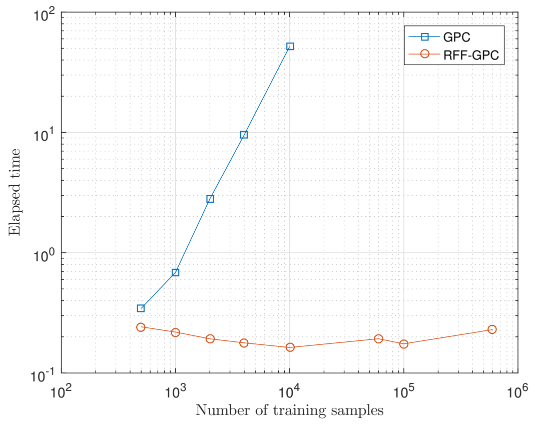


Figure 2: Elapsed prediction time per image for GPC and RFF-GPC. Notice the logarithmic scale in both axes.

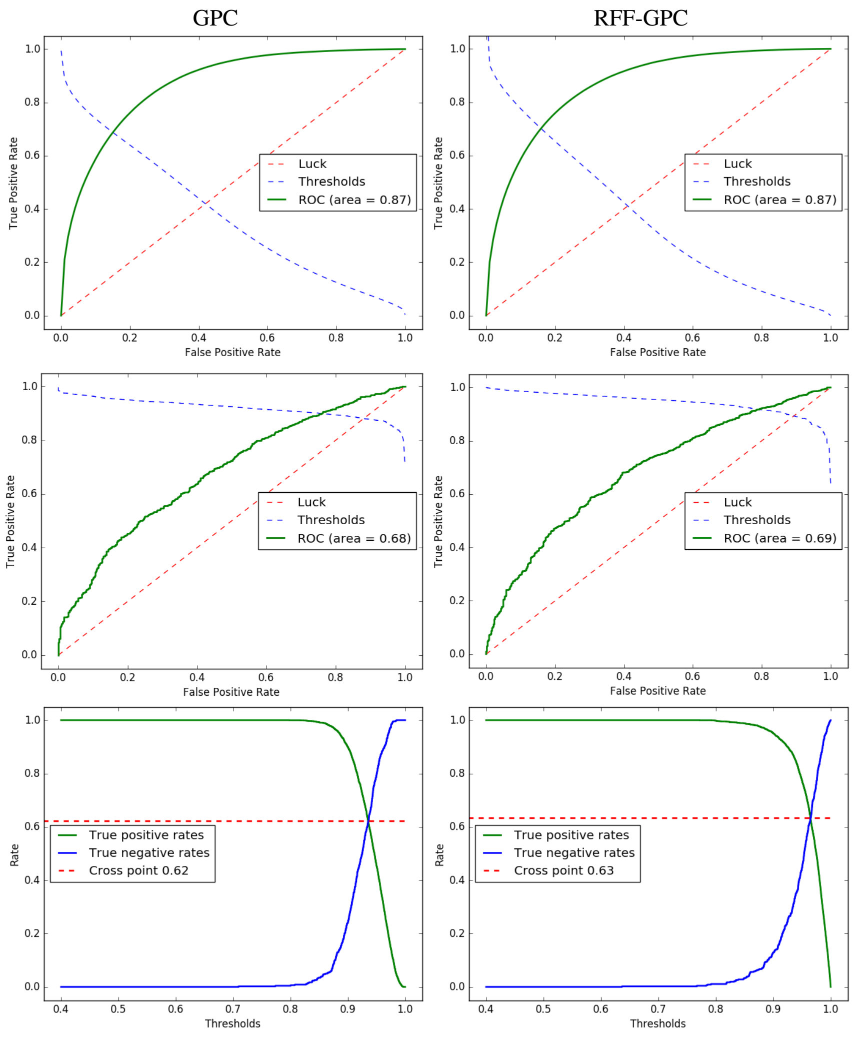


Figure 3: Comparative classification results for GPC (left column) and RFF-GPC (right column). First and second rows show the ROC curves (and AUC) at patch and image levels respectively. Third row explictly depicts the evolution of TPR, TNR, and their cross point.

The second experiment compares the classification performance of GPC and RFF-GPC at both patch and image levels. For the latter, each test image is associated to the highest probability of threat among all its patches. We use the full models, i.e. $n = 10^4$ for GPC and $n \approx 6 \cdot 10^5$ for RFF-GPC. In Figure 3 we show the ROC curves at patch and image levels. They represent the trade-off between true positive (TP) and false positive (FP) rates when considering different thresholds over the test probabilities of class 1 (having threat). The TPR-TNR evolution and their cross point at image level are also provided. We observe that both methods behave similarly, with RFF-GPC slightly outperforming GPC at image level.

We also carried out a preliminary experiment to assess the quality of our approach when using a higher number of random Fourier features D (see future work in Section 5). We used RFF-GPC with $D = 1750$ and $n = 1.5 \cdot 10^5$ training instances (both values can be still further increased). This significantly improves previous results, see Figure 4. Namely, AUC metric and TPR-TNR cross point improve from 0.69 and 0.63 to 0.72 and 0.65 respectively. These results suggest that increasing and wisely combining the values of D and n is a promising future work.

It is worth noticing that our novel approach is competitive with the state-of-the-art results over this data set presented in [Tapia et al., 2016]. The new method beats other kernel machines like SVM, and only Random Forest (RF) performs slightly better. Namely, at image level, TP and TN rates cross at 0.68 for RF (see [Tapia et al., 2016, Figure 6]) while, as we have already indicated, the crossing point for our method is at 0.65. The future use of prior models on the Fourier frequencies $\mathbf{w}_i$ and higher values of D and n are expected to surpass RF performance.

To sum up, our results are competitive with state-of-the-art ones. We already outperform classical GPC allowing for a much faster prediction (which is a remarkable benefit for real-world applications). More complex modelling of the classification problem will certainly lead to better performance.

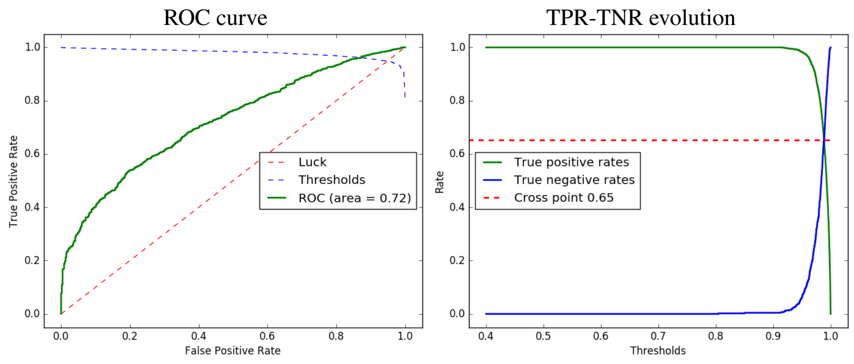


Figure 4: ROC curve (and AUC) and TPR-TNR evolution at image level for RFF-GPC with higher value of D.

5 Conclusions and future work

In this work we presented a new kernel-based approach to classify PMMWIs using a large scale training data set. The method works at patch level by extracting multiscale Haar features which are then summarized using PCA and LDA projections. The huge number of samples prevents us from using classical GPC methods on the complete data set. We resort here to the use of random Fourier features to linearize a non-linear kernel. A variational Bayes scheme is used to make inference tractable. All the model parameters are estimated. The proposed approach outperforms classical GPC, it is suitable for real-time applications, and produces competitive results with current state-of-the-art methods.

Since detection results are negatively affected by the poor quality of images, we are currently working on the use of image processing techniques to improve acquired images. Furthermore, we are also investigating the optimal relationship between the number of projected features d', Fourier frequencies D, and training examples n. The optimal estimation of Fourier frequencies in a probabilistic sense is also currently under study.

References

[Babacan et al., 2011] Babacan, S. D., Luessi, M., Spinoulas, L., Katsaggelos, A. K., Gopalsami, N., Elmer, T., Ahern, R., Liao, S., and Raptis, A. (2011). Compressive passive millimeter-wave imaging. In *2011 18th IEEE International Conference on Image Processing*, pages 2705–2708.

[Bazi and Melgani, 2010] Bazi, Y. and Melgani, F. (2010). Gaussian process approach to remote sensing image classification. *IEEE transactions on geoscience and remote sensing*, 48(1):186–197.

[Bishop, 2006] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Camps-Valls and Bruzzone, 2005] Camps-Valls, G. and Bruzzone, L. (2005). Kernel-based methods for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 43(6):1351–1362.

[Chapelle et al., 1999] Chapelle, O., Haffner, P., and Vapnik, V. N. (1999). Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055–1064.

[Damianou and Lawrence, 2014] Damianou, A. and Lawrence, N. (2014). Deep Gaussian Processes. *Journal of Machine Learning Research*, 31:207–215.

[Han et al., 2010] Han, B., Xiong, J., Li, L., Yang, J., and Wang, Z. (2010). Research on millimeter-wave image denoising method based on contourlet and compressed sensing. In *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, volume 2, pages V2–471–V2–475.

[Haworth et al., 2004] Haworth, C., Gonzalez, B., Tomsin, M., Appleby, R., Coward, P., Harvey, A., Lebart, K., Petillot, Y., and Trucco, E. (2004). Image analysis for object detection in millimetre-wave images. In *Passive Millimetre-wave and terahertz imaginh and technology*, volume 5619, pages 117–128.

[Haworth et al., 2006] Haworth, C., Petillot, Y., and Trucco, E. (2006). Image processing techniques for metallic object detection with millimetre-wave images. *Pattern Recognition Letters*, 27(15):1843 – 1851.

[Hensman et al., 2016] Hensman, J., Durrande, N., and Solin, A. (2016). Variational Fourier features for Gaussian processes. *arXiv preprint arXiv:1611.06740*.

[Lázaro-Gredilla et al., 2010] Lázaro-Gredilla, M., Quiñonero Candela, J., Rasmussen, C., and Figueiras-Vidal, A. (2010). Sparse spectrum Gaussian process regression. *Journal of Machine Learning Research*, (Jun):1865–1881.

[Maqueda et al., 2015] Maqueda, I. G., de la Blanca, N. P., Molina, R., and Katsaggelos, A. (2015). Fast millimetre wave threat detection algorithm. In *European Signal Processing Conference (EUSIPCO 2015)*, pages 599–603. Nice (France).

[Martínez and Kak, 2001] Martínez, A. M. and Kak, A. C. (2001). PCA versus LDA. *IEEE transactions on pattern analysis and machine intelligence*, 23(2):228–233.

[Martínez et al., 2010] Martínez, O., Ferraz, L., Binefa, X., Gómez, I., and Dorronsoro, C. (2010). Concealed object detection and segmentation over millimetric waves images. In *2010 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, pages 31–37.

[Mateos et al., 2016] Mateos, J., López, A., Vega, M., Molina, R., and Katsaggelos, A. K. (2016). Multiframe blind deconvolution of passive millimeter wave images using variational dirichlet blur kernel estimation. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 2678–2682. IEEE.

[Papageorgiou et al., 1998] Papageorgiou, C. P., Oren, M., and Poggio, T. (1998). A general framework for object detection. In *Computer vision, 1998. sixth international conference on*, pages 555–562. IEEE.

[Rahimi and Recht, 2007] Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press, New York.

[Rudin, 2011] Rudin, W. (2011). *Fourier analysis on groups*. John Wiley & Sons.

[Ruiz et al., 2016] Ruiz, P., Molina, R., and Katsaggelos, A. (2016). Joint data filtering and labeling using gaussian processes and alternating direction method of multipliers. *IEEE Transactions on Image Processing*, 25(7):3059–3072.

[Saafin et al., 2016] Saafin, W., Villena, S., Vega, M., Molina, R., and Katsaggelos, A. (2016). Compressive sensing super resolution from multiple observations with application to passive millimeter wave images. *Digital Signal Processing*, 50:180–190.

[Sheen et al., 2001] Sheen, D. M., McMakin, D. L., and Hall, T. E. (2001). Three-dimensional millimeter-wave imaging for concealed weapon detection. *IEEE Transactions on microwave theory and techniques*, 49(9):1581–1592.

[Tapia et al., 2016] Tapia, S. L., Molina, R., and de la Blanca, N. P. (2016). Detection and localization of objects in passivemillimeter wave images. In *European Signal Processing Conference (EUSIPCO 2016)*.

[Yang et al., 2010] Yang, J., Wang, J., and Li, L. (2010). A new algorithm for passive millimeter-wave image enhancement. In *Signal Processing Systems (ICSPS), 2010 2nd International Conference on*, volume 3, pages V3–507–V3–511.

[Yu et al., 2011] Yu, W., Chen, X., Dong, S., and Shao, W. (2011). Study on image enhancement algorithm applied to passive millimeter-wave imaging based on wavelet transformation. In *Electrical and Control Engineering (ICECE), 2011 International Conference on*, pages 856–859.

[Yujiri, 2006] Yujiri, L. (2006). Passive millimeter wave imaging. In *2006 IEEE MTT-S International Microwave Symposium Digest*, pages 98–101.

[Yujiri et al., 2003] Yujiri, L., Shoucri, M., and Moffa, P. (2003). Passive millimeter wave imaging. *IEEE Microwave Magazine*, 4(3):39–50.

CHAPTER 3

Gaussian Processes and Fourier features for remote sensing

3.1 Publication details

Authors: Pablo Morales-Álvarez, Adrián Pérez-Suay, Rafael Molina, Gustau Camps-Valls
Title: Remote Sensing Image Classification with Large Scale Gaussian Processes
Reference: IEEE Transactions on Geoscience and Remote Sensing 56 (2), 1103-1114, 2017
Status: Published
Quality indices:

- Impact Factor (JCR 2017): 4.662
- Rank: 28/260 (Q1) in Engineering, Electrical and Electronic.

3.2 Main contributions

- Building on RFF-GPC, we propose *VFF-GPC* (Variational Fourier features for Gaussian Processes Classification). The key novel contribution is that Fourier features are learned during the training step, and not just sampled at the beginning and fixed as in the case of RFF-GPC. This allows for greater flexibility and higher accuracy. However, the computational cost is also greater and the learned kernel may not be an approximation of the original one.
- The performance of VFF-GPC (and that of RFF-GPC) is illustated in complex problems of cloud detection from multispectral imagery and infrared sounding data. The results show that the proposed methods allow for scalability to large datasets, obtaining high accuracy at a reduced computational cost.

Remote Sensing Image Classification with Large Scale Gaussian Processes

Pablo Morales-Álvarez
Dept. of Computer Science and AI
University of Granada, Spain

Adrián Pérez-Suay
Image Processing Laboratory
University of Valencia, Spain

Rafael Molina
Dept. of Computer Science and AI
University of Granada, Spain

Gustau Camps-Valls
Image Processing Laboratory
University of Valencia, Spain

Abstract

Current remote sensing image classification problems have to deal with an unprecedented amount of heterogeneous and complex data sources. Upcoming missions will soon provide large data streams that will make land cover/use classification difficult. Machine learning classifiers can help at this, and many methods are currently available. A popular kernel classifier is the Gaussian process classifier (GPC), since it approaches the classification problem with a solid probabilistic treatment, thus yielding confidence intervals for the predictions as well as very competitive results to state-of-the-art neural networks and support vector machines. However, its computational cost is prohibitive for large scale applications, and constitutes the main obstacle precluding wide adoption. This paper tackles this problem by introducing two novel efficient methodologies for Gaussian Process (GP) classification. We first include the standard random Fourier features approximation into GPC, which largely decreases its computational cost and permits large scale remote sensing image classification. In addition, we propose a model which avoids randomly sampling a number of Fourier frequencies, and alternatively *learns* the optimal ones within a variational Bayes approach. The performance of the proposed methods is illustrated in complex problems of cloud detection from multispectral imagery and infrared sounding data. Excellent empirical results support the proposal in both computational cost and accuracy.

1 Introduction

"... Nature almost surely operates by combining chance with necessity, randomness with determinism..."
–Eric Chaisson, Epic of Evolution: Seven Ages of the Cosmos

Earth-observation (EO) satellites provide a unique source of information to address some of the challenges of the Earth system science [Berger et al., 2012]. Current EO applications for image classification have to deal with a huge amount of heterogeneous and complex data sources.

The super-spectral Copernicus Sentinels [Drusch et al., 2012, Donlon et al., 2012], as well as the planned EnMAP [Stuffler et al., 2007], HyspIRI [Roberts et al., 2012], PRISMA [Labate et al., 2009] and FLEX [Kraft et al., 2013], will soon provide unprecedented data streams to be analyzed. Very high resolution (VHR) sensors like Quickbird, Worldview-2 and the recent Worldview-3 [Longbotham et al., 2014] also pose big challenges to data processing. The challenge is not only attached to optical sensors. Infrared sounders, like the Infrared Atmospheric Sounding Interferometer (IASI) [Tournier et al., 2002] sensor on board the MetOp satellite series, impose even larger constraints: the orbital period of Metop satellites (101 minutes), the large spectral resolution (8461 spectral channels between 645 cm^{-1} and 2760 cm^{-1}), and the spatial resolution (60$\times$1530 samples) of the IASI instrument yield several hundreds of gigabytes of data to be processed daily. The IASI mission delivers approximately 1.3×10^6 spectra per day, which gives a rate of about 29 Gbytes/day to be processed. EO radar images also increased in resolution, and current platforms such as ERS-1/2, ENVISAT, RadarSAT-1/2, TerraSAR-X, and Cosmo-SkyMED give raise to extremely fine resolution data that call for advanced scalable processing methods. Besides, we should not forget the availability of the extremely large remote sensing data archives[1] already collected by several past missions. In addition, we should be also prepared for the near future in diversity and complementarity of sensors[2]. These large scale data problems require enhanced processing techniques that should be accurate, robust and fast. Standard classification algorithms cannot cope with this new scenario efficiently.

In the last decade, kernel methods have dominated the field of remote sensing image classification [Camps-Valls and Bruzzone, 2009, Camps-Valls et al., 2011]. In particular, a kernel method called support vector machine (SVM, [Huang et al., 2002, Camps-Valls et al., 2004, Melgani and Bruzzone, 2004, Foody and Mathur, 2004, Camps-Valls and Bruzzone, 2005]) was gradually introduced in the field, and quickly became a standard for image classification. Further SVM developments considered the simultaneous integration of spatial, spectral and temporal information [Benediktsson et al., 2005, Fauvel et al., 2008, Pacifici et al., 2009, Tuia et al., 2009, Camps-Valls et al., 2008], the richness of hyperspectral imagery [Camps-Valls and Bruzzone, 2005, Plaza et al., 2009], and exploited the power of clusters of computers [Plaza et al., 2008, Muñoz-Marí et al., 2009]. Undoubtedly, kernel methods have been the most widely studied classifiers, and became the preferred choice for users and practitioners. However, they are still not widely adopted in real practice because of the high computational cost when dealing with large scale problems. Roughly speaking, given n examples available for training, kernel machines need to store kernel matrices of size $n \times n$, and to process them using standard linear algebra tools (matrix inversion, factorization, eigen-decomposition, etc.) that typically scale cubically, $\mathcal{O}(n^3)$. This is an important constraint that hampers their applicability to large scale EO data processing.

An alternative kernel classifier to SVM is the Gaussian Process classifier (GPC) [Rasmussen and Williams, 2006a]. GPC has appealing theoretical properties, as it approaches the classification problem with a solid probabilistic treatment, and very good performance in practice. The GPC method was originally introduced in the field of remote sensing in [Bazi and Melgani, 2010], where very good capabilities for land cover classification from multi/hyperspectral imagery were illustrated. Since then, GPC has been widely used in practice and extended to many settings: hyperspectral image classification [Yang et al., 2015], semantic annotation of high-resolution remote

[1]The Earth Observing System Data and Information System (EOSDIS) for example is managing around 4 terabytes daily, and the flow of data to users is about 20 terabytes daily.

[2]Follow the links for an up-to-date list of current ESA, EUMETSAT, JAXA, CNSA and NASA EO missions.

sensing images [Chen et al., 2013a], change detection problems with semisupervised GPC [Chen et al., 2013b], or classification of images with the help of user's intervention in active learning schemes [Ruiz et al., 2014, Kalantari et al., 2016]. Unfortunately, like any other kernel method, its computational cost is very large. This is probably the reason why GPC has not yet been widely adopted by the geoscience and remote sensing community in large scale classification scenarios, despite its powerful theoretical background and excellent performance in practice.

In GP for classification we face two main problems. First, the non-conjugate observation model for classification (usually based on the sigmoid, probit, or Heaviside step functions) renders the calculation of the marginal distribution needed for inference impossible. The involved integrals are not analytically tractable, so one has to resort to numerical methods or approximations [Rasmussen and Williams, 2006a]. One could rely on Markov Chain Monte Carlo (MCMC) methods, but they are computationally far too expensive. By assuming a Gaussian approximation to the posterior of the latent variables, one can use the Laplace approximation (LA) and the (more accurate) expectation propagation (EP) [Minka, 2001, Kuss and Rasmussen, 2005]. The observation model can also be bounded, leading to the variational inference approach [Chen et al., 2014] that we use in this paper. Once the non-conjugacy of the observation model has been solved, the second problem is the inversion of huge matrices, which yields the unbearable $\mathcal{O}(n^3)$ complexity. Notice that this is the only difficulty that appears when GP is used for regression, where the observation model can be analytically integrated out. This efficiency problem could be addressed with recent Sparse GP approximations based on inducing points and approximate inference [Damianou, 2015], but they come at the price of a huge number of parameters to estimate.

In this paper, we introduce two alternative pathways to perform large scale remote sensing image classification with GPC. First, following the ideas in [Rahimi and Recht, 2007], we approximate the squared exponential (SE) kernel matrix of GPC by a linear one based on projections over a reduced set of random Fourier features (RFF). This novel method is referred to as RFF-GPC. It allows us to work in the primal space of features, which significantly reduces the computational cost of large scale applications. In fact, a recent similar approach allows for using millions of examples in SVM-based land cover classification and regression problems [Pérez-Suay et al., 2017]. The solid theoretical ground and the good empirical RFF-GPC performance make it a very useful method to tackle large scale problems in Earth observation. However, RFF-GPC can only approximate (theoretically and in practice) a predefined kernel (the SE in this work), and the approximation does not necessarily lead to a discriminative kernel. These shortcomings motivate our second methodological proposal: we introduce a novel approach to avoid *randomly sampling* a number of Fourier frequencies, and instead we propose *learning* the optimal ones within the variational approach. Therefore, Fourier frequencies are no longer *randomly sampled* and *fixed*, but *latent variables* estimated directly from data via variational inference. We refer to this method as VFF-GPC (Variational Fourier Features). The performance of RFF-GPC and VFF-GPC is illustrated in large and medium size real-world remote sensing image classification problems: (1) classification of clouds over landmarks from a long time series of Seviri/MSG (Spinning Enhanced Visible and Infrared Imager, Meteosat Second Generation) remote sensing images, and (2) cloud detection using IASI and AVHRR (Advanced Very High Resolution Radiometer) infrared sounding data, respectively. Excellent empirical results support the proposed large scale methods in both accuracy and computational efficiency. In particular, the extraordinary performance of VFF-GPC in the medium size data set justifies its use not only as a large scale method, but also as a general-purpose and scalable classification tool capable of learning an appropriate discriminative kernel.

The remainder of the paper is organized as follows. Section 2 reviews the RFF approximation and introduces it into GPC, deriving RFF-GPC and the more sophisticated VFF-GPC. Section 3 introduces the two real-world remote sensing data sets used for the experimental validation. Section 4 presents the experimental results comparing the two proposed methods and standard GPC in terms of accuracy and efficiency. Section 5 concludes the paper with some remarks and future outlook.

2 Large Scale Gaussian Process Classification

Gaussian Processes (GP) [Rasmussen and Williams, 2006b] is a probabilistic state-of-the-art model for regression and classification tasks. In the geostatistic community, GP for regression is usually referred to as *kriging*. For input-output data pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, a GP models the underlying dependence from a function-space perspective, i.e. introducing latent variables $\{f_i = f(\mathbf{x}_i) \in \mathbb{R}\}_{i=1}^n$ that jointly follow a normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{1\le i,j\le n})$. The kernel function k encodes the sort of functions f favored, and $\mathbf{K}$ is the so-called kernel matrix. The observation model of the output y given the latent variable f depends on the problem at hand. In binary classification (i.e. when $y \in \{0, 1\}$), the (non-conjugate) logistic observation model is widely used. It is given by the sigmoid function as $\mathrm{p}(y = 1|f) = \psi(f) = (1 + \exp(-f))^{-1} \in (0, 1)$.

2.1 Random Fourier Features

The main issue with large scale applications of GP is its $\mathcal{O}(n^3)$ cost at the training phase, which comes from the $n \times n$ kernel matrix inversion. The work [Rahimi and Recht, 2007] presents a general methodology (based on Bochner's theorem [Rudin, 2011]) to approximate any positive-definite shift-invariant kernel k by a linear one. This is achieved by explicitly projecting the original d-dimensional data $\mathbf{x}$ onto $\mathcal{O}(D)$ random Fourier features $\mathbf{z}(\mathbf{x})$, whose linear kernel k_L approximates k. This linearity will enable us to work in the primal space of features and substitute $n \times n$ matrix inversions by $\mathcal{O}(D) \times \mathcal{O}(D)$ ones, resulting in a total $\mathcal{O}(nD^2 + D^3)$ computational cost. In large-scale applications, one can set a $D \ll n$, and thus the obtained $\mathcal{O}(nD^2)$ complexity represents an important benefit over the original $\mathcal{O}(n^3)$. Moreover, the complexity at test is also reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(D^2)$, even becoming independent on n.

In this work we use the well-known SE (or Gaussian) kernel $k(\mathbf{x}, \mathbf{x}') = \gamma \cdot \exp(-||\mathbf{x} - \mathbf{x}'||^2/(2\sigma^2))$. Following the methodology in [Rahimi and Recht, 2007], this kernel can be linearly approximated as

$$k(\mathbf{x}, \mathbf{x}') \approx k_L(\mathbf{x}, \mathbf{x}') = \gamma \cdot \mathbf{z}(\mathbf{x})^\intercal \mathbf{z}(\mathbf{x}'), \tag{1}$$

where

$$\mathbf{z}(\mathbf{x})^\intercal = D^{-1/2} \cdot \left(\cos\left(\mathbf{w}_1^\intercal \mathbf{x}\right), \sin\left(\mathbf{w}_1^\intercal \mathbf{x}\right), \ldots, \cos\left(\mathbf{w}_D^\intercal \mathbf{x}\right), \sin\left(\mathbf{w}_D^\intercal \mathbf{x}\right)\right) \in \mathbb{R}^{2D}, \tag{2}$$

and the *Fourier frequencies* $\mathbf{w}_i$ must be sampled from a normal distribution $\mathcal{N}(\mathbf{0}, \sigma^{-2}\mathbf{I})$. As explained in [Rahimi and Recht, 2007, Claim 1], this approximation exponentially improves with the number D of Fourier frequencies used (and also exponentially worsens with d, the original dimension of $\mathbf{x}$). However, obviously, increasing D in our methods will go at the cost of increasing the

$\mathcal{O}(nD^2)$ and $\mathcal{O}(D^2)$ complexities. Other kernels different from the SE one could be used, but that would imply sampling from a different distribution.

Our novel RFF-GPC method considers a standard Bayesian linear model over these new features $\mathbf{z}$. Such a linear model corresponds to GP classification with the linear kernel k_L [Bishop, 2006, Chapter 6]. Since k_L approximates the SE kernel k, our RFF-GPC constitutes an approximation to GP classification with SE kernel. However, RFF-GPC is well suited for large scale applications, as it works in the primal space of $\mathcal{O}(D)$ features and thus presents a $\mathcal{O}(nD^2)$ (resp. $\mathcal{O}(D^2)$) train (resp. test) complexity.

Notice that RFF-GPC needs to sample the Fourier frequencies $\mathbf{w}_i$ from $\mathcal{N}(\mathbf{0}, \sigma^{-2}\mathbf{I})$ from the beginning, whereas hyperparameters σ and γ must be estimated during the learning process (just as in standard GP classification). In order to uncouple $\mathbf{w}_i$ and σ, in the sequel we consider the equivalent features

$$\mathbf{z}(\mathbf{x}|\sigma, \mathbf{W})^\intercal = D^{-1/2} \cdot \left(\cos\left(\sigma^{-1}\mathbf{w}_1^\intercal\mathbf{x}\right), \sin\left(\sigma^{-1}\mathbf{w}_1^\intercal\mathbf{x}\right), \ldots, \cos\left(\sigma^{-1}\mathbf{w}_D^\intercal\mathbf{x}\right), \sin\left(\sigma^{-1}\mathbf{w}_D^\intercal\mathbf{x}\right)\right), \tag{3}$$

with $\mathbf{w}_i$ now sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and fixed. Notice that we have collectively denoted $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_D]^\intercal \in \mathbb{R}^{D\times d}$.

At this point, it is natural to consider other possibilities for the Fourier frequencies $\mathbf{W}$, rather than just randomly sample and fix them from the beginning. The proposed VFF-GPC model treats them as hyperparameters to be estimated (so as to maximize the likelihood of the observed data), just like σ and γ in the case of RFF-GPC. This makes VFF-GPC more expressive, flexible, and tailored to the data, although it may no longer constitute an approximation to the SE kernel (for which the $\mathbf{w}_i$ must be normally distributed). More specifically, VFF-GPC does start with an approximated SE kernel (since the $\mathbf{w}_i$ are initialized with a normal distribution), but the maximum a posteriori (MAP) optimization on $\mathbf{w}_i$ makes it *learn* a new kernel which may no longer approximate a SE one. Therefore, for VFF-GPC we also use $\mathbf{z}(\mathbf{x}|\sigma, \mathbf{W})$ as in eq. (3), with now both σ and $\mathbf{W}$ to be estimated. Interestingly, VFF-GPC extends the Sparse Spectrum Gaussian Process model originally introduced for regression in [Lázaro-Gredilla et al., 2010], to GP classification.

More specifically, the authors there also sparsify the SE kernel by working on the primal space of $\cos$ and $\sin$ Fourier features, see [Lázaro-Gredilla et al., 2010, Equation 5]. However, our classification setting involves a sigmoid-based (logistic) observation model for the output given the latent variable (see the next eq. (4)), whereas in regression this is just given by a normal distribution. Therefore, VFF-GPC needs to additionally deal with the non-conjugacy of the sigmoid, which motivates the variational bound of eq. (6) and the consequent variational inference procedure described in Section 2.3. It is interesting to realize that VFF-GPC is introduced here as a natural extension of RFF-GPC, whereas there is not a regression analogous for RFF-GPC in [Lázaro-Gredilla et al., 2010].

Another possibility for the Fourier frequencies would be to estimate them (just as in VFF-GPC) but considering alternative prior distributions $\mathrm{p}(\mathbf{W})$ (which means utilizing alternative kernels). Moreover, instead of maximum a posteriori inference, we could address the marginalization of the Fourier frequencies $\mathbf{W}$. Alternatively, to promote sparsity, the use of Gaussian Scale Models (GSM) [Babacan et al., 2012] could also be investigated. These possibilities will be explored in future work, and here we will concentrate on RFF-GPC and VFF-GPC.

2.2 Models formulation

As anticipated in previous section, RFF-GPC and VFF-GPC are standard Bayesian linear models working on the explicitly mapped features $\mathbf{z}(\mathbf{x}|\sigma, \mathbf{W})$ of eq. (3). In the case of RFF-GPC, $\mathbf{W}$ is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ at the beginning and fixed, with σ to be estimated. In the case of VFF-GPC, both $\mathbf{W}$ and σ are estimated, with a $\mathcal{N}(\mathbf{0}, \mathbf{I})$ prior over $\mathbf{W}$. In order to derive both methods in a unified way, Φ will denote σ for RFF-GPC and both $(\mathbf{W}, \sigma)$ for VFF-GPC.

Since we are dealing with binary classification, we consider the standard logistic observation model

$$\mathrm{p}(y = 1|\boldsymbol{\beta}, \Phi, \mathbf{x}) = \psi(\boldsymbol{\beta}^\intercal \mathbf{z}) = (1 + \exp(-\boldsymbol{\beta}^\intercal \mathbf{z}))^{-1}, \tag{4}$$

where $\mathbf{z} = \mathbf{z}(\mathbf{x}|\Phi)$. For the weights $\boldsymbol{\beta} \in \mathbb{R}^{2D}$ we utilize the prior normal distribution $\mathrm{p}(\boldsymbol{\beta}|\gamma) = \mathcal{N}(\boldsymbol{\beta}|\mathbf{0}, \gamma\mathbf{I})$, with γ to be estimated, see eq. (1).

For an observed dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0, 1\}$, the joint p.d.f. reads

$$\mathrm{p}(\mathbf{y}, \boldsymbol{\beta}|\Phi, \gamma, \mathbf{X}) = \mathrm{p}(\mathbf{y}|\boldsymbol{\beta}, \Phi, \mathbf{X})\mathrm{p}(\boldsymbol{\beta}|\gamma) = \left(\prod_{i=1}^n \mathrm{p}(y_i|\boldsymbol{\beta}, \Phi, \mathbf{x}_i)\right) \mathrm{p}(\boldsymbol{\beta}|\gamma), \tag{5}$$

where we collectively denote $\mathbf{y} = (y_1, \ldots, y_n)^\intercal$ and $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n]^\intercal$. For the sake of brevity, from now on we will systematically omit the conditioning on $\mathbf{X}$.

2.3 Variational inference

Given the observed dataset $\mathcal{D}$, in this section we seek point estimates of γ and Φ by maximizing the marginal likelihood $\mathrm{p}(\mathbf{y}|\Phi, \gamma)$ (in VFF-GPC, the additional prior $\mathrm{p}(\mathbf{W}) = \mathcal{N}(\mathbf{W}|\mathbf{0}, \mathbf{I})$ yields maximum a posteriori inference, instead of maximum likelihood one, for $\mathbf{W}$). After that, we obtain (an approximation to) the posterior distribution $\mathrm{p}(\boldsymbol{\beta}|\mathbf{y}, \Phi, \gamma)$. Due to the non-conjugate observation model, the required integrals will be mathematically intractable, and we will resort to the variational inference approximation [Bishop, 2006, Section 10.6].

First, notice that integrating out $\boldsymbol{\beta}$ in eq. (5) is not analytically possible due to the sigmoid functions in the observation model $\mathrm{p}(\mathbf{y}|\boldsymbol{\beta}, \Phi)$. To overcome this problem, we use the variational bound

$$\log(1 + e^x) \leq \lambda(\xi)(x^2 - \xi^2) + \frac{x - \xi}{2} + \log\left(1 + e^\xi\right), \tag{6}$$

which is true for any real numbers x, ξ and where $\lambda(\xi) = (1/2\xi)\left(\psi(\xi) - 1/2\right)$ [Bishop, 2006, Section 10.6]. Applying it to every factor of $\mathrm{p}(\mathbf{y}|\boldsymbol{\beta}, \Phi)$, we have the following lower bound

$$\mathrm{p}(\mathbf{y}|\boldsymbol{\beta}, \Phi) \geq \exp\left(-\boldsymbol{\beta}^\intercal \mathbf{Z}^\intercal \boldsymbol{\Lambda} \mathbf{Z} \boldsymbol{\beta} + \mathbf{v}^\intercal \mathbf{Z} \boldsymbol{\beta}\right) \cdot C(\boldsymbol{\xi}). \tag{7}$$

Here we write $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_n]^\intercal \in \mathbb{R}^{n \times 2D}$ for the projected-data matrix (which depends on Φ), $\boldsymbol{\Lambda}$ is the diagonal matrix $\mathrm{diag}(\lambda(\xi), \ldots, \lambda(\xi_n))$, $\mathbf{v} = \mathbf{y} - (1/2) \cdot \mathbf{1}_{n\times 1}$, and the term $C(\boldsymbol{\xi}) = \prod_i \exp\left(\lambda(\xi_i)\xi_i^2 + (1/2)\xi_i - \log(1 + \exp(\xi_i))\right)$ only depends on $\boldsymbol{\xi}$. The key is that this lower bound for $\mathrm{p}(\mathbf{y}|\boldsymbol{\beta}, \Phi)$ is conjugate with the normal prior $\mathrm{p}(\boldsymbol{\beta}|\gamma)$ (since it is the exponential of a quadratic function on $\boldsymbol{\beta}$), and thus it allows us integrating out $\boldsymbol{\beta}$ in eq (5). In exchange, we have introduced n additional hyperparameters $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)^\intercal$ that will need to be estimated along with Φ and γ.

Therefore, substituting $\mathrm{p}(\mathbf{y}|\boldsymbol{\beta}, \Phi)$ for its bound, eq. (5) can be lower bounded as

$$\mathrm{p}(\mathbf{y}, \boldsymbol{\beta}|\Phi, \gamma) \geq F(\mathbf{y}, \Phi, \gamma, \boldsymbol{\xi}) \cdot \mathcal{N}(\boldsymbol{\beta}|\boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{8}$$

where we have denoted

$$F(\mathbf{y},\Phi,\gamma,\boldsymbol{\xi}) = C(\boldsymbol{\xi})\left|\gamma^{-1}\boldsymbol{\Sigma}\right|^{1/2}\exp\left(\frac{1}{2}\boldsymbol{\mu}^{\intercal}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\right),$$
$$\boldsymbol{\Sigma} = \left(\mathbf{Z}^{\intercal}(2\boldsymbol{\Lambda})\mathbf{Z}+\gamma^{-1}\mathbf{I}\right)^{-1}, \quad \boldsymbol{\mu} = \boldsymbol{\Sigma}\mathbf{Z}^{\intercal}\mathbf{v}. \tag{9}$$

Now it is clear that we can marginalize out $\boldsymbol{\beta}$, and iteratively estimate the optimal values of Φ, γ and $\boldsymbol{\xi}$ as those that maximize $F(\mathbf{y},\Phi,\gamma,\boldsymbol{\xi})$ (or equivalently $\log F(\mathbf{y},\Phi,\gamma,\boldsymbol{\xi})$) for the observed $\mathbf{y}$. Starting at $\boldsymbol{\xi}^{(1)}$, $\Phi^{(1)}$, and $\gamma^{(1)}$, we can calculate $\boldsymbol{\xi}^{(k)}$, $\Phi^{(k)}$, and $\gamma^{(k)}$ for $k \geq 1$. In the case of $\boldsymbol{\xi}$, we make use of the local maximum condition $\partial F/\partial\boldsymbol{\xi} = 0$. From there, it is not difficult to prove that the optimal value satisfies [Bishop, 2006]

$$\boldsymbol{\xi}^{(k+1)} = \sqrt{\operatorname{diag}\left(\mathbf{Z}^{(k)}\boldsymbol{\Sigma}^{(k)}\,(\mathbf{Z}^{(k)})^{\intercal}\right) + (\mathbf{Z}^{(k)}\boldsymbol{\mu}^{(k)})^{2}}, \tag{10}$$

where the square and square root of a vector are understood as element-wise. In the case of Φ and γ, we use nonlinear conjugate gradient methods [Fletcher and Reeves, 1964] and obtain (notice that for VFF-GPC we can collapse $\mathbf{W}$ and σ, removing the prior on $\mathbf{W}$)

$$\left(\Phi^{(k+1)},\gamma^{(k+1)}\right) = \arg\max_{\Phi,\gamma}\left\{-\log\left|2\gamma\mathbf{Z}^{\intercal}\boldsymbol{\Lambda}^{(k+1)}\mathbf{Z}+\mathbf{I}\right| + \mathbf{v}^{\intercal}\mathbf{Z}\left(2\mathbf{Z}^{\intercal}\boldsymbol{\Lambda}^{(k+1)}\mathbf{Z}+\gamma^{-1}\mathbf{I}\right)^{-1}\mathbf{Z}^{\intercal}\mathbf{v}\right\}. \tag{11}$$

Once the hyperparameters Φ, γ, and $\boldsymbol{\xi}$ have been estimated by $\hat{\Phi}$, $\hat{\gamma}$, and $\hat{\boldsymbol{\xi}}$ respectively, we need to compute the posterior $\mathrm{p}(\boldsymbol{\beta}|\mathbf{y},\hat{\Phi},\hat{\gamma})$. As before, this is mathematically intractable due to the sigmoids in the observation model. Therefore, we again resort to the variational bound in eq. (8) to get an optimal approximation $\hat{\mathrm{p}}(\boldsymbol{\beta})$ to the posterior $\mathrm{p}(\boldsymbol{\beta}|\mathbf{y},\hat{\Phi},\hat{\gamma})$. Namely, we do it by minimizing (an upper bound of) the KL divergence between both distributions:

$$\begin{aligned}\mathrm{KL}\left(\hat{\mathrm{p}}(\boldsymbol{\beta})||\mathrm{p}(\boldsymbol{\beta}|\mathbf{y},\hat{\Phi},\hat{\gamma})\right) &= \int\hat{\mathrm{p}}(\boldsymbol{\beta})\log\frac{\hat{\mathrm{p}}(\boldsymbol{\beta})}{\mathrm{p}(\boldsymbol{\beta}|\mathbf{y},\hat{\Phi},\hat{\gamma})}\mathrm{d}\boldsymbol{\beta}\\ &= \log\mathrm{p}(\mathbf{y}|\hat{\Phi},\hat{\gamma}) + \int\hat{\mathrm{p}}(\boldsymbol{\beta})\log\frac{\hat{\mathrm{p}}(\boldsymbol{\beta})}{\mathrm{p}(\mathbf{y},\boldsymbol{\beta}|\hat{\Phi},\hat{\gamma})}\mathrm{d}\boldsymbol{\beta}\\ &\leq \log\mathrm{p}(\mathbf{y}|\hat{\Phi},\hat{\gamma}) - \log F(\mathbf{y},\hat{\Phi},\hat{\gamma},\hat{\boldsymbol{\xi}}) + \mathrm{KL}\left(\hat{\mathrm{p}}(\boldsymbol{\beta})||\mathcal{N}(\boldsymbol{\beta}|\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\Sigma}})\right).\end{aligned}$$

Thus, the minimum is reached for $\hat{\mathrm{p}}(\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\beta}|\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\Sigma}})$, with $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ calculated in eq. (9) using $\hat{\Phi}$, $\hat{\gamma}$, and $\hat{\boldsymbol{\xi}}$.

In summary, at training time, our methods RFF-GPC and VFF-GPC run iteratively until convergence of the hyperparameters Φ, γ, and $\boldsymbol{\xi}$ to their optimal values $\hat{\Phi}$, $\hat{\gamma}$, and $\hat{\boldsymbol{\xi}}$ (see Algorithm 1). The computations involved there suppose a computational complexity of $\mathcal{O}(nD^2+D^3)$ (which equals $\mathcal{O}(nD^2)$ when $n \gg D$), whereas standard GPC scales as $\mathcal{O}(n^3)$. At test time, the probability of class 1 for a previously unseen instance $\mathbf{x}_* \in \mathbb{R}^d$ is:

$$\mathrm{p}(y_*=1) \approx \int\mathrm{p}(y=1|\boldsymbol{\beta},\hat{\Phi},\mathbf{x}_*)\hat{\mathrm{p}}(\boldsymbol{\beta})\mathrm{d}\boldsymbol{\beta} \approx \psi\left(\hat{\mathbf{z}}_*^{\intercal}\hat{\boldsymbol{\mu}}\cdot\left(1+(\pi/8)\hat{\mathbf{z}}_*^{\intercal}\hat{\boldsymbol{\Sigma}}\hat{\mathbf{z}}_*\right)^{-1/2}\right), \tag{12}$$

with ψ being the sigmoid function. Whereas GPC presents a computational cost of $\mathcal{O}(n^2)$ for each test instance, eq. (12) implies a complexity of $\mathcal{O}(D^2)$ in the case of our methods. In particular, notice that this is independent on the number of training instances. These significant reductions

Algorithm 1 Training RFF-GPC and VFF-GPC.

```
Require: Data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \{0,1\}$ and the number $D$ of Fourier frequencies.
  Only for RFF-GPC, sample the Fourier frequencies $\mathbf{w}_i$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and fix them.
  Initialize $\boldsymbol{\xi}^{(1)} = \mathbf{1}_{n\times 1}$, $\gamma^{(1)} = 1$, and $\Phi^{(1)}$. For RFF-GPC (where $\Phi = \sigma$), $\Phi^{(1)}$ is initialized
  as the mean distance between (a subset of) the training data points $\mathbf{x}_i$. For VFF-GPC (where
  $\Phi = (\sigma, \mathbf{W})$), $\sigma$ is initialized as described for RFF-GPC and $\mathbf{W}$ with a random sample from
  $\mathcal{N}(\mathbf{0}, \mathbf{I})$.
  repeat
    Update $\boldsymbol{\xi}^{(k+1)}$ with eq. (10).
    Update $\Phi^{(k+1)}$ and $\gamma^{(k+1)}$ with eq. (11), using $\Phi^{(k)}$ and $\gamma^{(k)}$ as initial values for the conjugate
    gradient method.
  until convergence
  return Optimal hyperparameter $\hat{\Phi}$ and the posterior distribution $\hat{\mathrm{p}}(\boldsymbol{\beta}) = \mathcal{N}(\boldsymbol{\beta}|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$.
```

in computational cost (both at training and test) make our proposal suitable for large scale and real-time applications in general, and in EO applications in particular.

Finally, regarding the convergence of the proposed methods, we cannot theoretically guarantee the convergence to a global optimum (only a local one), since we are using conjugate gradient methods to solve the non-convex optimization problem in eq. (11). However, from a practical viewpoint, we have experimentally checked that both methods have a satisfactory similar convergence pattern. Namely, in the first iterations, the hyperparameters experiment more pronounced changes, widely exploring the hyperparameters space. Then, once they reach a local optimum vicinity, these variations become smaller. Eventually, the hyperparameters values hardly change and the stop criterion is satisfied.

3 Data Collection and Preprocessing

This section introduces the datasets used for comparison purposes in the experiments. We considered (1) a continuous year of MSG data involving several hundred thousands of labeled pixels for cloud classification; and (2) a medium-size manually labeled dataset used to create the operational IASI cloud mask.

3.1 Cloud detection over landmarks with Seviri/MSG

We focus on the problem of cloud identification over landmarks using Seviri MSG data. This satellite mission constitutes a fundamental tool for weather forecasting, providing images of the full Earth disc every 15 minutes. Matching the landmarks accurately is of paramount importance in image navigation and registration models and geometric quality assessment in the Level 1 instrument processing chain. Detection of clouds over landmarks is an essential step in the MSG processing chain, as undetected clouds are one of the most significant sources of error in landmark matching (see Fig. 1).

The dataset used in the experiments was provided by EUMETSAT, and contains Seviri/MSG Level 1.5 acquisitions for 200 landmarks of variable size for a whole year (2010). Landmarks mainly cover coastlines, islands, or inland waters. We selected all multispectral images from a particular

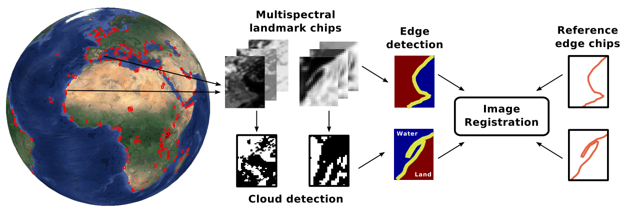


Figure 1: Landmarks are essential in image registration and geometric quality assessment. Misclassification of cloud contamination in landmarks degrades the correlation matching, which is a cornerstone for the image navigation and registration algorithms.

landmark location, Dakhla (Western Sahara), which involves 35,040 MSG acquisitions with a fixed resolution of 20×26 pixels. In addition, Level 2 cloud products were provided for each landmark observation, so the Level 2 cloud mask [Derrien and Le Gléau, 2005] is used as the best available 'ground truth' to validate the results. We framed the problem for this particular landmark as a pixel-wise classification one.

A total amount of 16 features were extracted from the images, involving band ratios, spatial, contextual and morphological features, and discriminative cloud detection scores. In particular, we considered: 7 channels converted to top of atmosphere (ToA) reflectance (R1, R2, R3, R4) and brightness temperature (BT7, BT9, BT10), 3 band ratios, and 6 spatial features. On the one hand, the three informative band ratios were: (i) a cloud detection ratio, $\mathrm{R}_{0.8\mu m}/\mathrm{R}_{0.6\mu m}$; (ii) a snow index, $(\mathrm{R}_{0.6\mu m} - \mathrm{R}_{1.7\mu m})/(\mathrm{R}_{0.6\mu m} + \mathrm{R}_{1.7\mu m})$; and (iii) the NDVI, $(\mathrm{R}_{0.8\mu m} - \mathrm{R}_{0.6\mu m})/(\mathrm{R}_{0.8\mu m} + \mathrm{R}_{0.6\mu m})$. On the other hand, the six spatial features were obtained by applying average filters of sizes 3×3 and 5×5, as well as a standard deviation filter of size 3×3, on both bands R1 and BT9.

Based on previous studies [Derrien and Le Gléau, 2005, Hocking et al., 2010], and in order to simplify the classification task, the different illumination conditions (and hence difficulty) over the landmarks are studied by splitting the day into four ranges (sub-problems) according to the solar zenith angle (SZA) values: *high* ($\mathrm{SZA}<\mathrm{SZA}_{\mathrm{median}}$), *mid* ($\mathrm{SZA}_{\mathrm{median}} < \mathrm{SZA} < 80°$), *low* ($80° < \mathrm{SZA} < 90°$), and *night* ($\mathrm{SZA} > 90°$). Therefore, different classifiers are developed for each SZA range.

The final amount of pixels available for each illumination condition is $n = 1500000$ for *high*, *mid*, and *night*, and $n = 1365083$ for *low*. Moreover, each problem has different dimensionality: all the $d = 16$ features were used for the three daylight problems, and $d = 6$ was used for the night one (some bands and ratios are meaningless at night).

3.2 Cloud detection with the IASI/AVHRR data

The IAVISA dataset is part of the study "IASI/AVHRR Visual Scenes Analysis and Cloud Detection" (http://www.brockmann-consult.de/iavisa-info-web/), whose aim is to improve the IASI cloud detection by optimizing the coefficients used for a predefined set of cloud tests performed in the IASI Level 2 processing. The dataset was derived by visual analysis of globally distributed data, and served as input for the optimization and validation of the IASI cloud detection. Each collected IASI sample was classified concerning its cloudiness based on the visual inspection of the AVHRR Level 1B inside the IASI footprint. Each sample classifies a single IASI instantaneous field of view (IFOV) as being cloud-free (clear sky, 28%), partly cloudy low (26%), partly cloudy

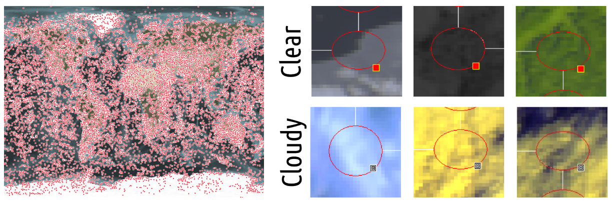


Figure 2: Global sample coverage for all seasons, times of day, and cloud cases (left), and examples of cloud-free and cloudy samples (right).

high (26%), or cloudy (20%). For the sake of simplicity, here we focus on discriminating between cloudy and cloud-free pixels.

In order to ensure the representativeness of the dataset for the natural variability of clouds, labeling further considered additional conditions depending on: 1) the surface type (see Table 1), 2) the climate zone (Köppen classification over land, geographical bands over sea), 3) the season and 4) day/night discrimination. First, the surface type database used as ancillary information was the IGBP (International Geosphere-Biosphere Programme) scene types in the CERES/SARB (Clouds and the Earth's Radiant Energy System, Surface and Atmospheric Radiation Budget) surface map. The 18-class map was used to identify surface properties of a given region. The distribution of the surface types in the map is given in Table 1, showing the even distribution of clouds across land cover types which ensures representativeness (natural variability) of the database. Second, the different climate zones were sampled as follows: tropical ($n = 7499$), dry ($n = 3237$), temperate ($n = 8150$), cold ($n = 2205$), and polar zones ($n = 3832$). Third, seasonality was also taken into account, and yielded the following distribution: Spring ($n = 5862$), Summer ($n = 6930$), Autumn ($n = 6662$), and Winter ($n = 5469$). The global sampling and some cloudy and cloud-free chips are shown in Fig. 2. The final database consists of $n = 24923$ instances and $d = 8461$ original features, which have been summarized to $d = 100$ more informative directions through a standard Principal Component Analysis.

Table 1: The samples distribution per surface types.

ID	Surface Type	Samples
1	Evergreen Needle Forrest	603
2	Evergreen Broad Forrest	876
3	Deciduous Needle Forrest	89
4	Deciduous Broad Forrest	324
5	Mixed Forest	602
6	Closed Shrubs	329
7	Open Shrubs	1484
8	Woody Savannas	768
9	Savannas	656
10	Grassland	886
11	Wetlands	147
12	Crops	1294
13	Urban	44
14	Crop/Mosaic	1436
15	Snow/Ice	1443
16	Barren/Desert	1379
17	Water	12150
18	Tundra	413

4 Experiments

In this section, we empirically demonstrate the performance of the proposed methods in the two real problems described above. Moreover, we carry out an exhaustive comparison to GPC with SE kernel (in the sequel, GPC for brevity).

In order to provide a deeper understanding of our methods, different values of D (number of Fourier frequencies) will be used. Different sizes n of the training dataset will be also considered, in order to analyze the scalability of the methods and to explore the trade-off between accuracy and computational cost. When increasing n (respectively, D), we will add training instances (respectively, Fourier frequencies) to those already used for lower values of n (respectively, D). We provide numerical (in terms of recognition rate), computational (training and test times), and visual (by inspecting classification maps) assessments.

4.1 Cloud detection in the landmarks dataset

In this large scale problem, the number of training examples is selected as $n \in \{10000, 50000, 100000, 300000\}$ for RFF-GPC and VFF-GPC, and $n \in \{5000, 10000, 15000\}$ for GPC. Notice that the improvement at training computational cost ($\mathcal{O}(nD^2)$ for the proposed methods against $\mathcal{O}(n^3)$ for GPC) enables us to consider much greater training datasets for our methods. In fact, as we will see in Figure 3, even with $n = 300000$ RFF-GPC and VFF-GPC are computationally cheaper than GPC with $n = 15000$. Indeed, higher values of n are not considered for GPC to avoid exceeding the (already expensive) 10^6 seconds of training CPU time needed with just $n = 15000$. Regarding the number D of Fourier frequencies, we use $D \in \{10, 25, 50, 100, 150, 200\}$.

The experimental results, which include predictive performance (test overall accuracy), training CPU time, and test CPU time, are shown in Figure 3. Every single value is the average of five independent runs under the same setting. Namely, for each illumination condition, a test dataset is fixed and five different balanced training datasets are defined with the remaining data. Notice that a general first observation across Figure 3 suggests that higher accuracies are obtained for higher illumination conditions (SZA).

Figure 3 reveals an overwhelming superiority of RFF-GPC and VFF-GPC over standard GPC: our proposed methods achieve a higher predictive performance while investing substantially lower training CPU time. This is very clear from the third column plots, where for any blue point we can find orange and yellow points which are placed more north-west (i.e. higher test OA and lower CPU training time). Furthermore, the fourth column shows an equally extraordinary reduction in test CPU time (production time), where the proposed methods are more than 100 times faster than GPC. In particular, this makes RFF-GPC and VFF-GPC better suited than standard GPC for real-time EO applications.

Regarding the practical differences between RFF-GPC and VFF-GPC, we observe that RFF-GPC is faster (at training) whereas VFF-GPC is more accurate. This is a natural consequence of their theoretical formulations: the estimation of the Fourier frequencies $\mathbf{W}$ in VFF-GPC makes it more flexible and expressive, but involves a heavier training. Therefore, in this particular problem, the final practical choice between the two proposed methods would depend on the relative importance that the user assigns to test accuracy (where VFF-GPC stands out) and training cost (where RFF-

GPC does so). In terms of test cost, both methods are very similar, as expected from the identical $\mathcal{O}(D^2)$ theoretical test complexity. The independence of this quantity on n is also intuitively reflected in the experiments, with all the RFF-GPC and VFF-GPC lines collapsing onto a single one in the fourth column plots of Figure 3.

At this point, it is worth to analyze a bit further the role of D in the performance of our methods. Recall (Section 2.1) that RFF-GPC is an approximation to GPC, with an error that exponentially decreases with the ratio D/d between the dimensions of the projected Fourier features space and the original one. Therefore, it is theoretically expected that the performance of RFF-GPC increases with D, becoming equivalent to GPC when $D \to \infty$. Actually, this is supported by the first column of Figure 3. Moreover, since our problem here presents a low d (16 for *high*, *mid*, and *low*, and 6 for *night*), it is natural that RFF-GPC with just $D = 200$ already gets very similar (even better in some cases) results to standard GPC with the same n (yet far much faster, compare GPC and RFF/VFF for $n = 10000$). In the case of VFF-GPC, where the Fourier frequencies are model parameters to be estimated, the number D is directly related to the complexity of the model. Therefore, its increase should not always mean a higher performance in test OA, since large values may provoke over-fitting to the training dataset (this will be clear in the next dataset, whereas it does not occur in LANDMARKS). Furthermore, unlike RFF-GPC, the performance of VFF-GPC is not directly affected by d.

It is also reasonable to expect that both test OA and training CPU time increase with the training dataset size n. More specifically, and from a practical perspective in which the computational resources are finite, the first column in Figure 3 shows that test OA becomes stalled when only one of n or D increases. However, greater improvements in test OA are achieved when n and D are jointly increased. Notice that this is also justifiable from a theoretical viewpoint: the higher the dimensionality of the projected Fourier features space (which is $2D$), the larger number n of examples are required to identify the separation between classes.

4.2 Cloud detection with the IAVISA dataset

As explained in Section 3.2, this problem involves a total amount of $n = 24923$ instances. We performed five-fold cross-validation, which produces five pairs of training/test datasets with (approximately) $20000/5000$ instances each. Results are then averaged. Since RFF-GPC and VFF-GPC are conceived for large-scale applications (they scale linearly with n, recall their $\mathcal{O}(nD^2)$ training cost), they will not be utilized with values of n lower than this training dataset size of ≈ 20000 (even GPC is able to cope with this size). Indeed, in the case of GPC we use the values $n \in \{1000, 5000, 10000, 15000, \mathrm{ALL} \approx 20000\}$. Regarding the number of Fourier frequencies D, we consider the grid $D \in \{1:10, 15:5:25, 50:25:150\}$. The experimental results, which include the same metrics as those used for the previous problem on landmarks, are shown in Figure 4.

In this case, we again observe a clear outperformance of VFF-GPC against GPC: it achieves higher test OA while requiring less training and test CPU times. Moreover, the improvement in test OA is greater than 3%, and train/test CPU times are around 100 and 1000 times lower respectively. However, unlike in the previous problem of cloud detection over landmarks, RFF-GPC does not exhibit such a clear superiority over GPC in this application. Whereas it does drastically decrease the train/test CPU times, it is not able to reach the test OA of GPC with $n = 10000$. Therefore, in practice, the optimal choice for this application is VFF-GPC. RFF-GPC would only be recommended if the training CPU time is a very strong limitation.

The main reason why RFF-GPC is not completely competitive in this problem is its theoretical scope: as an efficient approximation to GPC, it is conceived for large scale applications which are out of the reach of standard GPC. If the size of the problem allows for using GPC (as in this case), then RFF-GPC will only provide a more efficient alternative (less training and test CPU times), but its predictive performance will be always below that of GPC. Moreover, the difference in this performance is directly influenced by the original dimension d of data (recall that the kernel approximation behind RFF-GPC exponentially degrades with d, Section 2.1). This is precisely a second hurdle that RFF-GPC finds in IAVISA: the high $d = 100$ makes RFF-GPC with the full dataset be quite far from the corresponding GPC at predictive performance (test OA). In conclusion, the ideal setting for RFF-GPC is a large scale problem (high n) with few features (low d), precisely the opposite to the IAVISA dataset.

Interestingly, VFF-GPC bypasses these limitations of RFF-GPC by *learning a new kernel and not just approximating* the SE one. First, VFF-GPC is not just a GP adaptation well-suited for large scale applications, but a general-purpose, expressive, and very competitive kernel-based classifier that scales well with the number of training instances. Second, as it does not rely on the kernel approximation, VFF-GPC is not affected by the original dimension d of data. Both ideas are empirically supported by the results obtained in IAVISA.

The first plot of Figure 4 shows that the predictive performance of VFF-GPC does not necessarily improves by increasing D. This is the expected behavior from the theoretical formulation of VFF-GPC, where the Fourier frequencies are D parameters to be estimated. Thus, a higher amount of them confers VFF-GPC a greater flexibility to learn hidden patterns in the training dataset, but also the possibility to over-fit very particular structures of it which do not generalize to the test set. This is the classical problem of the *model complexity* in machine learning, and it is further illustrated in Figure 5. Together with the first plot in Figure 4, it shows the paradigmatic behaviour of train and test performance in presence of over-fitting: train OA grows with the model complexity (great flexibility allows for learning very particular structures of the training set, even reaching a 100% of train OA), whereas test OA initially grows (the first patterns are part of the ground truth and thus general to the test set) but then goes down (when the learned information is too specific to the training set). Notice that this over-fitting phenomena did not occur at LANDMARKS, where test OA monotonically increased with D. In addition to the different nature of the problems, the training dataset size n plays a crucial role at this: smaller datasets (like IAVISA) are more prone to over-fitting than larger ones (LANDMARKS) under the same model complexity.

Finally, it is worth noting that VFF-GPC achieves its maximum test OA when using just $D = 5$ Fourier frequencies. This reflects (i) a not very sophisticated internal structure of the IAVISA dataset (since just 5 directions are enough to correctly classify 85% of the data), and (ii) the VFF-GPC capability to learn those discriminative directions from data. In particular, this shows that VFF-GPC can be used not only as a classifier, but also as a method that learns the most relevant discriminative directions in a dataset. Unfortunately, RFF-GPC is not able to benefit from these privileged directions that may exist in some datasets, since it randomly samples and fix the Fourier frequencies from the beginning.

4.3 Explicit classification maps for cloud detection

The last two sections were dedicated to thoroughly analyze the performance of the proposed methods, empirically understand their behavior, weaknesses, and strengths, and compare them against GPC. In order to illustrate the explicit cloud detection behind the experiments, here we provide several explanatory classification maps obtained by the best model (in terms of predictive performance) for the LANDMARKS dataset: VFF-GPC with $n = 300000$ and $D = 200$.

The classification maps are obtained for the whole year 2010 at the Dakhla landmark, with a total of 34940 satellite acquisitions. The acquired window size is 26×20 pixels. Relying on the proposed feature extraction procedure, we trained the four necessary models (high, mid, low, night), and then proceed to predict over the whole available amount of chips acquired in the 2010 year.[3]

In Figure 6, several chips are provided with the aim of illustrating different behaviors. In the first situation (first row), we can see a characteristic error of the L2 cloud mask, which sometimes tends to wrongly label the coastline pixels as cloudy. However, VFF-GPC leads to a better classification, identifying just one cloudy pixel and thus avoiding this negative coastline effect[4]. In the second row, the visual channels show large clouds crossing the landmark. In the bottom-right of the image, a long cloud is unlabeled in the L2 cloud mask but correctly detected by VFF-GPC. While being formally accounted as an error, such discrepancy is actually positive for our method. Moreover, VFF-GPC shows an interesting cloud-sensitive behaviour at the top-left cloudy mass, identifying a larger cloudy area than that provided by EUMETSAT. This is a desirable propensity in cloud detection applications, where we prefer to identify larger clouds (and then thoroughly analyze them) rather than missing some of them. In the third row, the RGB channel allows for visually identifying three main cloudy masses at the landmark. The L2 mask poorly labels the central cloudy band, and does not detect the lower cloud. Both deficiencies are overcome by VFF-GPC. Finally, the fourth chip shows a huge cloudy mass that is undetected by the L2 mask but is correctly identified by VFF-GPC.

Therefore, although VFF-GPC is trained with an imperfect ground truth, we observe that it is able to bypass some of these deficiencies, and exhibits a desirable cloud-sensitive behavior. This improvement can be also related to the particular design of the training datasets, splitting the problem into four different cases depending on the illumination conditions.

5 Conclusions and Future Work

We presented two efficient approximations to Gaussian process classification to cope with big data classification problems in EO. The first one, RFF-GPC, performs standard GP classification by means of a fast approximation to the kernel (covariance) via D random Fourier features. The advantage of the method is mainly computational, as the training cost is $\mathcal{O}(nD^2)$ instead of the $\mathcal{O}(n^3)$ induced by the direct inversion of the $n \times n$ kernel matrix (the test cost is also reduced to be independent on n, from $\mathcal{O}(n^2)$ to $\mathcal{O}(D^2)$). The RFF method approximates the squared exponential (SE) covariance with Fourier features randomly sampled in the whole spectral domain. The solid theoretical grounds and good empirical performance makes it a very useful method to tackle large

[3]A full video with all the classification maps is available at http://decsai.ugr.es/vip/software.html and http://isp.uv.es/code/vff.html. RFF-GPC and VFF-GPC codes are also provided.

[4]As a clarification note, the coastline pixels were removed from the training dataset by applying a carefully designed morphological filter around coastlines.

scale problems. Actually, the use of RFF has been exploited before in other settings, from classification with SVMs to regression with the KRR. However, we emphasize two main shortcomings. Firstly, the RFF approach can only approximate (theoretically and in practice) a predefined kernel (the SE one in this work). Secondly, by sampling the Fourier domain from a Gaussian, one has no control about the expressive power of the representation since some frequency components of the signal can be better represented than others. As a consequence, the approximated kernel may not have good discrimination capabilities. Noting these two problems, we proposed here our second methodology: a variational GP classifier (VFF-GPC) which goes one step beyond by optimizing over the Fourier frequencies. It is shown to be not just a GP adaptation well-suited for large scale applications, but a whole novel, general-purpose, and very competitive kernel-based classifier that scales well (linearly, as RFF-GPC) with the number of training instances.

We illustrated the performance of the algorithms in two real remote sensing problems of large and medium size. In the first case study, a challenging problem dealt with the identification of clouds over landmarks using Seviri/MSG imagery. The problem involved several hundred thousands data points for training the classifiers. In the second case study, we used the IAVISA dataset, which exploits IASI/AVHRR data to identify clouds with the IASI infrared sounding data. Compared to the original GPC, the experimental results show a high competitiveness in accuracy, a remarkable decrease in computational cost, and an excellent trade-off between both.

These results encourage us to expand the experimentation to additional problems, trying to exploit the demonstrated potential of VFF-GPC when dealing with any value of n (training data set size) and d (original dimension of the data). Other prior distributions and inference methods, as explained at the end of Section 2.1, will be also explored in the future.

References

[Babacan et al., 2012] Babacan, S., Molina, R., Do, M., and Katsaggelos, A. (2012). Bayesian blind deconvolution with general sparse image priors. In *European Conference on Computer Vision (ECCV)*, pages 341–355.

[Bazi and Melgani, 2010] Bazi, Y. and Melgani, F. (2010). Gaussian process approach to remote sensing image classification. *IEEE Trans. Geosc. Rem. Sens.*, 48(1):186–197.

[Benediktsson et al., 2005] Benediktsson, J., Palmason, J., and Sveinsson, J. (2005). Classification of hyperspectral data from urban areas based on extended morphological profiles. *IEEE Trans. Geosc. Rem. Sens.*, 43:480–490.

[Berger et al., 2012] Berger, M., Moreno, J., Johannessen, J. A., Levelt, P., and Hanssen, R. (2012). ESA's sentinel missions in support of earth system science. *Rem. Sens. Env.*, 120:84–90.

[Bishop, 2006] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Camps-Valls and Bruzzone, 2005] Camps-Valls, G. and Bruzzone, L. (2005). Kernel-based methods for hyperspectral image classification. *IEEE Trans. Geosc. Rem. Sens.*, 43(6):1351–1362.

[Camps-Valls and Bruzzone, 2009] Camps-Valls, G. and Bruzzone, L. (2009). *Kernel Methods for Remote Sensing Data Analysis*. John Wiley and Sons.

[Camps-Valls et al., 2004] Camps-Valls, G., Gómez-Chova, L., Calpe, J., Soria, E., Martín, J. D., Alonso, L., and Moreno, J. (2004). Robust support vector method for hyperspectral data classification and knowledge discovery. *IEEE Trans. Geosc. Rem. Sens.*, 42(7):1530–1542.

[Camps-Valls et al., 2008] Camps-Valls, G., Gómez-Chova, L., Muñoz-Marí, J., Rojo-Álvarez, J. L., and Martínez-Ramón, M. (2008). Kernel-based framework for multi-temporal and multi-source remote sensing data classification and change detection. *IEEE Trans. Geosc. Rem. Sens.*, 46(6):1822–1835.

[Camps-Valls et al., 2011] Camps-Valls, G., Tuia, D., GÃ³mez-Chova, L., JimÃ©nez, S., and Malo, J., editors (2011). *Remote Sensing Image Processing*. Morgan & Claypool Publishers, LaPorte, CO, USA.

[Chen et al., 2013a] Chen, K., Jian, P., Zhou, Z., Guo, J., and Zhang, D. (2013a). Semantic annotation of high-resolution remote sensing images via gaussian process multi-instance multilabel learning. *IEEE Geoscience and Remote Sensing Letters*, 10(6):1285–1289.

[Chen et al., 2013b] Chen, K., Zhou, Z., Huo, C., Sun, X., and Fu, K. (2013b). A semisupervised context-sensitive change detection technique via gaussian process. *IEEE Geoscience and Remote Sensing Letters*, 10(2):236–240.

[Chen et al., 2014] Chen, Z., Babacan, S. D., Molina, R., and Katsaggelos, A. K. (2014). Variational bayesian methods for multimedia problems. *IEEE Transaction on Multimedia*, 16(4):1000–10017.

[Damianou, 2015] Damianou, A. (2015). *Deep Gaussian Processes and Variational Propagation of Uncertainty*. PhD thesis, University of Sheffield.

[Derrien and Le Gléau, 2005] Derrien, M. and Le Gléau, H. (2005). Msg/seviri cloud mask and type from safnwc. *International Journal of Remote Sensing*, 26(21):4707–4732.

[Donlon et al., 2012] Donlon, C., Berruti, B., Buongiorno, A., Ferreira, M.-H., Féménias, P., Frerick, J., Goryl, P., Klein, U., Laur, H., Mavrocordatos, C., Nieke, J., Rebhan, H., Seitz, B., Stroede, J., and Sciarra, R. (2012). The Global Monitoring for Environment and Security (GMES) Sentinel-3 mission. *Remote Sensing of Environment*, 120:37–57.

[Drusch et al., 2012] Drusch, M., Del Bello, U., Carlier, S., Colin, O., Fernandez, V., Gascon, F., Hoersch, B., Isola, C., Laberinti, P., Martimort, P., Meygret, A., Spoto, F., Sy, O., Marchese, F., and Bargellini, P. (2012). Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services. *Rem. Sens. Env.*, 120:25–36.

[Fauvel et al., 2008] Fauvel, M., Benediktsson, J. A., Chanussot, J., and Sveinsson, J. R. (2008). Spectral and spatial classification of hyperspectral data using svms and morphological profiles. *IEEE Trans. Geosc. Rem. Sens.*, 46(11):3804–3814.

[Fletcher and Reeves, 1964] Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154.

[Foody and Mathur, 2004] Foody, G. M. and Mathur, J. (2004). A relative evaluation of multiclass image classification by support vector machines. *IEEE Trans. Geosci. Rem. Sens.*, pages 1–9.

[Hocking et al., 2010] Hocking, J., Francis, P. N., and Saunders, R. (2010). Cloud detection in meteosat second generation imagery at the met office. Tech. Rep. 540, Universitat de València.

[Huang et al., 2002] Huang, C., Davis, L. S., and Townshend, J. R. G. (2002). An assessment of support vector machines for land cover classification. *Int. J. Rem. Sens.*, 23(4):725–749.

[Kalantari et al., 2016] Kalantari, L., Gader, P., Graves, S., and Bohlman, S. A. (2016). One-class gaussian process for possibilistic classification using imaging spectroscopy. *IEEE Geoscience and Remote Sensing Letters*, 13(7):967–971.

[Kraft et al., 2013] Kraft, S., Del Bello, U., Drusch, M., Gabriele, A., Harnisch, B., and Moreno, J. (2013). On the demands on imaging spectrometry for the monitoring of global vegetation fluorescence from space. In *Proceedings of SPIE - The International Society for Optical Engineering*, volume 8870.

[Kuss and Rasmussen, 2005] Kuss, M. and Rasmussen, C. E. (2005). Assessing approximate inference for binary gaussian process classification. *J. Mach. Learn. Res.*, 6:1679–1704.

[Labate et al., 2009] Labate, D., Ceccherini, M., Cisbani, A., De Cosmo, V., Galeazzi, C., Giunti, L., Melozzi, M., Pieraccini, S., and Stagi, M. (2009). The PRISMA payload optomechanical design, a high performance instrument for a new hyperspectral mission. *Acta Astronautica*, 65(9-10):1429–1436.

[Lázaro-Gredilla et al., 2010] Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., Figueiras-Vidal, A. R., et al. (2010). Sparse spectrum gaussian process regression. *Journal of Machine Learning Research*, 11(Jun):1865–1881.

[Longbotham et al., 2014] Longbotham, N., Pacifici, F., Baugh, B., and CampsValls, G. (2014). Pre-launch assessment of worldview-3 information content. pages 24–27, Lausanne, Switzerland.

[Melgani and Bruzzone, 2004] Melgani, F. and Bruzzone, L. (2004). Classification of hyperspectral remote sensing images with support vector machines. *IEEE Trans. Geosci. Rem. Sens.*, 42(8):1778–1790.

[Minka, 2001] Minka, T. (2001). Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 362–369, San Francisco, CA. Morgan Kaufmann.

[Muñoz-Marí et al., 2009] Muñoz-Marí, J., Plaza, A., Gualtieri, J., and Camps-Valls, G. (2009). Parallel programming and applications in grid, P2P and networking systems. In Xhafa, F., editor, *Parallel Implementation of SVM in Earth Observation Applications*. IOS Press, UK.

[Pacifici et al., 2009] Pacifici, F., Chini, M., and Emery, W. (2009). A neural network approach using multi-scale textural metrics from very high-resolution panchromatic imagery for urban land-use classification. *Remote Sens. Environ.*, 113(6):1276–1292.

[Pérez-Suay et al., 2017] Pérez-Suay, A., Gómez-Chova, L., Amorós, J., and Camps-Valls, G. (2017). Randomized kernels for large scale earth observation applications. *Remote Sensing of Environment*.

[Plaza et al., 2009] Plaza, A., Benediktsson, J. A., Boardman, J., Brazile, J., Bruzzone, L., Camps-Valls, G., Chanussot, J., Fauvel, M., Gamba, P., Gualtieri, A., and Tilton, J. (2009). Recent advances in techniques for hyperspectral image processing. *Remote Sensing of Environment*, 113(S1):110–122.

[Plaza et al., 2008] Plaza, J., Pérez, R., Plaza, A., Martínez, P., and Valencia, D. (2008). Parallel morphological/neural processing of hyperspectral images using heterogeneous and homogeneous platforms. *Cluster Comput.*, 11:17–32.

[Rahimi and Recht, 2007] Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184.

[Rasmussen and Williams, 2006a] Rasmussen, C. E. and Williams, C. K. I. (2006a). *Gaussian Processes for Machine Learning*. The MIT Press, New York.

[Rasmussen and Williams, 2006b] Rasmussen, C. E. and Williams, C. K. I. (2006b). *Gaussian Processes for Machine Learning*. The MIT Press, New York.

[Roberts et al., 2012] Roberts, D., Quattrochi, D., Hulley, G., Hook, S., and Green, R. (2012). Synergies between VSWIR and TIR data for the urban environment: An evaluation of the potential for the Hyperspectral Infrared Imager (HyspIRI) Decadal Survey mission. *Rem. Sens. Env.*, 117:83–101.

[Rudin, 2011] Rudin, W. (2011). *Fourier analysis on groups*. John Wiley & Sons.

[Ruiz et al., 2014] Ruiz, P., Mateos, J., Camps-Valls, G., Molina, R., and Katsaggelos, A. (2014). Bayesian active remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 52(4):2186–2196.

[Stuffler et al., 2007] Stuffler, T., Kaufmann, C., Hofer, S., Farster, K., Schreier, G., Mueller, A., Eckardt, A., Bach, H., Penné, B., Benz, U., and Haydn, R. (2007). The EnMAP hyperspectral imager-An advanced optical payload for future applications in Earth observation programmes. *Acta Astronautica*, 61(1-6):115–120.

[Tournier et al., 2002] Tournier, B., Blumstein, D., Cayla, F., , and Chalon, G. (2002). IASI level 0 and 1 processing algorithms description. In *Proc. of ISTCXII Conference*.

[Tuia et al., 2009] Tuia, D., Pacifici, F., Kanevski, M., and Emery, W. (2009). Classification of very high spatial resolution imagery using mathematical morphology and support vector machines. *IEEE Trans. Geosci. Rem. Sens.*, 47(11):3866 –3879.

[Yang et al., 2015] Yang, M. Y., Liao, W., Rosenhahn, B., and Zhang, Z. (2015). Hyperspectral image classification using gaussian process models. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 1717–1720.

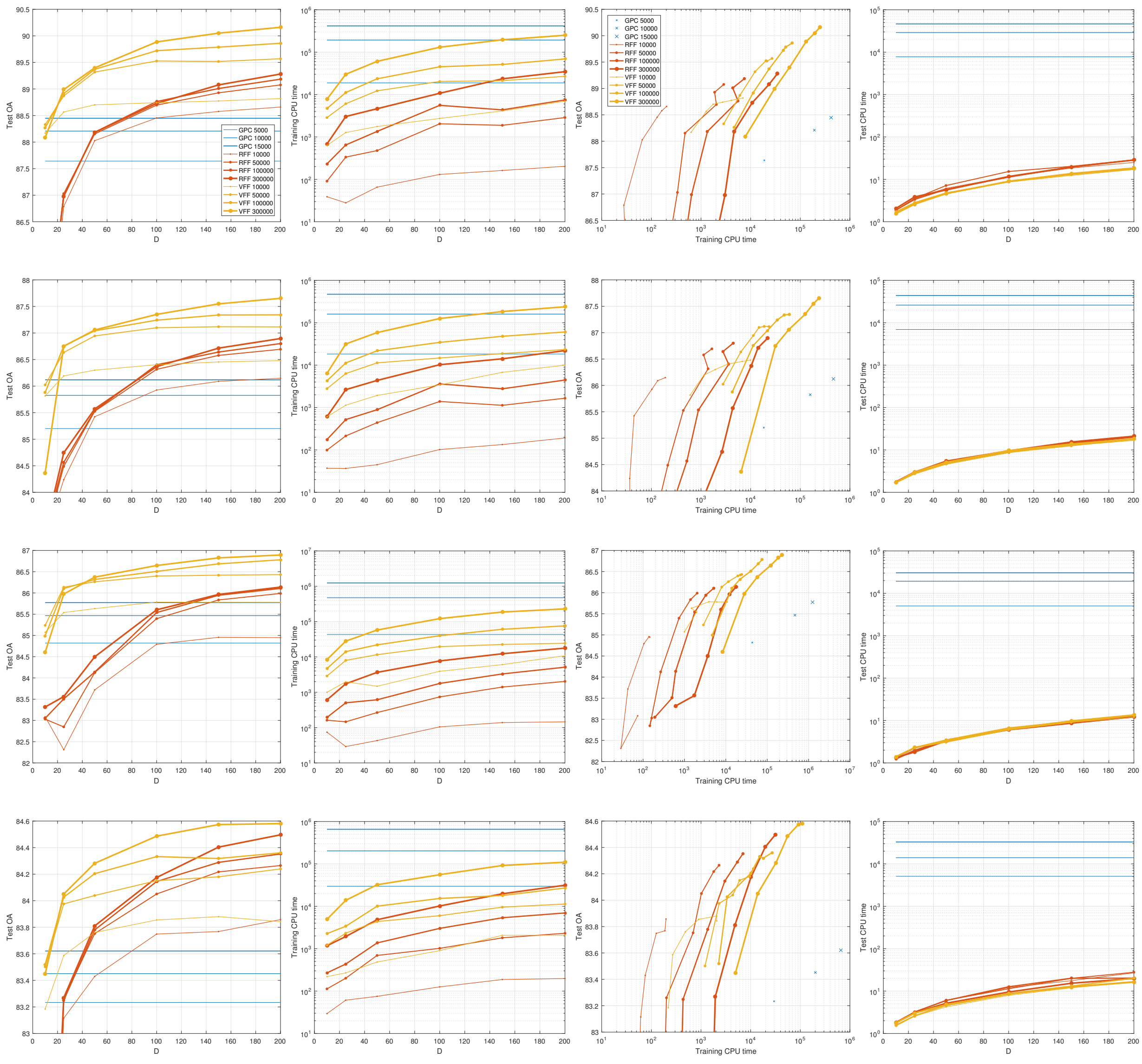


Figure 3: Experimental results for LANDMARKS dataset. From top to bottom, the rows correspond with the described *high*, *mid*, *low*, and *night* illumination conditions. For each row, the first column shows the test overall accuracy (OA) of RFF-GPC, VFF-GPC, and GPC for the different values of n (number of training examples) and D (number of Fourier frequencies) considered. The second column is analogous, but displays the CPU time (in seconds) needed to train each method (instead of the test OA). The third column summarizes the two previous ones, providing a trade-off between test OA and training CPU time. The last column is analogous to the first and second ones, but showing the CPU time used at the test step (production time). The legend for the second and fourth columns is the same as in the first one. However, notice that in the third column plots the GPC lines degenerate into single points (since GPC does not depend on D). In both legends, the numbers indicate the amount n of training examples used, which determines the width/size of the lines/points too. As further explained in the main text, shown results are the mean over five independent runs.

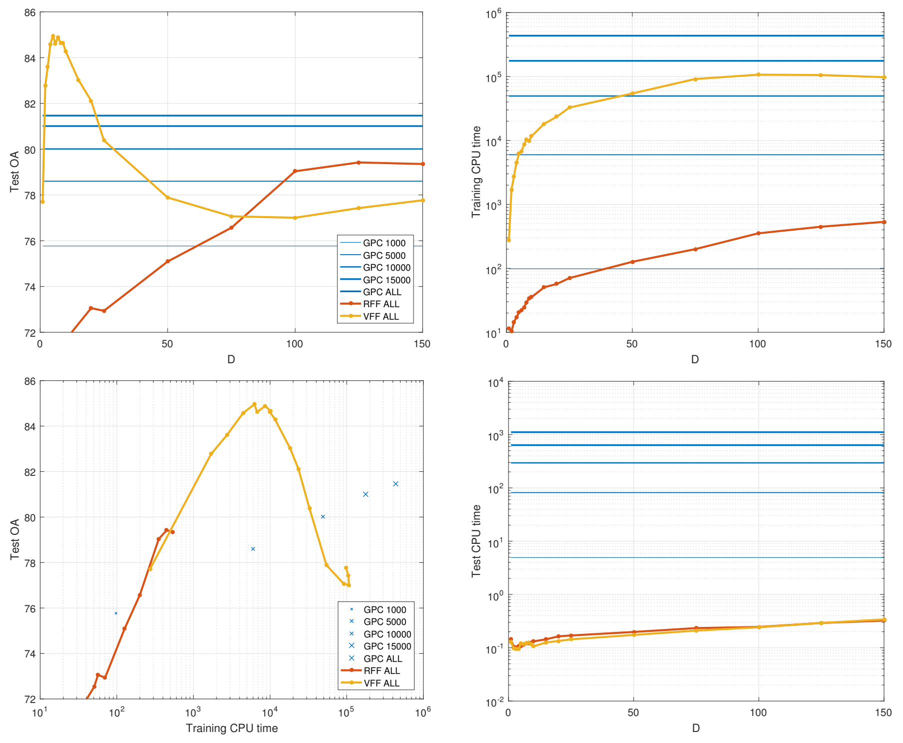


Figure 4: Experimental results for the IAVISA dataset. From left to right and top to bottom, the first plot shows the test overall accuracy (OA) of RFF-GPC, VFF-GPC, and GPC for the different values of n (number of training examples) and D (number of Fourier frequencies) considered. The second column is analogous, but displays the CPU time needed to train each method (instead of the test OA). The third column summarizes the two previous ones, providing a trade-off between test OA and training CPU time. The last column is analogous to the first and second ones, but showing the CPU time used at the test step. The legend for second and fourth plots is the same as the one in the first plot. However, in the third plot the GPC lines degenerate into single points (since GPC does not depend on D). In both legends, the numbers indicate the amount n of training examples used, which determines the width/size of the lines/points too (ALL means the whole training dataset, i.e. $n \approx 20000$). As explained in the main text, the results are the mean over five independent runs.

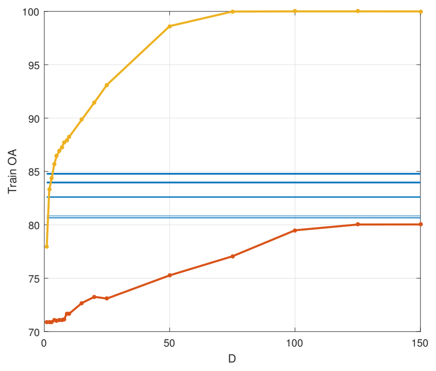


Figure 5: Train OA in the IAVISA dataset for RFF-GPC, VFF-GPC, and GPC with different values of n (number of training examples) and D (number of Fourier frequencies). These results complement the first plot in Figure 4, showing that high values of D make VFF-GPC over-fit to the training dataset. The legend and its interpretation are the same as there.

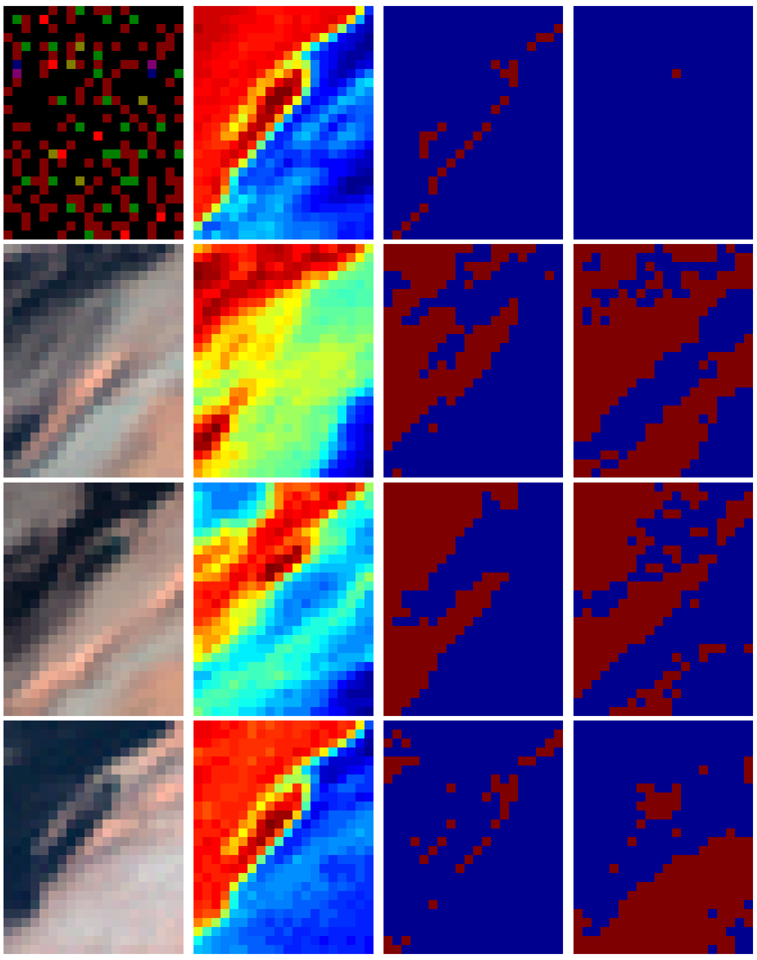


Figure 6: Explicit classification maps for the Dakhla landmark. The rows correspond with four different acquisitions. The first column shows the visible RGB channels (which are not informative for *night* acquisitions such us the first one), the second column is the infrared 10.8μm spectral band (very illustrative in night scenarios), the third column represents the ground truth obtained by EUMETSAT (the L2 cloud mask), and the last one is the VFF-GPC classification map. In the last two columns, the red color is used for cloudy pixels and blue for cloud-free ones.

CHAPTER 4

Variational inference for Gaussian Processes based crowdsourcing

4.1 Publication details

Authors: Pablo Ruiz*, Pablo Morales-Álvarez*, Rafael Molina, Aggelos K. Katsaggelos (* denotes equal contribution)
Title: Learning from crowds with variational Gaussian Processes
Reference: Pattern Recognition 88, 298-311, 2019
Status: Published
Quality indices:

- Impact Factor (JCR 2019): 7.196
- Rank: 12/136 (Q1 and D1) in Computer Science, Artificial Intelligence.

4.2 Main contributions

- We propose *VGPCR* (Variational Gaussian Processes for Crowdsourcing), which resorts to local variational methods to perform inference when modelling the crowdsourcing scenario with Gaussian Processes. The crowdsourcing annotations are modelled through the notions of sensitivity and specificity for each annotator, which describe their reliability when annotating instances from each class (here we focus on binary problems). This approach is presented as an alternative to the use of Expectation Propagation (EP), a more computationally intensive method that was the state-of-the-art for GP-based crowdsourcing.
- The proposed algorithm is evaluated at three different levels: on fully synthetic data, which allows for a controlled supervision of the estimated parameters, on semi-synthetic data, which provides additional insights into the crowdsourcing modelling, and on real data, which includes several crowdsourcing benchmarks on movies reviewing and music genre prediction.

LEARNING FROM CROWDS WITH VARIATIONAL GAUSSIAN PROCESSES

Pablo Ruiz[*]
Dept. of Electrical Engineering and Computer Science
Northwestern University, USA

Pablo Morales-Álvarez[*]
Dept. of Computer Science and AI
University of Granada, Spain

Rafael Molina
Dept. of Computer Science and AI
University of Granada, Spain

Aggelos K. Katsaggelos
Dept. of Electrical Engineering and Computer Science
Northwestern University, USA

ABSTRACT

Solving a supervised learning problem requires to label a training set. This task is traditionally performed by an expert, who provides a label for each sample. The proliferation of social web services (e.g., Amazon Mechanical Turk) has introduced an alternative crowdsourcing approach. Anybody with a computer can register in one of these services and label, either partially or completely, a dataset. The effort of labeling is then shared between a great number of annotators. However, this approach introduces scientifically challenging problems such as combining the unknown expertise of the annotators, handling disagreements on the annotated samples, or detecting the existence of spammer and adversarial annotators. All these problems require probabilistic sound solutions which go beyond the naive use of majority voting plus classical classification methods. In this work we introduce a new crowdsourcing model and inference procedure which trains a Gaussian Process classifier using the noisy labels provided by the annotators. Variational Bayes inference is used to estimate all unknowns. The proposed model can predict the class of new samples and assess the expertise of the involved annotators. Moreover, the Bayesian treatment allows for a solid uncertainty quantification. Since when predicting the class of a new sample we might have access to some annotations for it, we also show how our method can naturally incorporate this additional information. A comprehensive experimental section evaluates the proposed method with synthetic and real experiments, showing that it consistently outperforms other state-of-the-art crowdsourcing approaches.

[*]The first two authors contributed equally.

1 Introduction

The main goal in supervised learning is to find a mapping that predicts labels from features [Bishop, 2006, Murphy, 2012, Watt et al., 2016]. Most of the works in supervised learning assume that training samples have been labeled with no errors by an expert [Das et al., 2018, Ekambaram et al., 2016]. However, the recent advent of social web services has introduced a new approach to address the labeling problem. The term *crowdsourcing* was coined in 2006 by J. Howe [Howe, 2006] to describe *"the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined generally large group of people in the form of an open call"*.

In the last decade, many crowdsourcing services have proliferated in the Internet, where a dataset can be published and millions of people around the world can provide labels in exchange for a reward [Zhang et al., 2016]. Amazon Mechanical Turk (`www.amt.com`), Galaxy Zoo (`www.galaxyzoo.org`), Zooniverse (`www.zooniverse.org`), Crowdflowers (`www.crowdflower.com`) or Clickworker (`www.clickworker.com`) are among the most popular ones. Due to the great number of potential annotators, large data sets can be labeled in a very short time. However, this approach introduces new challenging problems: combining the unknown expertise of annotators, dealing with disagreements on the annotated samples, or detecting the existence of spammer and adversarial annotators [Zhang et al., 2016].

The first paper on crowdsourcing dates back to 1979 [Dawid and Skene, 1979]. Early contributions attempted to estimate the underlying true labels and the reliability of the annotators, but were not conceived to learn a classifier. This idea was explored by Raykar *et al.* [Raykar et al., 2010], who proposed to jointly estimate the coefficients of a logistic regression (LR) classifier and the annotators' expertise. The latter is modeled through the *sensitivity* and *specificity* concepts, which refer to the accuracy of the annotator when labelling instances from each class. Yan *et al.* [Yan et al., 2010] (see also the subsequent journal version [Yan et al., 2014]), introduced a crowdsourcing classifier (also based on LR) which considers a feature-dependent model for the annotators' expertise. The main limitation of these two approaches is the simple LR classification model, which can only deal with linearly separable data. Rodrigues *et al.* [Rodrigues et al., 2014] overcame this problem by introducing a crowdsourcing classifier based on Gaussian Processes (GP) [Rasmussen and Williams, 2006, Ruiz et al., 2016b, Morales-Álvarez et al., 2018]. GP theory makes use of the so-called *"kernel trick"* [Bishop, 2006, Chapter 6] to model complex classification problems where the decision boundary may be non-linear. Expectation Propagation (EP) [Rasmussen and Williams, 2006, Section 3.6] is used as inference procedure in [Rodrigues et al., 2014]. To the best of our knowledge, this is the most recent general-purpose probabilistic crowdsourcing approach (see also [Rodrigues et al., 2017, Section 2.2].)

Nowadays, crowdsourcing is a really active and promising research field, in which these general-purpose crowdsourcing methods are being tailored to a wide range of relevant problems (see the recent survey [Zhang et al., 2016] and related works [Wang et al., 2018, Triguero and Vens, 2016]). Crowdsourcing is being applied to modern areas such as ecological monitoring and conservation [Duyck et al., 2015], plant phenotyping [Giuffrida et al., 2018, Siegel et al., 2018], remote sensing [Fritz et al., 2017], mitosis detection in breast cancer histology images [Albarqouni et al., 2016], topic modeling from crowds [Rodrigues et al., 2017], and detection of glitches in signals acquired by the laureate Laser Interferometer Gravitational-Wave Observatory (LIGO) [Zevin et al., 2017].

Moreover, there exist some recent attempts to combine crowdsourcing with Deep Learning approaches [Albarqouni et al., 2016, Rodrigues and Pereira, 2018], and new challenges, such as the optimal expert validation of the crowdsourced labels [Liu et al., 2017], are emerging.

In this work we address the crowdsourcing classification problem. As in [Rodrigues et al., 2014], the true underlying training labels are modeled as latent variables by means of a GP. A sensitivity-specificity model is used for the annotators (as in [Raykar et al., 2010] and [Rodrigues et al., 2014]). However, there exist two main differences with [Rodrigues et al., 2014]: 1) we use Variational Bayes inference (VB) to estimate all unknowns (instead of EP), and 2) we model sensitivity and specificity as stochastic variables (instead of point parameters). Several reasons motivated our choice of Variational inference. First, it is well-known that the EP iterative procedure does not guarantee convergence, and it may not be able to capture complex posterior distributions (e.g., multi-modal) [Bishop, 2006, Section 10.7]. Second, as it will become clear in the experiments, the EP inference is usually slower in practice (which, in fact, has led to the introduction of some strategies to optimize it [Gerven et al., 2009]). A thorough experimentation (including comparisons with the aforementioned approaches in [Raykar et al., 2010], [Yan et al., 2010] and [Rodrigues et al., 2014] among others) will show that the proposed ideas can contribute to advance the current state-of-the-art in crowdsourcing classification. Moreover, the proposed model naturally lends itself to the integration of annotations that may have been provided for test instances in the prediction of their true class. The experiments will show that, if test annotations are available, this hybrid human-machine prediction is significantly more accurate than the one produced by either the machine or the annotators alone. To the best of our knowledge, this extension had not been addressed in any previous work.

This paper gathers together, clarifies, and significantly extends the ideas in our two conference contributions [Besler et al., 2016, Ruiz et al., 2016a]. The main novelties are: first, sensitivity and specificity are treated as stochastic variables (they are estimated through non-degenerate posterior distributions instead of point estimates). This allows for a better uncertainty quantification and, thus, an enhancement in the experimental results. Second, we show how our model can naturally integrate in the prediction annotations that may have been provided for test instances. If there are no such annotations, the new predictive distribution recovers the old one. Third, the experiments are exhaustively extended in several ways: the new methodology to integrate test set annotations is evaluated, a new type of data popular in crowdsourcing is introduced (*semi-synthetic* data), the computational cost is assessed, and the annotators' expertise estimations are reported in all experiments. Fourth, the experimental section does not restrict itself to the performance of the proposed method, but also examines the behavior of other state-of-the-art approaches that it is compared against. Thus, it can be useful as a brief experimental review of the main current crowdsourcing methods.

The rest of the paper is organized as follows. To facilitate the reading of the paper, an exhaustive glossary of all the symbols used in this work is included in Table 1. Section 2 presents the proposed probabilistic crowdsourcing model based on GP. The VB inference procedure is described in Section 3. The process to classify new samples (including the case when there are test annotations available) is described in Section 4. A comprehensive experimental validation is presented in Section 5. Section 6 concludes the paper and provides some future outlook.

Symbol	Description
$\mathcal{C}_0$ and $\mathcal{C}_1$	Classes 0 and 1, respectively.
N	Number of samples in the training set.
D	Dimension of the feature space.
R	Number of annotators who provided crowdsourcing labels.
$R_n \subseteq \{1,\dots,R\}$	Subset of annotators who labeled the n-th sample.
$N_r \subseteq \{1,\dots,N\}$	Subset of samples labeled by the r-th annotator.
$\mathbf{X} \in \mathbb{R}^{N\times D}$	Matrix containing all the samples in the training set.
$\mathbf{x}_n \in \mathbb{R}^D$	n-th sample of the training set.
$\mathbf{x}_* \in \mathbb{R}^D$	New sample whose class is predicted by the proposed method.
$\mathbf{Y}$	Set of annotations provided by the R annotators.
$y_n^r \in \{0,1\}$	Label provided by the r-th annotator for the n-th sample.
$\mathbf{y}_* = \{y_*^r : r \in R_*\}$	Annotations provided for the new sample $\mathbf{x}_*$.
$\mathbf{z} \in \{0,1\}^N$	Underlying real labels for the training set instances.
$z_n \in \{0,1\}$	Underlying real label for the n-th sample.
$z_* \in \{0,1\}$	Underlying real label for $\mathbf{x}_*$.
$\boldsymbol{\alpha} = [\alpha_1,\dots,\alpha_R]$	Sensitivity of each annotator.
$\boldsymbol{\beta} = [\beta_1,\dots,\beta_R]$	Specificity of each annotator.
a_0^α, b_0^α, a_0^β and b_0^β	Hyperparameters for α and β. (Default: All of them equal to 1.)
$\mathbf{f} = [f_1,\dots,f_N]^T$	GP modeling the relationship between $\mathbf{X}$ and $\mathbf{z}$.
$f_* \in \mathbb{R}$	GP modeling the relationship between $\mathbf{x}_*$ and z_*.
$k(\cdot,\cdot\vert\boldsymbol{\Omega})$	Kernel function depending on a set of parameters $\boldsymbol{\Omega}$.
$\mathbf{K} \in \mathbb{R}^{N\times N}$	Covariance matrix of the prior distribution of $\mathbf{f}$.
$\mathbf{h} \in \mathbb{R}^N$	Vector of prior covariances between f_* and $f_1,\dots,f_n$.
$c \in \mathbb{R}^+$	Prior variance of f_*.
$\boldsymbol{\xi} = (\xi_1,\dots,\xi_N)^T$	Variational parameters to be estimated.
$\boldsymbol{\Lambda} \in \mathbb{R}^{N\times N}$	Diagonal matrix calculated from the variational parameters $\boldsymbol{\xi}$.
m_* and s_*^2	Mean and variance of the approximated posterior distribution of f_*.
$\boldsymbol{\mu}_\mathbf{f}$ and $\boldsymbol{\Sigma}_\mathbf{f}$	Mean and covariance matrix of the posterior distribution of $\mathbf{f}$.
$\delta \in [0,1]$	Classification threshold.
$\boldsymbol{\Theta} = \{\mathbf{f},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\Omega}\}$	Subset of unknown variables of the model.
$\bar{\boldsymbol{\Theta}} = \{\mathbf{z},\mathbf{f},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\Omega}\}$	Set of the unknown variables of the model.
$\bar{\boldsymbol{\Theta}}_\theta = \bar{\boldsymbol{\Theta}} \setminus \theta$	Set $\bar{\boldsymbol{\Theta}}$ minus the element $\theta \in \bar{\boldsymbol{\Theta}}$.
$\sigma(\cdot)$ and $\psi(\cdot)$	Sigmoid and Digamma functions, respectively.
$\mathrm{KL}(\cdot\vert\vert\cdot)$	Kullback-Leibler divergence.
$\mathrm{p}(\cdot)$ and $\mathrm{q}(\cdot)$	Probability distributions: Assumed known (p) and approximated (q).
$\mathbf{0}$ and $\mathbf{1/2}$	Vector with all the components equal to 0 and 1/2, respectively.

Table 1: A comprehensive glossary of all the symbols used in this work.

2 Bayesian Modeling

Let $\mathbf{X} = [\mathbf{x}_1,\dots,\mathbf{x}_N]^T \in \mathbb{R}^{N\times D}$ be a training set of N D-dimensional samples, with unknown labels $\mathbf{z} = (z_1,\dots,z_N)^T \in \{0,1\}^N$. Let us assume there are R different annotators. Let $R_n \subseteq \{1,\dots,R\}$ denote the subset of annotators who labeled the n-th sample, and $N_r \subseteq \{1,\dots,N\}$ the subset of samples labeled by the r-th annotator. Finally, $\mathbf{Y} = \{y_n^r \in \{0,1\} \mid n = 1,\dots,N;\ r \in R_n\}$ is the set of labels provided by the R annotators.

Gaussian Processes (GP) model the relationship between samples $\mathbf{X}$ and the corresponding unknown true labels $\mathbf{z}$ in two steps. First, a set of latent variables $\mathbf{f} = [f_1,\dots,f_N]^T$ following a joint Gaussian distribution $\mathrm{p}(\mathbf{f}|\boldsymbol{\Omega}) = \mathcal{N}(\mathbf{f}|\mathbf{0},\mathbf{K})$ is introduced. The kernel matrix $\mathbf{K} = [k(\mathbf{x}_n,\mathbf{x}_m|\boldsymbol{\Omega})]_{nm}$ is computed with the kernel function k, which defines an inner product in a (possibly infinite-dimensional) transformed space [Bishop, 2006, Chapter 6]. Intuitively, the correlation between each pair of entries of $\mathbf{f}$ is calculated in a transformed space of the original feature space, which allows GP to estimate non-linear decision boundaries. In this work we use the well-known squared exponential (SE) kernel $k(\mathbf{x}_n,\mathbf{x}_m) = \gamma \cdot \exp(-||\mathbf{x}_n - \mathbf{x}_m||^2/(2l^2))$, although other kernels could be used. The kernel hyperparameters $\mathbf{\Omega} = \{\gamma, l\}$ are called *variance* and *length-scale*, respectively.

The second step is to relate the latent variables $\mathbf{f}$ to the unknown true labels $\mathbf{z}$ using a product of Bernoulli distributions:

$$\mathrm{p}(\mathbf{z}|\mathbf{f}) = \prod_{n=1}^{N} \sigma(f_n)^{z_n}(1-\sigma(f_n))^{1-z_n} = \prod_{n=1}^{N} \left(\frac{1}{1+e^{-f_n}}\right)^{z_n}\left(\frac{e^{-f_n}}{1+e^{-f_n}}\right)^{1-z_n}, \tag{1}$$

where the sigmoid function σ maps $\mathbb{R}$ into the interval $(0,1)$. In other words, the sigmoid function assigns the probability of belonging to a class depending on the value of the real number f_n. When $\mathbf{x}_n$ belongs to class 1 ($z_n = 1$), only the first factor is considered and a large positive value is expected for f_n. When $\mathbf{x}_n$ belongs to class 0 ($z_n = 0$), only the second factor appears and a large negative value is expected for f_n. Notice that, although a realization of a GP is a continuous real function on the feature space, the sigmoid function transforms it into a Bernoulli parameter. This is a natural generalization of logistic regression [Bootkrajang and Kabán, 2014]. While logistic regression uses a linear combination of the components of $\mathbf{x}_n$, with linear weights to be estimated, GP uses a linear combination of features in a transformed domain (this transformed domain depends on the kernel used) and denotes by f_n the corresponding linear combination. Moreover, the sigmoid function is an infinitely differentiable function, which allows VB to infer the posterior distribution of the latent variable $\mathbf{f}$.

The distributions $\mathrm{p}(\mathbf{f}|\mathbf{\Omega})$ and $\mathrm{p}(\mathbf{z}|\mathbf{f})$ define a standard GP classifier. Now we need to include the crowdsourcing labelling process in our model. Each annotator r is described by their *sensitivity* $\alpha_r := \mathrm{p}(y^r = 1|z = 1)$ and *specificity* $\beta_r := \mathrm{p}(y^r = 0|z = 0)$. Intuitively, α_r and β_r represent the reliability of the r-th annotator when labeling samples of class $\mathcal{C}_1$ and $\mathcal{C}_0$, respectively. This model is the same as in [Raykar et al., 2010, Rodrigues et al., 2014]. Assuming independence between annotators, we have the following product of Bernoulli distributions

$$\mathrm{p}(\mathbf{Y}|\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{r=1}^{R} \prod_{n \in N_r} \left[\alpha_r^{y_n^r}(1-\alpha_r)^{1-y_n^r}\right]^{z_n} \left[(1-\beta_r)^{y_n^r}\beta_r^{1-y_n^r}\right]^{1-z_n}, \tag{2}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_R)$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_R)$. Some observations are required at this point. First, this sensitivity-specificity model allows for scenarios where annotators might be non-experts. Second, *spammer* (resp. *adversarial*) annotators are those with α_r and β_r values close to (resp. much lower than) 0.5. Third, notice that exchanging the role of z_n and $1-z_n$ in eq. (2) corresponds to exactly the same model but with sensitivities $1-\beta_r$ and specificities $1-\alpha_r$. This means that, changing $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ accordingly, a certain set of underlying true training labels and their opposite are equally plausible. In section 3 we provide an initialization of our algorithm that accounts for this ambiguity.

In this work, sensitivities and specificities are treated as stochastic variables, and Beta distributions are used as hyper-priors. This treatment allows us to introduce prior knowledge about these parameters and weigh more certain configurations of them, yielding more accurate results and a better uncertainty quantification of the model. The independence between annotators yields $\mathrm{p}(\boldsymbol{\alpha}) = \prod_{r=1}^{R} \mathrm{Beta}(\alpha_r|a_0^\alpha, b_0^\alpha)$ and $\mathrm{p}(\boldsymbol{\beta}) = \prod_{r=1}^{R} \mathrm{Beta}(\beta_r|a_0^\beta, b_0^\beta)$,where we have removed the dependency on the parameters for simplicity. Recall that $\mathrm{Beta}(\omega|a,b) \propto \omega^{a-1}(1-\omega)^{b-1}$ with mean $<\omega> = a/(a+b)$. During inference, the following expectations will be required (see

[Bishop, 2006, Exercise 2.11])

$$<\log \omega> = \psi(a) - \psi(a+b), \ <\log(1-\omega)> = \psi(b) - \psi(a+b), \tag{3}$$

where ψ denotes the digamma function. The parameters a and b can be set to introduce prior knowledge about the expected values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and our confidence on them. When no prior knowledge is available, $a = b = 1$ produces uniform distributions. For instance, these hyper-priors on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are useful to deal with annotators who only provide labels for samples in one of the classes (see [Murphy, 2012] for more details about the so-called *"black swan paradox"*).

The proposed joint probabilistic model for the crowdsourcing problem is

$$\mathrm{p}(\mathbf{Y}, \mathbf{z}, \mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Omega}) = \mathrm{p}(\mathbf{Y}, \mathbf{z}, \boldsymbol{\Theta}) = \mathrm{p}(\mathbf{Y}|\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta})\mathrm{p}(\mathbf{z}|\mathbf{f})\mathrm{p}(\mathbf{f}|\boldsymbol{\Omega})\mathrm{p}(\boldsymbol{\alpha})\mathrm{p}(\boldsymbol{\beta})\mathrm{p}(\boldsymbol{\Omega}), \tag{4}$$

where $\boldsymbol{\Theta} = \{\mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Omega}\}$, and $\mathrm{p}(\boldsymbol{\Omega})$ is a flat prior on the kernel parameters $\boldsymbol{\Omega}$. The probabilistic graphical model is depicted in Figure 1. Yellow nodes correspond to observed variables, namely, the set of features $\mathbf{X}$ and the labels provided by the annotators $\mathbf{Y}$ (discrete). The unknown variables, to be inferred during training, are represented using blue nodes, namely, the real labels $\mathbf{z}$ (discrete), the GP latent variable $\mathbf{f}$ (continuous), the GP hyper-parameters $\boldsymbol{\Omega}$ (continuous), and sensitivity and specificity $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ (continuous).

Now that we have the full probabilistic modeling of our problem, let us briefly describe and explain in words its capabilities and limitations. First we utilize a GP (a prior on the set of functions defined over the feature space) which in combination with the sigmoid function is used to describe the real underlying classifier. Since we do not have access to the output of this classifier, the probability distributions of the labels provided by each annotator given the output of the true classifier is modeled using the sensitivity (when the true label is one) and specificity (when the true label is zero) parameters. These numbers quantify how close each annotator's behavior is to the true classifier. Finally, any additional information on each annotator's behavior can be included as prior information on the two aforementioned parameters. In summary, we are using a sound and robust to outliers probabilistic modeling of our crowdsourcing problem

One of the main limitations of the proposed framework is that the only supervised source of information consists in the labels provided by annotators, lacking some mechanism to introduce additional supervised knowledge which may prevail over the annotators. For instance, there may be some instances in the training set for which we know the real label instead of just (noisy) annotations, in which case we would rather rely on this more accurate knowledge. Second, a simple model has been considered for the annotators, based solely on their sensitivity and specificity. More complex (in particular, feature-dependent) behaviors could happen in real-world problems. For instance, there might be annotators who are much more skilled when labelling instances coming from a certain region of the feature space (because they have specialized in that type of instances), but are not that reliable in other regions. Third, there is the implicit assumption that all the annotators are not spammers. Notice that, for the scenario where all the annotators provide random labels for all the instances, there is no information to be able to infer the true decision boundaries, in which case the proposed method cannot train an accurate classifier.

Having explained the model, let us now see how inference is carried out, what problems will be found when estimating the posterior distribution of all the unknowns given the labels provided by the annotators, and how variational inference can be used to solve all of them.

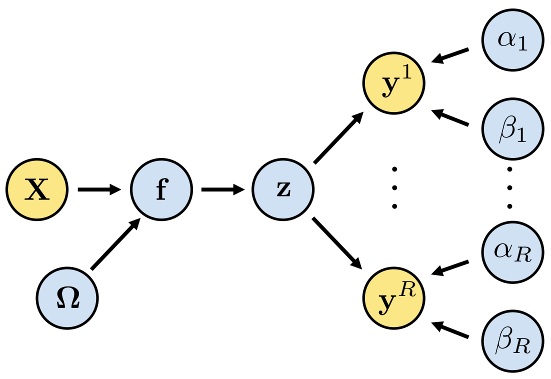


Figure 1: Probabilistic graphical representation for the proposed model. Yellow nodes denote observed variables, and blue nodes unknown variables (to be inferred during training). $\mathbf{Y} = \{\mathbf{y}^1, \dots, \mathbf{y}^R\}$ and $\mathbf{z}$ are discrete variables, whereas the rest are continuous.

3 Variational Bayes Inference

In Bayesian inference, the main goal is to find the posterior distribution $\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y}) = \mathrm{p}(\mathbf{Y}, \mathbf{z}, \boldsymbol{\Theta})/\mathrm{p}(\mathbf{Y})$. This models our certainty about the values of the different model variables once the annotations $\mathbf{Y}$ are observed, and allows us to make predictions on new samples as well as to assess the reliability of the annotators. However, notice that the marginal

$$\mathrm{p}(\mathbf{Y}) = \sum_{\mathbf{z}} \int_{\mathbf{f}} \int_{\boldsymbol{\alpha}} \int_{\boldsymbol{\beta}} \int_{\boldsymbol{\Omega}} \mathrm{p}(\mathbf{Y}, \mathbf{z}, \mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\Omega}) \mathrm{d}\mathbf{f} \mathrm{d}\boldsymbol{\alpha} \mathrm{d}\boldsymbol{\beta} \mathrm{d}\boldsymbol{\Omega} \tag{5}$$

is not tractable, and therefore we resort to the approximated Variational Bayes (VB) inference procedure.

In principle, inference for this model could be also addressed through Markov Chain Monte Carlo (MCMC) methods, which involve sampling from the posterior instead of approximating it with an explicit probability distribution. In fact, MCMC was one of the first approaches for approximate inference in GP [Neal, 1998], and its extension to our model is straightforward from a theoretical viewpoint. However, MCMC methods are computationally expensive. This issue is exacerbated when using GPs, since the large number of latent variables (at least one for each training instance) and the high correlation that may exist between them in the posterior usually requires sophisticated and slow MCMC sampling schemes [Titsias et al., 2011]. Moreover, analytical approximations (such as EP or VB) have obtained excellent results while being significantly faster [Kuss and Rasmussen, 2006].

In order to approximate the posterior $\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y})$, VB minimizes the Kullback-Leibler (KL) divergence with respect to a generic probability distribution $\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})$:

$$\begin{aligned}
\mathrm{KL}(\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta}) || \mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y})) &= \sum_{\mathbf{z}} \int \mathrm{q}(\mathbf{z}, \boldsymbol{\Theta}) \log \frac{\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})}{\mathrm{p}(\mathbf{z}, \boldsymbol{\Theta}|\mathbf{Y})} \mathrm{d}\boldsymbol{\Theta} \\
&= \sum_{\mathbf{z}} \int \mathrm{q}(\mathbf{z}, \boldsymbol{\Theta}) \log \frac{\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})}{\mathrm{p}(\mathbf{Y}, \mathbf{z}, \boldsymbol{\Theta})} \mathrm{d}\boldsymbol{\Theta} + \log \mathrm{p}(\mathbf{Y}).
\end{aligned}$$

The KL divergence between two distributions is always non negative, and is zero if and only if they coincide. Therefore, the optimal distribution $\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})$ in the sense of KL divergence minimization is unique and equals the exact $\mathrm{p}(\mathbf{z}, \mathbf{\Theta}|\mathbf{Y})$. Interestingly, notice that we do not need to know the real posterior $\mathrm{p}(\mathbf{z}, \mathbf{\Theta}|\mathbf{Y})$ to minimize the KL divergence on $\mathrm{q}(\mathbf{z}, \mathbf{\Theta})$: since $\log \mathrm{p}(\mathbf{Y})$ does not depend on $\mathrm{q}(\mathbf{z}, \mathbf{\Theta})$, only the joint distribution in eq. (4) is required.

However, the sigmoids in $\mathrm{p}(\mathbf{z}|\mathbf{f})$ (recall eq. (1)) prevents us from directly evaluating the KL divergence, since their expectation over a Gaussian cannot be obtained in closed-form. To overcome this problem, a variational lower bound for the sigmoid is used [Bishop, 2006, Section 10.6]. Namely, for any $\xi > 0$, we have $\sigma(f) = (1 + \exp(-f))^{-1} \geq \sigma(\xi) \exp\left((f - \xi)/2 - \lambda(\xi)(f^2 - \xi^2)\right)$, where $\lambda(\xi) = (2\xi)^{-1} (\sigma(\xi) - 1/2)$ [Bishop, 2006, Eq. (10.149)]. In our case, this bound yields $\mathrm{p}(\mathbf{z}|\mathbf{f}) \geq \mathbf{H}(\mathbf{z}, \mathbf{f}, \boldsymbol{\xi})$, where

$$\mathbf{H}(\mathbf{z}, \mathbf{f}, \boldsymbol{\xi}) = \prod_{n=1}^{N} \sigma(\xi_n) \exp\left\{ f_n \left(z_n - \frac{1}{2} \right) - \lambda(\xi_n) f_n^2 + \xi_n^2 \lambda(\xi_n) - \frac{\xi_n}{2} \right\}.$$

Plugging this bound into eq. (4), we have the following lower bound for the joint distribution

$$\mathrm{p}(\mathbf{z}, \mathbf{\Theta}, \mathbf{Y}) \geq \mathbf{M}(\mathbf{z}, \mathbf{f}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{\Omega}, \mathbf{Y}, \boldsymbol{\xi}) = \mathrm{p}(\mathbf{Y}|\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \mathbf{H}(\mathbf{z}, \mathbf{f}, \boldsymbol{\xi}) \mathrm{p}(\mathbf{f}|\mathbf{\Omega}) \mathrm{p}(\boldsymbol{\alpha}) \mathrm{p}(\boldsymbol{\beta}) \mathrm{p}(\mathbf{\Omega}),$$

which in turn produces

$$\mathrm{KL}(\mathrm{q}(\mathbf{z}, \mathbf{\Theta}) || \mathrm{p}(\mathbf{z}, \mathbf{\Theta}|\mathbf{Y})) \leq \mathrm{KL}(\mathrm{q}(\mathbf{z}, \mathbf{\Theta}) || \mathbf{M}(\mathbf{z}, \mathbf{\Theta}, \mathbf{Y}, \boldsymbol{\xi})) + \text{const.} \tag{6}$$

Interestingly, notice that $\mathbf{H}(\mathbf{z}, \mathbf{f}, \boldsymbol{\xi})$ is quadratic in $\mathbf{f}$, which allows us to compute the expectation over a Gaussian in closed-form. Therefore, we focus now on minimizing (with respect to $\mathrm{q}(\mathbf{z}, \Theta)$) the analytically tractable right-hand side term in eq. (6), which enforces the left-hand side term (intractable) to be small too. The price for using this bound is a new set of parameters $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)^T$ which need to be estimated.

So far, we have used a generic $\mathrm{q}(\mathbf{z}, \Theta)$ for the approximate posterior distribution. However, VB requires the specification of a particular family, from which the best distribution, in the sense of KL divergence, will be chosen. In this work we use the popular *mean field theory* [Bishop, 2006, Section 10.1], which assumes that the approximated distribution factorizes as $\mathrm{q}(\mathbf{z}, \mathbf{\Theta}) = \mathrm{q}(\mathbf{z})\mathrm{q}(\mathbf{f})\mathrm{q}(\boldsymbol{\alpha})\mathrm{q}(\boldsymbol{\beta})\mathrm{q}(\mathbf{\Omega})$. Let $\bar{\mathbf{\Theta}} = \{\mathbf{z}, \mathbf{\Theta}\}$ be the set $\mathbf{\Theta}$ expanded with the variable $\mathbf{z}$. For $\theta \in \bar{\mathbf{\Theta}}$, let us write $\bar{\mathbf{\Theta}}_\theta = \bar{\mathbf{\Theta}} \backslash \theta$ for the set $\bar{\mathbf{\Theta}}$ minus θ, and $\mathrm{q}(\bar{\mathbf{\Theta}}_\theta) = \prod_{\eta \in \bar{\mathbf{\Theta}}_\theta} \mathrm{q}(\eta)$. Then, for each $\theta \in \bar{\mathbf{\Theta}}$, it can be shown that the distribution $\mathrm{q}(\theta)$ that minimizes the KL-divergence is given by (see [Bishop, 2006, Eq. 10.9] for details)

$$\ln \mathrm{q}(\theta) = <\ln \mathbf{M}(\mathbf{z}, \mathbf{\Theta}, \mathbf{Y}, \boldsymbol{\xi})>_{\mathrm{q}(\bar{\mathbf{\Theta}}_\theta)} + \text{const.} \tag{7}$$

Alternating the estimation of $\mathrm{q}(\mathbf{z})$, $\mathrm{q}(\mathbf{f})$, $\mathrm{q}(\boldsymbol{\alpha})$, $\mathrm{q}(\boldsymbol{\beta})$ and $\mathrm{q}(\mathbf{\Omega})$ leads to an iterative algorithm where the KL divergence decreases after each iteration. Since it is always a non-negative number, the convergence is ensured.

To calculate $\mathrm{q}(\mathbf{z})$, we deduce from eq. (7) that it factorizes as $\mathrm{q}(\mathbf{z}) = \prod_{n=1}^{N} \mathrm{q}(z_n)$. Thus, we can compute each $\mathrm{q}(z_n)$ separately. Since z_n only takes two values, we have

$$\mathrm{q}(z_n = 0) \propto \prod_{r \in R_n} \exp\left\{ y_n^r < \log(1 - \beta_r) > + (1 - y_n^r) < \log \beta_r > \right\}, \tag{8}$$

$$\mathrm{q}(z_n = 1) \propto \exp(<f_n>) \prod_{r \in R_n} \exp\left\{ y_n^r < \log \alpha_r > + (1 - y_n^r) < \log(1 - \alpha_r) > \right\}.$$

For $\mathrm{q}(\mathbf{f})$ we observe that $<\ln \mathbf{M}(\mathbf{z}, \mathbf{\Theta}, \mathbf{Y}, \boldsymbol{\xi})>_{\mathrm{q}(\bar{\mathbf{\Theta}}_\mathbf{f})}$ cannot be calculated. To avoid this problem, we assume that $\mathrm{q}(\mathbf{\Omega})$ is a degenerate distribution. Then, $<\ln \mathbf{M}(\mathbf{z}, \mathbf{\Theta}, \mathbf{Y}, \boldsymbol{\xi})>_{\mathrm{q}(\bar{\mathbf{\Theta}}_\mathbf{f})}$ becomes a

Algorithm 1 VGPCR (Variational GP for CRowdsourcing)

```
Require: $\mathbf{X}, \mathbf{Y}, \boldsymbol{\xi}^0 = \mathbf{1}$, $\mathrm{q}^0(\mathbf{z})$, $k = 0$.
1: repeat
2:     Calculate $\boldsymbol{\Omega}^{k+1}$ as the minimizer of eq. (10) using $\mathrm{q}^k(\mathbf{z})$ and $\boldsymbol{\xi}^k$;
3:     Update $\mathrm{q}^{k+1}(\mathbf{f})$ with eq. (9) using $\mathrm{q}^k(\mathbf{z})$, $\boldsymbol{\xi}^k$, and $\boldsymbol{\Omega}^{k+1}$;
4:     Update $\mathrm{q}^{k+1}(\boldsymbol{\alpha})$ and $\mathrm{q}^{k+1}(\boldsymbol{\beta})$ with eqs. (11)-(12) using $\mathrm{q}^k(\mathbf{z})$ ;
5:     Update $\mathrm{q}^{k+1}(\mathbf{z})$ with eq. (8) using $\mathrm{q}^{k+1}(\mathbf{f})$. The expectations are calculated with eq. (3) using $\mathrm{q}^{k+1}(\boldsymbol{\alpha})$ and $\mathrm{q}^{k+1}(\boldsymbol{\beta})$.
6:     Calculate $\boldsymbol{\xi}^{k+1}$ with eq. (13) using $\mathrm{q}^{k+1}(\mathbf{f})$;
7:     $k = k + 1$;
8: until convergence
9: output $\mathrm{q}(\mathbf{z}, \boldsymbol{\Theta})$
```

quadratic function of $\mathbf{f}$ and, therefore, the approximate posterior $\mathrm{q}(\mathbf{f})$ is a Gaussian $\mathcal{N}(\mathbf{f}|\boldsymbol{\mu}_\mathbf{f}, \boldsymbol{\Sigma}_\mathbf{f})$. Mean and covariance are calculated by taking first and second order derivatives of $\log \mathrm{q}(\mathbf{f})$ to obtain:

$$\boldsymbol{\mu}_\mathbf{f} = \boldsymbol{\Sigma}_\mathbf{f}\left(<\mathbf{z}> - (\mathbf{1/2})\right), \quad \boldsymbol{\Sigma}_\mathbf{f} = \left(\mathbf{K}^{-1} + 2\boldsymbol{\Lambda}\right)^{-1}, \tag{9}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda(\xi_1), \ldots, \lambda(\xi_N))$, and $<\mathbf{z}> = (\mathrm{q}(z_1 = 1), \ldots, \mathrm{q}(z_n = 1))^T$.

Since $\mathrm{q}(\boldsymbol{\Omega})$ is a degenerate distribution, we only need the value of $\boldsymbol{\Omega}$ where $\mathrm{q}(\boldsymbol{\Omega})$ is not zero. For that, we minimize the following objective function

$$\mathcal{L}(\boldsymbol{\Omega}) = \ln|\mathbf{K} + (2\boldsymbol{\Lambda})^{-1}| + \mathbf{u}^T(\mathbf{K} + (2\boldsymbol{\Lambda})^{-1})^{-1}\mathbf{u}, \tag{10}$$

where $\mathbf{u} = (1/2)\cdot\boldsymbol{\Lambda}^{-1}(<\mathbf{z}> - (\mathbf{1/2}))$. Recall also that $\mathbf{K}$ depends on $\boldsymbol{\Omega}$.

To calculate $\mathrm{q}(\boldsymbol{\alpha})$ and $\mathrm{q}(\boldsymbol{\beta})$, we deduce from eq. (7) that both factorize as $\mathrm{q}(\boldsymbol{\alpha}) = \prod_{r=1}^R \mathrm{q}(\alpha_r)$ and $\mathrm{q}(\boldsymbol{\beta}) = \prod_{r=1}^R \mathrm{q}(\beta_r)$. Then we can calculate each $\mathrm{q}(\alpha_r)$ and $\mathrm{q}(\beta_r)$ separately. From eq. (7) we obtain the following Beta distributions:

$$\mathrm{q}(\alpha_r) = \mathrm{Beta}\left[\alpha_r \middle| a_0^\alpha + \sum_{n\in N_r} <z_n> y_n^r,\ b_0^\alpha + \sum_{n\in N_r} <z_n>(1 - y_n^r)\right], \tag{11}$$

$$\mathrm{q}(\beta_r) = \mathrm{Beta}\left[\beta_r \middle| a_0^\beta + \sum_{n\in N_r}(1 - <z_n>)(1 - y_n^r),\ b_0^\beta + \sum_{n\in N_r}(1 - <z_n>)y_n^r\right]. \tag{12}$$

For $\boldsymbol{\xi}$ we maximize $<\ln \mathbf{M}(\boldsymbol{\Theta}, \mathbf{Y}, \mathbf{X}, \boldsymbol{\xi})>_{\mathrm{q}(\bar{\boldsymbol{\Theta}})}$ w.r.t. each ξ_n, which yields

$$\xi_n = \sqrt{<f_n>^2 + \boldsymbol{\Sigma}_\mathbf{f}(n, n)}. \tag{13}$$

The whole estimation procedure is summarized in Algorithm 1. Notice that an initial approximated posterior for the true labels $\mathrm{q}^0(\mathbf{z})$ is required. We propose to initialize it with *soft majority voting*, that is, $\mathrm{q}^0(z_n = 1)$ is the proportion of annotators that assign label 1 to the sample $\mathbf{x}_n$. This initialization implicitly assumes that most of the annotators are not adversarial. Otherwise, due to the ambiguity of eq. (2), we would train a classifier predicting the opposite labels.

4 The predictive distribution

Once the model is trained, we are given a new sample $\mathbf{x}_*$ and we need to predict the probability of each class. In a crowdsourcing problem, we additionally might have access to a set of labels

$\mathbf{y}_* = \{y_*^r \in \{0,1\} : r \in R_* \subseteq \{1,\ldots,R\}\}$ provided by some annotators. To the best of our knowledge, this plausible scenario has not been addressed before in the crowdsourcing literature.

In our model, the prediction can be naturally obtained as the conditional of the hidden label z_* given the observed labels $\mathbf{y}_*$ and $\mathbf{Y}$, that is,

$$\mathrm{p}(z_*|\mathbf{y}_*,\mathbf{Y}) = \sum_{\mathbf{z}} \int \mathrm{p}(z_*,\mathbf{z},f_*,\mathbf{f},\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\Omega}|\mathbf{y}_*,\mathbf{Y})\mathrm{d}\boldsymbol{\Theta}\mathrm{d}f_* \approx \tag{14}$$
$$\text{const}\cdot\left[\int \mathrm{p}(z_*|f_*)\left(\int \mathrm{p}(f_*|\mathbf{f})\mathrm{q}(\mathbf{f})\mathrm{d}\mathbf{f}\right)\mathrm{d}f_*\right]\cdot\left[\int \mathrm{p}(\mathbf{y}_*|z_*,\boldsymbol{\alpha},\boldsymbol{\beta})\mathrm{q}(\boldsymbol{\alpha})\mathrm{q}(\boldsymbol{\beta})\mathrm{d}\boldsymbol{\alpha}\mathrm{d}\boldsymbol{\beta}\right].$$

The GP conditional is $\mathrm{p}(f_*|\mathbf{f}) = \mathcal{N}(f_*|\mathbf{h}^T\mathbf{K}^{-1}\mathbf{f}, c - \mathbf{h}^T\mathbf{K}^{-1}\mathbf{h})$, where $\mathbf{h} = [k(\mathbf{x}_1,\mathbf{x}_*), k(\mathbf{x}_2,\mathbf{x}_*),\ldots,k(\mathbf{x}_N,\mathbf{x}_*)]^T$, $c = k(\mathbf{x}_*,\mathbf{x}_*)$, and $\mathbf{K}$ is the kernel matrix in $\mathbf{X}$. Then, using $\mathrm{q}(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}_\mathbf{f},\boldsymbol{\Sigma}_\mathbf{f})$ (recall eq. (9)) we obtain $\int \mathrm{p}(f_*|\mathbf{f})\mathrm{q}(\mathbf{f})\mathrm{d}\mathbf{f} = \mathcal{N}(f_*|m_*,s_*^2)$, where $m_* = \mathbf{h}^T\mathbf{K}^{-1}\boldsymbol{\mu}_\mathbf{f}$ and $s_*^2 = c - \mathbf{h}^T\left(\mathbf{K} + (2\boldsymbol{\Lambda})^{-1}\right)^{-1}\mathbf{h}$.

Substituting back in eq. (14) and using eq. (4.153) in [Bishop, 2006], we obtain the following predictive distribution for z_*:

$$\mathrm{p}(z_* = 1|\mathbf{y}_*,\mathbf{Y}) \propto \sigma(\kappa(s_*^2)m_*)\prod_{r\in R_*} <\alpha_r>_{\mathrm{q}(\alpha_r)}^{y_*^r}(1 - <\alpha_r>_{\mathrm{q}(\alpha_r)})^{1-y_*^r}, \tag{15}$$

$$\mathrm{p}(z_* = 0|\mathbf{y}_*,\mathbf{Y}) \propto \left(1 - \sigma(\kappa(s_*^2)m_*)\right)\prod_{r\in R_*}(1 - <\beta_r>_{\mathrm{q}(\beta_r)})^{y_*^r}<\beta_r>_{\mathrm{q}(\beta_r)}^{1-y_*^r}, \tag{16}$$

where $\kappa(s_*^2) = (1 + \pi s_*^2/8)^{-1/2}$. Notice that this distribution generalizes the case where no information is provided by the annotators, that is $\mathbf{y}_* = \emptyset$. In such a case, the predictive distribution for z_* is the Bernoulli distribution $\mathrm{p}(z_*|\mathbf{Y}) = [\sigma(\kappa(s_*^2)m_*)]^{z_*}[1 - \sigma(\kappa(s_*^2)m_*)]^{1-z_*}$. Finally, a threshold δ is used on $\mathrm{p}(z_* = 1|\mathbf{y}_*,\mathbf{Y})$ to assign the new sample $\mathbf{x}_*$ to $\mathcal{C}_1$.

In the next section we will compare our novel crowdsourcing method against current state-of-the-art approaches. We will observe that the proposed method stands out as the most robust approach across a wide range of datasets. In particular, we will see that the proposed VB inference is better suited than EP for GP-based crowdsourcing classifiers. Finally, although the main goal is to illustrate the performance of the proposed method, we will also examine the behavior, strengths, and weaknesses of the other methods it is compared with. To some extent, this provides an up-to-date experimental review of the main crowdsourcing approaches in the literature.

5 Experimental Results

In this section, we provide a comprehensive experimentation that compares the proposed method with several state-of-the-art approaches on three different types of datasets. First, we make use of *fully synthetic* data, where crowdsourcing annotations are synthetically generated for an also synthetic underlying classification dataset. This constitutes a completely controlled framework where we can check the expected behavior of the compared algorithms. Second, we evaluate the methods on *semi-synthetic* data, where the underlying classification dataset comes from a real-world problem but the crowdsourcing annotations are obtained synthetically. This is an interesting and popular hybrid setting in crowdsourcing, where we can keep the influence of the real underlying classification dataset apart from the crowdsourced annotations, which remain under control.

Table 2: The three types of data used in this work.

	Fully synthetic	**Semi-synthetic**	**Fully real**
Classif. data set	Synthetic	Real	Real
Annotations	Synthetic	Synthetic	Real
Examples	1D cosine-based	Heart Sonar	Sentence Polarity Music Genre

Third, we evaluate the methods on *fully real* data, where both features and annotations come from a real problem. This is the most realistic setting for practical applications, although we have no knowledge about the data generation process. Table 2 summarizes the types of data used in this work.

The proposed method is referred to as *VGPCR* (Variational GP for CRowdsourcing). In all the experiments, *VGPCR* is compared against the state-of-the-art GP-based crowdsourcing method in [Rodrigues et al., 2014] (*Rodrigues*), which utilizes EP as inference procedure. In the comparison we also include the most straightforward manner to apply a GP to the crowdsourcing setting, *GP-MV*, which consists of a standard GP classifier trained with the Majority Voting (MV) labels. The last GP-based method included in the comparison is a GP classifier trained with the true labels (*GP-GOLD*)[2]. Notice that, intuitively, *GP-GOLD* and *GP-MV* provide (respectively) upper and lower bounds for the GP-based crowdsourcing methods *Rodrigues* and *VGPCR*. Finally, to obtain a more thorough comparison, we include the methods in [Raykar et al., 2010] (*Raykar*), and [Yan et al., 2010] (*Yan*), which are based in LR instead of GP (recall the third paragraph in Section 1).

If the annotators provide labels for the test set (that is, some $\mathbf{y}_*$ are available), then our method is referred to as *VGPCR**. As a baseline, we find interesting to compare *VGPCR** with the most straightforward way to predict with the test set annotations, which we refer to as *MV**, and whose predictions are based only on these annotations (no training step is needed). A brief summary of all the algorithms used in the experiments is provided in Table 3.

The predictive performance of the methods is compared using two popular metrics: the area under the ROC curve (AUC), and the overall accuracy (OA), which is calculated for the threshold $\delta = 1/2$. Moreover, in order to compare the computational cost, the CPU time needed to train each method is also provided.

We implemented *VGPCR*$^{(*)}$, *Raykar*, *Yan*, and GP-classification (necessary for *GP-GOLD* and *GP-MV*) in Matlab©, whereas a Matlab© implementation for *Rodrigues* can be downloaded from his website `http://www.fprodrigues.com`. All the code and datasets are available at `http://decsai.ugr.es/vip/software.html`. The experiments were run on the same machine Intel© Xeon© E5-4640 @ 2.40GHz.

5.1 Fully synthetic data

In this section we compare the performance of the methods with a controlled one-dimensional example. Figure 2a) shows the underlying synthetic classification dataset used. The features are uniformly sampled in the interval $[-\pi, \pi]$. The real labels are assigned according to the sign

[2]Clearly, *GP-GOLD* can only be trained if there are real labels available for the training set. Of course, this is not common in a real crowdsourcing application (otherwise it could be cast as a standard classification problem). However, in the two real datasets used here the true labels are also provided in order to compare with *GP-GOLD*.

Table 3: An overview of the methods compared in the experiments. From top to bottom, the thick horizontal lines separate non-crowdsourcing methods (*GP-GOLD*), crowdsourcing algorithms that do not use test set annotations, and approaches that do use them.

Algorithm	**Description**
GP-GOLD	Intuitive upper bound for the GP-based crowdsourcing methods. Trains a GP with the real labels (it is not a crowsourcing algorithm).
GP-MV	Simplest way to apply GP to crowdsourcing (intuitive lower bound). Trains a GP with the majority voting labels.
Rodrigues	State-of-the-art GP-based crowdsourcing method proposed in [Rodrigues et al., 2014]. EP inference is used.
VGPCR	GP-based crowdsourcing method *proposed here*. Variational inference is used.
Raykar	Based on logistic regression. EM for inference. Proposed in [Raykar et al., 2010]. First probabilistic model for crowdsourcing.
Yan	Based on logistic regression. EM for inference. Proposed in [Yan et al., 2010]. Annotators parameters depend on the instance they label.
*MV**	Simplest (naive) way to use test set annotations for prediction. It does not need a training step.
*VGPCR**	Straightforward extension to *VGPCR*. *Proposed here*. Probabilistically integrates test set annotations in the prediction.

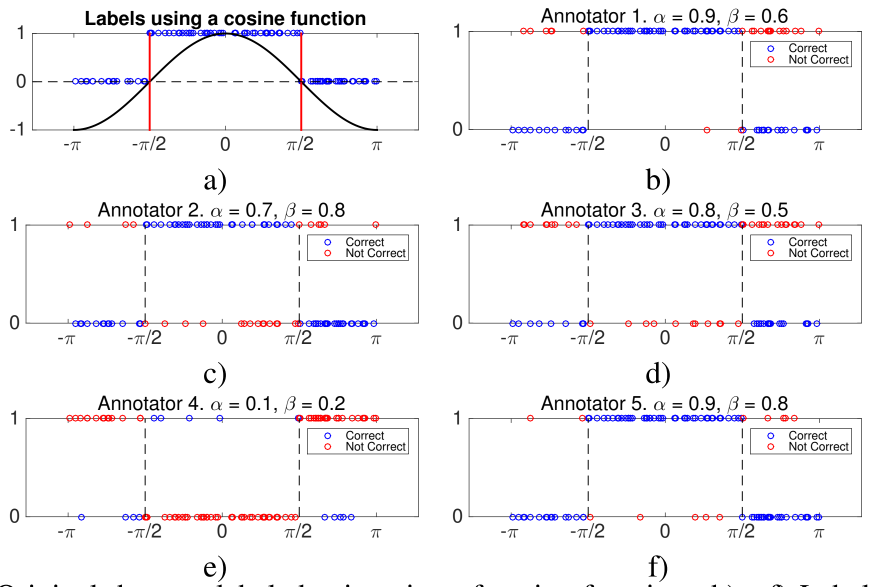


Figure 2: a) Original data set labeled using sign of cosine function. b) - f) Labels provided by annotators 1,2,3,4 and 5 respectively.

of the cosine function on each sample: class $\mathcal{C}_1$ (resp. class $\mathcal{C}_0$) if the cosine is positive (resp. negative). Then, we simulate $R = 5$ annotators by fixing the values of sensitivity and specificity to $\boldsymbol{\alpha} = \{0.9, 0.7, 0.8, 0.1, 0.9\}$ and $\boldsymbol{\beta} = \{0.6, 0.8, 0.5, 0.2, 0.8\}$, respectively. That is, if the true label of the n-th sample is $z_n = 1$ (resp. $z_n = 0$), the r-th annotator assigns it to class $\mathcal{C}_1$ (resp. $\mathcal{C}_0$) with probability α_r (resp. β_r). In Fig. 2 (b-f) we show the labels assigned by each annotator. As expected from the values of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, annotators 1, 2, 3, and 5 make fewer mistakes than annotator 4, who assigns most samples to the opposite class (it has an *adversarial* behavior).

The experiment is repeated 10 times with different training sets of 100 samples (50 of each class). In each realization we also generate a uniformly sampled test set with 200 instances (100 each

Table 4: Predictive performance of the compared methods for the 10 realizations of the fully synthetic experiment. The best mean performance among the crowdsourcing methods that do not use test set annotations (central columns) is bolded.

	GP-GOLD		*GP-MV*		*Rodrigues*		*VGPCR*		*Raykar*		*Yan*		*MV**		*VGPCR**	
Rea.	AUC	OA%	AUC	OA%	AUC	OA%	AUC	OA%	AUC	OA%	AUC	OA%	AUC	OA%	AUC	OA%
1	1.0000	100.00	0.9978	68.00	1.0000	100.00	1.0000	99.50	0.5800	50.00	0.5800	79.00	0.8520	77.00	1.0000	100.00
2	1.0000	98.50	1.0000	84.50	1.0000	98.50	1.0000	98.50	0.4400	50.00	0.5600	57.00	0.8572	78.00	0.9998	99.00
3	1.0000	100.00	1.0000	64.50	1.0000	100	1.0000	99.50	0.4200	50.00	0.7695	75.50	0.7836	76.00	1.0000	100.00
4	0.9998	98.50	0.9987	94.50	1.0000	94.50	1.0000	96.50	0.5000	50.00	0.5000	50.00	0.8357	75.50	1.0000	100.00
5	1.0000	99.00	0.9934	90.50	0.9965	94.50	0.9984	97.00	0.4700	50.00	0.5654	71.00	0.8435	79.00	1.0000	99.50
6	1.0000	100.00	0.9971	91.00	1.0000	100	1.0000	100.00	0.5100	69.00	0.7403	75.00	0.8261	77.00	1.0000	99.00
7	1.0000	99.00	0.9974	96.00	0.9993	98.00	0.9996	98.00	0.5700	76.00	0.6267	71.50	0.8171	76.50	1.0000	99.50
8	0.9990	97.50	0.9928	94.00	0.9972	95.50	0.9983	97.50	0.5500	46.50	0.7435	75.00	0.8436	77.00	0.9997	98.50
9	1.0000	99.50	0.9895	91.00	0.9997	98.00	1.0000	99.50	0.5100	41.50	0.5100	68.00	0.8368	78.50	1.0000	99.50
10	1.0000	98.00	0.9970	78.50	0.9999	98.00	0.9999	96.50	0.4900	50.00	0.7450	73.00	0.8419	79.50	1.0000	99.50
Mean	0.9999	99.00	0.9964	85.25	0.9993	97.70	**0.9996**	**98.25**	0.5040	53.30	0.6340	69.50	0.8337	77.40	1.0000	99.45

class). Moreover, test set annotations are also simulated in order to apply the *MV** and *VGPCR** algorithms.

Table 4 shows the predictive performance of all the methods for the 10 realizations. Let us focus first on the five crowdsourcing algorithms that do not use the test set annotations (i.e., the central columns of the table). The results show two clear groups: those based on GP (*GP-MV*, *Rodrigues*, *VGPCR*), whose results are competitive with *GP-GOLD*, and those that use LR (*Raykar*, *Yan*), whose performance is really poor. This is a reasonable behavior if we take into account that LR decision boundaries are hyperplanes, i.e., one point in this 1-D example. This is clearly insufficient to deal with our training dataset, where $\mathcal{C}_0$ has two disconnected parts with $\mathcal{C}_1$ in the middle (recall fig. 2a)).

Among the LR-based algorithms, we observe that *Yan* performs considerably better than *Raykar*. This means that *Yan*'s feature-dependent model for the annotations is, to some extent, helping to compensate for the insufficient LR model. It is worth noting that the mean result for *Raykar* is hardly above a random guess (around 0.5 of AUC and OA). This LR deficiency is clearly overcome by the GP-based methods, which manage to effectively separate the classes by using a non-linear kernel that allows for more complex decision boundaries (SE kernel in this work, recall Section 2). Among the GP-based methods, the proposed *VGPCR* obtains the best result, followed closely by *Rodrigues*. Notice that *GP-MV* is very close to them in AUC but not in OA, which implies that the threshold $\delta = 1/2$ is not the most appropriate one for class prediction in *GP-MV* (although the classes are well-separated by some other threshold). Nonetheless, this simple 1-D example turns out to be too easy for the GP-based methods, and further differences will be appreciated in subsequent experiments.

It is also interesting to check that, as theoretically intuited, *Rodrigues* and *VGPCR* performances are upper and lower bounded by *GP-GOLD* and *GP-MV*, respectively. Moreover, the differences with *GP-GOLD* are almost insignificant, which means that the crowdsourcing methods are able to extract from the noisy annotations almost the same information as a full GP does from the true labels.

Let us now concentrate on the methods that use test set annotations (*MV** and *VGPCR**). The latter reaches mean AUC and OA of 1.0000 and 99.45% respectively, and manages to totally separate the classes in 8 out of the 10 realizations. These results are better than those obtained by *VGPCR*, which supports the idea that crowdsourcing methods can benefit from the probabilistic integration of test set annotations if available. Moreover, notice that *VGPCR** even outperforms *GP-GOLD*

Table 5: Estimated values of sensitivity and specificity for the five annotators in the fully synthetic experiment. Only those methods that include these parameters in their formulation are shown. The values are the mean over the 10 realizations.

Annotator	Original		*Raykar*		*Rodrigues*		*VGPCR*	
	α	β	α	β	α	β	α	β
1	0.9	0.6	0.8910	0.6340	0.8183	0.5433	0.8993	0.6128
2	0.7	0.8	0.6855	0.8410	0.6058	0.7461	0.6828	0.8119
3	0.8	0.5	0.7916	0.4880	0.7607	0.4487	0.8019	0.4837
4	0.1	0.2	0.1362	0.1793	0.2081	0.2724	0.1015	0.1823
5	0.9	0.8	0.8908	0.8168	0.7986	0.6920	0.9042	0.7816

(and this will be also the case in subsequent experiments). In fact, this is expected when the annotations generation process follows the one proposed in the model (like in the synthetic and semi-synthetic experiments) and the sensitivity and specificity parameters ($\boldsymbol{\alpha}$, $\boldsymbol{\beta}$) are correctly estimated in the training step (as we will check in the following paragraph). In that case, test annotations are a very valuable source of information for *VGPCR**, as they directly depend on the true test label through $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. This has an interesting practical implication in real problems: as long as the crowdsourcing annotation model is representative for the data at hand, it is more useful to collect non-experts opinions for test instances than to collect expert true labels for train ones. Regarding *MV**, its performance is clearly below *VGPCR**. This is reasonable since *MV** does not consider any probabilistic model for the annotations, and thus it is very sensitive to the presence of noise in them (indeed, it performs better as $\boldsymbol{\alpha}, \boldsymbol{\beta} \to 1$, i.e., when the noise decreases and the annotations themselves become very representative of the underlying true labels).

Finally, Table 5 shows the estimated values of sensitivity and specificity for the models that include them in their formulation (i.e., *Rodrigues*, *VGPCR*, and *Raykar*, since *Yan* uses a more complex feature-dependent model). The proposed *VGPCR* method obtains the most accurate estimations: a maximum absolute difference of 0.0184, whereas it is 0.0410 for *Raykar* and 0.1081 for *Rodrigues* (in next Section 5.2 we will analyze the difficulties of *Rodrigues* to estimate $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$). As stated in the previous paragraph, these reliable estimations of $\boldsymbol{\alpha}/\boldsymbol{\beta}$ in *VGPCR* imply that *VGPCR** greatly benefits from test annotations. Moreover, the estimations of *VGPCR* for annotator 4 are quite accurate, which means that it has been able to recognize its adversarial behavior (*Rodrigues* and *Raykar* also detect it, although less accurately, especially *Rodrigues*). Finally, we stress that the poor performance reported for *Raykar* in Table 4 does not come from a wrong estimation of $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$, but from the underlying LR modeling.

For this simple synthetic experiment, CPU training time is not reported, since all values are very similar (there are only 100 training 1-D instances).

5.2 Semi-synthetic data

In this section we follow an analogous experimental approach as before, but focusing on two more complex semi-synthetic datasets. This allows us to gain additional insight into the behavior of the compared methods. In particular, we observe that the proposed method *VGPCR* (and *VGPCR**) stand out as the most effective and robust crowdsourcing approaches across the two experiments.

5.2.1 Heart dataset

This database, also known as *Heart Disease*, is a popular real classification problem donated by the Cleveland Clinic Foundation to the UCI Machine Learning repository, see `http://archive.ics.uci.edu/ml/datasets/Heart+Disease`. The goal is to predict the presence or absence (i.e., binary problem) of heart disease in the patient. For that, it contains 13 relevant explanatory variables (features), such as age, resting blood pressure, and maximum heart rate. After discarding 6 instances with missing features, the final dataset contains 297 samples (137 with disease and 160 without it).

With this real underlying classification problem, we simulate $R = 5$ crowdsourcing annotators with the same sensitivity and specificity values as before, i.e., $\boldsymbol{\alpha} = \{0.9, 0.7, 0.8, 0.1, 0.9\}$ and $\boldsymbol{\beta} = \{0.6, 0.8, 0.5, 0.2, 0.8\}$. Notice that the crowdsourcing setting is very appropriate for this medical domain, where different doctors (annotators) may have different opinions (annotations) about the presence/absence of heart disease based on the 13 provided features. The adversarial behavior of annotator 4 represents the meddling of a non-expert annotator who is confusing both classes. We will see that crowdsourcing methods are able to identify this type of undesirable annotator, and take advantage of their opinions in light of their degree of expertise.

To average the results over different runs, we consider 10 independent random train/test partitions with 208/89 instances respectively. The results are shown in Table 6 and Figure 3. The table contains the AUC and OA mean values for both the test and train datasets. Moreover, it shows the mean CPU time needed to train each method. In the figure we focus on two of these quantities, analyzing the trade-off between generalization capability (in terms of test AUC) and computational cost (in terms of CPU train time). The figure does not include the methods that use test set annotations. Moreover, the figure displays plus/minus one standard deviation of the shown mean quantities. Finally, Table 7 presents the estimated specificity and sensitivity values.

Table 6: Results in the *heart* semi-synthetic dataset. Test and train performances (in terms of AUC and OA) and the CPU time needed to train each method are provided. The results are the mean over the 10 runs. The best generalization (test) performance among the crowdsourcing methods that do not use test set annotations (central rows) is bolded.

Methods	**Test set**		**Train set**		**CPU time (s)**
	AUC	OA%	AUC	OA%	
GP-GOLD	0.8898	81.91	0.9349	86.25	120.65
GP-MV	0.8633	69.33	0.9133	75.91	46.36
Rodrigues	0.8239	78.09	0.9827	93.46	913.06
VGPCR	**0.8870**	**82.02**	0.9298	86.20	29.21
Raykar	0.8853	80.34	0.9287	86.01	0.54
Yan	0.7396	63.37	0.7944	72.69	625.06
*MV**	0.8211	74.04	0.8350	76.88	0
*VGPCR**	0.9921	95.62	0.9935	96.30	29.21

We observe that, among the five crowdsourcing methods, the proposed *VGPCR* gets the best generalization (test) performance in both AUC and OA. In fact, in the latter metric it even outperforms *GP-GOLD*, which is trained with the true labels. This means that our method is making the most of the noisy labels that is provided with, reaching the level of its intuitive upper bound. Table 7 also supports that *VGPCR* is able to accurately figure out the model that generates the annotations. The estimated values for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are very close to the true original ones (better than those obtained by

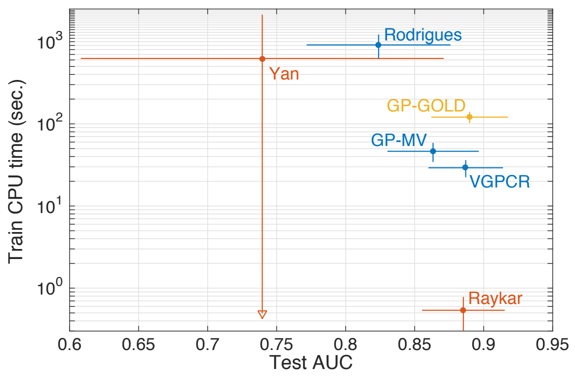


Figure 3: Trade-off between generalization capability (in terms of test AUC) and computational cost (in terms of CPU train time) in the *heart* dataset. For each method, the mean plus/minus one standard deviation is shown. The color indicates the family of the algorithm, i.e. yellow for the non-crowdsourcing method *GP-GOLD*, blue for the GP-based crowdsourcing methods, and red for the LR-based ones. Notice also the logarithmic scale in the y-axis.

Table 7: Estimated values of sensitivity and specificity for the five annotators in the *heart* semi-synthetic experiment. Only those methods that include these parameters in their formulation are shown. The values are the mean over the 10 realizations.

Annotator	Original		*Raykar*		*Rodrigues*		*VGPCR*	
	α	β	α	β	α	β	α	β
1	0.9	0.6	0.8862	0.6070	0.7058	0.4417	0.8901	0.6065
2	0.7	0.8	0.7052	0.7995	0.5144	0.6310	0.7123	0.8021
3	0.8	0.5	0.8151	0.4772	0.7151	0.3821	0.8177	0.4772
4	0.1	0.2	0.1065	0.1789	0.3808	0.4259	0.1006	0.1790
5	0.9	0.8	0.8807	0.7834	0.6271	0.5554	0.8904	0.7877

Rodrigues and similar to *Raykar*'s). In particular, it manages to detect the adversarial behavior of annotator 4. Moreover, it obtains the second shortest CPU train time, 29.21 s.

Regarding its GP-based competitors, the intuitive lower bound *GP-MV* exhibits a worse predictive capacity, as expected. The behavior of *Rodrigues* is, however, more surprising and worth analyzing. *Rodrigues* obtains quite poor test performance, far from *GP-GOLD*, *VGPCR*, and even its supposed lower bound *GP-MV*. The key is given by its performance in the training set. There, we observe that *Rodrigues* is fitting very well the training instances (e.g. 0.9827 of train AUC), much better than the rest of algorithms (even *GP-GOLD*). This is the so-called *over-fitting* problem, which happens when a machine learning method fits the training data too faithfully, at the expense of its generalization capability. It is also worth pointing that *Rodrigues* is the most computationally heavy method (and with a reduced standard deviation, see Figure 3). We will see that this inefficiency of the EP inference is recurrent across all experiments, and in Section 5.3.1 we will analyze it in more detail. In the comparison with *VGPCR*, it is also interesting to note that the α and β estimates provided by *Rodrigues* are clearly less accurate than those obtained by the proposed method, see Table 7. From a practical viewpoint, this means that *Rodrigues* faces difficulties to identify the annotators reliability, which is a very relevant information for the user. We hypothesize that all this enhancement is due to the two main differences between *Rodrigues* and *VGPCR*: the use of variational inference and the more refined modeling of the annotators. In subsequent experiments, we will further support this idea and analyze some other subtle differences in the modeling of *VGPCR* and *Rodrigues*.

Next, let us analyze the results of the LR-based methods. Interestingly, the simpler model of *Raykar* obtains an excellent test performance, very close to the GP-based *VGPCR* and even *GP-GOLD*. This suggests some latent linear structure in the *heart* dataset. Otherwise, as we saw in the fully synthetic experiment, the LR hyperplanes could not produce as good results as the more complex GP boundaries. In order to confirm this linearity in the underlying *heart* dataset, we trained a standard LR classifier with the true labels (following the same scheme as for the GP-based *GP-GOLD*). As expected, the mean test AUC is 0.8884, almost the same as *GP-GOLD* (recall Table 6). Moreover, *Raykar*'s estimations for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are quite accurate (Table 7), and its computational cost is insignificant (it is the fastest method). Therefore, it could be stated that, in this close-to-linear dataset, the LR-based *Raykar* gets the best trade-off between predictive performance and computational cost. Even better than the proposed GP-based *VGPCR* (see Figure 3). This supports the common practice in Machine Learning that, when data is simple, a good model does not need to be a complicated one. However, in the remaining experiments we will find more complex datasets where *Raykar* cannot keep up with the level of the proposed *VGPCR*.

On the other hand, in spite of the aforementioned linearity, *Yan* obtains a very poor (the worst) test performance in this dataset. This must be a consequence of its more complex feature-dependent model for the annotations, which makes the convergence at the training step more challenging. This is reflected in the large standard deviations exhibited by *Yan* in Figure 3, which show that different runs have converged to very different parameters, leading to very heterogeneous results[3]. Moreover, recall that the synthetic generation process used for the annotations does not depend on the features. Therefore, this scenario seems more favorable to *Raykar*, and it will be convenient to compare both methods in the fully real datasets.

Regarding the methods that use test set annotations, the conclusions are the same as in the previous section. Again, *VGPCR** obtains an almost perfect separation between classes, which is mainly caused by the accurate estimation of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (recall Table 7). We also observe that the baseline *MV** is not competitive against *VGPCR**, as it is very sensitive to the noisy labels. Recall that *MV** does not need a training step (therefore, its CPU train time is 0).

5.2.2 Sonar dataset

This database, also known as *Sonar, Mines vs Rocks*, is a real classification problem donated by R.P. Gorman and T.J. Sejnowski to the UCI Machine Learning repository, see `http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks)`. The goal is to distinguish between rocks and mines (metal cylinders) by analyzing the sonar signals bounced off these materials. The features are 60 numbers in the range $[0, 1]$, where each number represents the energy within a particular frequency band. The dataset includes 208 records, 97 samples correspond to rocks and 111 to mines. For this real underlying classification problem, we simulate $R = 5$ crowdsourcing annotators with the same sensitivity and specificity values as before, i.e., $\boldsymbol{\alpha} = \{0.9, 0.7, 0.8, 0.1, 0.9\}$ and $\boldsymbol{\beta} = \{0.6, 0.8, 0.5, 0.2, 0.8\}$.

[3]In particular, we observed that the surprisingly high mean training CPU time for *Yan* is mainly caused by 2 of the 10 runs, which really struggled to converge. Without them, the mean would be 26.14 seconds, more in accordance with the other LR-based *Raykar*.

To average the results over different runs, we consider 10 independent random train/test partitions with 146/62 instances. As in the *heart* dataset, the results are shown in Table 8 and Figure 4. The table contains the AUC and OA mean values in both the test and train datasets. Moreover, it shows the mean CPU time needed to train each method. The figure analyzes the trade-off between generalization capability (test AUC) and computational cost (CPU train time). Finally, Table 9 shows the estimated specificity and sensitivity values.

Table 8: Results in the *sonar* semi-synthetic dataset. Test and train performances (in terms of AUC and OA) and the CPU time needed to train each method are provided. The results are the mean over the 10 runs. The best generalization (test) performance among the crowdsourcing methods that do not use test set annotations (central rows) is bolded.

Methods	**Test set**		**Train set**		**CPU time (s)**
	AUC	OA%	AUC	OA%	
GP-GOLD	0.9043	80.16	0.9901	94.32	98.09
GP-MV	0.7779	60.97	0.8822	71.51	34.05
Rodrigues	0.8574	71.77	0.9843	91.10	153.67
VGPCR	**0.8668**	**74.84**	0.9680	89.59	172.09
Raykar	0.6974	65.65	0.9115	88.08	52.65
Yan	0.6592	57.58	0.7698	75.41	228.72
*MV**	0.8452	77.42	0.8449	77.60	0
*VGPCR**	0.9890	94.19	0.9851	93.15	172.09

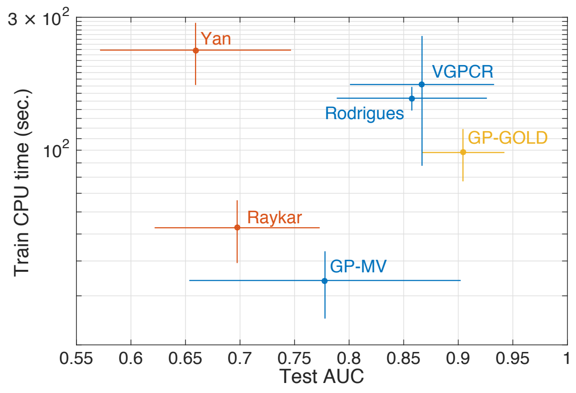


Figure 4: Trade-off between generalization capability (in terms of test AUC) and computational cost (in terms of CPU train time) in the *sonar* dataset. For each method, the mean plus/minus one standard deviation is shown. The color indicates the family of the algorithm, i.e. yellow for the non-crowdsourcing method *GP-GOLD*, blue for the GP-based crowdsourcing methods, and red for the LR-based ones. Notice also the logarithmic scale in the y-axis.

We observe again that *VGPCR* obtains the best generalization performance among the five crowdsourcing methods that do not use test set annotations (in both test AUC and test OA). Moreover, it clearly obtains the most accurate estimations of specificity and sensitivity: the maximum absolute difference in Table 9 is 0.0266 for *VGPCR*, whereas it is 0.1081 for *Raykar* and 0.3039 for *Rodrigues*. The training CPU time is similar to the one obtained by *Rodrigues* (the only crowdsourcing method that is competitive with it in test performance).

Let us analyze the results for the LR-based methods, which will again shed some light on the internal structure of the underlying classification dataset. As opposed to the *heart* dataset, here

Table 9: Estimated values of sensitivity and specificity for the five annotators in the *sonar* semi-synthetic experiment. Only those methods that include these parameters in their formulation are shown. The values are the mean over the 10 realizations.

Annotator	Original		*Raykar*		*Rodrigues*		*VGPCR*	
	α	β	α	β	α	β	α	β
1	0.9	0.6	0.8889	0.7081	0.6768	0.4374	0.8734	0.6244
2	0.7	0.8	0.6786	0.8548	0.4890	0.6221	0.7027	0.8167
3	0.8	0.5	0.8215	0.5296	0.7003	0.3717	0.8051	0.4761
4	0.1	0.2	0.2054	0.2580	0.3922	0.4850	0.0961	0.2079
5	0.9	0.8	0.8359	0.8253	0.5961	0.5334	0.8815	0.7997

the test results for *Raykar* (and *Yan*) are distinctly worse than those for the GP-based methods (specially the more elaborated *VGPCR* and *Rodrigues*, see Figure 4, which clearly shows that blue points are to the right of red ones in the x-axis). This suggests that this *sonar* database is not as linearly-separable as the one before. Again, this can be confirmed by training a standard LR classifier with the true labels. Indeed, it obtains a mean test AUC value of 0.7546, very far from the more complex decision boundary of *GP-GOLD* (0.9043, see Table 8). Notice also that this LR classifier is an intuitive upper bound for the LR-based crowdsourcing methods (just as *GP-GOLD* is for the GP-based ones). This is in accordance with the test AUC values obtained by *Raykar* and *Yan* (0.6974 and 0.6592, respectively), which are below 0.7546. Regarding the comparison between them, *Yan* is again significantly outperformed by *Raykar*, which is also much faster. The justification is as before: *Yan*'s feature-dependent model is too complex for the simple generation process of the annotations, which follows the simpler model of *Raykar*. This makes the convergence more difficult for *Yan*, whereas *Raykar* logically gets pretty good estimations of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ (see Table 9).

The behavior of the GP-based methods is the expected one. Unlike in the *heart* dataset, where it suffered over-fitting, here *Rodrigues* (and also *VGPCR*) exhibit better predictive performance than their intuitive lower bound *GP-MV*. They are also upper bounded by their natural limit *GP-GOLD*. Interestingly, we see that the difference here between *GP-GOLD* and *GP-MV* (in test AUC and OA) is significantly larger than in *heart*. This is connected with the aforementioned non-linearity of the dataset: a close-to-linear boundary can be well identified with low-quality labels, but a complex one needs more accurate data.

The two GP-based methods *VGPCR* and *Rodrigues* present a very similar trade-off between predictive performance and computational cost (Figure 4). However, the estimations of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are much poorer for *Rodrigues*. In principle this is certainly surprising, because the formulas that define $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in this work (recall eqs. (11)-(12)) are the same as in *Rodrigues* (see eqs. (8)-(10) in [Rodrigues et al., 2014])[4]. There are two explanations for this: 1) the treatment of the latent variable $\mathbf{z}$ (which appears in the formulas for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$), and 2) the modeling of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ themselves. The first one is pretty subtle but very relevant, and refers to the fact that $\mathbf{z}$ is integrated out from the beginning in the model of [Rodrigues et al., 2014] whereas it is included in our model as a latent variable. That allows us to compute sounder estimates for $\mathbf{z}$, which is the basis of the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ update formulas. The second one is clearer, as our posterior distributions over $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ account for the uncertainty in the model (whereas the point estimates in [Rodrigues et al., 2014] do not).

[4]More precisely, recall that we model $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as stochastic variables whereas they are treated as parameters in [Rodrigues et al., 2014]. Thus, it is the *mean* of the beta distributions in eqs.(11)-(12) what equals the formulas in [Rodrigues et al., 2014].

Table 10: Examples of positive and negative samples in *Sentence Polarity* dataset.

Sentence	True Label
"An original gem about an obsession with time." "A taut, intelligent psychological drama." "Clever, brutal and strangely soulful movie."	"positive"
"This is amusing for about three minutes." "The film can depress you about life itself." "The pool drowned me in boredom."	"negative"

The conclusions for the methods that make use of the test annotations is the same as in the *heart* dataset: *VGPCR** obtains extraordinarily good results thanks to the accurate estimation of α and β, whereas *MV** is not competitive with it because it does not model the noise in the annotations.

5.3 Fully real data

In this section we compare the performance of *VGPCR* and its competitors on two real crowdsourcing datasets. True labels for the training instances are provided by the datasets contributors. This allows us to compare also with *GP-GOLD*. However, no test annotations $\mathbf{y}_*$ are provided, so *VGPCR** and *MV** are not included in this section. The obtained results support that the novel *VGPCR* is also the most competitive approach in these practical applications.

5.3.1 Sentence Polarity dataset

The *Sentence Polarity* dataset first was presented by Pang and Lee [Pang and Lee, 2005]. It consists of 10427 sentences extracted from movie reviews in "Rotten Tomatoes" website `http://www.rottentomatoes.com/`. The goal is to decide whether a sentence corresponds to a "positive" or "negative" review. In Table 10 we show six sentences in the dataset. Preprocessing and feature extraction were carried out by Rodrigues *et al.* [Rodrigues et al., 2013], which resulted in feature vectors with 1200 components. The dataset is divided into train and test sets, with 4999 and 5428 samples, respectively. To obtain crowdsourcing labels, the train set was made available in Amazon Mechanical Turk. A total amount of 27746 labels were obtained from 203 different annotators.

Results are shown in Table 11 and Figure 5. The table contains the AUC and OA for both test and train datasets. Moreover, it shows the CPU time needed to train each method. The figure analyzes the trade-off between generalization capability (test AUC) and computational cost (CPU train time). Finally, Figure 6 shows the estimated specificity and sensitivity values.

Table 11: Results in the *Sentence Polarity* fully real dataset. Test and train performances (in terms of AUC and OA) and the CPU time needed to train each method are provided. The results are the mean over the 10 runs. The best generalization (test) performance among the crowdsourcing methods is bolded.

Methods	**Test set**		**Train set**		**CPU time (s)**
	AUC	OA%	AUC	OA%	
GP-GOLD	0.8037	73.07	0.9130	83.76	3.3089140×10^4
GP-MV	0.7932	72.03	0.8706	79.22	2.9595670×10^4
Rodrigues	0.7815	72.07	0.9415	89.44	1.0685530×10^4
VGPCR	**0.8000**	**72.53**	0.8861	81.32	3.9638080×10^4
Raykar	0.7141	68.22	0.9100	90.68	1.8156210×10^4
Yan	0.7530	69.45	0.8974	84.28	1.4089233×10^5

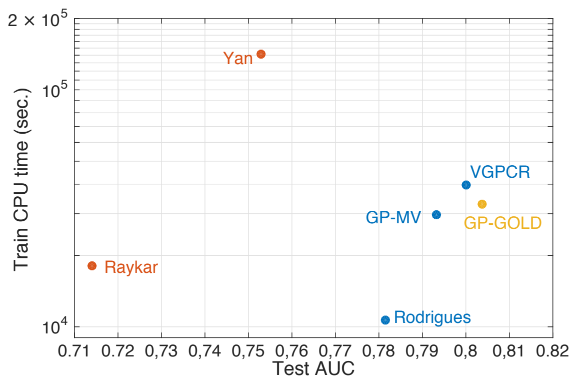


Figure 5: Trade-off between generalization capability (test AUC) and computational cost (CPU train time) in the *Sentence Polarity* dataset. The color denotes the family of the algorithm: yellow for the non-crowdsourcing method *GP-GOLD*, blue for the GP-based crowdsourcing methods, and red for the LR-based ones. Notice the logarithmic scale in the y-axis.

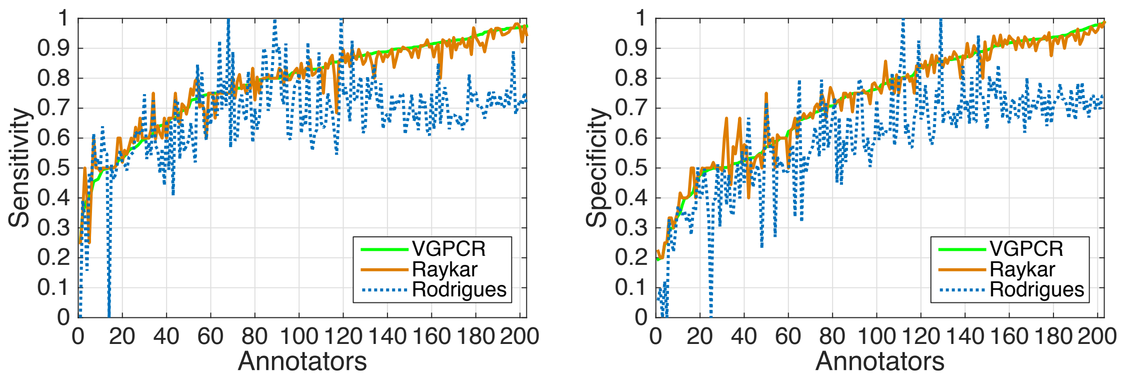


Figure 6: Sensitivity α (left) and specificity β (right) estimations for the 203 annotators in the *Sentence Polarity* dataset. Only those methods that include these parameters in their formulation are shown. For a clearer display, in each figure the annotators are arranged in ascending order of the *VGPCR* estimated value.

Again, *VGPCR* is the best crowdsourcing method in terms of predictive performance (0.8 of AUC and 72.53% of OA). These values place the proposed method really close to its natural upper bound *GP-GOLD* (0.8037 of AUC and 73.07% of OA). As expected, it is also lower bounded by *GP-MV*.

It is important here to analyze the behavior of *Rodrigues*. Although it is below its intuitive lower bound *GP-MV* (it is clearly suffering from over-fitting, see its high training performance), its generalization capability is not very far from *VGPCR*, and it seems that it might be the method of choice in certain applications because it is around four (resp. three) times faster than *VGPCR* (resp. *GP-MV*). This low computational cost (the lowest in this dataset) seems certainly surprising, since the EP inference is quite expensive (as both semi-synthetic experiments have shown). The key is that, in this application, the code provided by the authors fixes the kernel hyperparameters from the beginning[5], and they are not estimated during training (which is the most time-consuming

[5]Specifically, the length-scale l is fixed to 1.5 and the variance γ to 1.3 (recall Section 2).

step). However, the other two GP-based methods do estimate them. Therefore, the CPU training costs should not be compared. If we fix the kernel hyperparameters of *VGPCR* to its previously estimated values, then its CPU training time falls down to 2.1312×10^3 seconds (around 5 times less than *Rodrigues*), whereas its predictive performance remains unchanged.

The problem is that, because of the EP inference procedure, estimating the kernel hyperparameters with *Rodrigues* in this dataset (4999 training instances) is computationally prohibitive. Indeed, Figure 7 shows the experimental CPU time needed to train *Rodrigues* (including the hyperparameters estimation) with increasingly larger subsets of the original set. The rapid growth makes training with $n = 4999$ instances infeasible. Interestingly, the theoretical complexity of each EP iteration is $\mathcal{O}(n^3)$, the same as the variational inference used here. However, EP usually requires many more iterations for convergence, and that makes it computationally heavier in practice.

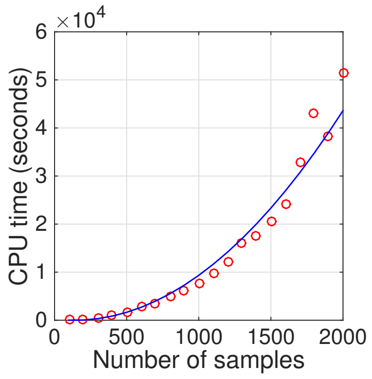


Figure 7: Mean CPU time in three runs of *Rodrigues*. The x-axis shows the number of training samples.

Regarding the estimation of specificity and sensitivity, Figure 6 shows very similar estimations for *VGPCR* and *Raykar*, whereas *Rodrigues* deviates from this common tendency. This is in accordance with all the previous experiments, where *Rodrigues* estimations were less reliable. The reasons behind this were analyzed in Section 5.2.2.

Figure 5 shows a clear separation between GP- and LR-based methods in the x-axis (i.e., the generalization capability). As explained in Section 5.2, this may reveal a non-linear underlying structure in the dataset.

Finally, as opposed to the semi-synthetic datasets, notice that *Yan* significantly outperforms *Raykar* here. This is in accordance with the fact that the annotations generation process does not necessarily imitate *Raykar*'s one in this real dataset, and the feature-dependent model of *Yan* seems to adapt well. However, this is at the expense of a really heavy training step, being the only method (together with Rodrigues) beyond 10^5 seconds of CPU train time.

5.3.2 Music Genre dataset

In this experiment we use the Music Genre dataset presented in [Tzanetakis and Cook, 2002], which consists of 1000 fragments (30 secs. length) of songs. The goal is to distinguish between 10 music genres: *classical, country, disco, hiphop, jazz, rock, blues, reggae, pop*, and *metal*. We use an *one-vs-all* strategy to address this multi-class classification problem, and the results are averaged over the 10 experiments.

For preprocessing and feature extraction, the authors in [Rodrigues et al., 2013] used Marsyas music information tool (`http://marsyas.info/`) to extract 124 features from the original dataset. These features include relevant technical metrics such us means and variances of timbral features, time-domain zero-crossings, spectral centroid, rolloff, flux, and Mel-Frequency Cepstral Coefficients (MFCC).

Table 12: Results in the *Music Genre* fully real dataset. Test and train performances (in terms of AUC and OA) and the CPU time needed to train each method are provided. The results are the mean over the 10 runs. The best generalization (test) performance among the crowdsourcing methods is bolded.

Methods	**Test set**		**Train set**		**CPU Time (s)**
	AUC	OA%	AUC	OA%	
GP-GOLD	0.9426	94.60	0.9713	95.69	3.283342×10^3
GP-MV	0.8865	91.50	0.8809	91.89	2.170080×10^3
Rodrigues	0.8795	85.43	0.9429	92.16	6.520268×10^3
VGPCR	**0.9152**	**92.70**	0.9259	93.70	1.712601×10^3
Raykar	0.8806	90.40	0.9414	95.84	7.201810×10^2
Yan	0.8614	91.90	0.8913	93.96	1.088944×10^3

The dataset contains 100 samples from each genre, which were randomly divided in 70 samples for training and 30 for testing. Crowdsourcing labels were obtained with Amazon Mechanical Turk. Each annotator listened to a subset of fragments and labeled them as one of the ten genres listed above. A total amount of 2945 labels were provided by 44 different annotators.

The results are shown in Table 12 and Figure 8. The table contains the AUC and OA for both test and train datasets. Moreover, it shows the CPU time needed to train each method. The figure analyzes the trade-off between generalization capability (test AUC) and computational cost (CPU train time). Finally, Figure 9 shows the estimated specificity and sensitivity values.

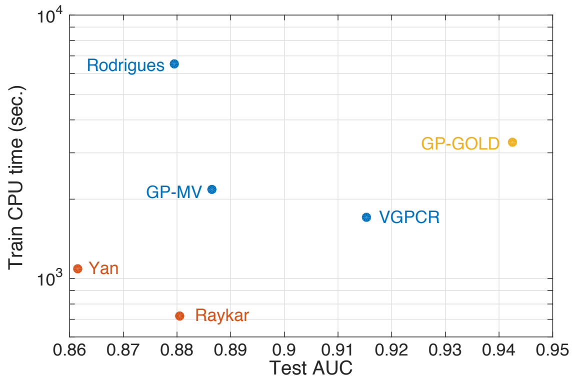


Figure 8: Trade-off between generalization capability (test AUC) and computational cost (CPU train time) in the *Music Genre* dataset. The color indicates the family of the algorithm: yellow for the non-crowdsourcing method *GP-GOLD*, blue for the GP-based crowdsourcing methods, and red for the LR-based ones. Notice the logarithmic scale in the y-axis.

Once more, the novel *VGPCR* exhibits the best generalization capability, keeping a considerable distance with the next one (*GP-MV*). Moreover, *VGPCR* is also the fastest among the GP-based methods. This implies an unbeatable trade-off in Figure 8. Furthermore, as theoretically expected, its performance lies between that of *GP-GOLD* and *GP-MV*.

As opposed to the previous experiment, *Rodrigues* is now the most computationally expensive method (around three times more than the next one, *GP-MV*). This difference is due to the fact that the kernel hyperparameters are estimated during the training step. Test performance for *Rodrigues* is clearly below *VGPCR*, being only competitive with *GP-MV*. This is due to over-fitting (see

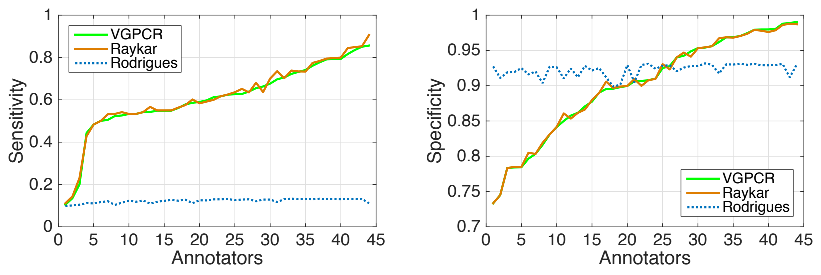


Figure 9: Sensitivity $\boldsymbol{\alpha}$ (left) and specificity $\boldsymbol{\beta}$ (right) estimations for the 44 annotators in the *Music Genre* dataset. Only those methods that include these parameters in their formulation are shown. For a clearer display, in each figure the annotators are arranged in ascending order of the *VGPCR* estimated value.

its high training performance in comparison with the test one), and the very poor estimation of sensitivity/specificity (see Figure 9). In turn, as explained in Section 5.2, these follow from a less subtle modeling (marginalization of $\mathbf{z}$, point estimates for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) and the use of a different inference approach.

Once more, Figure 9 shows similar estimates for *VGPCR* and *Raykar*, whereas *Rodrigues* exhibits a quite bizarre behavior with almost constant estimates. This definitely confirms its difficulties for calculating $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. In this particular case, the problem may come from the unbalanced setting (recall the *one-vs-all* strategy, which implies a 90%-10% balance between negative and positive classes). In fact, *Rodrigues* is the only method with test OA below 90%, which would be the OA for a naive classifier that assigns every instance to the majority class.

In the comparison between GP- and LR-based methods, the x-axis of Figure 8 does not show a clear separation in predictive performance. This suggests that linear boundaries may be representative for the classes of this set. Indeed, notice that both families are much better separated in the y-axis. Interestingly, this is precisely connected with the aforementioned underlying linear structure, which allows for a fast convergence of the LR-based methods. The same behavior could be appreciated in the close-to-linear *heart* set, recall Figure 3.

Regarding the LR-based methods, *Raykar* and *Yan* obtain similar results (the former performs better with respect to AUC and the latter with respect to OA). However, the complex modeling of *Yan* makes it computationally heavier, and thus less competitive in practice.

6 Conclusions

We have introduced a new crowdsourcing classification methodology. As previous approaches, it is based on a Gaussian Process classifier, which allows for the description of complex data. However, a novel Variational Bayes (VB) inference procedure is proposed here (instead of Expectation Propagation, EP). The modeling of the annotators is also refined with respect to previous GP-based methods: the level of expertise is treated as a stochastic variable, and the underlying true training labels $\mathbf{z}$ are not marginalized out from the model. Moreover, the proposed method allows for integrating in the prediction (possibly non-expert) annotations that may have been provided for test instances.

The experimental results have shown that the novel VB-based approach is really competitive and robust across very different types of datasets, ranking always first among its competitors in terms of predictive performance. On the contrary, the EP-based method has suffered over-fitting in three out of the five datasets used. The computational cost of the proposed method is competitive with the rest of the crowdsourcing classifiers (and considerably lower than the EP-based one). Our refined model for the annotators is also reflected in the experiments. Indeed, our sensitivity-specificity estimations are significantly more accurate than those by the EP-based approach. If there are test annotations available (which is not always possible, see the fully real datasets used here), we have seen that the proposed method largely benefits from its probabilistic integration within the model. It would be interesting to study to what extent this generalizes to fully real datasets, where the annotations generation process does not necessarily follow the one proposed in the model. Other lines of future work are i) development of alternative and more accurate feature-dependent crowdsourcing models, ii) a probabilistic multi-class generalization of the proposed model, and iii) extension of GP-based crowdsourcing methods to large-scale datasets.

References

[Albarqouni et al., 2016] Albarqouni, S., Baur, C., Achilles, F., Belagiannis, V., Demirci, S., and Navab, N. (2016). Aggnet: Deep Learning from crowds for mitosis detection in breast cancer histology images. *IEEE T. Med. Imaging*, 35(5):1313–1321.

[Besler et al., 2016] Besler, E., Ruiz, P., Molina, R., and Katsaggelos, A. K. (2016). Classification of multiple annotator data using variational Gaussian process inference. In *EUSIPCO*, pages 2025–2029.

[Bishop, 2006] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York, NJ, USA.

[Bootkrajang and Kabán, 2014] Bootkrajang, J. and Kabán, A. (2014). Learning kernel logistic regression in the presence of class label noise. *Pattern Recogn.*, 47(11):3641 – 3655.

[Das et al., 2018] Das, S., Datta, S., and Chaudhuri, B. B. (2018). Handling data irregularities in classification: Foundations, trends, and future challenges. *Pattern Recogn.*, 81:674 – 693.

[Dawid and Skene, 1979] Dawid, A. and Skene, A. (1979). Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Appl. Stat.-J. Roy. St.*, 28(1):20.

[Duyck et al., 2015] Duyck, J., Finn, C., Hutcheon, A., Vera, P., Salas, J., and Ravela, S. (2015). Sloop: A pattern retrieval engine for individual animal identification. *Pattern Recogn.*, 48(4):1059–1073.

[Ekambaram et al., 2016] Ekambaram, R., Fefilatyev, S., Shreve, M., Kramer, K., Hall, L. O., Goldgof, D. B., and Kasturi, R. (2016). Active cleaning of label noise. *Pattern Recogn.*, 51:463 – 480.

[Fritz et al., 2017] Fritz, S., See, L., Perger, C., McCallum, I., Schill, C., Schepaschenko, D., Duerauer, M., Karner, M., Dresel, C., Laso-Bayas, J.-C., Lesiv, M., Moorthy, I., Salk, C. F., Danylo, O., Sturn, T., Albrecht, F., You, L., Kraxner, F., and Obersteiner, M. (2017). A global dataset of crowdsourced land cover and land use reference data. *Scientific Data*, 4:170075.

[Gerven et al., 2009] Gerven, M. V., Cseke, B., Oostenveld, R., and Heskes, T. (2009). Bayesian source localization with the multivariate Laplace prior. In *NIPS*, pages 1901–1909.

[Giuffrida et al., 2018] Giuffrida, M. V., Chen, F., Scharr, H., and Tsaftaris, S. A. (2018). Citizen crowds and experts: observer variability in image-based plant phenotyping. *Plant Methods*, 14(1):12.

[Howe, 2006] Howe, J. (2006). The rise of crowdsourcing. *Wired magazine*, 14(6):1–4.

[Kuss and Rasmussen, 2006] Kuss, M. and Rasmussen, C. E. (2006). Assessing approximations for Gaussian process classification. In *NIPS*, pages 699–706.

[Liu et al., 2017] Liu, M., Jiang, L., Liu, J., Wang, X., Zhu, J., and Liu, S. (2017). Improving Learning-from-Crowds through Expert Validation. In *IJCAI*, pages 2329–2336.

[Morales-Álvarez et al., 2018] Morales-Álvarez, P., Pérez-Suay, A., Molina, R., and Camps-Valls, G. (2018). Remote sensing image classification with large-scale Gaussian processes. *IEEE T. Geoscie. Remote.*, 56(2):1103–1114.

[Murphy, 2012] Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective*. MIT.

[Neal, 1998] Neal, R. M. (1998). Regression and classification using Gaussian process priors. In Bernardo, J. M., Berger, J. O., Dawid, A. P., and Smith, A. F. M., editors, *Bayesian Statistic 6*, pages 475–502. Oxford University Press.

[Pang and Lee, 2005] Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. of ACL*, pages 115–124.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT.

[Raykar et al., 2010] Raykar, V., Yu, S., Zhao, L., Hermosillo Valadez, G., Florin, C., Bogoni, L., and Moy, L. (2010). Learning from crowds. *J. Mach. Learn. Res.*, 11:1297–1322.

[Rodrigues et al., 2017] Rodrigues, F., Lourenco, M., Ribeiro, B., and Pereira, F. (2017). Learning supervised topic models for classification and regression from crowds. *IEEE T. Pattern Anal.*, 39(12):2409–2422.

[Rodrigues and Pereira, 2018] Rodrigues, F. and Pereira, F. (2018). Deep learning from crowds. In *AAAI*, pages 81–92.

[Rodrigues et al., 2013] Rodrigues, F., Pereira, F., and Ribeiro, B. (2013). Learning from multiple annotators: Distinguishing good from random labelers. *Pattern Recog. Lett.*, 34(12):1428–1436.

[Rodrigues et al., 2014] Rodrigues, F., Pereira, F., and Ribeiro, B. (2014). Gaussian process classification and active learning with multiple annotators. In *ICML*, pages 433–441.

[Ruiz et al., 2016a] Ruiz, P., Besler, E., Molina, R., and Katsaggelos, A. (2016a). Variational Gaussian process for missing label crowdsourcing classification problems. In *MLSP*, pages 1–6.

[Ruiz et al., 2016b] Ruiz, P., Molina, R., and Katsaggelos, A. (2016b). Joint data filtering and labeling using Gaussian processes and Alternating Direction Method of Multipliers. *IEEE T. Image Process.*, 25(7):3059–3072.

[Siegel et al., 2018] Siegel, Z., Zhou, N., Zarecor, S., Lee, N., Campbell, D., Andorf, C., Nettleton, D., Lawrence-Dill, C., Ganapathysubramanian, B., Friedberg, I., and Kelly, J. (2018). Crowdsourcing Image Analysis for Plant Phenomics to Generate Ground Truth Data for Machine Learning. *Mechanical Engineering Publications*, page 271.

[Titsias et al., 2011] Titsias, M. K., Rattray, M., and Lawrence, N. D. (2011). *Markov chain Monte Carlo algorithms for Gaussian processes*, pages 295—316. Cambridge University Press.

[Triguero and Vens, 2016] Triguero, I. and Vens, C. (2016). Labelling strategies for hierarchical multi-label classification techniques. *Pattern Recogn.*, 56:170 – 183.

[Tzanetakis and Cook, 2002] Tzanetakis, G. and Cook, P. (2002). Musical genre classification of audio signals. *IEEE T. Speech Audi. P.*, 10(5):293–302.

[Wang et al., 2018] Wang, S., Chen, S., Chen, T., and Shi, X. (2018). Learning with privileged information for multi-label classification. *Pattern Recogn.*, 81:60 – 70.

[Watt et al., 2016] Watt, J., Borhani, R., and Katsaggelos, A. (2016). *Machine Learning Refined: Foundations, Algorithms, and Applications*. Cambridge University Press.

[Yan et al., 2010] Yan, Y., Rosales, R., Fung, G., Schmidt, M., Hermosillo Valadez, G., Bogoni, L., Moy, L., and Dy, J. (2010). Modeling annotator expertise: Learning when everybody knows a bit of something. In *AISTATS*, pages 932–939.

[Yan et al., 2014] Yan, Y., Rosales, R., Fung, G., Subramanian, R., and Dy, J. (2014). Learning from multiple annotators with varying expertise. *Mach. Learn.*, 95(3):291–327.

[Zevin et al., 2017] Zevin, M., Coughlin, S., Bahaadini, S., Besler, E., Rohani, N., Allen, S., et al. (2017). Gravity Spy: integrating advanced LIGO detector characterization, Machine Learning, and citizen science. *Classical Quant. Grav.*, 34(6):064003.

[Zhang et al., 2016] Zhang, J., Wu, X., and Sheng, V. S. (2016). Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576.

CHAPTER 5

Fourier features for Gaussian Processes based crowdsourcing

5.1 Publication details

Authors: Pablo Morales-Álvarez, Pablo Ruiz, Raúl Santos-Rodríguez, Rafael Molina, Aggelos K Katsaggelos
Title: Scalable and efficient learning from crowds with Gaussian Processes
Reference: Information Fusion 52, 110-127, 2019
Status: Published
Quality indices:

- Impact Factor (JCR 2019): 13.669
- Rank: 2/136 (Q1 and D1) in Computer Science, Artificial Intelligence.

5.2 Main contributions

- In this work we introduce RFFGPCR and VFFGPCR (Random and Variational Fourier Features for Gaussian Processes CRowdsourcing). The idea is to extend the methods RFF-GPC (Chapter 2) and VFF-GPC (Chapter 3) to the crowdsourcing scenario. As in VGPCR (Chapter 4), inference resorts to local variational methods and the annotators modelling is based on their specificity and sensitivity. This allows for scalability to medium-large datasets in GP-based crowdsourcing for the first time.
- The proposed approach is evaluated on several synthetic and real datasets, including *sphere* (Sensor Platform for HEalthcare in Residential Environment). This is a recognition dataset owned by the University of Bristol (UK) that aims at monitoring the well functioning of British residences, based on RBG-D video, a tri-axial accelerometer, and environmental sensors.

SCALABLE AND EFFICIENT LEARNING FROM CROWDS WITH GAUSSIAN PROCESSES

Pablo Morales-Álvarez
Dept. of Computer Science and AI
University of Granada, Spain

Pablo Ruiz
Dept. of Electrical Engineering and Computer Science
Northwestern University, USA

Raúl Santos-Rodríguez
Intelligent Systems Laboratory
University of Bristol, UK

Rafael Molina
Dept. of Computer Science and AI
University of Granada, Spain

Aggelos K. Katsaggelos
Dept. of Electrical Engineering and Computer Science
Northwestern University, USA

ABSTRACT

Over the last few years, multiply-annotated data has become a very popular source of information. Online platforms such as Amazon Mechanical Turk have revolutionized the labelling process needed for any classification task, sharing the effort between a number of annotators (instead of the classical single expert). This *crowdsourcing* approach has introduced new challenging problems, such as handling disagreements on the annotated samples or combining the unknown expertise of the annotators. Probabilistic methods, such as Gaussian Processes (GP), have proven successful to model this new crowdsourcing scenario. However, GPs do not scale up well with the training set size, which makes them prohibitive for medium-to-large datasets (beyond 10K training instances). This constitutes a serious limitation for current real-world applications. In this work, we introduce two scalable and efficient GP-based crowdsourcing methods that allow for processing previously-prohibitive datasets. The first one is an efficient and fast approximation to GP with squared exponential (SE) kernel. The second allows for learning a more flexible kernel at the expense of a heavier training (but still scalable to large datasets). Since the latter is not a GP-SE approximation, it can be also considered as a whole new scalable and efficient crowdsourcing method, useful for any dataset size. Both methods use Fourier features and variational inference, can predict the class of new samples, and estimate the expertise of the involved annotators. A complete experimentation compares them with state-of-the-art probabilistic approaches in synthetic and real crowdsourcing datasets of different sizes. They stand out as the best performing approach for large scale problems. Moreover, the second method is competitive with the current state-of-the-art for small datasets.

1 Introduction

The term *crowdsourcing* was coined in 2006 by J. Howe [Howe, 2006] to refer to "the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined generally large group of people in the form of an open call". In the last decade, many crowdsourcing services have proliferated in the Internet, where a dataset can be published and millions of people around the world can provide labels in exchange for a reward [Zhang et al., 2016]. Amazon Mechanical Turk (`www.amt.com`), Galaxy Zoo (`www.galaxyzoo.org`), Zooniverse (`www.zooniverse.org`), Crowdflowers (`www.crowdflower.com`) or Clickworker (`www.clickworker.com`) are among the most popular ones. Due to the great number of potential annotators, large data sets can be labeled in a very short time, overcoming one of the main limitations of the classical expert-alone labelling process. However, this crowdsourcing approach has introduced new challenging problems, such as combining the unknown expertise of annotators, dealing with disagreements on the annotated samples, or detecting the existence of spammer and adversarial annotators [Zhang et al., 2016]. All these problems have required probabilistic sound solutions, beyond the naive use of majority voting plus classical classification methods.

Crowdsourcing applications are growing rapidly. Since the early innovative use to detect small volcanoes in Magellan SAR images of Venus [Smyth et al., 1995], crowdsourcing techniques have been applied to a wide range of modern problems such as mitosis detection in breast cancer histology images [Albarqouni et al., 2016], topic modelling from crowds [Rodrigues et al., 2017], and detection of glitches in signals acquired by the laureate Laser Interferometer Gravitational-Wave Observatory (LIGO) [Zevin et al., 2017]. There also exist some recent attempts to combine crowdsourcing with Deep Learning approaches [Albarqouni et al., 2016, Rodrigues and Pereira, 2018]. Interestingly, the growth of social websites based on user-generated content (TripAdvisor, Twitter, YouTube) has turned multiple-annotation into a very natural way of labeling reviews, opinions, or videos. This relates crowdsourcing to the emerging *explainable-AI* [Goebel et al., 2018] which, in addition to predict a label for a given sample, explains the decision process in a human understandable and reconstructable way.

The first paper on crowdsourcing dates back to 1979 [Dawid and Skene, 1979]. Early contributions addressed the estimation of the underlying true labels and the reliability of the annotators, but were not conceived to learn a classifier. This idea was explored by Raykar *et al.* [Raykar et al., 2010], who proposed to jointly estimate the coefficients of a logistic regression (LR) classifier and the annotators' expertise. The latter is modelled through the *sensitivity* and *specificity* concepts, which refer to the accuracy of the annotator when labelling instances from each class. Yan *et al.* [Yan et al., 2010] (see also the subsequent journal version [Yan et al., 2014]), introduced a crowdsourcing classifier (also based on LR) which considers a feature-dependent model for the annotators' expertise. The main limitation of these two approaches is the simple LR classification model, which can only deal with linearly separable data. Rodrigues *et al.* [Rodrigues et al., 2014] overcame this problem by introducing a crowdsourcing classifier based on Gaussian Processes (GP) [Rasmussen and Williams, 2006, Ruiz et al., 2016, Morales-Álvarez et al., 2018]. GP is a probabilistic state-of-the-art model for functions, which uses the so-called *"kernel trick"* [Bishop, 2006, Chapter 6] to deal with complex non-linear decision boundaries. Moreover, its Bayesian formulation excels at uncertainty quantification [Rasmussen and Williams, 2006]. Expectation Propagation (EP) [Minka, 2001b] (see also [Rasmussen and Williams, 2006, Section 3.6]) was used as inference procedure for GP in [Rodrigues et al., 2014]. Recently, Variational Inference (VI) [Jaakkola, 2000, Blei et al., 2017] was used as an alternative to EP in crowdsourcing, outperforming it in both predictive performance and computational cost [Besler et al., 2016, Ruiz et al.,]. These probabilistic GP-based methods have proven very successful in the crowdsourcing literature. However, the poor scalability of standard GP models hampers their applicability to current medium-to-large scale real-world problems. Therefore, the development of scalable and efficient methods is one of the main research lines in crowdsourcing.

More specifically, classical[1] GPs operate with $N \times N$ kernel matrices, where N is the training set size. This implies a $\mathcal{O}(N^2)$ cost in RAM memory, a $\mathcal{O}(N^3)$ computational complexity at the training step (since the kernel matrix must be inverted), and $\mathcal{O}(N^2)$ cost in the test step. As a consequence, $N = 10^4$ instances is generally considered the practical limit of standard GPs [Rasmussen and Williams, 2006]. Since current real-world problems usually involve larger datasets, many *sparse GP approximations* have been developed in the Machine Learning community during the last years. The first approaches focused on selecting a convenient subset of the training set and applying standard GP there [Lawrence et al., 2003], see also [Rasmussen and Williams, 2006, Chapter 8]. Later on, *pseudo-inputs* and *inducing points* were proposed as a smarter way to reduce the computational cost of classical GP without completely loosing the information provided by the discarded points [Snelson and Ghahramani, 2006]. This approach has become very popular, and many works have been devoted to analyze it in depth and advance it further [Quiñonero-Candela and Rasmussen, 2005, Titsias, 2009, Hensman et al., 2013, Bauer et al., 2016]. Another recent promising approach is based on the random Fourier features approximation to the kernel matrix [Rahimi and Recht, 2008], which was proposed for GP-regression in [Lázaro-Gredilla et al., 2010] and further improved in [Gal and Turner, 2015]. Moreover, it was recently extended to GP-classification in [Morales-Álvarez et al., 2018].

In this work, we start by applying the aforementioned Fourier features methodology to approximate the squared exponential (SE) kernel of the GP-based crowdsourcing method proposed in [Besler et al., 2016, Ruiz et al.,]. This approach is referred to as *RFF* (Random Fourier Features). Then, we also propose *VFF* (Variational Fourier Features), which does not approximate a SE kernel but learns a new one well-suited for the data at hand. The training cost and RAM memory requirements for both approaches, including the computation of the Fourier features, scale linearly with N, and their test cost is independent on N. These are very significant reductions with respect to previous approaches. Whereas *RFF* is a large-scale approximation of the previous approach in [Besler et al., 2016, Ruiz et al.,], *VFF* is a whole new scalable crowdsourcing method, whose additional flexibility allows one to capture new relevant patterns (even in previously-reachable small datasets). However, *VFF* is more prone to overfitting, and slower in practice. A complete experimentation with real and synthetic crowdsourcing datasets of different sizes will show that i) the proposed methods can handle much larger training sets than previous approaches, ii) they have better generalization capability with a faster training step, iii) the test computational cost is extraor-

[1]Throughout this work, we will refer to *classical* and *standard* GP interchangeably to denote the typical and well-known formulation in [Rasmussen and Williams, 2006].

dinarily reduced, iv) the estimations of annotators' sensitivity and specificity are very accurate, and v) *VFF* is competitive with other state-of-the-art methods in small datasets.

The rest of the paper is organized as follows. Section 2 introduces the probabilistic modelling of the proposed methods. Section 3 presents the variational inference scheme used to estimate the posterior distributions and all the parameters of the model. Section 4 shows the predictive distribution to be used in the test step. Section 5 includes a complete experimentation evaluating the proposed methods. Finally, the main conclusions and some future outlook are provided in Section 6.

2 Probabilistic modelling

Formally, a crowdsourcing classification problem involves a training dataset $\{\mathbf{X}, \mathbf{Y}\}$, where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\intercal \in \mathbb{R}^{N\times D}$ is the set of features, and $\mathbf{Y} = \{y_n^r \in \{0,1\} | n = 1,\ldots,N,\ r \in R_n\}$ is the set annotations. N, D, and R denote, respectively, the number of training instances, their dimension (i.e. the number of features), and the number of annotators. $R_n \subseteq \{1,\ldots,R\}$ denotes the set of annotators that labelled the n-th instance. Analogously, we define $N_r \subseteq \{1,\ldots,N\}$ as the set of instances annotated by the r-th annotator.

The most successful probabilistic crowdsourcing approaches model the set of annotations $\mathbf{Y}$ by introducing a set of underlying unknown *real* labels $\mathbf{z} = (z_1,\ldots,z_N)^\intercal \in \{0,1\}^N$. Given z_n and $r \in R_n$, the r-th annotator's label is modelled with the conditional Bernoulli distributions

$$\mathrm{p}(y_n^r = 1 | z_n = 1) = \alpha_r, \quad \mathrm{p}(y_n^r = 0 | z_n = 0) = \beta_r, \tag{1}$$

where $\alpha_r, \beta_r \in [0,1]$ are called *sensitivity* and *specificity* for the r-th annotator, respectively. These numbers represent the reliability of that annotator when labelling instances in each class. Assuming independence between annotators and across their annotations, we have

$$\mathrm{p}(\mathbf{Y}|\mathbf{z},\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{r=1}^{R}\prod_{n\in N_r}\left[\alpha_r^{y_n^r}(1-\alpha_r)^{1-y_n^r}\right]^{z_n}\left[(1-\beta_r)^{y_n^r}\beta_r^{1-y_n^r}\right]^{1-z_n}, \tag{2}$$

where we denote $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_R)^\intercal$, and $\boldsymbol{\beta} = (\beta_1,\ldots,\beta_R)^\intercal$.

In this work, as in [Ruiz et al.,], all the α_r and β_r are treated in a Bayesian way, i.e. they are assumed to be stochastic variables. More specifically, they are assigned prior beta distributions $\alpha_r \to \text{Beta}(a_\alpha^r, b_\alpha^r)$ and $\beta_r \to \text{Beta}(a_\beta^r, b_\beta^r)$. Recall that $\text{Beta}(x|a,b) \propto x^{a-1}(1-x)^{b-1}$ for $0 < x < 1$, with $\mathbb{E}(x) = a/(a+b)$. During inference, the following expectations of a beta distribution will be also required

$$\mathbb{E}(\log x) = \psi(a) - \psi(a+b), \quad \mathbb{E}(\log(1-x)) = \psi(b) - \psi(a+b), \tag{3}$$

where ψ denotes the digamma function (see [Bishop, 2006, Exercise 2.11]). In a beta distribution, the hyper-parameters a and b can be set to introduce prior knowledge about the variable (in our case, the reliability of each annotator labelling instances in each class). When no prior knowledge is available, $a = b = 1$ produces an uniform prior distribution. Since the *specificity* and *sensitivity* of the different annotators are assumed independent, we have the joint priors:

$$\mathrm{p}(\boldsymbol{\alpha}) = \prod_{r=1}^{R}\text{Beta}(\alpha_r|a_\alpha^r, b_\alpha^r), \quad \mathrm{p}(\boldsymbol{\beta}) = \prod_{r=1}^{R}\text{Beta}(\beta_r|a_\beta^r, b_\beta^r). \tag{4}$$

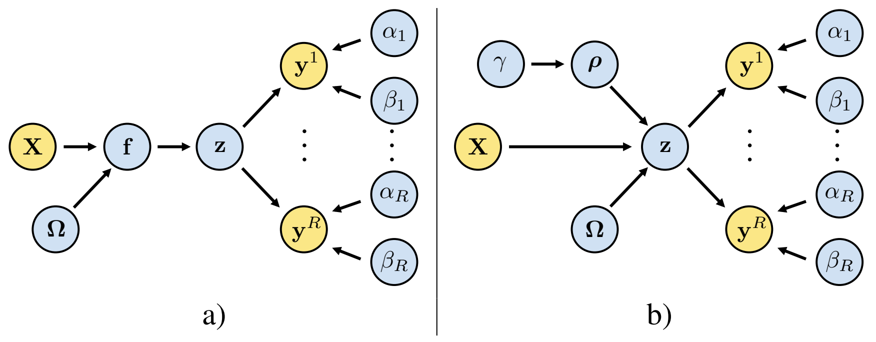


Figure 1: Graphical representations of the classical GP-based probabilistic model for crowdsourcing (left) and the new one proposed here (right). Yellow nodes represent the observed variables, and blue nodes represent the variables to be estimated. Notice that the only difference is in the connection between the features $\mathbf{X}$ and the underlying real labels $\mathbf{z}$ (a GP is used on the left and a Bayesian logistic-regression model based on Fourier features on the right). In the latter, we have $\mathbf{\Omega} = \omega$ for *RFF* and $\mathbf{\Omega} = \mathbf{W}$ for *VFF*.

Finally, to model the underlying real labels $\mathbf{z}$ given the features $\mathbf{X}$, Gaussian Processes (GP) has proven to be the most successful probabilistic approach, mainly because of its great flexibility and excellent uncertainty quantification [Rodrigues et al., 2014, Besler et al., 2016, Ruiz et al.,]. A GP introduces N latent variables $(f_1 = f(\mathbf{x}_1), \dots, f_N = f(\mathbf{x}_N)) =: \mathbf{f}$ that jointly follow a multivariate normal distribution whose covariance matrix (the *kernel matrix*) depends on $\mathbf{X}$, i.e. the distribution of $\mathbf{f}$ is $\mathcal{N}(\mathbf{0}, \mathbf{K} = (k(\mathbf{x}_n, \mathbf{x}_m))_{1\leq n,m\leq N})$. The kernel function $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ encodes the properties (like smoothness) of the functions $f(\mathbf{x})$ considered. Then, given each latent variable f_n, the underlying real label z_n is modelled with the sigmoid function σ, $\mathrm{p}(z_n = 1|f_n) = \sigma(f_n) = (1 + \exp(-f_n))^{-1}$. Under this common classical model, the main difference between the previous approaches [Rodrigues et al., 2014] and [Besler et al., 2016, Ruiz et al.,] is the inference procedure used: Expectation Propagation [Minka, 2001a] in the former and Variational Inference [Jaakkola, 2000, Blei et al., 2017] in the latter (recall the second paragraph of Section 1). Figure 1a) shows a graphical representation of this GP-based classical model, which is in the basis of our proposal.

Although standard GP is well-known for modelling very complex data and accurately quantifying and propagating uncertainty, it does not scale up well to large datasets (recall the fourth paragraph in Section 1). Therefore, different sparse GP approximations have been proposed over the last years in the Machine Learning community [Hensman et al., 2013, Bauer et al., 2016, Lázaro-Gredilla et al., 2010, Gal and Turner, 2015, Morales-Álvarez et al., 2018]. Here, as it is done for regression in [Lázaro-Gredilla et al., 2010, Gal and Turner, 2015] and for classification in [Morales-Álvarez et al., 2018], we will resort to the interesting Fourier features approximation [Rahimi and Recht, 2008] and will apply it to crowdsourcing.

2.1 Fourier features

The work [Rahimi and Recht, 2008] presents a general methodology to approximate any positive-definite shift-invariant kernel k by a linear one. This is achieved by projecting the original D-dimensional data $\mathbf{x}$ into $2D_f$ Fourier features $\boldsymbol{\phi}(\mathbf{x})$, whose linear kernel k_L approximates the original k. In the case of GP, this linearity enables one to *undo* the so-called kernel trick and work in the primal space of features [Bishop, 2006, Chapter 6]. With this, $N \times N$ matrix inversions are substituted by $2D_f \times 2D_f$ ones, yielding a total $\mathcal{O}(ND_f^2 + D_f^3)$ training cost. In large-scale applications we can set $D_f \ll N$, and the resulting $\mathcal{O}(ND_f^2)$ complexity, which is *linear* in N, constitutes an important reduction over the original $\mathcal{O}(N^3)$. Moreover, both the test complexity and the memory cost reduce to $\mathcal{O}(D_f^2)$, which is *independent* on N. Of course, the main drawback of this process is that we work with an *approximation* to the original kernel.

More specifically, let us consider the well-known SE kernel $k(\mathbf{x}, \mathbf{y}) = \gamma \cdot \exp(-||\mathbf{x}-\mathbf{y}||^2/(2\omega^2))$, where the hyper-parameters γ and ω are called *variance* and *length-scale*, respectively. Following [Rahimi and Recht, 2008], this kernel can be approximated as $k(\mathbf{x}, \mathbf{y}) \approx k_L(\mathbf{x}, \mathbf{y}) := \gamma \cdot \boldsymbol{\phi}(\mathbf{x})^\intercal \boldsymbol{\phi}(\mathbf{y})$, where the *Fourier features* $\boldsymbol{\phi}$ are given by

$$\boldsymbol{\phi}(\mathbf{x})^\intercal = D_f^{-1/2} \cdot \left(\cos(\mathbf{w}_1^\intercal \mathbf{x}), \sin(\mathbf{w}_1^\intercal \mathbf{x}), \ldots, \cos(\mathbf{w}_{D_f}^\intercal \mathbf{x}), \sin(\mathbf{w}_{D_f}^\intercal \mathbf{x})\right) \in \mathbb{R}^{2D_f}, \tag{5}$$

and the D_f *Fourier frequencies* $\mathbf{w}_i$ must be sampled from a normal distribution $\mathcal{N}(\mathbf{0}, \omega^{-2}\mathbf{I})$. This approximation exponentially improves with the number D_f of Fourier frequencies used [Rahimi and Recht, 2008, Claim 1]. However, increasing D_f will go at the expense of increasing train and test computational cost and memory requirements in our methods. Other kernels could also be used, but that would involve sampling from a different distribution.

2.2 The proposed models

Our first proposal consists of introducing this Fourier features approximation for the SE kernel in the variational GP-based crowdsourcing method *VGPCR* [Besler et al., 2016, Ruiz et al.,]. Notice that, as explained above, the Fourier frequencies $\mathbf{w}_i$ must be sampled from $\mathcal{N}(\mathbf{0}, \omega^{-2}\mathbf{I})$ and fixed, whereas the length-scale hyper-parameter ω must be estimated during training (just as for standard GPs). To uncouple $\mathbf{w}_i$ and ω, we resort to the following equivalent expression for the Fourier features, which makes explicit the dependence on ω

$$\boldsymbol{\phi}(\mathbf{x}|\omega)^\intercal = D_f^{-1/2} \cdot \left(\cos(\omega^{-1}\mathbf{w}_1^\intercal \mathbf{x}), \sin(\omega^{-1}\mathbf{w}_1^\intercal \mathbf{x}), \ldots, \cos(\omega^{-1}\mathbf{w}_{D_f}^\intercal \mathbf{x}), \sin(\omega^{-1}\mathbf{w}_{D_f}^\intercal \mathbf{x})\right), \tag{6}$$

where now $\mathbf{w}_i$ must be sampled now from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Then, undoing the kernel trick and passing to the primal space of features, we change the GP for the equivalent[2] Bayesian logistic-regression model $\mathrm{p}(z_n = 1|\mathbf{x}_n, \omega, \boldsymbol{\rho}) = (1 + \exp(-\boldsymbol{\phi}(\mathbf{x}_n|\omega)^\intercal \boldsymbol{\rho}))^{-1}$, where the logistic-regression weights $\boldsymbol{\rho}$ follow a normal prior $\mathcal{N}(\mathbf{0}, \gamma\mathbf{I})$ (more details about the kernel trick in [Bishop, 2006, Chapter 6]). Finally, assuming independence between the different instances given $\boldsymbol{\rho}$, we have

$$\mathrm{p}(\mathbf{z}|\boldsymbol{\rho}, \omega, \mathbf{X}) = \prod_{n=1}^{N} \left(\frac{1}{1 + e^{-\boldsymbol{\rho}^\intercal \boldsymbol{\phi}(\mathbf{x}_n|\omega)}}\right)^{z_n} \left(\frac{e^{-\boldsymbol{\rho}^\intercal \boldsymbol{\phi}(\mathbf{x}_n|\omega)}}{1 + e^{-\boldsymbol{\rho}^\intercal \boldsymbol{\phi}(\mathbf{x}_n|\omega)}}\right)^{1-z_n}. \tag{7}$$

This model will be refered to as *RFFGPCR* (Random Fourier Features Gaussian Processes for Crowdsourcing), or *RFF* for short. In *RFF*, the Fourier frequencies $\mathbf{W} = (\mathbf{w}_1, \ldots, \mathbf{w}_{D_f})^\intercal \in \mathbb{R}^{D_f \times D}$ are *randomly* sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and fixed from the beginning, whereas ω is estimated during training (to maximize the marginal likelihood, see Section 3).

Our second proposal follows the same rationale as *RFF*, but *optimizes* the Fourier frequencies $\mathbf{w}_i$ in eq. (5). Since they are estimated to maximize the marginal likelihood within a variational scheme

[2]Again, we stress that this new model is equivalent to GP with the Fourier features *approximation* for the SE kernel.

(see Section 3), this approach is refered to as *VFFGPCR* (Variational Fourier Features Gaussian Processes for Crowdsourcing), or *VFF* for short. Therefore, the *VFF* model for $\mathbf{z}$ is identical to that of *RFF*, eq. (7), but with $\mathbf{W}$ playing the role of ω and with the original Fourier features expression in eq. (5) instead of the modified eq. (6). To unify the notation, we will indistinctly write $\mathbf{\Omega}$ for ω (*RFF*) or $\mathbf{W}$ (*VFF*), and therefore

$$\mathrm{p}(\mathbf{z}|\boldsymbol{\rho}, \mathbf{\Omega}, \mathbf{X}) = \prod_{n=1}^{N} \left(\frac{1}{1 + e^{-\boldsymbol{\rho}^\intercal \boldsymbol{\phi}(\mathbf{x}_n|\mathbf{\Omega})}} \right)^{z_n} \left(\frac{e^{-\boldsymbol{\rho}^\intercal \boldsymbol{\phi}(\mathbf{x}_n|\mathbf{\Omega})}}{1 + e^{-\boldsymbol{\rho}^\intercal \boldsymbol{\phi}(\mathbf{x}_n|\mathbf{\Omega})}} \right)^{1-z_n}, \tag{8}$$

with $\boldsymbol{\phi}(\mathbf{x}_n|\mathbf{\Omega})$ as in eq. (6) (*RFF*) or eq. (5) (*VFF*).

Unlike *RFF*, notice that *VFF* is no longer an approximation to *VGPCR* (for which the Fourier frequencies must be sampled from $\mathcal{N}(\mathbf{0}, \omega^{-2}\mathbf{I})$), but a whole new probabilistic crowdsourcing method that *learns* an appropriate kernel. Moreover, its computational cost is similar to *RFF*'s. More specifically, we will see that the theoretical training complexity for *VFF* is $\mathcal{O}(ND_fD + ND_f^2)$ (whereas it is $\mathcal{O}(ND_f^2)$ for *RFF*). This is linear in N (like for *RFF*), and therefore much more scalable than the original *VGPCR* ($\mathcal{O}(N^3)$). Nonetheless, the experimentation will show that the Fourier frequencies optimization significantly slows down *VFF* when compared to *RFF* in practice. Moreover, whereas D_f has a clear influence in *RFF* performance (the higher, the better it is the kernel approximation), it is related to the complexity of the model (the degrees of freedom) in *VFF*. Therefore, in *VFF*, large values of D_f may lead to overfitting to the training set.

In summary, the proposed probabilistic crowdsourcing model is

$$\mathrm{p}(\mathbf{Y}, \mathbf{z}, \boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}|\mathbf{\Omega}, \gamma) = \mathrm{p}(\mathbf{Y}|\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta})\mathrm{p}(\mathbf{z}|\boldsymbol{\rho}, \mathbf{\Omega})\mathrm{p}(\boldsymbol{\rho}|\gamma)\mathrm{p}(\boldsymbol{\alpha})\mathrm{p}(\boldsymbol{\beta}), \tag{9}$$

with $\mathrm{p}(\mathbf{Y}|\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ as in eq. (2), $\mathrm{p}(\mathbf{z}|\boldsymbol{\rho}, \mathbf{\Omega})$ as in eq. (8), $\mathrm{p}(\boldsymbol{\rho}|\gamma) = \mathcal{N}(\boldsymbol{\rho}|\mathbf{0}, \gamma\mathbf{I})$, and $\mathrm{p}(\boldsymbol{\alpha}), \mathrm{p}(\boldsymbol{\beta})$ as in eq. (4). Notice that, for clarity, we have omitted $\mathbf{X}$ from the notation. Figure 1b) shows a graphical representation of the proposed model.

3 Variational Bayes inference

Once the training set $\{\mathbf{X}, \mathbf{Y}\}$ is observed, Bayesian inference seeks to calculate the maximum-likelihood hyperparameters $(\hat{\mathbf{\Omega}}, \hat{\gamma}) = \arg\max_{\mathbf{\Omega},\gamma} \mathrm{p}(\mathbf{Y}|\mathbf{\Omega}, \gamma)$, and the posterior distribution $\mathrm{p}(\mathbf{z}, \boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}|\mathbf{Y}, \hat{\mathbf{\Omega}}, \hat{\gamma})$. However, in our case, the marginal likelihood $\mathrm{p}(\mathbf{Y}|\mathbf{\Omega}, \gamma) = \int_{\mathbf{z},\boldsymbol{\rho},\boldsymbol{\alpha},\boldsymbol{\beta}} \mathrm{p}(\mathbf{Y}, \mathbf{z}, \boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}|\mathbf{\Omega}, \gamma)$ cannot be obtained in closed form (for simplicity, the sum in the discrete variable $\mathbf{z}$ is denoted with integration). Variational inference [Jaakkola, 2000, Blei et al., 2017], see also [Bishop, 2006, Section 10.1], is a very popular approach to obtain an approximation to the posterior distribution in Bayesian inference. It consists of finding, inside a predefined family $\mathcal{Q}$, the distribution $\mathrm{q} \in \mathcal{Q}$ that minimizes the Kullback-Leibler divergence (KL) from q to the real posterior. Recall that the KL divergence from a distribution $\mathrm{q}(\mathbf{x})$ to another $\mathrm{p}(\mathbf{x})$ is defined as $\mathrm{KL}(\mathrm{q}||\mathrm{p}) = \int \mathrm{q}(\mathbf{x}) \log(\mathrm{q}(\mathbf{x})/\mathrm{p}(\mathbf{x}))\mathrm{d}\mathbf{x}$, which is always greater or equal to zero, and vanishes if and only if $\mathrm{q} = \mathrm{p}$. A different popular approach to approximate the posterior distribution is called *Expectation Propagation* [Minka, 2001b]. However, to the best of our knowledge, variational inference has achieved better results in classical GP-based probabilistic crowdsourcing methods, being also significantly more efficient (which is specially relevant in large-scale scenarios like ours) [Besler et al., 2016, Ruiz et al.,].

Here, for $\mathbf{\Omega}$ and γ fixed, we propose an approximate posterior of the form

$$\mathrm{q}(\mathbf{z}, \boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathrm{q}(\boldsymbol{\rho})\mathrm{q}(\boldsymbol{\alpha})\mathrm{q}(\boldsymbol{\beta})\mathrm{q}(\mathbf{z})^3. \tag{10}$$

The reason for "uncoupling" $\mathbf{z}$ is that integrating it out in the true posterior $\mathrm{p}(\mathbf{z}, \boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}|\mathbf{Y}, \mathbf{\Omega}, \gamma)$ is analytically intractable. Notice that this also applies to $\boldsymbol{\rho}$ due to the sigmoids in eq. (8), for which we will additionally resort to the local variational bound of the sigmoid [Bishop, 2006, Section 10.5]. Using the factorization proposed in eq. (10), the well-known mean-field formula [Bishop, 2006, Section 10.1.1, eq. (10.9)] yields the following update for $\mathrm{q}(\mathbf{z})$ (which factorizes along data points):

$$\begin{aligned}
\mathrm{q}(z_n = 0) &\propto \prod_{r \in R_n} \exp\left\{y_n^r \mathbb{E}_\mathrm{q}(\log(1-\beta_r)) + (1-y_n^r)\mathbb{E}_\mathrm{q}(\log \beta_r)\right\}, \\
\mathrm{q}(z_n = 1) &\propto \exp(\boldsymbol{\phi}(\mathbf{x}_n|\mathbf{\Omega})^\intercal \mathbb{E}_\mathrm{q}(\boldsymbol{\rho})) \cdot \prod_{r \in R_n} \exp\left\{y_n^r \mathbb{E}_\mathrm{q}(\log \alpha_r) + (1-y_n^r)\mathbb{E}_\mathrm{q}(\log(1-\alpha_r))\right\},
\end{aligned} \tag{11}$$

where the expectations are with respect to the current values of $\mathrm{q}(\boldsymbol{\alpha})$, $\mathrm{q}(\boldsymbol{\beta})$ and $\mathrm{q}(\boldsymbol{\rho})$. For the terms of the form $\mathbb{E}_\mathrm{q}(\log(\cdot))$, recall eq. (3). Analogously, the updates for $\mathrm{q}(\boldsymbol{\alpha})$ and $\mathrm{q}(\boldsymbol{\beta})$ factorize along annotators and are given by:

$$\mathrm{q}(\alpha_r) = \text{Beta}\left(\alpha_r \middle| a_\alpha^r + \sum_{n \in N_r} \mathbb{E}_\mathrm{q}(z_n) y_n^r, b_\alpha^r + \sum_{n \in N_r} \mathbb{E}_\mathrm{q}(z_n)(1-y_n^r)\right), \tag{12}$$

$$\mathrm{q}(\beta_r) = \text{Beta}\left(\beta_r \middle| a_\beta^r + \sum_{n \in N_r} (1-\mathbb{E}_\mathrm{q}(z_n))(1-y_n^r), b_\beta^r + \sum_{n \in N_r} (1-\mathbb{E}_\mathrm{q}(z_n)) y_n^r\right), \tag{13}$$

where the expectations are with respect to the current distribution $\mathrm{q}(\mathbf{z})$.

In order to update $\mathrm{q}(\boldsymbol{\rho})$, we find analytic intractability in $\boldsymbol{\rho}$ due to the sigmoids in $\mathrm{p}(\mathbf{z}|\boldsymbol{\rho}, \mathbf{\Omega})$, recall eq. (8). To overcome this, we use the local variational bound of the sigmoid [Bishop, 2006, Section 10.5, eq. (10.144)], which yields

$$\mathrm{p}(\mathbf{z}|\boldsymbol{\rho}, \mathbf{\Omega}) \geq \exp\left(\mathbf{v}^\intercal \mathbf{\Phi} \boldsymbol{\rho} - \boldsymbol{\rho}^\intercal \mathbf{\Phi}^\intercal \mathbf{\Lambda} \mathbf{\Phi} \boldsymbol{\rho} + C(\boldsymbol{\xi})\right) =: H(\mathbf{z}, \boldsymbol{\rho}, \mathbf{\Omega}, \boldsymbol{\xi}). \tag{14}$$

Notice that this lower bound is exponentially-quadratic in $\boldsymbol{\rho}$, which will allow us to identify a Gaussian distribution in $\boldsymbol{\rho}$. In exchange, we are introducing N additional hyper-parameters $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)$ to be estimated. Here we are writing $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_N)^\intercal \in \mathbb{R}^{N \times (2D_f)}$ for the matrix of Fourier features, $\mathbf{v} = \mathbf{z} - (1/2)\mathbf{1}$, $\mathbf{\Lambda} = \mathrm{diag}(\lambda(\xi_1), \ldots, \lambda(\xi_N))$, and $\lambda(\xi) = (2\xi)^{-1}\left((1+\exp(-\xi))^{-1} - 1/2\right)$. The term $C(\boldsymbol{\xi}) = \sum_{n=1}^N \left(\lambda(\xi_n)\xi_n^2 + \xi_n/2 - \log\left(1 + e^{\xi_n}\right)\right)$ only depends on $\boldsymbol{\xi}$.

Using eq. (14) we have, up to a constant, the following upper bound for the KL divergence (which must be minimized, instead of the intractable KL itself, in $\mathrm{q}(\boldsymbol{\rho})$, with $\mathrm{q}(\mathbf{z})$, $\mathrm{q}(\boldsymbol{\alpha})$ and $\mathrm{q}(\boldsymbol{\beta})$ fixed):

$$\begin{aligned}
&\mathrm{KL}(\mathrm{q}(\boldsymbol{\rho})\mathrm{q}(\boldsymbol{\alpha})\mathrm{q}(\boldsymbol{\beta})\mathrm{q}(\mathbf{z}) || \mathrm{p}(\mathbf{z}, \boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}|\mathbf{Y}, \mathbf{\Omega}, \gamma)) \leq \\
&\qquad \int_{\mathbf{z}, \boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{\Omega}, \gamma} \mathrm{q}(\boldsymbol{\rho})\mathrm{q}(\boldsymbol{\alpha})\mathrm{q}(\boldsymbol{\beta})\mathrm{q}(\mathbf{z}) \log \frac{\mathrm{q}(\boldsymbol{\rho})\mathrm{q}(\boldsymbol{\alpha})\mathrm{q}(\boldsymbol{\beta})\mathrm{q}(\mathbf{z})}{\mathrm{p}(\mathbf{Y}|\mathbf{z}, \boldsymbol{\alpha}, \boldsymbol{\beta}) H(\mathbf{z}, \boldsymbol{\rho}, \mathbf{\Omega}, \boldsymbol{\xi}) \mathrm{p}(\boldsymbol{\rho}|\gamma)\mathrm{p}(\boldsymbol{\alpha})\mathrm{p}(\boldsymbol{\beta})}.
\end{aligned} \tag{15}$$

[3]This is equivalent to the more general form $\mathrm{q}(\boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta})\mathrm{q}(z)$, since the variables $\boldsymbol{\rho}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ are coupled in the joint model of eq. (9) only through $\mathbf{z}$.

Following the standard mean-field procedure [Bishop, 2006, Section 10.1.1], this minimization yields $\mathrm{q}(\boldsymbol{\rho}) \propto H(\mathbb{E}_{\mathrm{q}}(\mathbf{z}), \boldsymbol{\rho}, \boldsymbol{\Omega}, \boldsymbol{\xi})\mathrm{p}(\boldsymbol{\rho}|\gamma)$. Since H is exponentially-quadratic in $\boldsymbol{\rho}$, we have $\mathrm{q}(\boldsymbol{\rho}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with

$$\boldsymbol{\Sigma} = \left(\gamma^{-1}\mathbf{I} + \boldsymbol{\Phi}^{\intercal}(2\Lambda)\boldsymbol{\Phi}\right)^{-1}, \quad \boldsymbol{\mu} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^{\intercal}\mathbb{E}_{\mathrm{q}}(\mathbf{v}). \tag{16}$$

Then, approximating $\mathrm{p}(\mathbf{z}|\boldsymbol{\rho}, \boldsymbol{\Omega})$ by its lower bound $H(\mathbf{z}, \boldsymbol{\rho}, \boldsymbol{\Omega}, \boldsymbol{\xi})$ in the full model $\mathrm{p}(\mathbf{Y}, \mathbf{z}, \boldsymbol{\rho}, \boldsymbol{\alpha}, \boldsymbol{\beta}|\boldsymbol{\Omega}, \gamma)$, we find that $\boldsymbol{\rho}$ can be marginalized out (again, H being exponentially-quadratic in $\boldsymbol{\rho}$ is essential here). Using the current distribution $\mathrm{q}(\mathbf{z})$, the maximum-likelihood estimators for $\boldsymbol{\Omega}$ and γ are

$$(\hat{\boldsymbol{\Omega}}, \hat{\gamma}) = \arg\max_{\boldsymbol{\Omega},\gamma} \left(-\log|2\gamma\boldsymbol{\Phi}^{\intercal}\boldsymbol{\Lambda}\boldsymbol{\Phi} + \mathbf{I}| + \mathbb{E}_{\mathrm{q}}(\mathbf{v})^{\intercal}\boldsymbol{\Phi}\left(\gamma^{-1}\mathbf{I} + 2\boldsymbol{\Phi}^{\intercal}\boldsymbol{\Lambda}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\intercal}\mathbb{E}_{\mathrm{q}}(\mathbf{v})\right). \tag{17}$$

Finally, the hyper-parameters $\boldsymbol{\xi}$ are estimated to minimize the right-hand side of eq. (15), which yields (notice that the square is element-wise)

$$\boldsymbol{\xi} = \sqrt{\mathrm{diag}\left(\boldsymbol{\Phi}\boldsymbol{\Sigma}\boldsymbol{\Phi}^{\intercal}\right) + (\boldsymbol{\Phi}\boldsymbol{\mu})^2}. \tag{18}$$

In summary, the proposed methods calculate sequences $\{\boldsymbol{\xi}^k\}$, $\{\boldsymbol{\Omega}^k, \gamma^k\}$, $\{\mathrm{q}^k(\boldsymbol{\rho})\}$, $\{\mathrm{q}^k(\boldsymbol{\alpha})\}$, $\{\mathrm{q}^k(\boldsymbol{\beta})\}$, $\{\mathrm{q}^k(\mathbf{z})\}$ until convergence, following the formulas derived in this section. The training process is summarized in Algorithm 1. The computational cost of the algorithms is dominated by $2D_f \times 2D_f$ matrix inversions (e.g. eq. (17)) and $(2D_f \times N) \cdot (N \times 2D_f)$ matrix multiplications (e.g. $\boldsymbol{\Sigma}$ in eq. (16)). This yields a theoretical complexity of $\mathcal{O}(D_f^3 + ND_f^2)$ which, in large scale scenarios (where D_f will be taken $D_f \ll N$), is $\mathcal{O}(ND_f^2)$. In the case of VFF, the optimization with respect to the $D_f \cdot D$ components of $\mathbf{W}$ introduces an additional dependence on D, and yields $\mathcal{O}(ND_f^2 + ND_fD)$ cost.

Algorithm 1 Training of RFF and VFF

```
Require: $\mathbf{X}, \mathbf{Y}, \boldsymbol{\Omega}^0, \mathrm{q}^0(\boldsymbol{\rho}), \mathrm{q}^0(\mathbf{z}), k = 0$.
  repeat
    Update $\boldsymbol{\xi}^{k+1}$ with eq. (18) using $\mathrm{q}^k(\boldsymbol{\rho})$ and $\boldsymbol{\Omega}^k$;
    Update $\gamma^{k+1}$ and $\boldsymbol{\Omega}^{k+1}$ with eq. (17) using $\boldsymbol{\xi}^{k+1}$ and $\mathrm{q}^k(\mathbf{z})$;
    Update $\mathrm{q}^{k+1}(\boldsymbol{\rho})$ with eq. (16) using $\boldsymbol{\xi}^{k+1}$, $\boldsymbol{\Omega}^{k+1}$, $\gamma^{k+1}$ and $\mathrm{q}^k(\mathbf{z})$;
    Update $\mathrm{q}^{k+1}(\boldsymbol{\alpha})$ and $\mathrm{q}^{k+1}(\boldsymbol{\beta})$ with eqs. (12)-(13) using $\mathrm{q}^k(\mathbf{z})$;
    Update $\mathrm{q}^{k+1}(\mathbf{z})$ with eq. (11) using $\boldsymbol{\Omega}^{k+1}$, $\mathrm{q}^{k+1}(\boldsymbol{\rho})$, $\mathrm{q}^{k+1}(\boldsymbol{\alpha})$ and $\mathrm{q}^{k+1}(\boldsymbol{\beta})$.
    $k = k + 1$;
  until convergence
  Output: Final values $\hat{\boldsymbol{\xi}}$, $\hat{\boldsymbol{\Omega}}$, $\hat{\gamma}$, $\hat{\mathrm{q}}(\boldsymbol{\rho})$, $\hat{\mathrm{q}}(\boldsymbol{\alpha})$, $\hat{\mathrm{q}}(\boldsymbol{\beta})$, $\hat{\mathrm{q}}(\mathbf{z})$.
```

It is interesting to examine and understand how the proposed methodology mitigates the effect of weak annotators (i.e. those who may provide unreliable labels). Recall from eq. (1) that each annotator reliability is modelled through sensitivity and specificity parameters α and β. These parameters are estimated during the training step, see eqs. (12) and (13). Then, these estimations of α and β are used in eq. (11) in order to update the distribution of the underlying real label z for each training instance. Importantly, note that the influence of each annotation y_n^r is appropriately modulated by the estimations of α and β for the corresponding annotator.

This becomes even clearer when degenerate posterior distributions are assumed for α_r and β_r. In this case, the posterior distribution approximation $\mathrm{q}(z_n)$ in eq. (11) is proportional to $\prod_r(1-\beta_r)^{y_n^r}\beta_r^{(1-y_n^r)}$ and $\prod_r \alpha_r^{y_n^r}(1-\alpha_r)^{(1-y_n^r)}$ for $z_n = 0$ and $z_n = 1$, respectively. Suppose that an annotator labels an instance as $y_n^r = 1$. Then, this implies a factor (which can be understood as a "multiplicative" weight) of $(1-\beta_r)$ for the probability of $z_n = 0$, and a factor of α_r for $z_n = 1$. If the annotator is a reliable one, then α_r and β_r are close to 1, which implies a much greater weight for $z_n = 1$ than for $z_n = 0$. However, if the annotator is a weak one (for both classes), then α_r and β_r will be close to 0, and the weight for $z_n = 0$ will be much greater than for $z_n = 1$, making it very likely to correctly switch the (very likely) wrong label provided by this weak annotator. The weaker the annotator is, the more likely it is to switch the annotation. An analogous interpretation applies when the annotator labels $y_n^r = 0$, or when the annotator is weak only for one of the two classes. Observe also that a spammer annotator (i.e. $\alpha_r = \beta_r = 0.5$), will not influence the probability of z_n, as both weights will be identical.

Finally, notice that the Bayesian modelling allows for naturally specifying the available prior knowledge on the annotators. For instance, if a particular annotator is known to be weak (even only for one of the two classes), the corresponding Beta prior distribution (recall the paragraph before eq. (3)) can be conveniently set to integrate in the model this valuable information.

4 The predictive distribution

Once the model is trained, the final distributions $\hat{\mathrm{q}}(\boldsymbol{\alpha})$ and $\hat{\mathrm{q}}(\boldsymbol{\beta})$ represent the estimated sensitivity and specificity for the annotators (as well as their uncertainty). Analogously, $\hat{\mathrm{q}}(\mathbf{z})$ describes the estimated uncertainty for the underlying real labels of the training instances. The most common problem is to, based on the training data, obtain the *predictive distribution* for the real class of a new instance $\mathbf{x}_* \in \mathbb{R}^D$, i.e. compute $\mathrm{p}(z_* = 1|\mathbf{Y})$ (obviously, $\mathrm{p}(z_* = 0|\mathbf{Y}) = 1 - \mathrm{p}(z_* = 1|\mathbf{Y})$). Using the standard approximation for the expectation of the sigmoid under a Gaussian [Bishop, 2006, Section 4.5.2, eq. (4.153)], we have

$$\mathrm{p}(z_* = 1|\mathbf{Y}) = \mathbb{E}_{\hat{\mathrm{q}}(\boldsymbol{\rho})}\mathrm{p}(z_* = 1|\boldsymbol{\rho}, \hat{\boldsymbol{\Omega}}) \approx \sigma\left(\frac{\hat{\boldsymbol{\phi}}_*^\intercal \hat{\boldsymbol{\mu}}}{\sqrt{1 + (\pi/8)\hat{\boldsymbol{\phi}}_*^\intercal \hat{\boldsymbol{\Sigma}} \hat{\boldsymbol{\phi}}_*}}\right), \tag{19}$$

where $\sigma(x) = (1 + \exp(-x))^{-1}$ is the sigmoid, $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ are the mean and covariance of the posterior $\hat{\mathrm{q}}(\boldsymbol{\rho})$ (which are obtained in the training step), and $\hat{\boldsymbol{\phi}}_* = \boldsymbol{\phi}(\mathbf{x}_*|\hat{\boldsymbol{\Omega}})$ (using eq. (6) in the case of *RFF* and eq. (5) for *VFF*).

The theoretical computational complexity for the test step is dominated by the computation $\hat{\boldsymbol{\phi}}_*^\intercal \Sigma \hat{\boldsymbol{\phi}}_*$. This implies a $\mathcal{O}(D_f^2)$ cost per test instance. Unlike classical GP, whose corresponding complexity is $\mathcal{O}(N^2)$, this is independent on the number of training instances N. In large scale scenarios (where N is large), this will translate into an overwhelming superiority of the proposed methods in terms of "production" time (i.e. time needed for prediction), which is essential in real-world applications.

Finally, since eq. (19) is one of the key ingredients for Active Learning (AL) techniques, let us conclude this section by commenting on the use of AL for our RFF and VFF models. In section 1 we motivated the use of crowdsourcing in labeling tasks as a very efficient way to annotate large datasets. In order to further speed up this process, crowdsourcing can be combined with AL. For a classic (non-crowdsourcing) classifier, AL selects the most informative instance from a set of unlabeled samples, and the expert provides the corresponding label. The new labeled sample is included in the training set, and the classifier is retrained (updated). It has been shown that AL significantly reduces the number of samples to be labeled in order to train an accurate classifier (see, for instance, [Ruiz et al., 2014]).

In crowdsourcing labeling problems, AL becomes an even more interesting (and challenging) problem, since the best annotator/s to provide the label must also be selected. Interestingly, the majority of probabilistic crowdsourcing AL methods in the literature are based on different combinations of the same two key ingredients: the uncertainty of the model when labeling a new instance (in our case given by the predictive distribution in eq. (19)), and the estimated expertise for each annotator (in our case the sensitivity and specificity posteriors given in eqs. (12) and (13)). Rodrigues *et al.* [Rodrigues et al., 2014] first select the closest sample to the decision boundary and then the annotator who maximizes the expected probability of success. Yan *et al.* [Yan et al., 2011] minimize an objective function to simultaneously find the closest sample to the decision boundary and the annotator who minimizes the probability of mistake. More recently, Yang *et al.* [Yang et al., 2018] select the sample that maximizes the Shannon entropy of the predictive distribution and the annotator who maximizes the probability of success. All these approaches can be naturally used with our predictive distribution in eq. (19) and our estimated sensitivities and specificities in eqs. (12) and (13). However, since the use of AL in crowdsourcing is not the goal of this work, the comparison and development of AL techniques will not be explored here.

5 Experiments

In this section, we evaluate the performance of our methods and compare them with current state-of-the-art probabilistic crowdsourcing approaches. These include the GP-based *VGPCR* [Ruiz et al.,] and *Rodrigues* [Rodrigues et al., 2014]. We also include the most straightforward manner to apply a GP to the crowdsourcing setting, *GP-MV*, which consists of a standard GP classifier trained with the majority voting (MV) labels. Finally, to obtain a more thorough comparison, the classical LR-based methods *Raykar* [Raykar et al., 2010] and *Yan* [Yan et al., 2010] are also considered (recall the second paragraph in Section 1).

Since the main goal is to illustrate the scalability and performance of *RFF* and *VFF* in previously-prohibitive settings, we include two such datasets (where classical approaches must be trained with a subset). The first one, with 28000 training instances, comes from a real health-care activity-recognition problem. The second one, synthetic and with 100000 training samples, shows the potential of the proposed methods in even larger scale problems. Finally, two real datasets with 700 and 4999 training instances, respectively, are included to illustrate the performance of the proposed methods on small-scale problems. They cover different application domains such as audio recognition and sentiment analysis.

The predictive performance of the methods is compared using the area under the ROC curve (AUC). This metric deals well with imbalance scenarios (it penalizes errors in the minority class), and is independent on the threshold used for the final prediction. In order to compare the computational cost, the CPU time needed for both train and test steps will be reported. Please notice that the train CPU time includes the optimization of all the model parameters, including the Fourier frequencies for *VFF* (recall that *RFF* does not estimate them).

We implemented *RFF*, *VFF*, *VGPCR*, *GP-MV*, *Raykar*, and *Yan* in Matlab©, whereas a Matlab© implementation for *Rodrigues* can be downloaded from his website `http://www.fprodrigues.com`. All the code and datasets will be made available at `http://decsai.ugr.es/vip/software.html` upon acceptance of the paper. The experiments were run on the same machine Intel© Xeon© E5-2630 v4 @ 2.20GHz.

5.1 The sphere dataset

Sphere (Sensor Platform for HEalthcare in Residential Environment) is a recognition dataset where activity predictions are made based on RBG-D video, a tri-axial accelerometer, and environmental sensors [Twomey et al., 2016]. Data was collected from 10 people on two different occasions. There were 8 males and 2 females, with 8 between the ages of 18 to 29 and 2 within the ages of 30 to 39. Each participant was wearing a wrist-worn accelerometer and was asked to perform a series of scripted activities, taking around 25 to 30 minutes in total. These activities are categorized into ambulation actions (e.g. walking), posture actions (e.g. standing), and transitional actions (e.g. sit to stand). The script was carried out twice in full by each participant on different days.

Labeling this data is challenging, since the annotations are inherently noisy. For instance, the precise selection of start and end time is inherently ambiguous, as is the distinction among closely related actions (e.g., "bending" and "kneeling"). In order to mitigate these issues, the full dataset was annotated at least twice by a team of $R = 12$ annotators that were recruited and trained to annotate the set of activities. Our experiments consider the binary task of classifying between ambulatory and sedentary activities based on $D = 12$ statistical features (mean, minimum, maximum, standard deviation, variance) extracted from the acceleration data. This yields a final dataset with 31050 instances.

A set with 3050 instances was left for test[4], yielding a maximum number of 28000 training instances. In order to study the scalability of the compared methods, increasing training set sizes were considered, namely $N \in \{1000, 5000, 10000, 15000, 20000, 28000\}$. As classical GP-based methods are limited in practice to 10000-15000 training points, *VGPCR* and *GP-MV* could not be trained beyond $N = 15000$[5]. A special grid $N \in \{100, 500, 1000, 2500\}$ was used for *Rodrigues*, since it did not manage to converge properly and therefore its computational training cost exploded as N increased (as we will see in Figure 4). Different values of D_f (number of Fourier frequencies) were also considered for *RFF*, $D_f \in \{10, 50, 100, 200, 300, 400, 500, 600, 700\}$, and for *VFF*, $D_f \in \{1, 5, 10, 30, 50, 70, 90, 110, 130, 150\}$. Notice that, since *VFF* optimizes over the Fourier frequencies, it is natural to train it with smaller values of D_f.

The main ideas and interpretations will be provided in this section together with the most relevant figures. For completeness, additional information is included in the tables in A. Namely, Table 1 contains the test AUC for all the compared methods (except for *Rodrigues*, see its own Table 2). Mean and covariance over five independent runs[6] are shown. Analogously, Table 3 (Table 4 for

[4]Since true underlying labels were not available in this real problem, test instances were selected among those not having discrepancies between different annotators.

[5]When trying with $N = 20000$ for any of these methods, the RAM memory requirements exceeded the possibilities of the considered machine.

[6]These independent runs differ in the training subset if $N < 28000$, and also in the Fourier frequencies initialization for *RFF* and *VFF*.

Rodrigues) shows the CPU time needed for train, and Table 5 (Table 6 for *Rodrigues*) the CPU time needed for test.

First, let us examine the trade-off between generalization capability and training CPU time for the compared methods, see Figure 2. Notice that *Rodrigues* does not appear in the figure, since its predictive performance in this problem is around 0.5 in AUC, see Table 2. Among the rest of methods, (the x-axis of) Figure 2 shows a clear distinction between LR-based ones (*Raykar* and *Yan*, which are below 0.7 in AUC) and GP-based ones (the other four, which reach around 0.79). Of course, this is to be expected due to the more complex non-linear boundaries provided by GP-based methods, and reveals an underlying non-linear structure for the *sphere* dataset (otherwise, the gap between LR- and GP-based methods would be smaller).

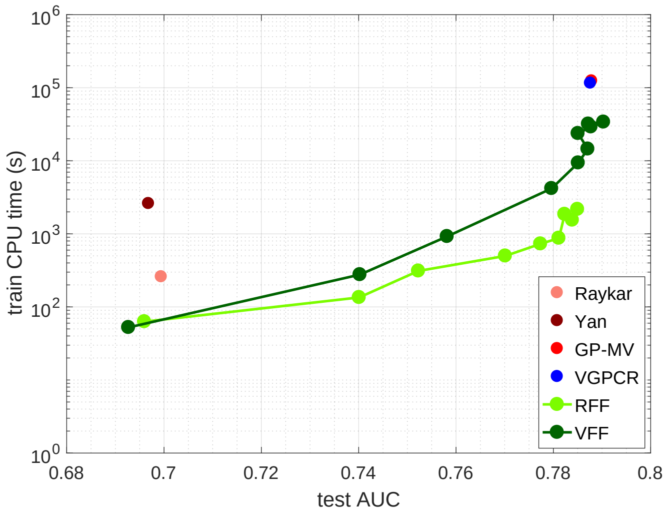


Figure 2: Trade-off between predictive performance (test AUC) and training cost (training CPU time) in the *sphere* dataset. Each method is trained with its maximum possible number of training points. For *RFF* and *VFF*, the full D_f-grids specified in the text are used. We observe that both *RFF* and *VFF* are significantly (more than three times) faster than the other competitive methods (*GP-MV* and *VGPCR*). Indeed, *VFF* manages to slightly outperform them, whereas *RFF* is around 50 times faster (and very close in predictive performance). Notice the logarithmic scale in the y-axis.

Now, among the four outstanding methods in terms of test AUC, the y-axis of Figure 2 shows a clear difference in the CPU time needed to train each one (recall the logarithmic scale in this axis). Namely, the proposed *RFF* and *VFF* are around three and fifty times faster than *GP-MV*/*VGPCR*, respectively. Notice also that, in terms of predictive performance, *RFF* is slightly below *GP-MV*/*VGPCR*, whereas *VFF* is slightly above them. This is the natural and logical behavior of the proposed pair of methods: since *VFF* optimizes over the Fourier frequencies, it is more computationally demanding than *RFF*; on the other hand, it manages to achieve more accurate results. This latter advantage will be more noticeable in the next experiment, where many more training points will be available to learn from.

Second, an essential aspect in real-world applications is the test CPU time, also known as *production time*. This amounts to the actual time that the system needs to make a prediction once it is trained. Depending on the problem at hand, test CPU time might be more relevant than train one, since the latter affects only once whereas the former is involved in any new prediction. In our case, a fast prediction is essential to develop a practical health-care activity-recognition system that can be deployed in real nursing or retirement homes. Thus, let us now analyze the compared methods in terms of test CPU time, see Figure 3.

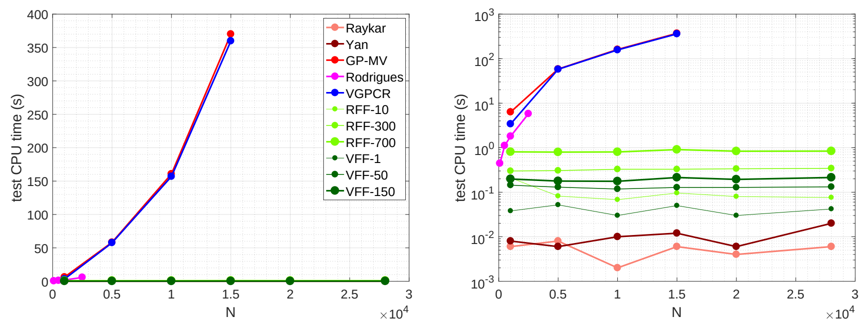


Figure 3: CPU time needed at test step (production time) as a function of the training set size in the *sphere* dataset. The linear (standard) scale in the left plot allows for a more intuitive perception of the methods scalability. The logarithmic scale in the right plot shows the differences between the fastest ones. Different representative values of D_f are shown for *RFF* and *VFF*. These are more than 350 times faster than *GP-MV* and *VGPCR* (the only competitive methods in terms of predictive performance). Moreover, as theoretically expected, their test cost is independent on N.

The difference between the proposed *RFF*/*VFF* and *GP-MV*/*VGPCR* (the only competitive methods in predictive performance, recall Figure 2) is overwhelming. Whereas the latter need more than 350 seconds to provide a prediction (for all the 3050 test instances), the former take less than 1 second. But, actually, the difference goes beyond these "absolute" numbers in this particular problem: whereas the test CPU time for the classical GP-based methods grows as $\mathcal{O}(N^2)$ with the training set size N, the novel *RFF*/*VFF* are independent on N (as expected from their theoretical formulation, recall Section 4). This fact makes classical GP-based probabilistic crowdsourcing methods prohibitive in practice for any medium-size real-world application where the production time plays an important role. Indeed, the new *RFF*/*VFF* might be the only choice for these scenarios (of course, as shown in the figure, test CPU time for LR-based methods is also independent on N, but their linear boundaries usually limit their applicability to real-world problems).

Among the proposed methods, notice that test CPU time grows with D_f (indeed, recall from Section 4 that their theoretical complexity is $\mathcal{O}(D_f^2)$). Therefore, and since *RFF* usually works with larger values of D_f, it is usually slightly slower than *VFF* in production time. Nonetheless, the difference is normally insignificant.

We have just seen that our methods are scalable in terms of test CPU time (in fact, they are independent on N). Let us now analyze the scalability with N in terms of training CPU time. Figure 2 already showed that *RFF*/*VFF* can be trained with $N = 28000$ instances significantly faster than *GP-MV*/*VGPCR* with $N = 15000$ (their maximum possible N). Now we examine more carefully the explicit dependence on N, see Figure 4.

This figure confirms in practice the theoretical linear-in-N training cost of the novel *RFF* and *VFF*, as well as the cubic of the classical GP-based methods (*GP-MV*, *Rodrigues*, *VGPCR*). This means

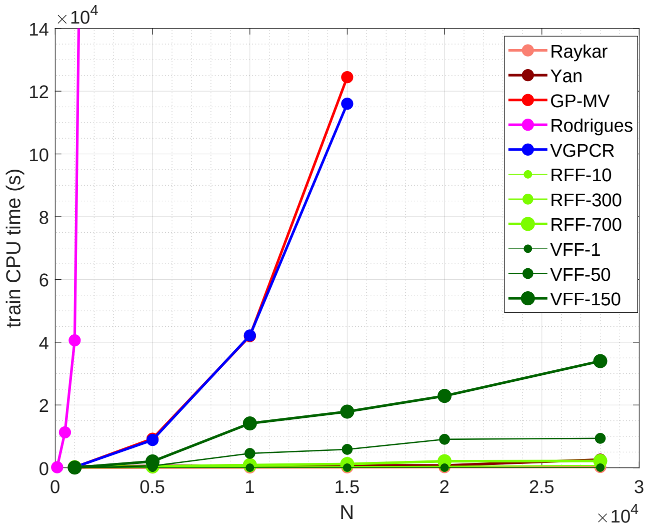


Figure 4: Training computational cost as a function of the training set size in the *sphere* dataset. Different representative values of D_f are shown for *RFF* and *VFF*. As theoretically expected, we observe a linear growth with N for the proposed methods, which makes them suitable for large-scale applications. On the contrary, classical GP-based methods cubic growth is prohibitive for that setting.

that our methods can still scale up to pretty larger datasets (in fact, in the next experiment they will reach $N = 10^5$), whereas classical ones are not suitable for such scenarios. Moreover, this training CPU time explosion is not the only limitation of classical approaches. Even if we did not have training time restrictions (which, of course, is not realistic in practical applications), classical methods need to deal with $N \times N$ matrices, which implies a $\mathcal{O}(N^2)$ RAM memory cost. However, *RFF* and *VFF* substitute these matrices with $2D_f \times 2D_f$ ones, removing the quadratic dependence on N.

Figure 4 also shows that, although both *RFF* and *VFF* are linear in N, the latter is computationally more expensive than the former (because of the Fourier frequencies optimization). Finally, the extraordinary long training CPU time of *Rodrigues* is explained because the convergence process oscillates and the maximum number of iterations is reached. This might be related to the different inference procedure.

Finally, it is interesting to analyze the role of D_f in *RFF* and *VFF*, that is, how it influences their predictive performance in practice. Figure 5 addresses this question. According to their theoretical formulation (recall second-to-last paragraph in Section 2.2), increasing D_f in *RFF* improves its approximation to a GP with SE kernel. However, in *VFF* it regulates the complexity of the model and, therefore, large values might lead to overfitting to the training set. The left plot in Figure 5 confirms the simple behavior of *RFF*. Analogously, the right plot shows a more complicated behavior for *VFF*, with a slightly decreasing tendency after reaching a maximum performance. This will be also observed in the next experiment. Finally, as is natural, the performance of both methods improves with the number of training instances N.

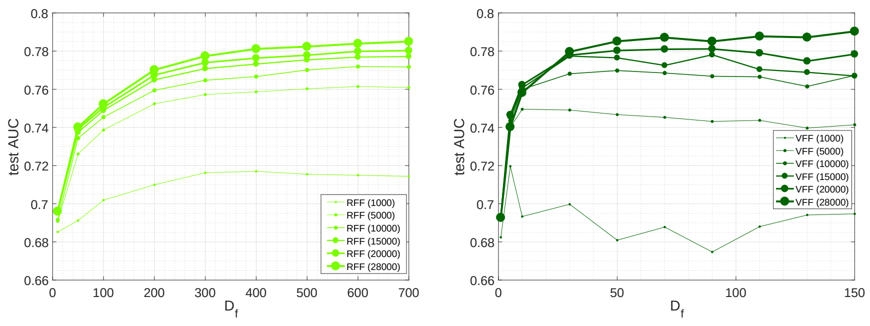


Figure 5: Predictive performance as a function of the number of Fourier frequencies used in *RFF* (left) and *VFF* (right) for the *sphere* dataset. In both cases, different training set sizes N are used. As theoretically hypothesized, *RFF* performance increases with D_f (regardless of N). However, *VFF* may suffer from over-fitting when D_f exceeds some complexity limit (which usually increases with the training set size N).

5.2 The cubes dataset

This experiment shows that the proposed methods scale up to even larger datasets, reaching $N = 100000$ training instances. Moreover, its synthetic nature allows us to i) have access to the true labels for test instances, and ii) have true *sensitivity* and *specificity* values for the annotators (and, therefore, evaluate the accuracy of their estimation). In order to analyze the differences with the previous experiment, we simulated a classification dataset with similar dimensionality, $D = 15$. Its structure is simple[7], and consists of a cube fitted inside a bigger one. Figure 7 shows the intuitive idea in $\mathbb{R}$ and $\mathbb{R}^2$.

More specifically, the *cubes* dataset is defined in $[-1, 1]^{15} \subset \mathbb{R}^{15}$, i.e. the 15 features are in the interval $[-1, 1]$. Training and test datasets are sampled from $[-1, 1]^{15}$ uniformly and independently. In order to define the probability that $\mathbf{x} \in [-1, 1]^{15}$ belongs to class 1, we resort to the so-called *infinity* norm, $||\mathbf{x}||_\infty = \max(|x_1|, \ldots, |x_{15}|)$. The level hyper-surfaces of this norm (i.e. the points that satisfy $||\mathbf{x}||_\infty = \text{ct.}$) are (the border of) the hyper-cubes inside $[-1, 1]^{15}$. Therefore, defining $\mathrm{p}(y = 1|\mathbf{x}) = \varphi(||\mathbf{x}||_\infty)$ with $\varphi : [0, 1] \to \mathbb{R}$ an increasing function, we obtain a dataset in which class 1 is mainly located in the border of the $[-1, 1]^{15}$ hyper-cube whereas class 0 is mainly located in its center. More specifically, we used the function $\varphi(w) = \max(0, 128(w - 0.5)^7)$, which is represented in Figure 6. The reasons for this choice is that $\varphi(0) = 0$, $\varphi(1) = 1$, and that it generates a balanced dataset (because the measures of the subsets $\{\mathbf{x} \in [-1, 1]^{15} : 0 \leq \varphi(||\mathbf{x}||_\infty) \leq 0.5\}$ and $\{\mathbf{x} \in [-1, 1]^{15} : 0.5 \leq \varphi(||\mathbf{x}||_\infty) \leq 1\}$ are very similar).

Then, five annotators, with sensitivities $\boldsymbol{\alpha} = \{0.9, 0.7, 0.8, 0.1, 0.9\}$ and specificities $\boldsymbol{\beta} = \{0.6, 0.8, 0.5, 0.2, 0.8\}$, are simulated. This produces both very reliable annotators (e.g. the fifth) and adversarial ones (e.g. the fourth). Training and test sets with 100000 and 200000 instances, respectively, were generated. As in the previous experiment, training sets of increasing size were considered in order to examine the scalability of the compared methods,

[7]The more complex the dataset structure is, the more relevant it is to have a large training dataset which can reveal more detailed patterns (in other words, if the structure of the dataset is really simple, say linear, the amount of training data needed to puzzle it out reduces). Therefore, by avoiding complex dataset, we prevent the introduction of artificial complexities that could favor the proposed methods.

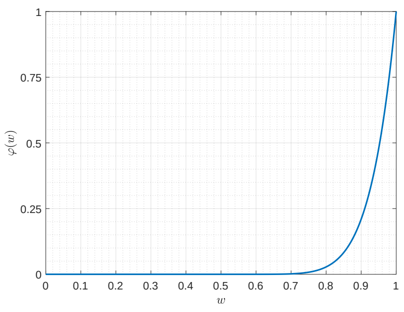


Figure 6: Graphical representation of φ, the function used to define the separation between classes in the synthetic dataset *cubes*.

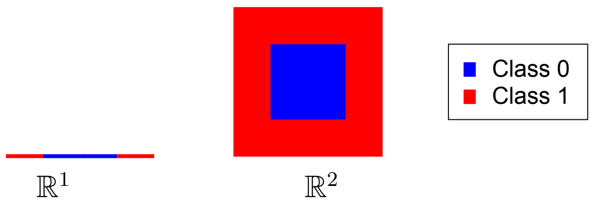


Figure 7: Structure of the two classes in the synthetic dataset *cubes*. It consists of a cube (more generally an *hyper-cube*, since we work in $\mathbb{R}^{15}$) fitted inside a bigger one. The probability of class 1 grows as we approach the border.

namely $N \in \{1000, 5000, 10000, 15000, 50000, 100000\}$. As before, classical GP-based methods *GP-MV* and *VGPCR* could not be trained beyond $N = 15000$, and *Rodrigues* used its own grid $N \in \{100, 500, 1000, 2500\}$ (although this time it did not exhibit convergence problems, its inference procedure is again slow in practice). Finally, the same grids $D_f \in \{10, 50, 100, 200, 300, 400, 500, 600, 700\}$ and $D_f \in \{1, 5, 10, 30, 50, 70, 90, 110, 130, 150\}$ were used for the number of Fourier frequencies in *RFF* and *VFF*, respectively. For completeness, all the raw results are shown in A, Tables 7 and 8 (test AUC), 9 and 10 (train CPU time), and 11 and 12 (test CPU time).

First, let us analyze the trade-off between generalization capability and training computational cost for the compared methods, see Figure 8. Again, in terms of predictive performance (see x-axis), we observe a clear distinction between LR-based methods (*Raykar*, *Yan*), which can only provide linear boundaries, and GP-related ones (*GP-MV*, *VGPCR*, *RFF*, *VFF*). *Rodrigues* is located in the middle since, although it also provides non-linear boundaries, its inference procedure limited its application to $N = 2500$ training instances (Figure 10 will analyze its lack of scalability).

Most importantly, the novel *RFF* and *VFF* exhibit the expected complementary behavior that was already observed in the previous experiment: whereas *RFF* is significantly more efficient and faster (it does not optimize over the Fourier frequencies), the flexibility of *VFF* allows it to capture additional relevant patterns and, therefore, achieve a superior predictive performance. Here, notice that *RFF* with $D_f = 200$ is around 500 times faster than *VGPCR* (the only competitive method in terms of predictive performance), while it is already (slightly) better in that aspect. Moreover, *VFF* reaches a 0.788 in test AUC, whereas classical approaches get a maximum of 0.704 (*VGPCR*).

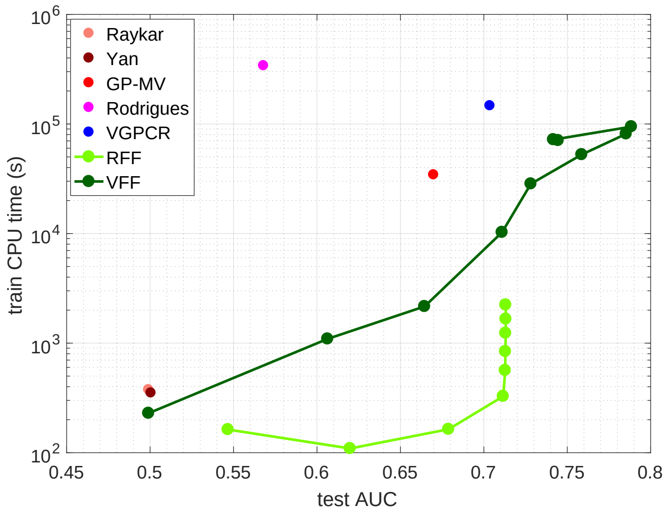


Figure 8: Trade-off between predictive performance (test AUC) and training cost (training CPU time) in the dataset *cubes*. Each method is trained with the maximum possible number of training points. For *RFF* and *VFF*, the D_f-grids specified in the text are used. We observe that *RFF* is more than 50 times faster than *VGPCR* (the other competitive method in terms of predictive performance). Moreover, it slightly outperforms *VGPCR* in that aspect. It is precisely in predictive performance where *VFF* achieves an overwhelming superiority (more than 8 points of test AUC better than *VGPCR*). Moreover, it is also faster than *VGPCR*. Notice the logarithmic scale in the y-axis.

It is also interesting to observe that *VFF* with $D_f = 1$ obtains a very similar result (in test AUC) to LR-based methods (*Raykar*, *Yan*). This is reasonable according to its formulation, since it is optimizing one Fourier frequency that plays the role of the linear regression coefficients. Moreover, in Figure 11 we will analyze how the number of Fourier frequencies D_f influences the behavior of *RFF* and *VFF*.

The second main idea is the overwhelming superiority of *RFF* and *VFF* in test CPU time, see Figure 9. As in the previous experiment, their theoretical independence on N is confirmed here in practice, as opposed to the $\mathcal{O}(N^2)$ growth of the classical GP-based crowdsourcing methods. This makes the latter prohibitive for any real-world problem where the test time plays an important role. Again, we observe that the test CPU time for *RFF* and *VFF* grows with D_f, as theoretically expected.

Third, Figure 10 analyzes the train CPU time scalability of the compared methods in this large dataset. As theoretically justified, recall also the previous experiment, we confirm here that *RFF*/*VFF* growth depends linearly on N, whereas classical GP-based approaches increase with N^3. In fact, notice that our slowest method (*VFF* with $D_f = 150$ and $N = 100000$) is twice faster than *VGPCR* with $N = 15000$ (the best among the competitors, and still 9 points below in predictive performance), and very similar to *VGPCR* with $N = 10000$. This suggests that our methods can be applied to even larger datasets, whereas classical GP-based ones have already achieved their maximum capabilities in a standard machine (recall their $\mathcal{O}(N^2)$ cost in RAM memory).

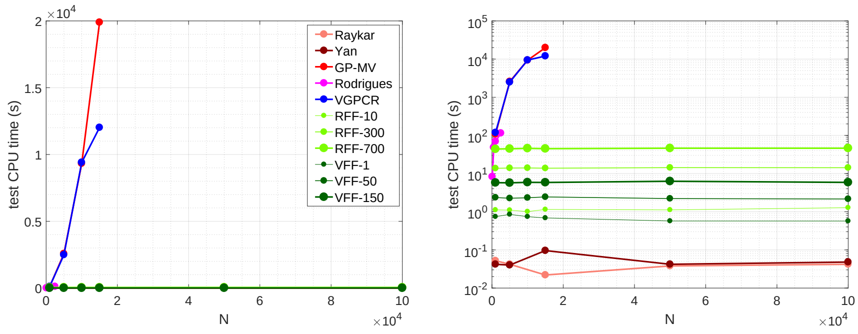


Figure 9: CPU time needed at test step (production time) as a function of the training set size in the *cubes* dataset. The linear (standard) scale in the left plot allows for a more intuitive perception of the methods scalability. The logarithmic scale in the right plot shows the differences between the fastest ones. Different representative values of D_f are shown for *RFF* and *VFF*. These are more than 250 times faster than *VGPCR* (the only competitive method in terms of generalization capability). Moreover, as theoretically expected, their test cost is independent on N.

Figure 10 also confirms that the Fourier frequencies optimization of *VFF* makes it significantly slower than *RFF*. Finally, notice the prohibitive growth of *Rodrigues*, which was conceived to deal with small datasets. Although the RAM memory requirements did not prevent us from training *Rodrigues* until $N = 15000$ (just like the rest of classical GP-based methods), we did not try beyond $N = 2500$ because of this very large training CPU time.

Let us now analyze how the number of Fourier frequencies D_f influences the predictive performance of the proposed methods, see Figure 11. Again, this is in accordance with their theoretical formulation (recall the second-to-last paragraph of Section 2.2) and the results obtained in the previous experiment. For *RFF*, it is simple: increasing D_f improves its approximation to a GP with SE kernel, and therefore enhances its predictive performance. For *VFF*, large values of D_f may lead to excessively complex models which overfit the training data and lose generalization capability. This produces the characteristic evolution observed in Figure 11, in which test AUC increases until an optimal value of D_f and then decreases. Naturally, a greater training set size N usually requires a greater complexity D_f to overfit.

Finally, since we have available the real *sensitivity* and *specificity* values of the annotators, it is interesting to assess the quality of the estimations provided by *RFF* and *VFF*. We describe the case $N = 100000$, since the main goal of this work is to deal with large-scale scenarios. The results obtained for $N < 100000$ were almost identical. Table 1 show the estimations of the proposed methods for *sensitivity* and *specificity*. We observe that both *RFF* and *VFF* provide very accurate estimations for all the annotators in both *sensitivity* and *specificity*. Namely, the maximum absolute difference in *sensitivity* is 0.0118 for *RFF*, and 0.0039 for *VFF*. In *specificity*, it is 0.0123 for *RFF*, and 0.0022 for *VFF*. Moreover, the accuracy in the estimation does not depend on D_f. Notice that this is natural from a theoretical viewpoint, since D_f (the number of Fourier frequencies) is not related to the model of the annotations $\mathbf{Y}$ given the latent true labels $\mathbf{z}$ (but to the model of $\mathbf{z}$ given the features $\mathbf{X}$, recall Figure 1).

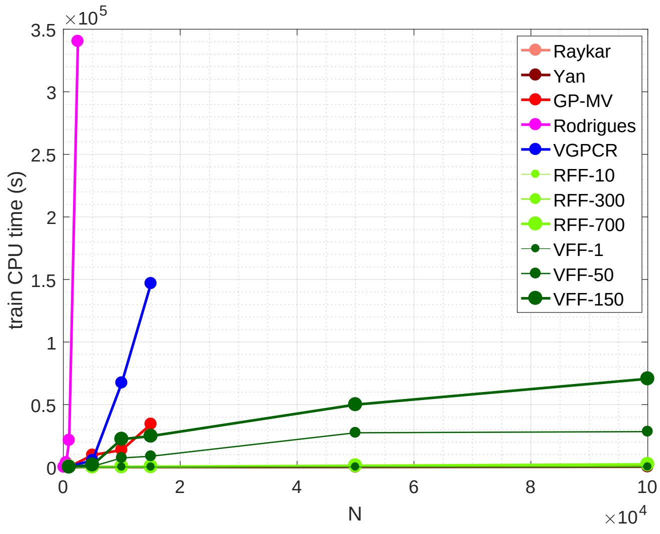


Figure 10: Training computational cost as a function of the training set size in dataset *cubes*. Different values of D_f are shown for *RFF* and *VFF*. As theoretically expected, we observe a linear growth with N for the proposed methods, which makes them suitable for large-scale applications. On the contrary, classical GP-based methods cubic growth is prohibitive for that setting. In fact, notice that our methods training with 100000 data points is faster than *VGPCR* (the best method among the competitors in terms of predictive performance) with 15000 instances (and already analogous to *VGPCR* with 10000 instances).

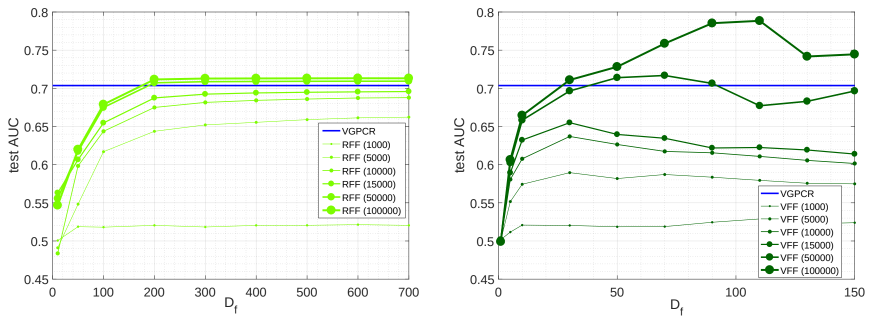


Figure 11: Predictive performance as a function of the number D_f of Fourier frequencies used in *RFF* (left) and *VFF* (right) for the *cubes* dataset. In both cases, different training set sizes N are used. As theoretically hypothesized, *RFF* performance increases with D_f (regardless of N). However, *VFF* may suffer from over-fitting when D_f exceeds some complexity limit (which usually increases with the training set size N).

5.3 Music genre dataset

In this experiment we use the Music Genre dataset presented in [Tzanetakis and Cook, 2002], which consists of 1000 fragments (30 secs. length) of songs. The goal is to distinguish between 10 music genres: classical, country, disco, hiphop, jazz, rock, blues, reggae, pop, and metal. We use an *one-vs-all* strategy to address this multi-class crowdsourcing classification problem, and

Sensitivity (α), RFF										
Annot.	Real	D_f								
		10	50	100	200	300	400	500	600	700
1	0.9	0.903	0.897	0.894	0.893	0.893	0.893	0.893	0.893	0.893
2	0.7	0.704	0.696	0.692	0.691	0.691	0.691	0.691	0.691	0.691
3	0.8	0.799	0.795	0.794	0.793	0.793	0.793	0.793	0.793	0.793
4	0.1	0.091	0.101	0.107	0.108	0.108	0.108	0.108	0.108	0.108
5	0.9	0.903	0.894	0.889	0.888	0.888	0.888	0.888	0.888	0.888

Specificity (β), RFF										
Annot.	Real	D_f								
		10	50	100	200	300	400	500	600	700
1	0.6	0.594	0.603	0.608	0.609	0.609	0.609	0.609	0.609	0.609
2	0.8	0.795	0.801	0.805	0.806	0.806	0.806	0.806	0.806	0.806
3	0.5	0.500	0.505	0.508	0.509	0.509	0.509	0.509	0.509	0.509
4	0.2	0.205	0.194	0.189	0.188	0.188	0.188	0.188	0.188	0.188
5	0.8	0.792	0.804	0.810	0.811	0.811	0.811	0.811	0.811	0.811

Sensitivity (α), VFF											
Annot.	Real	D_f									
		1	5	10	30	50	70	90	110	130	150
1	0.9	0.900	0.899	0.899	0.900	0.900	0.900	0.899	0.899	0.900	0.900
2	0.7	0.700	0.700	0.699	0.700	0.700	0.700	0.700	0.700	0.700	0.700
3	0.8	0.797	0.797	0.797	0.798	0.798	0.797	0.797	0.797	0.797	0.797
4	0.1	0.096	0.097	0.097	0.096	0.096	0.096	0.097	0.097	0.096	0.097
5	0.9	0.898	0.898	0.898	0.899	0.899	0.899	0.899	0.899	0.899	0.899

Specificity (β), VFF											
Annot.	Real	D_f									
		1	5	10	30	50	70	90	110	130	150
1	0.6	0.599	0.599	0.599	0.598	0.598	0.599	0.599	0.599	0.599	0.599
2	0.8	0.798	0.798	0.798	0.798	0.798	0.798	0.799	0.799	0.798	0.798
3	0.5	0.503	0.503	0.503	0.503	0.503	0.503	0.503	0.503	0.503	0.503
4	0.2	0.199	0.199	0.198	0.200	0.200	0.199	0.199	0.199	0.199	0.199
5	0.8	0.799	0.799	0.800	0.799	0.799	0.799	0.800	0.800	0.799	0.799

Table 1: Sensitivity and specificity estimations of *RFF* and *VFF* for the five annotators in the *cubes* dataset. Different values of D_f are used, and N is set to 100000. The results are the mean over five independent runs. We observe very accurate estimations, independently on D_f.

the results are averaged over the 10 experiments. For preprocessing and feature extraction, the Marsyas music information tool (`http://marsyas.info/`) was used to extract 124 features from the original dataset [Rodrigues et al., 2013]. These features include relevant technical metrics such us means and variances of timbral features, time-domain zero-crossings, spectral centroid, rolloff, flux, and Mel-Frequency Cepstral Coefficients (MFCC). The dataset contains 100 samples from each genre, which were randomly divided in 70 samples for training and 30 for testing. Crowd-sourcing labels were obtained with Amazon Mechanical Turk [Snow et al., 2008]. Each annotator listened to a subset of fragments and labeled them as one of the ten genres listed above. A total amount of 2945 labels were provided by 44 different annotators.

Although *RFF* and *VFF* are initially conceived for large-scale problems out of the reach of classical GP-based crowdsourcing methods, it is interesting to analyze their behavior when applied in a small (700 training instances) real crowdsourcing problem. Figure 12 shows the predictive performance (left) and training computational cost (right) for the compared methods, using different values of D_f for *RFF*/*VFF*. Since the training is much faster now, the same fine grid

$D_f = \{1, 5, 10, 20, 40, 60, 80, 100, 120, ..., 460, 480, 500\}$ was used for both methods. In all cases, the whole training set was used (i.e. $N = 700$).

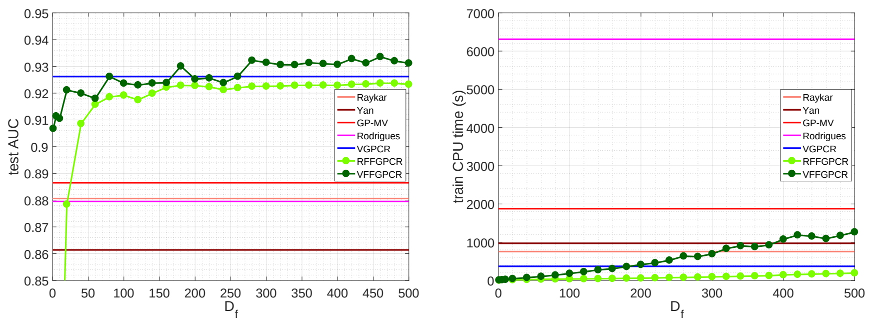


Figure 12: Left: predictive performance of the compared methods in the *Music* dataset. As theoretically expected, *RFF* constitutes an (efficient and scalable) approximation to *VGPCR*. However, *VFF* is a whole new crowdsourcing method which is also competitive with (even outperforms for some values of D_f) the state-of-the-art in small datasets. Right: training computational cost for the compared methods in the *Music* dataset. The approximation *RFF* stands out for its efficiency, whereas the new *VFF* is competitive with the rest of state-of-the-art approaches.

Figure 12 is in accordance with the theoretical formulation of the proposed methods. *RFF* is an (efficient and scalable) approximation to *VGPCR* and, therefore, its predictive performance is limited by that of *VGPCR* (as long as they are trained with the same set, like here; the advantage of *RFF* is precisely that i) it can scale up to larger datasets, and ii) it is faster than *VGPCR* even in this small set). Consequently, in practice, and provided that *VGPCR* can handle the dataset at hand, it should be preferred to *RFF* if we are only interested in generalization capability. If training CPU time is an issue, the right plot shows that *RFF* becomes an interesting more efficient alternative.

For its part, since it does not approximate a GP with SE kernel but learns its own one, *VFF* is not limited by the performance of *VGPCR* (i.e., it is a whole novel approach). In fact, the left plot shows that *VFF* can outperform *VGPCR* for many choices of D_f. In any case, we observe that *VFF* is a new probabilistic crowdsourcing method that is competitive with the current ones in previously-reachable datasets. Moreover, its scalability to larger datasets makes it push further the state-of-the-art in this field.

5.4 Sentence polarity dataset

Finally, in order to further assess the robustness of the proposed methods, let us evaluate their performance in an additional application domain: sentiment analysis. More specifically, the *sentence polarity* dataset is a real crowdsourcing problem that consists of 10427 sentences extracted from movie reviews in "Rotten Tomatoes" website `http://www.rottentomatoes.com/`. The goal is to decide whether a sentence corresponds to a *positive* or *negative* review. In Table 2 we show six sentences in the dataset. Preprocessing and feature extraction were carried out by Rodrigues *et al.* [Rodrigues et al., 2013], which resulted in feature vectors with 1200 components. The dataset is divided into train and test sets, with 4999 and 5428 samples, respectively. To obtain crowdsourcing labels, the train set was made available in Amazon Mechanical Turk. A total amount of 27746 labels were obtained from 203 different annotators.

Table 2: Examples of positive and negative samples in *sentence polarity* dataset.

Sentence	True Label
"An original gem about an obsession with time." "A taut, intelligent psychological drama." "Clever, brutal and strangely soulful movie."	"positive"
"This is amusing for about three minutes." "The film can depress you about life itself." "The pool drowned me in boredom."	"negative"

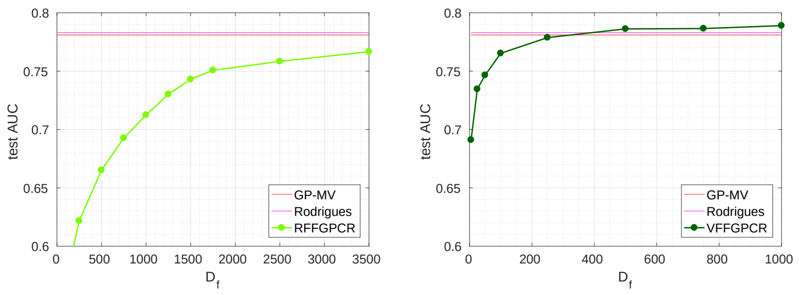


Figure 13: Predictive performance of *RFF* (left) and *VFF* (right) in the *sentence polarity* dataset, compared to the results reported in [Rodrigues et al., 2014]. As theoretically expected, the approximated *RFF* stays below classical methods in previously-reachable datasets, becoming closer as D_f grows. However, *VFF* is a whole new crowdsourcing algorithm which is competitive with (even outperforms for some values of D_f) the previous approaches in small datasets.

This dataset was used to evaluate *Rodrigues* method in [Rodrigues et al., 2014]. With 4999 training instances, it is within the reach of classical crowdsourcing approaches. Yet, let us check that our large-scale-oriented methods obtain consistent test results in this setting, comparing them to those reported in [Rodrigues et al., 2014, Table 3] (namely, 0.783 and 0.781 in test AUC for *GP-MV* and *Rodrigues* respectively).

First, since *RFF* approximates the SE kernel, its performance is expected to be below that of classical methods when the same amount of training instances is used (its power is, precisely, the ability to scale up to large datasets, as shown in previous experiments). Moreover, its performance should increase with the number D_f of Fourier frequencies used, since the SE kernel is recovered when $D_f \to \infty$. These hypotheses are confirmed in the results shown in Figure 13, left plot. Notice how the test performance grows with D_f and approaches that of the previously-reported methods. Observe also that high values of D_f have been used for *RFF* (up to 3500), since the high original dimension of the data (1200 features) requires a large number of Fourier frequencies to approximate the kernel.

Second, as *VFF* learns a new kernel (which might be better suited for the data at hand), its behavior is more difficult to predict from a theoretical viewpoint. In any case, it is expected to be competitive with previous approaches in non-large-scale settings. Indeed, Figure 13, right plot, shows that it outperforms the methods reported in [Rodrigues et al., 2014], reaching a test AUC of 0.7862 for $D_f = 500$ and 0.789 for $D_f = 1000$. Unlike *RFF*, observe that *VFF* achieves good results with significantly less Fourier frequencies, since they are optimized and therefore have a weaker dependence on the original dimension of the data.

6 Conclusions and future work

We have introduced two new scalable and efficient probabilistic crowdsourcing methods that can deal with previously-prohibitive datasets. Both are closely related to Gaussian Processes (GP), rely on the Fourier features approximation to achieve scalability, and utilize variational inference to estimate all the model unknowns. Unlike classical GP-based crowdsourcing approaches, whose training computational cost and RAM memory requirements grow as $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ respectively, the proposed methods scale up linearly with the training set size N in both aspects. This allows them to go beyond the GP practical limit of $N = 10000$, reaching datasets with up to $N = 100000$ samples. In turn, this allows them to outperform the previous approaches in terms of predictive performance, while still remaining more efficient and faster. Moreover, an overwhelming superiority is achieved in test computational cost (i.e. production time), where the novel methods are independent on N whereas classical ones grow as $\mathcal{O}(N^2)$. The novel *RFF* is a large-scale approximation to the recent GP-based crowdsourcing method *VGPCR*, while *VFF* is capable of estimating a new kernel (different to the squared exponential one that *VGPCR* is equipped with) tailored to the training data. In exchange, *VFF* is slower in practice, and more prone to overfitting. The proposed methods have proven to be the leading approach for medium-to-large scale problems. They are complementary approaches, and the final choice strongly depends on the application: whereas *RFF* guarantees a very fast and efficient training, *VFF* may achieve a higher predictive performance. Finally, the number of Fourier frequencies used, D_f, is an essential quantity in the novel approaches. As theoretically expected, more frequencies are always better for *RFF*, whereas it might lead to overfitting in *VFF*.

This is precisely the main future research line. A Bayesian treatment of the Fourier features in *VFF* could contribute to *weight* them across a wide posterior probability distribution, instead of relying on a single maximum likelihood estimation. An analogous idea has been successfully applied for regression in [Gal and Turner, 2015]. A multi-class crowdsourcing formulation and the use of inducing points (instead of Fourier features) to sparsify the underlying GP will also be explored in the future.

A Tables of results

This appendix contains all the results obtained in the previously-prohibitive datasets *sphere* and *cubes*. Tables 1, 3 and 5 show the test AUC, train CPU time and test CPU time, respectively, in the *sphere* dataset for all the methods except for *Rodrigues* (respectively, Tables 2, 4 and 6 are dedicated to *Rodrigues*). Analogously, Tables 7, 9 and 11 show the test AUC, train CPU time and test CPU time, respectively, in the *cubes* dataset for all the methods except for *Rodrigues* (respectively, Tables 8, 10 and 12 are dedicated to *Rodrigues*).

	N					
	1000	5000	10000	15000	20000	28000
Raykar	0.693±0.010	0.699±0.003	0.699±0.001	0.699±0.001	**0.700±0.001**	0.699±0.000
Yan	0.679±0.018	0.683±0.013	0.696±0.005	**0.698±0.003**	0.697±0.001	0.697±0.000
GP-MV	0.717±0.006	0.765±0.006	0.780±0.005	**0.788±0.004**	-	-
VGPCR	0.718±0.005	0.767±0.006	0.780±0.004	**0.788±0.003**	-	-
RFF-10	0.685±0.014	0.691±0.012	0.692±0.008	0.695±0.008	0.696±0.007	**0.696±0.008**
RFF-50	0.691±0.010	0.726±0.003	0.734±0.005	0.737±0.004	0.739±0.004	**0.740±0.004**
RFF-100	0.702±0.002	0.739±0.006	0.745±0.006	0.749±0.004	0.750±0.004	**0.752±0.003**
RFF-200	0.710±0.004	0.752±0.006	0.759±0.005	0.765±0.005	0.767±0.003	**0.770±0.002**
RFF-300	0.716±0.005	0.757±0.006	0.765±0.005	0.771±0.004	0.774±0.003	**0.777±0.003**
RFF-400	0.717±0.005	0.759±0.006	0.767±0.002	0.773±0.003	0.776±0.004	**0.781±0.003**
RFF-500	0.715±0.004	0.760±0.007	0.770±0.005	0.775±0.004	0.778±0.004	**0.782±0.003**
RFF-600	0.715±0.003	0.761±0.007	0.772±0.005	0.777±0.004	0.780±0.002	**0.784±0.001**
RFF-700	0.714±0.003	0.761±0.007	0.772±0.006	0.777±0.004	0.780±0.003	**0.785±0.001**
VFF-1	0.682±0.011	0.691±0.004	0.692±0.004	**0.693±0.003**	0.692±0.002	0.692±0.000
VFF-5	0.720±0.009	0.740±0.011	0.744±0.005	0.746±0.002	**0.747±0.010**	0.740±0.003
VFF-10	0.693±0.012	0.750±0.002	0.760±0.004	0.761±0.007	**0.762±0.007**	0.758±0.009
VFF-30	0.700±0.020	0.749±0.004	0.768±0.009	0.777±0.003	0.778±0.003	**0.780±0.004**
VFF-50	0.681±0.014	0.747±0.004	0.770±0.003	0.776±0.002	0.780±0.002	**0.785±0.001**
VFF-70	0.688±0.008	0.745±0.010	0.769±0.009	0.772±0.006	0.781±0.004	**0.787±0.002**
VFF-90	0.675±0.012	0.743±0.008	0.767±0.010	0.778±0.005	0.781±0.004	**0.785±0.005**
VFF-110	0.688±0.004	0.744±0.004	0.767±0.005	0.770±0.006	0.779±0.005	**0.788±0.002**
VFF-130	0.694±0.010	0.740±0.008	0.761±0.004	0.769±0.009	0.775±0.005	**0.787±0.004**
VFF-150	0.695±0.006	0.741±0.011	0.767±0.005	0.767±0.005	0.778±0.003	**0.790±0.003**

Table 1: *Sphere* dataset. Mean and standard deviation of test AUC (i.e. generalization capability) over five independent runs, except for *Rodrigues* method. For each method, the highest value is bolded.

	N			
	100	500	1000	2500
Rodrigues	**0.507±0.019**	0.490±0.009	0.498±0.010	0.495±0.003

Table 2: *Sphere* dataset. Mean and standard deviation of test AUC (i.e. generalization capability) over five independent runs for *Rodrigues*. The highest value is bolded.

	N					
	1000	5000	10000	15000	20000	28000
Raykar	13.0 ± 4.1	76.8 ± 8.2	103.9 ± 5.3	143.1 ± 7.0	199.9 ± 33.3	259.5 ± 48.5
Yan	425.4 ± 89.1	582.0 ± 114.4	695.0 ± 117.9	793.4 ± 121.8	746.8 ± 78.7	2609.0 ± 230.6
GP-MV	160.8 ± 18.2	9262.3 ± 1014.9	41953.7 ± 1725.8	124456.9 ± 3922.0	–	–
VGPCR	201.7 ± 67.0	8853.4 ± 595.5	42144.1 ± 2894.0	115996.4 ± 6836.8	–	–
RFF-10	10.9 ± 3.3	15.8 ± 4.6	19.6 ± 1.2	37.9 ± 17.5	43.5 ± 17.2	62.9 ± 28.6
RFF-50	15.5 ± 2.2	29.1 ± 4.1	53.2 ± 4.5	69.7 ± 7.0	95.8 ± 27.5	135.0 ± 62.4
RFF-100	25.3 ± 3.2	54.4 ± 4.3	112.6 ± 67.6	149.1 ± 46.7	132.5 ± 12.7	311.2 ± 185.2
RFF-200	42.1 ± 7.4	78.7 ± 3.7	154.6 ± 37.1	193.3 ± 9.6	262.2 ± 48.5	499.4 ± 158.9
RFF-300	56.2 ± 7.3	141.0 ± 31.5	264.6 ± 72.1	322.7 ± 18.5	449.0 ± 52.7	731.2 ± 193.5
RFF-400	89.0 ± 10.5	186.6 ± 8.1	344.5 ± 15.9	496.2 ± 39.0	602.3 ± 53.7	874.6 ± 166.6
RFF-500	111.0 ± 17.5	270.7 ± 11.8	548.4 ± 91.1	670.6 ± 33.3	866.9 ± 109.3	1857.0 ± 950.5
RFF-600	144.5 ± 23.6	348.6 ± 16.1	625.8 ± 4.0	1085.2 ± 349.2	1226.1 ± 189.1	1545.5 ± 137.8
RFF-700	184.0 ± 28.0	427.5 ± 31.6	866.6 ± 184.0	1198.5 ± 72.5	2079.3 ± 750.7	2171.6 ± 279.6
VFF-1	0.9 ± 0.3	7.5 ± 1.3	36.8 ± 7.6	50.0 ± 16.7	47.4 ± 15.4	52.5 ± 8.1
VFF-5	6.3 ± 1.7	44.9 ± 16.5	219.0 ± 62.1	322.9 ± 94.1	310.3 ± 64.2	277.1 ± 33.7
VFF-10	20.4 ± 5.6	105.5 ± 21.9	568.1 ± 171.9	642.3 ± 219.9	801.4 ± 170.3	922.8 ± 373.4
VFF-30	35.2 ± 3.7	337.0 ± 52.4	2478.7 ± 843.6	3596.8 ± 501.3	3418.9 ± 781.6	4187.2 ± 1159.9
VFF-50	47.3 ± 3.0	591.3 ± 71.3	4582.3 ± 1115.0	5862.0 ± 545.2	9075.0 ± 2773.2	9383.2 ± 1194.1
VFF-70	52.2 ± 4.8	815.4 ± 146.0	6786.7 ± 2407.5	9589.2 ± 1262.4	11574.1 ± 3645.6	14535.1 ± 2575.1
VFF-90	62.1 ± 2.5	1012.5 ± 165.2	9984.4 ± 2053.7	11908.1 ± 1549.2	15682.6 ± 3789.9	23716.8 ± 3709.6
VFF-110	75.8 ± 15.1	1320.9 ± 275.5	10240.8 ± 1503.8	14653.5 ± 4986.6	16443.2 ± 2009.3	29169.3 ± 6151.4
VFF-130	81.7 ± 7.0	1883.8 ± 442.7	13186.8 ± 2328.1	16614.2 ± 2184.3	22481.6 ± 4992.3	31939.8 ± 3517.6
VFF-150	80.1 ± 9.5	2027.5 ± 420.0	14133.8 ± 4156.4	17888.4 ± 2623.9	22889.4 ± 1343.3	33984.3 ± 1992.8

Table 3: *Sphere* dataset. Mean and standard deviation of CPU train time over five independent runs, except for *Rodrigues* method.

	N			
	100	500	1000	2500
Rodrigues	163.6±79.7	11285.1±7749.3	40609.9±13363.6	757778.4±455399.7

Table 4: *Sphere* dataset. Mean and standard deviation of CPU train time over five independent runs for *Rodrigues*.

	N					
	1000	5000	10000	15000	20000	28000
Raykar	0.006 ± 0.005	0.008 ± 0.004	0.002 ± 0.004	0.006 ± 0.008	0.004 ± 0.005	0.006 ± 0.008
Yan	0.008 ± 0.012	0.006 ± 0.005	0.010 ± 0.006	0.012 ± 0.010	0.006 ± 0.005	0.020 ± 0.011
GP-MV	6.322 ± 3.403	58.286 ± 11.929	160.682 ± 14.796	370.072 ± 33.214	–	–
VGPCR	3.412 ± 2.055	57.636 ± 13.149	156.912 ± 10.616	359.774 ± 7.491	–	–
RFF-10	0.210 ± 0.250	0.082 ± 0.016	0.068 ± 0.012	0.096 ± 0.026	0.080 ± 0.020	0.076 ± 0.010
RFF-50	0.138 ± 0.054	0.134 ± 0.023	0.120 ± 0.021	0.120 ± 0.015	0.134 ± 0.027	0.128 ± 0.015
RFF-100	0.152 ± 0.040	0.148 ± 0.015	0.146 ± 0.016	0.146 ± 0.019	0.160 ± 0.011	0.174 ± 0.019
RFF-200	0.258 ± 0.064	0.230 ± 0.020	0.234 ± 0.014	0.234 ± 0.027	0.248 ± 0.012	0.236 ± 0.019
RFF-300	0.300 ± 0.023	0.306 ± 0.010	0.328 ± 0.031	0.328 ± 0.044	0.336 ± 0.024	0.344 ± 0.031
RFF-400	0.414 ± 0.034	0.412 ± 0.024	0.420 ± 0.028	0.416 ± 0.024	0.422 ± 0.030	0.416 ± 0.021
RFF-500	0.522 ± 0.034	0.558 ± 0.051	0.556 ± 0.053	0.542 ± 0.053	0.532 ± 0.032	0.544 ± 0.036
RFF-600	0.642 ± 0.052	0.660 ± 0.043	0.648 ± 0.026	0.664 ± 0.051	0.644 ± 0.012	0.668 ± 0.024
RFF-700	0.806 ± 0.031	0.796 ± 0.047	0.802 ± 0.037	0.906 ± 0.114	0.834 ± 0.030	0.838 ± 0.031
VFF-1	0.038 ± 0.004	0.052 ± 0.031	0.030 ± 0.006	0.050 ± 0.045	0.030 ± 0.006	0.042 ± 0.020
VFF-5	0.092 ± 0.029	0.066 ± 0.016	0.060 ± 0.011	0.062 ± 0.015	0.070 ± 0.011	0.076 ± 0.036
VFF-10	0.088 ± 0.015	0.086 ± 0.014	0.074 ± 0.010	0.090 ± 0.017	0.100 ± 0.025	0.068 ± 0.007
VFF-30	0.116 ± 0.008	0.116 ± 0.014	0.116 ± 0.024	0.114 ± 0.008	0.104 ± 0.014	0.090 ± 0.009
VFF-50	0.144 ± 0.005	0.130 ± 0.014	0.118 ± 0.017	0.128 ± 0.019	0.128 ± 0.021	0.132 ± 0.017
VFF-70	0.160 ± 0.013	0.146 ± 0.030	0.134 ± 0.008	0.136 ± 0.012	0.128 ± 0.016	0.150 ± 0.017
VFF-90	0.144 ± 0.008	0.156 ± 0.015	0.138 ± 0.022	0.136 ± 0.012	0.152 ± 0.019	0.194 ± 0.027
VFF-110	0.178 ± 0.017	0.154 ± 0.015	0.166 ± 0.024	0.168 ± 0.019	0.164 ± 0.017	0.166 ± 0.015
VFF-130	0.178 ± 0.019	0.170 ± 0.024	0.178 ± 0.019	0.280 ± 0.177	0.164 ± 0.017	0.194 ± 0.015
VFF-150	0.198 ± 0.013	0.178 ± 0.007	0.176 ± 0.023	0.214 ± 0.020	0.194 ± 0.030	0.214 ± 0.019

Table 5: *Sphere* dataset. Mean and standard deviation of CPU test time (i.e. production time) over five independent runs, except for *Rodrigues* method.

	N			
	100	500	1000	2500
Rodrigues	0.444±0.153	1.112±0.576	1.806±0.428	5.788±0.396

Table 6: *Sphere* dataset. Mean and standard deviation of CPU test time (i.e. production time) over five independent runs for *Rodrigues*.

	N					
	1000	5000	10000	15000	50000	100000
Raykar	0.501±0.001	**0.501±0.001**	0.501±0.002	0.501±0.002	0.501±0.001	0.499±0.000
Yan	0.500±0.002	0.501±0.002	0.501±0.002	**0.501±0.002**	0.501±0.001	0.501±0.000
GP-MV	0.489±0.004	0.493±0.001	0.631±0.031	**0.670±0.010**	-	-
VGPCR	0.616±0.031	0.670±0.009	0.694±0.002	**0.704±0.001**	-	-
RFF-10	0.501±0.038	0.491±0.040	0.484±0.050	**0.563±0.005**	0.556±0.009	0.547±0.012
RFF-50	0.519±0.075	0.548±0.061	0.598±0.014	0.607±0.019	0.617±0.021	**0.620±0.020**
RFF-100	0.518±0.041	0.617±0.014	0.644±0.014	0.655±0.014	0.675±0.013	0.679±0.013
RFF-200	0.521±0.045	0.644±0.010	0.675±0.004	0.687±0.003	0.707±0.001	**0.712±0.001**
RFF-300	0.518±0.079	0.652±0.010	0.682±0.003	0.692±0.002	0.709±0.000	**0.713±0.000**
RFF-400	0.520±0.081	0.655±0.008	0.684±0.002	0.694±0.002	0.709±0.000	**0.713±0.000**
RFF-500	0.521±0.079	0.659±0.008	0.686±0.002	0.695±0.002	0.709±0.001	**0.713±0.000**
RFF-600	0.521±0.079	0.661±0.007	0.687±0.002	0.695±0.002	0.709±0.000	**0.713±0.000**
RFF-700	0.520±0.080	0.662±0.007	0.688±0.002	0.696±0.002	0.709±0.000	**0.713±0.000**
VFF-1	**0.502±0.009**	0.501±0.001	0.501±0.002	0.501±0.002	0.501±0.001	0.499±0.000
VFF-5	0.512±0.005	0.552±0.008	0.581±0.007	0.589±0.004	0.602±0.003	**0.606±0.002**
VFF-10	0.521±0.011	0.574±0.009	0.608±0.013	0.632±0.007	0.658±0.003	**0.664±0.001**
VFF-30	0.520±0.009	0.590±0.012	0.637±0.007	0.655±0.007	0.697±0.002	**0.711±0.004**
VFF-50	0.519±0.004	0.582±0.007	0.626±0.008	0.640±0.002	0.714±0.014	**0.728±0.010**
VFF-70	0.519±0.009	0.587±0.012	0.617±0.005	0.635±0.004	0.717±0.013	**0.759±0.011**
VFF-90	0.525±0.008	0.584±0.005	0.616±0.006	0.622±0.008	0.706±0.017	**0.785±0.011**
VFF-110	0.529±0.010	0.579±0.011	0.611±0.005	0.622±0.003	0.677±0.007	**0.788±0.014**
VFF-130	0.522±0.006	0.576±0.004	0.606±0.003	0.619±0.002	0.683±0.014	**0.742±0.021**
VFF-150	0.524±0.008	0.575±0.009	0.602±0.008	0.614±0.004	0.696±0.010	**0.745±0.032**

Table 7: Mean and standard deviation of test AUC (i.e. generalization capability) over five independent runs, except for *Rodrigues* method. For each method, the highest value is bolded. Dataset: *cubes*.

	N			
	100	500	1000	2500
Rodrigues	0.501±0.003	0.528±0.006	0.544±0.011	**0.568±0.011**

Table 8: Mean and standard deviation of test AUC (i.e. generalization capability) over five independent runs for *Rodrigues*. The highest value is bolded. Dataset: *cubes*

	N					
	1000	5000	10000	15000	50000	100000
Raykar	9.7 ± 1.3	79.7 ± 8.9	113.1 ± 10.3	143.2 ± 11.0	245.5 ± 15.1	376.4 ± 7.8
Yan	29.4 ± 11.9	72.1 ± 10.3	80.9 ± 7.3	89.0 ± 2.7	174.5 ± 1.7	351.6 ± 6.3
GP-MV	293.6 ± 306.5	9662.4 ± 6025.3	13738.3 ± 3357.0	34390.9 ± 503.5	–	–
VGPCR	639.4 ± 601.4	5390.8 ± 256.0	67373.8 ± 11094.2	146773.5 ± 31379.0	–	–
RFF-10	11.5 ± 5.2	34.3 ± 10.1	55.6 ± 17.6	61.8 ± 19.0	114.4 ± 11.8	162.7 ± 32.0
RFF-50	31.2 ± 15.8	20.2 ± 20.5	31.2 ± 21.2	31.8 ± 9.4	58.0 ± 3.1	109.3 ± 6.8
RFF-100	22.2 ± 21.4	19.3 ± 4.3	30.0 ± 4.7	39.0 ± 7.7	90.7 ± 6.0	163.7 ± 13.0
RFF-200	32.4 ± 28.8	22.9 ± 2.3	43.4 ± 2.9	59.0 ± 4.4	169.3 ± 3.9	327.3 ± 8.8
RFF-300	71.9 ± 51.6	37.5 ± 6.2	64.1 ± 3.2	87.5 ± 7.6	298.4 ± 13.3	564.2 ± 9.2
RFF-400	122.8 ± 83.0	56.2 ± 10.0	96.3 ± 9.2	137.1 ± 17.5	426.5 ± 35.3	839.5 ± 12.4
RFF-500	195.7 ± 157.4	96.8 ± 29.2	153.3 ± 27.6	215.9 ± 33.1	613.0 ± 42.5	1233.3 ± 30.2
RFF-600	277.5 ± 202.9	130.5 ± 34.9	210.8 ± 41.7	284.1 ± 53.1	813.6 ± 87.2	1661.3 ± 48.9
RFF-700	399.1 ± 304.0	167.1 ± 59.4	260.4 ± 52.9	402.0 ± 38.3	1127.6 ± 83.1	2234.7 ± 30.9
VFF-1	3.0 ± 1.2	14.4 ± 3.0	116.2 ± 34.1	124.4 ± 33.4	184.9 ± 27.6	229.2 ± 37.5
VFF-5	8.7 ± 1.1	130.6 ± 35.2	791.1 ± 497.6	608.5 ± 150.7	893.3 ± 298.6	1092.7 ± 389.3
VFF-10	21.0 ± 5.1	173.1 ± 25.3	1480.2 ± 273.7	1460.6 ± 488.5	1726.8 ± 288.2	2160.9 ± 583.5
VFF-30	48.7 ± 11.4	399.2 ± 53.2	5039.8 ± 1208.3	6020.8 ± 1427.5	7832.2 ± 2032.4	10261.9 ± 3239.3
VFF-50	61.5 ± 11.2	638.8 ± 117.8	7388.7 ± 2115.0	8720.7 ± 2936.3	27493.7 ± 15726.5	28411.7 ± 11184.2
VFF-70	70.1 ± 12.3	809.5 ± 59.5	11065.6 ± 3504.0	12704.8 ± 4026.7	32159.1 ± 11781.5	52529.9 ± 13354.0
VFF-90	77.3 ± 15.2	1292.3 ± 247.0	12141.8 ± 3080.7	17052.8 ± 4579.0	33085.1 ± 8452.8	81205.7 ± 11555.1
VFF-110	86.6 ± 18.5	1662.5 ± 383.0	19402.3 ± 5435.1	21365.0 ± 5921.8	27909.0 ± 6329.1	94428.9 ± 33174.8
VFF-130	101.7 ± 31.5	2207.8 ± 404.7	20206.9 ± 5174.2	20747.1 ± 4638.8	46683.2 ± 24627.5	71927.6 ± 18965.0
VFF-150	110.1 ± 28.5	1837.7 ± 331.1	22362.4 ± 6844.6	24831.1 ± 6319.1	49974.3 ± 16474.8	70633.5 ± 32070.8

Table 9: Mean and standard deviation of CPU train time over five independent runs, except for *Rodrigues* method. Dataset: *cubes*

	N			
	100	500	1000	2500
Rodrigues	67.0±23.2	3646.7±919.1	21471.2±1611.1	340339.7±48419.3

Table 10: Mean and standard deviation of CPU train time over five independent runs for *Rodrigues*. Dataset: *cubes*

	N					
	1000	5000	10000	15000	50000	100000
Raykar	0.052 ± 0.007	0.042 ± 0.010	0.040 ± 0.011	0.022 ± 0.004	0.038 ± 0.007	0.042 ± 0.004
Yan	0.042 ± 0.004	0.040 ± 0.009	0.036 ± 0.005	0.096 ± 0.117	0.042 ± 0.004	0.048 ± 0.012
GP-MV	108.418 ± 2.068	2572.136 ± 72.938	9325.962 ± 27.316	19895.748 ± 661.243	–	–
VGPCR	118.398 ± 2.023	2489.040 ± 103.969	9407.908 ± 230.540	12019.232 ± 3653.075	–	–
RFF-10	1.118 ± 0.121	1.102 ± 0.172	1.010 ± 0.171	1.148 ± 0.146	1.118 ± 0.137	1.276 ± 0.176
RFF-50	2.332 ± 0.359	2.294 ± 0.177	2.436 ± 0.257	2.478 ± 0.103	2.388 ± 0.165	2.360 ± 0.119
RFF-100	3.900 ± 0.336	4.228 ± 0.334	4.460 ± 0.332	4.322 ± 0.223	4.288 ± 0.221	4.396 ± 0.274
RFF-200	7.976 ± 0.290	8.298 ± 0.218	8.360 ± 0.433	8.656 ± 0.450	8.898 ± 0.282	9.068 ± 0.286
RFF-300	13.738 ± 1.553	13.958 ± 0.416	14.048 ± 0.755	13.818 ± 0.405	14.314 ± 0.617	14.252 ± 0.339
RFF-400	19.826 ± 1.940	19.732 ± 0.838	20.694 ± 0.913	20.406 ± 0.746	21.114 ± 0.701	20.890 ± 0.302
RFF-500	27.160 ± 2.920	27.116 ± 2.087	27.178 ± 0.520	27.354 ± 2.912	29.026 ± 1.834	28.456 ± 0.630
RFF-600	34.788 ± 3.241	35.050 ± 3.523	35.674 ± 1.310	37.036 ± 1.893	37.512 ± 0.362	36.772 ± 1.957
RFF-700	43.928 ± 3.407	44.634 ± 1.275	45.790 ± 2.873	44.960 ± 2.203	46.230 ± 0.872	46.236 ± 2.113
VFF-1	0.744 ± 0.066	0.852 ± 0.180	0.744 ± 0.108	0.688 ± 0.106	0.572 ± 0.097	0.570 ± 0.061
VFF-5	0.824 ± 0.125	0.872 ± 0.122	0.942 ± 0.188	0.744 ± 0.082	0.770 ± 0.119	0.758 ± 0.068
VFF-10	1.092 ± 0.106	1.154 ± 0.159	1.102 ± 0.115	0.908 ± 0.101	0.840 ± 0.117	1.016 ± 0.300
VFF-30	1.804 ± 0.100	1.662 ± 0.156	1.694 ± 0.051	1.712 ± 0.118	1.554 ± 0.083	1.634 ± 0.122
VFF-50	2.362 ± 0.076	2.254 ± 0.088	2.326 ± 0.092	2.442 ± 0.184	2.214 ± 0.152	2.160 ± 0.189
VFF-70	2.842 ± 0.107	2.760 ± 0.154	2.974 ± 0.065	2.992 ± 0.151	3.004 ± 0.079	2.956 ± 0.190
VFF-90	3.576 ± 0.132	3.504 ± 0.259	3.552 ± 0.130	3.688 ± 0.279	3.872 ± 0.352	3.622 ± 0.178
VFF-110	4.206 ± 0.175	4.062 ± 0.213	4.264 ± 0.181	4.556 ± 0.443	4.514 ± 0.313	4.284 ± 0.174
VFF-130	5.072 ± 0.430	4.868 ± 0.306	5.030 ± 0.237	5.008 ± 0.158	5.332 ± 0.166	5.240 ± 0.293
VFF-150	5.770 ± 0.321	5.716 ± 0.122	5.892 ± 0.281	5.850 ± 0.267	6.290 ± 0.915	5.878 ± 0.425

Table 11: Mean and standard deviation of CPU test time (i.e. production time) over five independent runs, except for *Rodrigues* method. Dataset: *cubes*

	N			
	100	500	1000	2500
Rodrigues	8.294±4.201	48.528±34.869	69.630±31.114	114.842±88.935

Table 12: Mean and standard deviation of CPU test time (i.e. production time) over five independent runs for *Rodrigues*. Dataset: *cubes*

References

[Albarqouni et al., 2016] Albarqouni, S., Baur, C., Achilles, F., Belagiannis, V., Demirci, S., and Navab, N. (2016). AggNet: Deep Learning From Crowds for Mitosis Detection in Breast Cancer Histology Images. *IEEE Transactions on Medical Imaging*, 35(5):1313–1321.

[Bauer et al., 2016] Bauer, M., van der Wilk, M., and Rasmussen, C. E. (2016). Understanding probabilistic sparse gaussian process approximations. In *Advances in neural information processing systems (NIPS)*, pages 1533–1541.

[Besler et al., 2016] Besler, E., Ruiz, P., Molina, R., and Katsaggelos, A. K. (2016). Classification of multiple annotator data using variational gaussian process inference. In *European Signal Processing Conference (EUSIPCO)*, pages 2025–2029.

[Bishop, 2006] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Blei et al., 2017] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

[Dawid and Skene, 1979] Dawid, A. and Skene, A. (1979). Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Journal of the Real Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

[Gal and Turner, 2015] Gal, Y. and Turner, R. (2015). Improving the gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *International Conference on Machine Learning (ICML)*, pages 655–664.

[Goebel et al., 2018] Goebel, R., Chander, A., Holzinger, K., Lecue, F., Akata, Z., Stumpf, S., Kieseberg, P., and Holzinger, A. (2018). Explainable ai: The new 42? In Holzinger, A., Kieseberg, P., Tjoa, A. M., and Weippl, E., editors, *Machine Learning and Knowledge Extraction*, Cham. Springer International Publishing.

[Hensman et al., 2013] Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 282–290.

[Howe, 2006] Howe, J. (2006). The rise of crowdsourcing. *Wired Magazine*, 14(6).

[Jaakkola, 2000] Jaakkola, T. S. (2000). Tutorial on variational approximation methods. In *Advanced Mean Fields Methods: Theory and Practice*, pages 129–159. MIT Press.

[Lawrence et al., 2003] Lawrence, N. D., Seeger, M., and Herbrich, R. (2003). Fast sparse gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems (NIPS)*, pages 625–632. MIT Press.

[Lázaro-Gredilla et al., 2010] Lázaro-Gredilla, M., Quiñonero Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. R. (2010). Sparse spectrum gaussian process regression. *Journal of Machine Learning Research*, 11:1865–1881.

[Minka, 2001a] Minka, T. P. (2001a). Expectation propagation for approximate bayesian inference. In *Seventeenth Conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc.

[Minka, 2001b] Minka, T. P. (2001b). *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Cambridge, MA, USA.

[Morales-Álvarez et al., 2018] Morales-Álvarez, P., Pérez-Suay, A., Molina, R., and Camps-Valls, G. (2018). Remote sensing image classification with large-scale gaussian processes. *IEEE Transactions on Geoscience and Remote Sensing*, 56(2):1103–1114.

[Quiñonero-Candela and Rasmussen, 2005] Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959.

[Rahimi and Recht, 2008] Rahimi, A. and Recht, B. (2008). Random features for large-scale kernel machines. In *Advances in neural information processing systems (NIPS)*, pages 1177–1184.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT.

[Raykar et al., 2010] Raykar, V., Yu, S., Zhao, L., Hermosillo Valadez, G., Florin, C., Bogoni, L., and Moy, L. (2010). Learning from crowds. *Journal of Machine Learning Research*, 11(Apr):1297–1322.

[Rodrigues et al., 2017] Rodrigues, F., Lourenco, M., Ribeiro, B., and Pereira, F. (2017). Learning supervised topic models for classification and regression from crowds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2409–2422.

[Rodrigues and Pereira, 2018] Rodrigues, F. and Pereira, F. (2018). Deep learning from crowds. In *Conference on Artificial Intelligence (AAAI)*, pages 81–92.

[Rodrigues et al., 2013] Rodrigues, F., Pereira, F., and Ribeiro, B. (2013). Learning from multiple annotators: Distinguishing good from random labelers. *Pattern Recognition Letters*, 34(12):1428–1436.

[Rodrigues et al., 2014] Rodrigues, F., Pereira, F., and Ribeiro, B. (2014). Gaussian process classification and active learning with multiple annotators. In *International Conference on Machine Learning (ICML)*, pages 433–441.

[Ruiz et al., 2014] Ruiz, P., Mateos, J., Camps-Valls, G., Molina, R., and Katsaggelos, A. (2014). Bayesian active remote sensing image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 52(4):2186–2196.

[Ruiz et al., 2016] Ruiz, P., Molina, R., and Katsaggelos, A. (2016). Joint data filtering and labeling using gaussian processes and alternating direction method of multipliers. *IEEE Transactions on Image Processing*, 25(7):3059–3072.

[Ruiz et al.,] Ruiz, P., Morales-Álvarez, P., Molina, R., and Katsaggelos, A. Learning from crowds with variational Gaussian Processes. *Accepted for publication in Pattern Recognition (2018).*

[Smyth et al., 1995] Smyth, P., Fayyad, U. M., Burl, M. C., Perona, P., and Baldi, P. (1995). Inferring ground truth from subjective labelling of Venus images. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1085–1092.

[Snelson and Ghahramani, 2006] Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems (NIPS)*, pages 1257–1264.

[Snow et al., 2008] Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. (2008). Cheap and fast, but is it good?: evaluating non-expert annotations for natural language tasks. In *Conference on Empirical Methods in Language Processing (EMNLP)*, pages 254–263.

[Titsias, 2009] Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, volume 5, pages 567–574.

[Twomey et al., 2016] Twomey, N., Diethe, T., Kull, M., Song, H., Camplani, M., Hannuna, S., Fafoutis, X., Zhu, N., Woznowski, P., Flach, P., and Craddock, I. (2016). The sphere challenge: Activity recognition with multimodal sensor data. *arXiv preprint arXiv:1603.00797.*

[Tzanetakis and Cook, 2002] Tzanetakis, G. and Cook, P. (2002). Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302.

[Yan et al., 2011] Yan, Y., Fung, G., Rosales, R., and Dy, J. (2011). Active learning from crowds. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1161–1168.

[Yan et al., 2010] Yan, Y., Rosales, R., Fung, G., Schmidt, M., Hermosillo Valadez, G., Bogoni, L., Moy, L., and Dy, J. (2010). Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 932–939.

[Yan et al., 2014] Yan, Y., Rosales, R., Fung, G., Subramanian, R., and Dy, J. (2014). Learning from multiple annotators with varying expertise. *Machine Learning*, 95(3):291–327.

[Yang et al., 2018] Yang, J., Drake, T., Damianou, A., and Maarek, Y. (2018). Leveraging crowdsourcing data for deep active learning an application: Learning intents in alexa. In *Proc. of the 2018 World Wide Web Conference*, pages 23–32. International World Wide Web Conferences Steering Committee.

[Zevin et al., 2017] Zevin, M., Coughlin, S., Bahaadini, S., Besler, E., Rohani, N., Allen, S., et al. (2017). Gravity spy: integrating advanced ligo detector characterization, machine learning, and citizen science. *Classical and Quantum Gravity*, 34(6):064003.

[Zhang et al., 2016] Zhang, J., Wu, X., and Sheng, V. S. (2016). Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46(4):543–576.

CHAPTER 6

Inducing points, Normalizing Flows and Inference Networks for Gaussian Processes based crowdsourcing

6.1 Publication details

Authors: Pablo Morales-Álvarez, Pablo Ruiz, Scott Coughlin, Rafael Molina, Aggelos K. Katsaggelos
Title: Scalable Variational Gaussian Processes for Crowdsourcing: Glitch Detection in LIGO
Reference: Under review at IEEE Transactions on Pattern Analysis and Machine Intelligence (first revision submitted on April, 30th 2020).
Status: Under review (preprint published in arXiv, see https://arxiv.org/abs/1911.01915)
Quality indices:

- Impact Factor (JCR 2019): 17.861
- Rank: 1/136 (Q1 and D1) in Computer Science, Artificial Intelligence and 2/266 (Q1 and D1) in Engineering, Electrical and Electronic.

6.2 Main contributions

- First, we extend the use of sparse GP approaches based on inducing points to the crowdsourcing setting. This allows for mini-batch training, achieving scalability to datasets of virtually any size. The derived training objective naturally integrates the crowdsourcing and sparse GP ones. Variational inference is used to estimate the model parameters.
- Second, we leverage Normalizing Flows (NF) to describe a more complex variational posterior on the inducing point values (which originally is considered as a Gaussian distribution). NFs sequentially apply invertible transformations with cheap Jacobian to obtain more expressive distributions.

- Third, we generalize the use of inference networks for scalable inference of GPs to the crowdsourcing scenario. This is a very recent approach that allows for expressive predictive distributions which are not limited by the number of inducing points used.
- Finally, we evaluate the three approaches on a challenging dataset to detect glitches in the search for gravitational waves within the laureate LIGO project. The inducing points based method outperforms all previous algorithms, NFs slightly improves the results a bit further, and inference networks consolidate the outperformance even for smaller values of M (which is now the size of the measurement set, used to match the predictions of the predictive and the true distributions at each step).

SCALABLE VARIATIONAL GAUSSIAN PROCESSES FOR CROWDSOURCING: GLITCH DETECTION IN LIGO

Pablo Morales-Álvarez
Dept. of Computer Science and AI
University of Granada, Spain

Pablo Ruiz
Dept. of Electrical Engineering and Computer Science
Northwestern University, USA

Scott Coughlin
Center for Interdisciplinary Exploration and Research in Astrophysics
Northwestern University, USA

Rafael Molina
Dept. of Computer Science and AI
University of Granada, Spain

Aggelos K. Katsaggelos
Dept. of Electrical Engineering and Computer Science
Northwestern University, USA

ABSTRACT

In the last years, crowdsourcing is transforming the way classification training sets are obtained. Instead of relying on a single expert annotator, crowdsourcing shares the labelling effort among a large number of collaborators. For instance, this is being applied in the laureate Laser Interferometer Gravitational Waves Observatory (LIGO), in order to detect glitches which might hinder the identification of true gravitational-waves. The crowdsourcing scenario poses new challenging difficulties, as it has to deal with different opinions from a heterogeneous group of annotators with unknown degrees of expertise. Probabilistic methods, such as Gaussian Processes (GP), have proven successful in modeling this setting. However, GPs do not scale up well to large data sets, which hampers their broad adoption in real-world problems (in particular LIGO). This has led to the very recent introduction of deep learning based crowdsourcing methods, which have become the state-of-the-art for this type of problems. However, the accurate uncertainty quantification provided by GPs has been partially sacrificed. This is an important aspect for astrophysicists in LIGO, since a glitch detection system should provide very accurate probability distributions of its predictions. In this work, we first leverage a standard sparse GP approximation (SVGP) to develop a GP-based crowdsourcing method that factorizes into mini-batches. This makes it able to cope with previously-prohibitive data sets. This first approach, which we refer to as Scalable Variational Gaussian Processes for Crowdsourcing (SVGPCR), brings back GP-based methods to a state-of-the-art level, and excels at uncertainty quantification. SVGPCR is shown to outperform deep learning based methods and previous

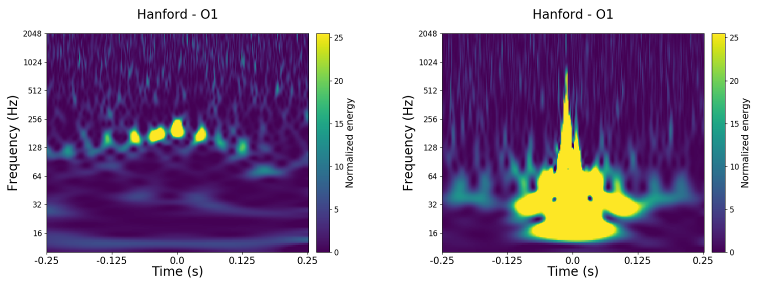


Figure 1: Two examples of glitches observed by the LIGO detector. The fifteen types considered in this work will be carefully described in section 3, see also figure 3.

probabilistic ones when applied to the LIGO data. Its behavior and main properties are carefully analyzed in a controlled experiment based on the MNIST data set. Moreover, recent GP inference techniques are also adapted to crowdsourcing and evaluated experimentally.

1 Introduction

Crowdsourcing, also known as citizen science, is revolutionizing the way real-world data sets are obtained nowadays [Irwin, 2018, Guerrini et al., 2018]. Traditionally, the task of labelling has been accomplished by a single expert annotator in a process that is time-consuming, expensive and difficult to scale. The proliferation of web services such as Amazon Mechanical Turk (`www.mturk.com`) and Figure-Eight (`www.figure-eight.com`, formerly Crowdflower) allows for outsourcing this process to a distributed workforce that can collaborate virtually, sharing the effort among a huge number of annotators [Snow et al., 2008, Buhrmester et al., 2011]. This approach is rapidly growing in popularity, and is being applied to many different fields such as medical imaging [Albarqouni et al., 2016], genetics [Saez-Rodriguez et al., 2016], remote sensing [Fritz et al., 2017], topic modelling [Rodrigues et al., 2017], and object segmentation [Heim et al., 2018].

A very recent application of crowdsourcing in the field of astrophysics is the GravitySpy project [Zevin et al., 2017], which aims at detecting glitches in the Laser Interferometer Gravitational Waves Observatory (LIGO). The LIGO collaboration is one of the most exciting and recognized scientific international initiatives [Abramovici et al., 1992]. It was awarded the 2017 Physics Nobel Prize for the first empirical detection of a gravitational-wave in September 2015 [B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration), 2016]. These waves are ripples in the fabric of spacetime, their existence was theoretically predicted by Einstein's General Relativity theory in 1916, and open a whole new way to explore the universe (beyond the electromagnetic signals available so far) [Castelvecchi and Witze, 2016]. However, the LIGO detector is equipped with extremely delicate technology, which is sensitive to many different sources of noise. This produces a wide variety of glitches, see figure 1, which make the detection of true gravitational-waves difficult. The goal of GravitySpy is to leverage citizen science to label the large data set of glitches produced by LIGO, and then develop a machine learning system (based on crowdsourcing methods) to help astrophysicists classify them [Zevin et al., 2017].

The crowdsourcing scenario introduces new challenges in machine learning, such as combining the unknown expertise of annotators, dealing with disagreements on the labelled samples, or detecting the existence of spammer and adversarial annotators [Sheng et al., 2008, Donmez and Carbonell, 2008]. The first approaches to deal with multiple-annotated data used to rely on some kind of label aggregation mechanism prior to training. The most straightforward one is majority voting, which assumes that every annotator is equally reliable. More elaborated methods consider the biases of the different annotators, yielding a better calibrated set of training labels, see [Dawid and Skene, 1979] (which is usually considered the first crowdsourcing work) and [Ipeirotis et al., 2010, Whitehill et al., 2009]. In all these cases, the idea is to obtain a set of clean true labels, which are then fed to the preferred standard (no-crowdsourcing) classification algorithm.

However, recent works show that jointly modelling the classifier and the annotators behavior leads to superior performance, since the features provide information to puzzle out the true labels [Raykar et al., 2010, Yan et al., 2014]. In this joint model, Bayesian methods based on Gaussian Processes (GPs) have proved extremely successful to accurately quantify uncertainty [Rodrigues et al., 2014, Ruiz et al., 2019, Rodrigues et al., 2017]. However, in real-world applications they have been gradually replaced by deep learning based approaches [Albarqouni et al., 2016, Rodrigues and Pereira, 2018, Guan et al., 2018], since GPs do not scale well to large data sets [Rodrigues et al., 2017, Rasmussen and Williams, 2006]. As a result, the sound probabilistic formulation of GPs has been sacrificed. However, large scale problems could greatly benefit from such a solid modelling. In particular, in order to develop a reliable glitch detection system, astrophysicists with the GravitySpy project are particularly interested in the Bayesian formulation given by GPs [Zevin et al., 2017]. Therefore, their scalability issues must be addressed.

GP is a popular Bayesian non-parametric model for supervised learning that excels at uncertainty quantification [Rasmussen and Williams, 2006]. Due to the kernel matrix inversion, its computational cost at training is $\mathcal{O}(N^3)$, where N is the size of the training set. To overcome this problem, different sparse GP approximations have been proposed in the last years [Bauer et al., 2016]. Many of them rely on the notion of inducing points, a reduced set of M ($M \ll N$) instances which condense the information contained in the whole data set [Snelson and Ghahramani, 2006, Titsias, 2009][1]. One of the most widespread methods is the Scalable Variational Gaussian Processes (SVGP) method [Hensman et al., 2015a], which uses Variational Inference (VI) [Blei et al., 2017] and is inspired by the earlier regression method [Hensman et al., 2013]. SVGP can be trained through mini-batches and yields a training computational cost of $\mathcal{O}(N_b M^2)$, with N_b the mini-batch size. This allows SVGP to handle data sets of practically any size [Hensman et al., 2015a]. In fact, very interesting rates of convergence have been obtained recently for sparse GP regression problems [Burt et al., 2019]. They provide theoretical guarantees of scalability, showing that the increase of M can be kept slower than N, specially for large datasets. Although these proofs do not directly apply for classification or crowdsourcing problems, in practice we will also observe that a small value of M can deal with the LIGO problem successfully.

[1]There exist other sparse GP approximations which alternatively rely on Fourier features [Morales-Álvarez et al., 2018], and which have been already used for crowdsourcing problems [Morales-Álvarez et al., 2019]. In the experiments, the proposed method will be shown to clearly outperform these alternative approaches too.

In this work, we we start by extending the well-established sparse GP approximation behind SVGP to the multiple-annotated crowdsourcing setting. Importantly, the form of the Evidence Lower Bound (ELBO) is still suitable for Stochastic VI [Hoffman et al., 2013], which allows for training through mini-batches. To the best of our knowledge, this allows GPs to be used for crowdsourcing problems of virtually any size for the first time. This method is refered to as Scalable (or Sparse) Variational Gaussian Processes for Crowdsourcing (SVGPCR). The annotators noise model is also fully Bayesian, described by per-user confusion matrices which are assigned Dirichlet priors. The underlying true labels are modelled in a probabilistic manner as well. VI is used to approximate the posterior distribution of the model.

In order to deal with the LIGO data, SVGPCR is modelled and implemented as a multi-class method. The implementation is based on GPflow, a very popular GP library that benefits from GPU acceleration through TensorFlow [Matthews et al., 2017]. Three sets of experiments are provided. First, a controlled crowdsourcing problem specified for MNIST illustrates the main properties and behavior of SVGPCR. Among these, we may highlight its accurate identification of annotators' expertise degree, reconstruction of the real underlying label, and how the number of inducing points influences its performance. Secondly, SVGPCR is compared against previous probabilistic crowdsourcing methods in a relevant binary LIGO problem[2]. SVGPCR stands out as the best performing approach, thanks to its innovative scalability through mini-batches. Third, SVGPCR is shown to outperform state-of-the-art DL-based methods in the full LIGO data set, specially in terms of test likelihood, due to the more robust uncertainty control.

Once SVGPCR has been successfully developed, we explore more recent GP inference techniques in the LIGO problem (beyond the standard SVGP). For instance, GPs can be combined with inference networks [Shi et al., 2019] and the amortized setting of Variational Autoencoders [Casale et al., 2018]. More expressive posteriors can be described by Normalizing Flows [Rezende and Mohamed, 2015, Papamakarios et al., 2019], and the use of sampling is an alternative to VI [Hensman et al., 2015b, Tegner et al., 2018]. While the number of possibilities is large and worth exploring, here we restrict ourselves to two approaches. First, we resort to Normalizing Flows to allow for richer posterior within the SVGPCR model. The results are only slightly better than SVGPCR, which suggests that a unimodal Gaussian posterior might be enough in this application. Then, we further modify the model by extending to crowdsourcing the use of inference networks for GPs [Shi et al., 2019]. In this case, the results are clearly better for low values of M (measurement points), which is precisely one of the main benefits of [Shi et al., 2019].

The rest of the paper is organized as follows. Section 2 describes the proposed model and inference procedure. Section 3 presents the LIGO data available in the GravitySpy project. The experimental results for SVGPCR are discussed in Section 4. More recent inference approaches are explored in Section 5. Finally, Section 6 contains some remarks and future outlook.

2 Probabilistic model and inference

This section introduces the theoretical formulation of the proposed method. Figure 2 shows a graphical representation of the proposed model, which will be useful here.

[2]Most of these previous probabilistic crowdsourcing approaches were originally proposed for binary problems, and the code is available accordingly.

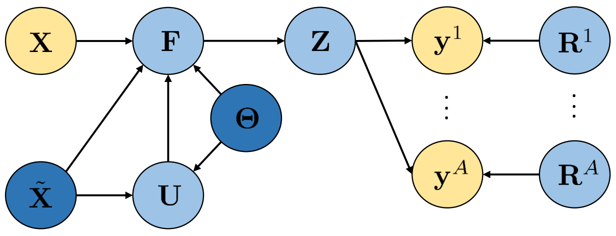


Figure 2: Probabilistic graphical model for SVGPCR. Observed variables are depicted in yellow, and those to be estimated in blue. In the latter case, the intensity indicates whether the estimation is through a posterior distribution (light blue) or a point value (dark).

2.1 The model

In a crowdsourcing problem with K classes, we observe training data $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{Y}_n^a) : n = 1, \dots, N;\ a \in A_n\}$, where $\mathbf{x}_n \in \mathbb{R}^D$ are the training features, and $\mathbf{Y}_n^a$ is the set[3] of annotations provided by the a-th annotator for the n-th instance. That is, each $\mathbf{y} \in \mathbf{Y}_n^a$ is an one-hot encoded vector in $\{\mathbf{e}_1, \dots, \mathbf{e}_K\}$ that represents the k-th class (i.e., all elements of $\mathbf{e}_k$ are zero but the k-th one, which is one). There are N data points, A annotators, and $A_n \subseteq \{1, \dots, A\}$ contains the annotators that labelled the n-th instance. All training instances will be grouped in $\mathbf{X} = \{\mathbf{x}_n : n = 1, \dots, N\}$, and analogously all annotations in $\mathbf{Y} = \{\mathbf{Y}_n^a : n = 1, \dots, N; a \in A_n\}$.

As with previous approaches [Rodrigues et al., 2017, Raykar et al., 2010, Yan et al., 2014, Rodrigues et al., 2014, Ruiz et al., 2019], the proposed model assumes an (unknown) real label for each instance, $\mathbf{z}_n \in \{\mathbf{e}_1, \dots, \mathbf{e}_K\}$. The actual annotations depend on this real label and the degree of expertise of each annotator, which is modelled by the confusion matrix $\mathbf{R}^a = (r_{ij}^a)_{1 \le i,j \le K}$. Each $r_{ij}^a \in [0, 1]$ represents the probability that the a-th annotator labels as class i an instance whose real class is j. Notice that this matrix must add up to one by columns. Mathematically, this is given by

$$\mathrm{p}(\mathbf{Y}_n^a | \mathbf{z}_n, \mathbf{R}^a) = \prod_{\mathbf{y} \in \mathbf{Y}_n^a} \mathbf{y}^\intercal \mathbf{R}^a \mathbf{z}_n. \tag{1}$$

If $\mathbf{y} = \mathbf{e}_i$ and $\mathbf{z}_n = \mathbf{e}_j$, the product $\mathbf{y}^\intercal \mathbf{R}^a \mathbf{z}_n$ yields r_{ij}^a. Other variants of crowdsourcing likelihoods could be explored. For instance, the values in $\mathbf{R}^a$ could depend on the input $\mathbf{x}$ (i.e. the annotator degree of expertise might vary depending on the specific features of the input).

Assuming that all annotators label the different instances independently, we have

$$\mathrm{p}(\mathbf{Y} | \mathbf{Z}, \mathbf{R}) = \prod_{n=1}^{N} \prod_{a \in A_n} \mathrm{p}(\mathbf{Y}_n^a | \mathbf{z}_n, \mathbf{R}^a), \tag{2}$$

where $\mathbf{Z} = \{\mathbf{z}_n : n = 1, \dots, N\}$ and $\mathbf{R} = \{\mathbf{R}^a : a = 1, \dots, A\}$ group the corresponding individual variables, and $\mathrm{p}(\mathbf{Y}_n^a | \mathbf{z}_n, \mathbf{R}^a)$ is given by eq. (1).

The prior distribution for the annotators behavior is modelled through (independent) Dirichlet distributions, which are conjugate to the categorical one in eq. (1) [Bishop, 2006]. This yields

$$\mathrm{p}(\mathbf{R}) = \prod_{a=1}^{A} \prod_{j=1}^{K} \mathrm{p}(\mathbf{r}_j^a) = \prod_{a=1}^{A} \prod_{j=1}^{K} \mathrm{Dir}(\mathbf{r}_j^a | \alpha_{1j}^a, \dots, \alpha_{Kj}^a), \tag{3}$$

[3]Notice that annotators are allowed to label the same instance more than once (possibly with different labels). This happens in a few cases in the LIGO data.

where $\mathbf{r}_j^a = (r_{1j}^a, \ldots, r_{Kj}^a)^\intercal$ denotes the j-th column of the confusion matrix $\mathbf{R}^a$. The hyperparameters $\boldsymbol{\alpha} = \{\alpha_{ij}^a : i,j = 1,\ldots,K,\ a = 1,\ldots,A\}$ codify any prior belief on the behavior of the annotators. The use of a prior protects from the so-called *black swan paradox* [Murphy, 2012, Section 3.3.4.1], i.e. when trying to estimate $\mathbf{r}_j^a$ for an annotator who provided no annotations for samples in the j-th class. As we will see in eq. (15), the prior affects the training through a KL divergence term that couples it to the posterior. If no prior knowledge is available, the default choice $\alpha_{ij}^a = 1$ corresponds to uniform distributions. This is the most standard scenario, and the one that is considered here. Yet, notice that this would not be the optimal choice if there were very few annotations, since the prior could hide the effect of the observed data. This is not the case of LIGO, where there is an average of over 500 annotations per user (see Section 3). Additionally, the prior allows for more informative modelling. For instance, an annotator who is known to mix up two classes can be modelled by setting the corresponding values of α_{ij}^a.

For each instance, the true underlying label $\mathbf{z}_n$ is modelled through K latent variables $f_{n1}, \ldots, f_{nK}$. Both parts are related by means of the likelihood model

$$\mathrm{p}(\mathbf{z}_n | f_{n1}, \ldots, f_{nK}) = \mathbf{z}_n^\intercal \boldsymbol{\nu}(f_{n1}, \ldots, f_{nK}), \tag{4}$$

where $\boldsymbol{\nu}(f_{n1}, \ldots, f_{nK})$ is any vector with K positive components that add up to 1. In this work we will use the popular robust-max likelihood [Hernández-Lobato et al., 2011]. It is given by $\boldsymbol{\nu}(a_1, \ldots, a_K) = (\nu_1, \ldots, \nu_K)$, with $\nu_i = 1 - \varepsilon$ for $i = \arg\max(a_1, \ldots, a_K)$ and $\nu_j = \varepsilon/(K-1)$ for $j \neq i$. The value of ε is fixed to the default value 10^{-3}. This likelihood is implemented in the GPflow library [Matthews et al., 2017], and in practice it can be substituted by any other one available in GPflow. For instance the soft-max likelihood, which generalizes the sigmoid likelihood to multi-class, i.e. $\nu_i = e^{a_i} / \sum_j e^{a_j}$.

Assuming that the underlying real labels for the different instances are independent given the latent variables, it is

$$\mathrm{p}(\mathbf{Z}|\mathbf{F}) = \prod_{n=1}^{N} \mathrm{p}(\mathbf{z}_n | \mathbf{f}_{n,:}), \tag{5}$$

where $\mathrm{p}(\mathbf{z}_n|\mathbf{f}_{n,:})$ is given by eq. (4), and $\mathbf{F}$ gathers the K latent variables for the N instances. Specifically, $\mathbf{F}$ is a $N \times K$ matrix, whose (n,k) term is the value of the k-th latent variable for the n-th instance. As usual, the n-th row of $\mathbf{F}$ is denoted by $\mathbf{f}_{n,:}$, and the k-th column by $\mathbf{f}_k$.

Finally, independent GP priors are utilized for the latent variables $f_1, \ldots, f_K$. This yields the joint prior

$$\mathrm{p}(\mathbf{F}|\boldsymbol{\Theta}, \mathbf{X}) = \prod_{k=1}^{K} \mathrm{p}(\mathbf{f}_k | \boldsymbol{\theta}_k, \mathbf{X}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{f}_k | \mathbf{0}, \mathbf{K}_{\boldsymbol{\theta}_k}(\mathbf{X}, \mathbf{X})), \tag{6}$$

where $\boldsymbol{\theta}_k$ are the kernel hyperparameters for the k-th GP. In this work we will use the well-known squared exponential kernel, $k(\mathbf{x}, \mathbf{y}) = \gamma \cdot \exp(-||\mathbf{x} - \mathbf{y}||^2/(2\sigma^2))$, which has the hyperparameters of variance γ and length-scale σ. However, as before, GPflow allows us to use any other kernel [Matthews et al., 2017].

In summary, the full probabilistic model is given by

$$\mathrm{p}(\mathbf{Y}, \mathbf{Z}, \mathbf{F}, \mathbf{R}|\boldsymbol{\Theta}) = \mathrm{p}(\mathbf{Y}|\mathbf{Z}, \mathbf{R})\mathrm{p}(\mathbf{R})\mathrm{p}(\mathbf{Z}|\mathbf{F})\mathrm{p}(\mathbf{F}|\boldsymbol{\Theta}), \tag{7}$$

with the four factors on the right hand side defined through eqs. (2), (3), (5) and (6), respectively. Here, the dependency on the observed features $\mathbf{X}$ has been omitted for simplicity.

In order to introduce the sparse GP approximation, let us expand this model by introducing M inducing points for each GP. Namely, each GP prior $\mathrm{p}(\mathbf{f}_k)$ can be naively rewritten as the marginal of $\mathrm{p}(\mathbf{f}_k, \mathbf{u}_k)$, where $\mathbf{u}_k = (u_{1k}, \ldots, u_{Mk})$ are M inducing points. These represent the value of the k-th GP on M new locations called inducing inputs, $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_M\}$, just like $\mathbf{f}_k$ does for $\mathbf{X}$, i.e. $\mathrm{p}(\mathbf{u}_k) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\boldsymbol{\theta}_k}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}))$. [4] Analogously to $\mathbf{F}$, we write $\mathbf{U}$ for the $M \times K$ matrix gathering all the inducing points, whose rows and columns are denoted by $\mathbf{u}_{m,:}$ and $\mathbf{u}_k$ respectively. The idea is that, in sparse GPs, these inducing points are used to summarize the information from the training data. By taking $M \ll N$, computational tractability can be achieved.

Then, if the joint GP $\mathrm{p}(\mathbf{f}_k, \mathbf{u}_k)$ is factorized as $\mathrm{p}(\mathbf{f}_k|\mathbf{u}_k)\mathrm{p}(\mathbf{u}_k)$, the model in eq. (7) can be analogously rewritten as

$$\mathrm{p}(\mathbf{Y}, \mathbf{Z}, \mathbf{F}, \mathbf{U}, \mathbf{R}|\boldsymbol{\Theta}) = \mathrm{p}(\mathbf{Y}|\mathbf{Z}, \mathbf{R}) \cdot \mathrm{p}(\mathbf{R}) \cdot \mathrm{p}(\mathbf{Z}|\mathbf{F}) \cdot \mathrm{p}(\mathbf{F}|\mathbf{U}, \boldsymbol{\Theta}) \cdot \mathrm{p}(\mathbf{U}|\boldsymbol{\Theta}), \tag{8}$$

where the Gaussian conditional $\mathrm{p}(\mathbf{F}|\mathbf{U}, \boldsymbol{\Theta})$ is given by $\prod_k \mathcal{N}(\mathbf{f}_k|\mathbf{B}_k\mathbf{u}_k, \mathbf{K}_{\boldsymbol{\theta}_k}(\mathbf{X}, \mathbf{X}) - \mathbf{B}_k\mathbf{K}_{\boldsymbol{\theta}_k}(\tilde{\mathbf{X}}, \mathbf{X}))$ and $\mathbf{B}_k = \mathbf{K}_{\boldsymbol{\theta}_k}(\mathbf{X}, \tilde{\mathbf{X}})[\mathbf{K}_{\boldsymbol{\theta}_k}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}})]^{-1}$.

It is worth stressing that, by marginalizing out $\mathbf{U}$, this model is equivalent to the one in eq. (7). This is important because sparse GP approximations are grouped into two big categories: those which approximate the model and perform exact inference (like FITC [Snelson and Ghahramani, 2006]), and those which keep the model unaltered and introduce the approximation at the inference step. Our approach, like SVGP, belongs to the second group, and the approximation is carried out next.

2.2 Variational inference

Given the model in eq. (8), an exact solution would involve calculating the marginal likelihood $\mathrm{p}(\mathbf{Y}|\boldsymbol{\Theta})$, in order to estimate the optimal kernel hyperparameters $\tilde{\boldsymbol{\Theta}}$ and then obtain the posterior $\mathrm{p}(\mathbf{Z}, \mathbf{F}, \mathbf{U}, \mathbf{R}|\mathbf{Y}, \tilde{\boldsymbol{\Theta}})$. However, integrating out $\mathbf{Z}$, $\mathbf{F}$, $\mathbf{U}$ and $\mathbf{R}$ is analytically intractable, and we resort to variational inference to approximate the computations [Blei et al., 2017].

The core of variational inference is the following decomposition of the log marginal likelihood, which is straightforward and holds for any distribution $\mathrm{q}(\mathbf{z}, \mathbf{F}, \mathbf{U}, \mathbf{R})$[5]:

$$\begin{aligned}\log \mathrm{p}(\mathbf{Y}|\boldsymbol{\Theta}) &= \mathrm{KL}(\mathrm{q}(\mathbf{Z}, \mathbf{F}, \mathbf{U}, \mathbf{R})||\mathrm{p}(\mathbf{Z}, \mathbf{F}, \mathbf{U}, \mathbf{R}|\mathbf{Y}, \boldsymbol{\Theta})) \\ &\quad + \underbrace{\int \mathrm{q}(\mathbf{Z}, \mathbf{F}, \mathbf{U}, \mathbf{R}) \log \frac{\mathrm{p}(\mathbf{Y}, \mathbf{Z}, \mathbf{F}, \mathbf{U}, \mathbf{R}|\boldsymbol{\Theta})}{\mathrm{q}(\mathbf{Z}, \mathbf{F}, \mathbf{U}, \mathbf{R})} \mathrm{d}\mathbf{Z}\mathrm{d}\mathbf{F}\mathrm{d}\mathbf{U}\mathrm{d}\mathbf{R}.}_{\text{ELBO}}\end{aligned} \tag{9}$$

This distribution q must be understood as an approximation to the true posterior $\mathrm{p}(\mathbf{Z}, \mathbf{F}, \mathbf{U}, \mathbf{R}|\mathbf{Y}, \boldsymbol{\Theta})$. The second term in the right hand side of eq. (9) is called the Evidence Lower Bound (ELBO), since it is a lower bound for the model evidence or log marginal likelihood $\log \mathrm{p}(\mathbf{Y}|\boldsymbol{\Theta})$ (recall that the first term, the KL divergence, is always non-negative, and is zero if and only if both distributions coincide).

The idea of variational inference is to propose a parametric form for q. Then, the ELBO in eq. (9) is maximized with respect to these new variational parameters, the kernel hyperparameters $\boldsymbol{\Theta}$, and

[4]Notice that the inducing locations $\tilde{\mathbf{X}}$ do not depend on k. Although different inducing locations could be used for each GP, in practice they are usually considered the same. However, the inducing points $\mathbf{u}_k$ do depend on k, as each GP models a different function.

[5]Observe that, in order to "lighten" the notation, we use the integral symbol also for the discrete variable $\mathbf{Z}$.

the inducing locations $\tilde{\mathbf{X}}$ (which are not usually considered fixed). Notice that, by maximizing the ELBO, we are at the same time considering the log marginal likelihood $\log \mathrm{p}(\mathbf{Y}|\boldsymbol{\Theta})$ and the KL divergence between q and the real posterior (just solve for the ELBO in eq. (9)). Thus, variational inference converts the problem of posterior distribution approximation into an optimization one [Blei et al., 2017], which in practice is addressed through optimization algorithms such as Adam Optimizer [Kingma and Ba, 2015].

Here, the following parametric form is proposed for q:

$$\mathrm{q}(\mathbf{Z},\mathbf{F},\mathbf{U},\mathbf{R}) = \mathrm{q}(\mathbf{Z})\mathrm{q}(\mathbf{F}|\mathbf{U},\boldsymbol{\Theta})\mathrm{q}(\mathbf{U})\mathrm{q}(\mathbf{R}), \text{ with} \tag{10}$$

$$\mathrm{q}(\mathbf{Z}) = \prod_{n=1}^{N} \mathrm{q}(\mathbf{z}_n) = \prod_{n=1}^{N} \mathbf{z}_n^\intercal \mathbf{q}_n, \tag{11}$$

$$\mathrm{q}(\mathbf{F}|\mathbf{U},\boldsymbol{\Theta}) = \mathrm{p}(\mathbf{F}|\mathbf{U},\boldsymbol{\Theta}), \tag{12}$$

$$\mathrm{q}(\mathbf{U}) = \prod_{k=1}^{K} \mathrm{q}(\mathbf{u}_k) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{u}_k|\mathbf{m}_k,\mathbf{S}_k), \tag{13}$$

$$\mathrm{q}(\mathbf{R}) = \prod_{a=1}^{A}\prod_{k=1}^{K} \mathrm{q}(\mathbf{r}_k^a) = \prod_{a=1}^{A}\prod_{j=1}^{K} \mathrm{Dir}(\mathbf{r}_j^a|\tilde{\alpha}_{1j}^a,\dots,\tilde{\alpha}_{Kj}^a). \tag{14}$$

The proposed posterior on $\mathbf{Z}$ factorizes across data points, and each $\mathbf{q}_n = (q_{n1},\dots,q_{nK}) \in [0,1]^K$ describes the probability that K is the real class for $\mathbf{x}_n$ (i.e., $\sum_k q_{nk} = 1$). The prior conditional $\mathbf{F}|\mathbf{U}$ does not introduce any new variational parameter. The posterior on $\mathbf{U}$ factorizes across dimensions, and each one is given by a Gaussian with mean $\mathbf{m}_k \in \mathbb{R}^M$ and (positive-definite) covariance matrix $\mathbf{S}_k \in \mathbb{R}^{M\times M}$. Finally, $\mathrm{q}(\mathbf{R})$ factorizes across annotators and dimensions, and they are assigned Dirichlet distributions with parameters $\tilde{\alpha}_{ij}^a > 0$. In the sequel, all these variational parameters $\{\mathbf{q}_n : n = 1,\dots,N\}$, $\{\mathbf{m}_k,\mathbf{S}_k : k = 1,\dots,K\}$, $\{\tilde{\alpha}_{ij}^a : i,j = 1,\dots,K;\ a = 1,\dots,A\}$ will be denoted by $\mathbf{V}$.

In the proposed form described by eqs. (10)–(14), the prior conditional $\mathrm{p}(\mathbf{F}|\mathbf{U},\boldsymbol{\Theta})$ arises in a natural way if the GP values are assumed conditionally independent on any other value given the inducing points $\mathbf{U}$. This is the original assumption of Titsias in [Titsias, 2009], and intuitively implies that all the information is condensed by and propagated through the inducing points $\mathbf{U}$. This form of $\mathbf{F}|\mathbf{U}$, plus that of $\mathrm{q}(\mathbf{U})$, are at the core of the sparse GP approximation that we are inspired by, SVGP [Hensman et al., 2015a]. The distributions $\mathrm{q}(\mathbf{Z})$ and $\mathrm{q}(\mathbf{R})$ are given the functional form that would arise if a mean-field approach was applied [Bishop, 2006, Eq. (10.9)]. For that, the conjugacy between the Dirichlet distribution in $\mathrm{p}(\mathbf{R})$ and the categorical in $\mathrm{p}(\mathbf{Y}|\mathbf{Z},\mathbf{R})$ is essential.

Table 1: Specifying the dependence of the ELBO on the variational parameters, the kernel hyperparameters, and the inducing locations through its five terms in eq. (15).

ELBO term	Parameters it depends on
$\sum \mathrm{q}_{nk}\mathbb{E}_{\mathrm{q}(\mathbf{r}_k^a)}\left[\log \mathrm{p}(\mathbf{y}\|\mathbf{e}_k,\mathbf{r}_k^a)\right]$	$q_{nk}, \tilde{\alpha}_{ij}^a$
$\sum \mathrm{q}_{nk}\mathbb{E}_{\mathrm{q}(\mathbf{f}_{n,:})}\left[\log \mathrm{p}(\mathbf{e}_k\|\mathbf{f}_{n,:})\right]$	$q_{nk}, \mathbf{m}_k, \mathbf{S}_k, \mathbf{\Theta}, \tilde{\mathbf{X}}$
$\sum q_{nk}\log q_{nk}$	q_{nk}
$\sum \mathrm{KL}(\mathrm{q}(\mathbf{u}_k)\|\|\mathrm{p}(\mathbf{u}_k))$	$\mathbf{m}_k, \mathbf{S}_k, \mathbf{\Theta}, \tilde{\mathbf{X}}$
$\sum \mathrm{KL}(\mathrm{q}(\mathbf{r}_k^a)\|\|\mathrm{p}(\mathbf{r}_k^a))$	$\tilde{\alpha}_{ij}^a$

Now, we can compute the explicit expression for the ELBO in our case, which must be maximized w.r.t. $\mathbf{V}$, $\mathbf{\Theta}$, and $\tilde{\mathbf{X}}$:

$$\begin{aligned}\mathrm{ELBO}(\mathbf{V},\mathbf{\Theta},\tilde{\mathbf{X}}) &= \mathbb{E}_{\mathrm{q}(\mathbf{Z})\mathrm{p}(\mathbf{F}|\mathbf{U})\mathrm{q}(\mathbf{U})\mathrm{q}(\mathbf{R})}\log \frac{\mathrm{p}(\mathbf{Y}|\mathbf{Z},\mathbf{R})\mathrm{p}(\mathbf{Z}|\mathbf{F})\cancel{\mathrm{p}(\mathbf{F}|\mathbf{U})}\mathrm{p}(\mathbf{U})\mathrm{p}(\mathbf{R})}{\mathrm{q}(\mathbf{Z})\cancel{\mathrm{p}(\mathbf{F}|\mathbf{U})}\mathrm{q}(\mathbf{U})\mathrm{q}(\mathbf{R})} \\ &= \sum_{n=1}^{N}\sum_{a\in A_n}\sum_{\mathbf{y}\in\mathbf{Y}_n^a}\sum_{k=1}^{K} q_{nk}\mathbb{E}_{\mathrm{q}(\mathbf{r}_k^a)}\left[\log \mathrm{p}(\mathbf{y}|\mathbf{e}_k,\mathbf{r}_k^a)\right] + \sum_{n=1}^{N}\sum_{k=1}^{K} q_{nk}\mathbb{E}_{\mathrm{q}(\mathbf{f}_{n,:})}\left[\log \mathrm{p}(\mathbf{e}_k|\mathbf{f}_{n,:})\right] \\ &- \sum_{n=1}^{N}\sum_{k=1}^{K} q_{nk}\log q_{nk} - \sum_{k=1}^{K}\mathrm{KL}(\mathrm{q}(\mathbf{u}_k)||\mathrm{p}(\mathbf{u}_k)) - \sum_{a=1}^{A}\sum_{k=1}^{K}\mathrm{KL}(\mathrm{q}(\mathbf{r}_k^a)||\mathrm{p}(\mathbf{r}_k^a)),\end{aligned} \tag{15}$$

A detailed derivation of this expression is provided in the supplemental material. Notice that the inclusion of the prior conditional $\mathrm{p}(\mathbf{F}|\mathbf{U})$ in the approximate posterior makes the highlighted cancellation possible, which is essential for the scalability of the method. All these five terms in eq. (15) but the second one can be expressed in closed-form as a function of $\mathbf{V}$, $\mathbf{\Theta}$, and $\tilde{\mathbf{X}}$. Similarly, the second one can be approximated explicitly through Gaussian-Hermite quadrature [Olver et al., 2010a], which is already implemented in GPflow for many different likelihoods (like the robust-max used here) [Matthews et al., 2017]. Further details and the specific expressions can be found in the supplemental material. As a summary, Table 1 shows which parameters each term in eq. (15) depends on.

Importantly, observe that the expression for the ELBO factorizes across data points, which allows for stochastic optimization through mini-batches [Hoffman et al., 2013]. To the best of our knowledge, this allows GP-based crowdsourcing methods to scale up to previously prohibitive data sets for the first time. More specifically, the computational complexity to evaluate the ELBO in eq. (15) in terms of the training set size is $\mathcal{O}(N_b(M^2 + A_bK))$, where N_b is the mini-batch size, M the number of inducing points, K the number of classes, and A_b the number of annotations per instance in the mini-batch. Although this is theoretically linear in N, the amount of inducing points M might grow with N. An interesting alternative, which is based on inference networks and addresses this issue, will be presented in Section 5.2. It is also interesting to compare eq. (15) with the expression for the ELBO in SVGP [Hensman et al., 2015a, Eq. (19)]. The second and fourth terms, which come from the prior and the classification likelihood, are analogous to the two terms in [Hensman et al., 2015a]. The other three terms arise naturally from the crowdsourcing modelling.

Once the ELBO is maximized w.r.t. $\mathbf{V}$, $\boldsymbol{\Theta}$ and $\tilde{\mathbf{X}}$, we can make predictions for previously unseen data points. Given a new $\mathbf{x}^*$, we have

$$\begin{aligned}\mathrm{p}(f_k^*|\mathbf{x}^*, \mathcal{D}) = \int \mathrm{p}(f_k^*|\mathbf{u}_k)\mathrm{p}(\mathbf{u}_k|\mathcal{D})\mathrm{d}\mathbf{u} &\approx \mathbb{E}_{\mathrm{q}(\mathbf{u}_k)}\mathrm{p}(f_k^*|\mathbf{u}_k) \\ &= \mathcal{N}\left(f_k^*|\mathbf{B}_{\mathbf{x}^*\tilde{\mathbf{X}}}\mathbf{m}_k, k_{\mathbf{x}^*\mathbf{x}^*} + \mathbf{B}_{\mathbf{x}^*\tilde{\mathbf{X}}}(\mathbf{S}_k - \mathbf{K}_{\tilde{\mathbf{X}}\tilde{\mathbf{X}}})\mathbf{B}_{\tilde{\mathbf{X}}\mathbf{x}^*}\right), \quad (16)\end{aligned}$$

where $\mathbf{B}_{\mathbf{x}^*\tilde{\mathbf{X}}}$ stands for $\mathbf{K}_{\mathbf{x}^*\tilde{\mathbf{X}}}\mathbf{K}_{\tilde{\mathbf{X}}\tilde{\mathbf{X}}}^{-1}$, and we are using the values of $\mathbf{m}_k$, $\mathbf{S}_k$, $\boldsymbol{\Theta}$, and $\tilde{\mathbf{X}}$ estimated after training. The predictive distribution on the real label $\mathbf{z}^*$ is obtained as $\mathrm{p}(\mathbf{z}^*) = \int \mathrm{p}(\mathbf{z}^*|\mathbf{f}^*)\mathrm{p}(\mathbf{f}^*)\mathrm{d}\mathbf{f}^*$. For classification likelihoods like ours, this is computed by GPflow through Gaussian-Hermite quadrature. Moreover, as we will illustrate in the experiments, the posterior distributions $\mathrm{q}(\mathbf{Z})$ and $\mathrm{q}(\mathbf{R})$ provide an estimation for the underlying real label of the training points and for the annotators degree of expertise, respectively. Finally, in order to exploit GPU acceleration through TensorFlow, the novel SVGPCR is implemented within the popular GP framework GPflow [Matthews et al., 2017]. The code will be made publicly available in GitHub upon acceptance of the paper, and will be listed in the "projects using GPflow" section of the GPflow site `https://github.com/GPflow/GPflow`.

3 LIGO data description

The Laser Interferometer Gravitational-Waves Observatory (LIGO) is a large-scale physics experiment and observatory to detect gravitational waves (GWs) [Abramovici et al., 1992]. These are ripples in the space-time produced by non-symmetric movements of masses, being their energy much higher for events such as binary black holes or neutron stars mergers. Their existence is a direct consequence of the General Relativity theory postulated in 1916. However, Albert Einstein himself believed they would be extremely difficult to detect by any technology foreseen at that time [Kennefick, 2016].

The first direct observation of GWs was made one hundred years later by LIGO, on September 14th, 2015. The discovery had a tremendous impact in the scientific community. Not only as an empirical validation of one of the most recognized Physics theories, but also as a whole new way to explore the universe. So far, astrophysicists could perceive the outer space only through one "sense" (electromagnetic radiation), but were "deaf" to GWs. This detection has inaugurated a new era of the so-called GWs astronomy, and has been awarded the 2017 Physics Nobel Prize [B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration), 2016].

To identify GWs, LIGO is able to detect changes of the length of a 4 kilometers arm by a thousandth of the width of a proton [Abramovici et al., 1992]. This is proportionally equivalent to changing the distance to the nearest star outside the Solar System by one hair's width. Such precision requires cutting-edge technology that is also extremely sensitive to different instrumental and environmental sources of noise. In the spectrograms that astrophysicists analyze to search for GWs, this contamination manifests itself in the form of glitches, which are noisy patterns that adopt many different morphologies [Zevin et al., 2017]. The presence of these glitches hinders the detection of true GWs. Figure 3 shows the 15 types of glitches considered in this work, which will be later described.

The goal of the GravitySpy project is to develop a system to accurately classify the different types of glitches [Zevin et al., 2017]. This would help astrophysicists to gain insights on their taxonomy and potential causes, enhancing detection of true GWs. Since LIGO produces a constant stream of data, GravitySpy leverages crowdsourcing techniques through the Zooniverse platform in order to label a training set `https://www.zooniverse.org/projects/zooniverse/gravity-spy`. Then, machine learning crowdsourcing algorithms that can learn from this multiple-annotated data must be applied (like the SVGPCR presented here).

Our training set contains 173565 instances (glitches) and 1828981 annotations (i.e., a mean value of more than 10 labels per instance), which have been provided by 3443 collaborators through the Zooniverse platform. These instances are time-frequency plots (spectrograms) like those in figures 1 and 3, taken with four time windows. For each one, we will use 256 relevant features extracted in [Bahaadini et al., 2018]. These glitches have been classified into 15 different classes proposed by astrophysicists (recall figure 3). Next, we provide a brief description of them (see [Bahaadini et al., 2018] for a more detailed explanation).

1080 Line: It appears as a string of short yellow dots, always around 1080Hz. It was reduced after an update on 2017, although it is still present.

1400 Ripple: Glitches of 0.05s or longer around 1400Hz. So far, their origin is unknown. They are commonly confused with 1080Line and Violin Mode Harmonic.

Blip: Short glitches with a symmetric “teardrop” shape in time-frequency. Blips are extremely important since they hamper the detection of binary black hole mergers [Abbott et al., 2016].

Extremely Loud: These are caused by major disturbances, such as an actuator reaching the end of its range and “railing”, or a photodiode “saturating”. They look very bright, due to their very high energy.

Koifish: Similar to Blips, but resemble a fish with the head at the low frequency end, pectoral fins around 30 Hz, and a thin tail around 500Hz. LIGO scientists do not understand the physical origin of this glitch.

Low Frequency Burst: Resembles a hump with a nearly triangular shape growing from low frequency to a peak, and then dying back down in one or two seconds. It is caused by scattered light driven by motion of the output mirrors.

Low Frequency Lines: These appear as horizontal lines at low frequencies. Can be confused with Scattered Light (the latter shows some curvature) and Low Frequency Bursts (the former continues to look like a line in the 4s window).

No Glitch: No glitch refers to images that do not have any glitch visible at all. The spectrograms would appear dark blue with only small fluctuations.

Other: This category is a catch-all for glitches that do not fit into the other categories. Therefore, it presents a great variability in its morphology.

Power-line 60Hz: In US, the mains power is alternating current at 60Hz. When equipment running on this power switches on or off, glitches can occur at 60Hz or harmonics (120, 180...). These glitches usually look narrow in frequency, centered around 60Hz or harmonics.

Repeating blips: Analogous to blips, but repeat at regular intervals, usually every 0.125, 0.25 or 0.5 seconds.

Scattered Light: After hitting optical components, some light from LIGO beam is scattered. It may then reflect off of other objects and re-enter the beam with a different phase. It usually looks like upward humps, with frequency below 30 Hz. It hinders searches of binary neutron stars, neutron star black hole binaries, and binary black holes.

Scratchy: Wide band of mid-frequency signals that looks like a ripply path through time-frequency space. This glitch hampers searches for binary black hole mergers.

Violin Mode Harmonic: Test masses in LIGO are suspended from fibers with resonances. These are called violin modes, as they resemble violin strings resonances. Thermal excitations of the fibers produce movements at the violin mode frequencies, centered around 500Hz. Thus, these glitches are short and located around 500 Hz and harmonics.

Whistle: Usually appear with a characteristic W or V shape. Caused by radio frequency signals beating with the LIGO Voltage Controlled Oscillators. Whistles mainly contaminate searches for binary black hole mergers [Nuttall et al., 2015].

For testing purposes, the astrophysicists at GravitySpy have labelled a set of 9997 instances, including glitches from all the 15 types explained above.

4 Experimental results

In this section, the proposed SVGPCR is empirically validated and compared against current crowdsourcing methods, with a special focus on the LIGO data introduced in the previous section. Three blocks of experiments are presented in sections 4.1, 4.2 and 4.3. Firstly, the behavior of SVGPCR is thoroughly analyzed in a controlled crowdsourcing experiment based on the popular MNIST set. Secondly, SVGPCR is compared with previous probabilistic (mainly GP-based) approaches on the LIGO data. Since most of these methods were proposed for binary problems, we consider a binary task relevant to the GravitySpy project. Thirdly, SVGPCR is compared against state-of-the-art DL-based crowdsourcing methods in the full LIGO data set.

4.1 Understanding the proposed method

Before comparing against other crowdsourcing methodologies, let us analyze the behavior and main properties of SVGPCR. To do so, we simulate five different crowdsourcing annotators for the well-known MNIST data set. The availability of simulated annotators and real training labels on this graphic data set constitutes a controlled setting that allows for a comprehensive analysis.

We use the standard train/test split of MNIST with 60K/10K hand-written digits from 0 to 9 (multi-class problem with 10 classes) [LeCun et al., 1998]. Notice that 60K training instances is already prohibitive for standard GPs. Five decreasingly reliable annotators are considered. The first one has a 95% accuracy for each class, that is, $r^1_{jj} = 0.95$ for $j = 0, \dots, 9$ (the rest of values, r^1_{ij}, $i \neq j$, are randomly assigned to add the remaining 0.05 probability by columns). The second and third ones are defined analogously, but with 90% and 80% accuracy, respectively. The fourth one is a *spammer* annotator, that is, $r^4_{ij} = 0.1$ for all $i, j = 0, \dots, 9$. This implies that, regardless of the real class, this annotator assigns a random label. The fifth one is an *adversarial* annotator. Specifically,

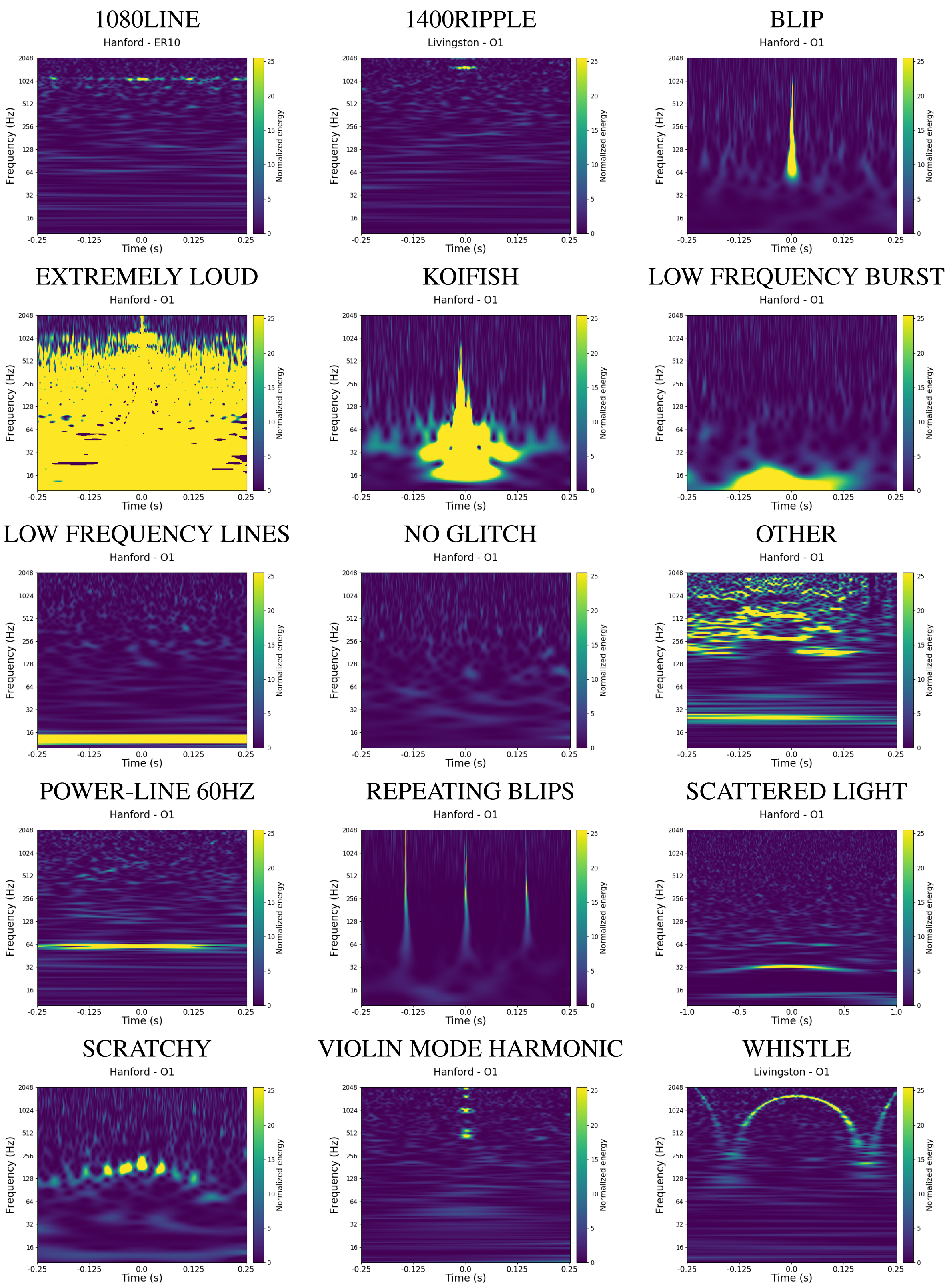


Figure 3: Representative spectrograms for the 15 different types of glitches considered in this work. Hanford and Livinsgton refer to the two observatories that LIGO comprises, and ER10/O1 to two different observation runs. A brief description of each glitch is provided in the text. The goal of the GravitySpy project is to learn a machine learning system to automatically classify these glitches. The labels for the training set are obtained through crowdsourcing.

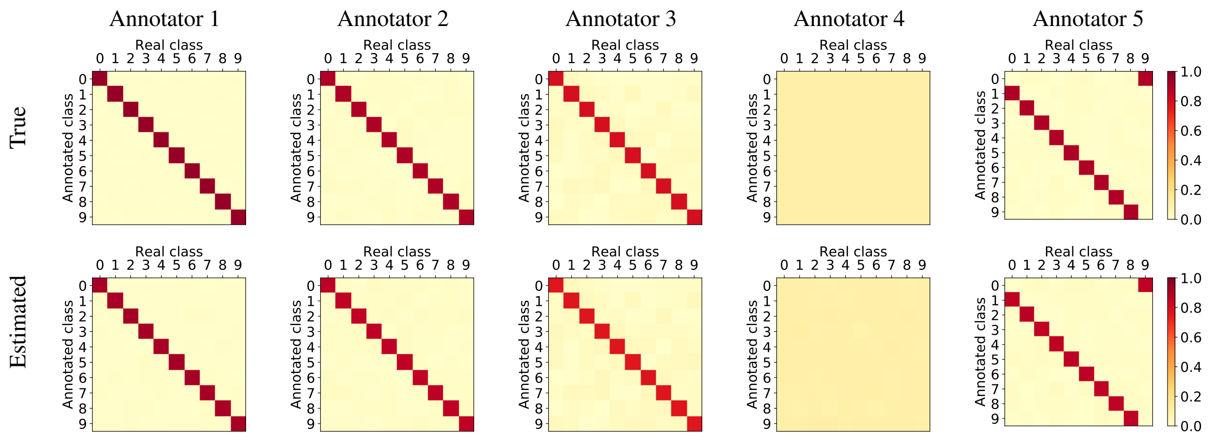


Figure 4: Estimation of the expertise degree of the simulated annotators in the MNIST problem by SVGPCR. Upper row: true confusion matrices. Lower row: mean of the estimated distribution for the confusion matrices. Notice that the proposed method perfectly identifies adversarial (fifth) and spammer (fourth) annotators. Moreover, not only the structure of the matrices is well identified, but also the actual values (the intensity of color is very similar).

Table 2: Per-class and global test performance of SVGPCR and SVGP-gold in the MNIST problem. In spite of the corrupted labels, the proposed method almost recovers the golden results.

	Test accuracy		Test likelihood	
	SVGP-gold	SVGPCR	SVGP-gold	SVGPCR
0	0.9867	**0.9898**	0.9777	**0.9781**
1	**0.9885**	0.9877	**0.9853**	0.9845
2	0.9525	**0.9535**	**0.9368**	0.9345
3	0.9703	**0.9733**	0.9475	**0.9555**
4	0.9715	**0.9735**	0.9548	**0.9570**
5	**0.9630**	0.9540	**0.9397**	0.9352
6	**0.9749**	**0.9749**	**0.9604**	0.9585
7	**0.9591**	0.9543	0.9418	**0.9432**
8	**0.9620**	**0.9620**	**0.9458**	0.9445
9	**0.9346**	0.9316	**0.9217**	0.9186
Global	**0.9665**	0.9657	**0.9515**	0.9514

in this case, with a 90% of probability, the annotator labels as $(i+1)$-th class an instance whose real class is the ith (samples in class 9 are assigned to class 0). The confusion matrices for these annotators are depicted in the first row of figure 4. The five annotators label all the instances, which yields 300K annotations that are used to train SVGPCR.

Since we have available the true labels for the training instances, let us start by comparing SVGPCR with its theoretical upper bound, namely SVGP trained with the true labels, which we refer to as SVGP-gold. Table 2 shows the global and per-class test accuracy and test likelihood for both approaches. Importantly, notice that the results are very similar for all classes and both metrics, and SVGPCR almost reaches the same global performance as SVGP-gold (in spite of the corrupted labels provided by annotators).

This excellent performance of SVGPCR can be explained by its accurate prediction of the annotators behavior, which in turn allows SVGPCR to properly reconstruct the underlying true labels

Table 3: Per-class and global performance of SVGPCR to reconstruct the underlying true label for training instances in the MNIST problem. An excellent result is obtained across all the classes, with only 20 (out of the 60000 training examples) not correctly predicted.

	0	1	2	3	4	5	6	7	8	9	Global
Accuracy	0.9998	0.9997	1.0000	0.9995	0.9995	0.9993	1.0000	0.9997	0.9997	0.9995	0.9997
Likelihood	0.9997	0.9996	0.9997	0.9992	0.9994	0.9993	0.9999	0.9995	0.9996	0.9991	0.9995

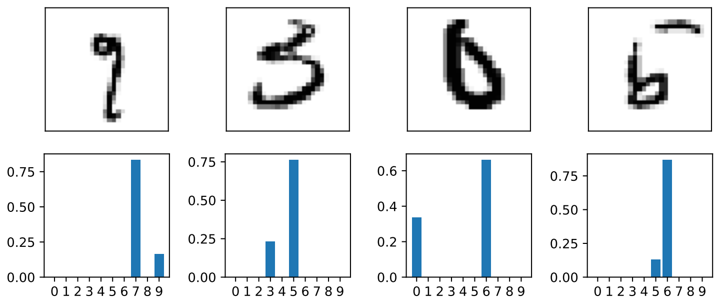


Figure 5: Upper row: four (out of the 20) examples for which SVGPCR is not able to reconstruct the real underlying class in the MNIST problem. Lower row: the corresponding probabilities assigned by SVGPCR. In all cases, the proposed method assigns the second highest probability to the real class. Notice that some of these examples are not easy, and have some features which might lead to confusion.

from the noisy annotations. Indeed, firstly, figure 4 shows the exceptional estimations obtained by SVGPCR for the annotators confusion matrices. Recall from eq. (14) that the expertise degree of annotators is estimated through posterior Dirichlet distributions. The bottom row of figure 4 shows the mean of those distributions. Interestingly, the maximum variance was 0.0016, which implies a high degree of certainty about the predictions in figure 4. Secondly, as previously mentioned, this allows SVGPCR to correctly puzzle out the underlying true labels from the noisy annotations. In fact, table 3 shows the excellent per-class and global performance of SVGPCR in this sense (recall that SVGPCR estimates the underlying true labels through the approximate posterior $\mathrm{q}(\mathbf{z})$ in eq. (11)).

More in depth, we have analyzed the 20 examples where SVGPCR fails to reconstruct the true label, and some of them can be certainly considered as not-easy ones. Figure 5 shows four of them, along with the probabilities assigned by SVGPCR for each one. In all cases, the true label is assigned the second highest probability by SVGPCR, and the digit presents some feature which certainly leads to confusion with the class that SVGPCR assigns more probability to.

Another key aspect of SVGPCR is the role of the inducing points. In this example we are using $M = 100$, and the next experiment will be devoted to analyze the influence of M in the performance of SVGPCR. But before, figure 6 shows the locations to which 30 out of the 100 inducing points have converged after training (recall that the ELBO in eq. (15) is also maximized w.r.t. the inducing locations $\tilde{\mathbf{X}}$). For instance, the first column shows the locations of three inducing points which are classified as 0 by SVGPCR (according to the estimated $\mathbf{m}_k$, recall eq. (13)), and analogously for the rest of the columns. It is very interesting to notice that the inducing point locations

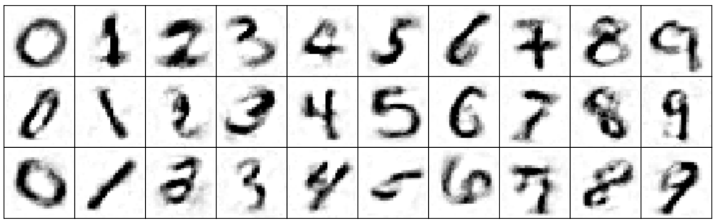


Figure 6: Some of the inducing point locations learned by SVGPCR in the MNIST problem. They have been arranged by columns based on their classification. Notice that, for each digit (column), different representative patterns are learned in terms of shape, orientation and thickness.

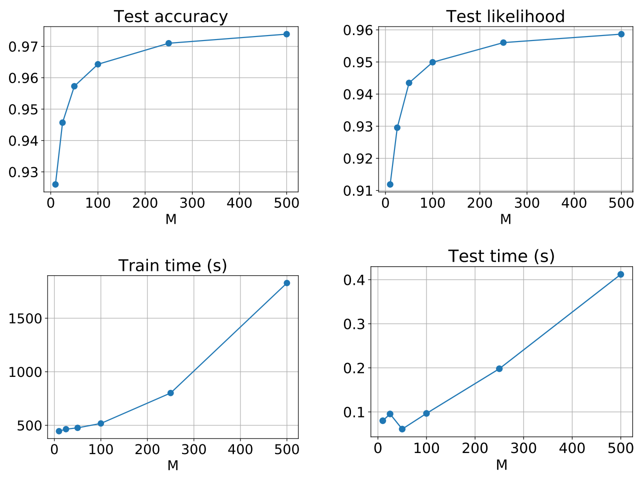


Figure 7: Influence of M (number of inducing points) on the test performance (accuracy and likelihood) and computational cost (elapsed time at training and testing) for SVGPCR in the MNIST problem. As theoretically expected, more inducing points lead to better test performance at the expense of a higher computational cost.

comprise different calligraphic representations (in terms of shape, orientation and thickness) of the same digit. This is related to their intuitive role of entities that summarize the training data.

Next, let us study the influence of M (the number of inducing points) on the behavior of SVGPCR. Figure 7 shows the dependence on M of four different metrics: two measures of the test performance (accuracy and mean likelihood), and two related to the computational cost (at training and test steps). As expected from the theoretical formulation in section 2, a greater number of inducing points implies a higher performance at test (in both accuracy and mean likelihood), since the expressiveness of the model is higher. However, this also leads to heavier train and test costs, since there are more parameters to be estimated (inducing locations $\tilde{\mathbf{X}}$, $\mathbf{m}_k$, and $\mathbf{S}_k$), and the size of several matrices increase.

Moreover, for a given M, the model is expected to obtain better test performance as the training time evolves (i.e., when more epochs are run). In order to further investigate this, figure 8 shows the test accuracy of SVGPCR as the training time evolves, for different values of M. It is interesting to observe that, the more inducing points, the higher values of test accuracy can be potentially reached, but also a greater amount of training time is needed to reach that precision (notice that the

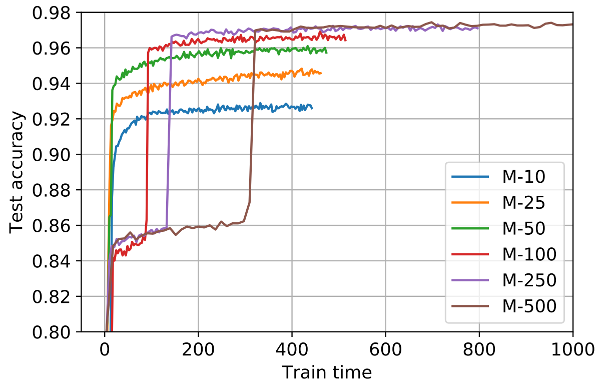


Figure 8: Evolution of test accuracy as a function of the training time for different values of M in the MNIST problem. If there is no limit on the available training time, then high values of M must be selected (as long as it allows for the kernel matrix inversion). However, lower values would be more appropriate for a fast training, since the amount of parameters to be trained significantly reduces. Moreover, when a certain (problem-dependent) M has been reached, there is no a significant benefit by increasing it (observe the difference from $M = 250$ to $M = 500$).

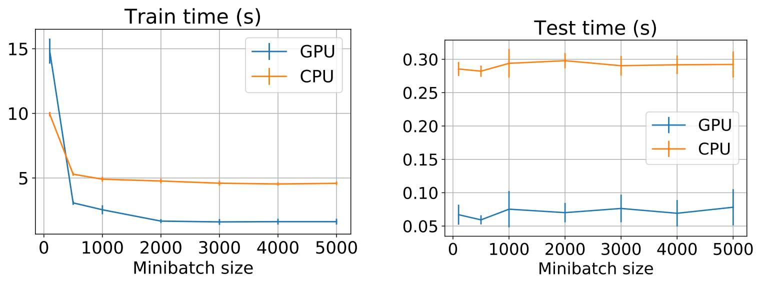


Figure 9: Comparison between CPU and GPU implementations of SVGPCR in terms of computational cost for the MNIST problem. At training, the time depends on the minibatch size, since a greater minibatch implies computations with larger matrices and less memory copies, which benefits the GPU. With a minibatch of size 500 (the default used in this section), the GPU is around two times faster. These values are per epoch. At testing (production time), the GPU is over three times faster, and logically does not depend on the minibatch size. In this case, the shown values are for the whole test set.

steps which take $M = 100, 250, 500$ to the level of their final precision happen increasingly later). The conclusion is that, for a given computational budget, the M to be selected is the highest one that can reach convergence in that time (logically, assuming that it allows for the inversion of the associated kernel matrix, i.e., usually $M < 10^4$).

Finally, since the associated code can leverage GPU acceleration through GPflow [Matthews et al., 2017], let us compare CPU and GPU implementations. Figure 9 shows that, for training, the GPU is usually the preferred choice, unless the minibatch size is very small, in which case the amount of memory copies from CPU to GPU does not compensate the advantage provided by the latter. In test, the GPU is always faster, since it involves much less data transfers from CPU to GPU.

4.2 Comparison to classical probabilistic approaches

As explained in section 1, the most popular approaches to crowdsourcing jointly model a classifier for the underlying true labels along with the annotators' behavior. The first works used basic logistic regression as the classifier, e.g. *Raykar* [Raykar et al., 2010] and *Yan* [Yan et al., 2014] (the difference between them is the noise model considered for the annotators). However, they struggled when dealing with complex non-linear data sets. Then, Gaussian Processes became the preferred choice, since their non-parametric form and accurate uncertainty quantification yielded much better results, e.g. *Rodr14* [Rodrigues et al., 2014] and VGPCR [Ruiz et al., 2019] (they differ in the inference procedure used, Expectation Propagation [Minka, 2001] and Variational Inference [Blei et al., 2017], respectively). However, the poor scalability of GPs hampered the wide adoption of these approaches in practice. This motivated the development of the so-called RFF and VFF algorithms, which leverage Random Fourier features approximations to GPs to propose two more scalable GP-based crowdsourcing methods [Morales-Álvarez et al., 2019]. These approaches significantly improve the scalability, reducing it from cubic $\mathcal{O}(N^3)$ to linear $\mathcal{O}(ND^2)$ (with D the number of Fourier frequencies used, $D \ll N$, see [Morales-Álvarez et al., 2019]). In practice, this implies moving from manageable data sets of $N \approx 10^4$ up to $N \approx 10^5/10^6$. However, RFF and VFF do not factorize in mini-batches, which prevents them from reaching data sets of virtually any size.

In the last few years, these classical (mainly GP-based) approaches have been replaced by crowdsourcing methods based on Deep Learning (DL) [Albarqouni et al., 2016, Rodrigues and Pereira, 2018]. These achieve excellent scalability through mini-batches, and can handle data sets of almost any size. Because of this, they have become the state of the art approach for real-world crowdsourcing problems. In the next section 4.3, we will bring GP-based methods back to a state of the art level. We will show that SVGPCR is competitive with DL-based methods, and additionally provides a very accurate control of uncertainty. But before this, it is worth to analyze here the advances that SVGPCR introduces over its predecessors classical (mainly GP-based) crowdsourcing approaches.

More specifically, let us compare SVGPCR with the aforementioned *Raykar*, *Yan* (based on logistic-regression), *Rodr14*, VGPCR (based on GPs), and RFF, VFF (based on scalable approximations to GP). Since most of them were formulated for binary problems, we consider a binary task relevant to astrophysicists in GravitySpy. Using the data set presented in section 3, the goal is to distinguish between the glitch called "Other" and the rest of types. This is important in order to identify potential overlaps between that catch-all class and the rest of glitches. Moreover, it introduces an imbalanced scenario, since "Other" represents only a 10.12% of the total amount of annotations. We will use the area under the ROC curve (AUC) as test performance metric.

Figure 10 compares the scalability of the compared methods as the training set grows. SVGPCR clearly stands out as the most scalable approach. This can be attributed to its training scheme through mini-batches, which considerably alleviates the dependence on the training set size. The rest of methods explode at different moments: the heavy EP inference of *Rodr14* only allows for training with up to $N = 2500$, the GP-based formulation of VGPCR and the complex annotators noise model of *Yan* make them reach $N = 25000$ with difficulties. In spite of the GP approximation, VFF does not go beyond $N = 10^5$ in this problem, because of the expensive optimization of Fourier features. Finally, *Raykar* (which is based on cheap logistic regression) and RFF (which

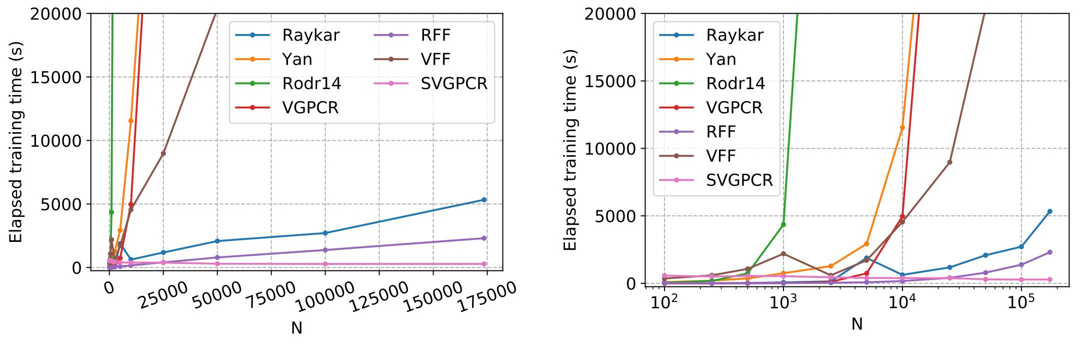


Figure 10: Elapsed training time as a function of the training set size N in the binary LIGO data set. The mean over five independent runs is shown. On the left, a standard linear scale is used for the x-axis. Notice that SVGPCR exhibits a significantly better scalability than classical probabilistic methods, which is due to its factorization in mini-batches. Moreover, among previous approaches, we can distinguish between those that have already exploded for $N = 25000$ (*Yan*, *Rodr14*, VGPCR, VFF), and those which have not yet for the full set size (*Raykar*, RFF). In order to better appreciate the differences, a logarithmic scale is used for the x-axis on the figure at the right. This further shows that *Rodr14* shoots up as early as $N = 1000$, *Yan*, VGPCR and VFF do it around $N = 10000$, and *Raykar* and RFF are starting beyond $N = 10^5$.

does not optimize over the Fourier features) can cope with the full data set, although they are significantly slower than SVGPCR.

Moreover, figure 11 shows that their test performance is pretty far from that of SVGPCR. Indeed, the logistic regression model underlying *Raykar* is not sufficient for the nonlinear problem at hand, and the GP approximation provided by RFF is known to be poor when the dimensionality of the problem is high [Morales-Álvarez et al., 2019] (like here, where we are working with 256 features, recall section 3). The rest of methods are also clearly outperformed, since their limited scalability prevents them from processing the full data set. Interestingly, figure 11 shows an intuitive and logical structure: the more simple logistic-regression based methods are located on the left (less test AUC), the classical GP-based ones in the central part, and SVGPCR on the right.

4.3 Comparison with state of the art DL-based methods

In the last years, Deep Learning has emerged as a scalable alternative to model crowdsourcing problems. Two of the most popular approaches are AggNet [Albarqouni et al., 2016] and the various crowd layers proposed in [Rodrigues and Pereira, 2018]. The former considers a deep neural network (DNN) as underlying classifier, and a probabilistic noise model for annotators based on per-user confusion matrices. Then, the training step follows an iterative expectation-maximization (EM) scheme between both parts of the model [Bishop, 2006, Section 9.4]. Alternatively, the crowd layers in [Rodrigues and Pereira, 2018] allow for end-to-end training of the DNN, without the need for the EM scheme. This is significantly cheaper in terms of computational cost, although the probabilistic formulation of AggNet allows for a better uncertainty quantification. The three crowd layers studied in [Rodrigues and Pereira, 2018] will be considered here: CL-VW, CL-VWB and CL-MW. They differ in the parametric form of the annotator noise model, which is increas-

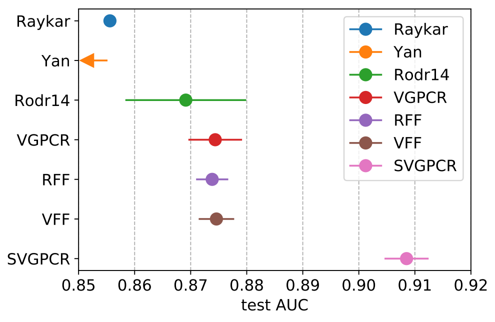


Figure 11: Test AUC achieved in the binary LIGO data set by the compared models (in each case, the largest one that could be trained). The mean and one standard deviations are shown. SVGPCR clearly outperforms methods which cannot cope with the full data set (*Yan*, *Rodr14*, VGPCR, VFF). Moreover, RFF and *Raykar* are also beaten because of their more limited formulation (see the main text).

ingly complex: a vector of per-class weights for CL-VW, an additional bias for CL-VWB, and a whole confusion matrix for CL-MW.

These four DL-based methods (AggNet, CL-VW, CL-VWB, CL-MW) are compared against three increasingly complex SVGPCR models: SVGPCR-10, SVGPCR-50, SVGPCR-100, where each number represents the amount of inducing points used. As all these approaches are defined for multi-class tasks, the full LIGO problem in section 3 can be addressed now.

Tables 4 and 5 show the global and per-class test performance of the compared methods. Table 4 is devoted to the test accuracy, which relies only on the mode of the predictive distribution and is less influenced by the uncertainty quantification of the model. Table 5 shows the test likelihood, which additionally depends on the uncertainty of the predictive distribution, and therefore depends more heavily on its accurate control within the model.

In both tables, SVGPCR stands out as the best-performing method globally. The difference is greater in the case of the test likelihood, which is logically explained by the better uncertainty quantification of GPs. Indeed, the better control of uncertainty also justifies that AggNet outperforms CL-based methods in test likelihood (whereas they are very similar in accuracy). Moreover, observe that the global superiority of SVGPCR is not due to a great result in only one or two very populated classes. SVGPCR performs consistently well across the 15 glitch types in both tables, winning in few of them (a bit more in test likelihood, as logically expected), and avoiding dramatic failures on difficult classes. This will be also observed for the alternative GP-based methods introduced in Section 5. According to astrophysicists at GravitySpy, this regularity across classes is a desirable property for a reliable glitch detection system.

It is also worth to notice that $M = 50$ inducing points seem enough for the problem at hand. In both tables 4 and 5, a significant improvement is observed from $M = 10$ to $M = 50$, but $M = 100$ produces very similar results. This small value of $M = 50$ hints at a not very complex internal structure of the data. It is also interesting to observe that, in general, the most difficult classes are "Repeating Blips" and "Other" (recall the 15 types in figure 3). This discovery is not surprising for astrophysicists in GravitySpy, since the former is usually confused with "Blips", and the latter is a

Table 4: Per-class and global test accuracy for the compared methods in the LIGO experiment. Mean and standard deviation over ten runs are shown. Globally, SVGPCR with enough inducing points outperforms DL-based methods by 2%. In per-class results, notice the regularity of SVGPCR, which performs well across all classes without standing out in many of them.

	AggNet	CL-VW	CL-VWB	CL-MW	SVGPCR (M=10)	SVGPCR (M=50)	SVGPCR (M=100)
1080LINE	**.9791(.0045)**	.9676(.0063)	.9732(.0059)	.9746(.0063)	.9781(.0075)	.9720(.0076)	.9720(.0069)
1400RIPPLE	**.9447(.0237)**	.1666(.3335)	.3243(.4001)	.0000(.0000)	.0000(.0000)	.8528(.0203)	.8577(.0171)
BLIP	.9737(.0078)	.9754(.0079)	.9741(.0135)	**.9826(.0062)**	.8712(.2904)	.9643(.0044)	.9622(.0052)
EXTR.LOUD	**.8704(.0566)**	.7193(.0652)	.8034(.0660)	.5272(.3502)	.3681(.3705)	.7261(.0398)	.7295(.0427)
KOIFISH	.8760(.0634)	.8749(.0151)	**.9043(.0296)**	.9023(.0203)	.8007(.2673)	.8844(.0129)	.8828(.0115)
L.F.BURST	.7649(.0802)	.8639(.0299)	.9023(.0211)	**.9146(.0174)**	.7912(.2638)	.8864(.0087)	.8861(.0105)
L.F.LINE	.8603(.0742)	.8603(.0274)	.8514(.0293)	.8566(.0279)	.8354(.2785)	.9150(.0105)	**.9156(.0111)**
NOGLITCH	**.9174(.0354)**	.8835(.0201)	.8887(.0241)	.9141(.0274)	.7048(.2366)	.7941(.0223)	.7951(.0162)
OTHER	.2786(.0294)	**.4167(.0348)**	.3660(.0376)	.3503(.0297)	.3622(.1213)	.3977(.0083)	.4011(.0091)
P.L.60HZ	.8885(.0676)	.8591(.0290)	**.9468(.0104)**	.9365(.0179)	.7417(.2476)	.8438(.0102)	.8425(.0127)
REP.BLIPS	.5487(.0730)	.0470(.1410)	.0572(.1717)	.0000(.0000)	.0000(.0000)	.6649(.0215)	**.6700(.0210)**
SCATT.LIGHT	.9088(.0492)	.9497(.0126)	.9601(.0109)	**.9645(.0053)**	.8657(.2886)	.9600(.0050)	.9562(.0056)
SCRATCHY	.8980(.0400)	.4433(.4437)	.4426(.4452)	.4440(.4444)	.8093(.2702)	.8953(.0206)	**.9000(.0165)**
VIOLIN	.9755(.0057)	**.9932(.0032)**	.9921(.0033)	.9930(.0018)	.8915(.2971)	.9899(.0027)	.9914(.0017)
WHISTLE	.9175(.0236)	.9359(.0136)	**.9438(.0137)**	.9377(.0230)	.8122(.2712)	.9166(.0070)	.9201(.0047)
GLOBAL	.8957(.0227)	.8886(.0126)	.8985(.0105)	.8956(.0104)	.8355(.1919)	**.9184(.0031)**	.9183(.0027)

Table 5: Per-class and global test likelihood for the compared methods in the LIGO experiment. Mean and standard deviation over ten independent runs are shown. Globally, SVGPCR with enough inducing points outperforms DL-based methods by almost 3%. It also exhibits a desirable regularity across the different classes. Moreover, notice that, compared to the accuracy in Table 4, there exists here a greater advantage against methods that do not quantify uncertainty (i.e. CL-based ones).

	AggNet	CL-VW	CL-WVB	CL-MW	SVGPCR (M=10)	SVGPCR (M=50)	SVGPCR (M=100)
1080LINE	**.9781(.0048)**	.9515(.0091)	.9597(.0076)	.9649(.0074)	.8811(.2715)	.9689(.0082)	.9688(.0075)
1400RIPPLE	**.9416(.0242)**	.1644(.3290)	.3196(.3937)	.0118(.4874)	.0067(.0199)	.8475(.0182)	.8509(.0156)
BLIP	.9709(.0077)	.9753(.0079)	.9735(.0123)	**.9777(.0043)**	.8746(.2694)	.9606(.0044)	.9587(.0055)
EXTR.LOUD	**.8626(.0574)**	.7214(.0645)	.8024(.0684)	.5276(.3503)	.3651(.3541)	.7266(.0435)	.7242(.0408)
KOIFISH	.8688(.0679)	.8752(.0161)	**.9013(.0300)**	.8959(.0190)	.8022(.2457)	.8788(.0109)	.8784(.0117)
L.F.BURST	.7575(.0799)	.7249(.0139)	.8035(.0177)	.8419(.0207)	.7966(.2434)	**.8851(.0083)**	.8838(.0098)
L.F.LINE	.8502(.0786)	.5808(.0192)	.6200(.0178)	.6917(.0242)	.8378(.2571)	**.9125(.0094)**	.9118(.0103)
NOGLITCH	**.9091(.0402)**	.8109(.0247)	.8196(.0342)	.8196(.0244)	.7062(.2146)	.7919(.0238)	.7932(.0146)
OTHER	.2692(.0285)	**.4123(.0322)**	.3439(.0362)	.3268(.0275)	.3686(.1012)	.3959(.0095)	.3999(.0091)
P.L.60HZ	.8700(.0665)	.8565(.0291)	**.8993(.0108)**	.8929(.0117)	.7435(.2260)	.8384(.0107)	.8380(.0107)
REP.BLIPS	.5323(.0664)	.0466(.1400)	.0584(.1752)	.4878(.8447)	.0067(.0199)	.6581(.0191)	**.6651(.0198)**
SCATT.LIGHT	.8716(.0480)	.8782(.0165)	.8947(.0196)	.9416(.0067)	.8683(.2673)	**.9558(.0047)**	.9520(.0057)
SCRATCHY	.8823(.0399)	.4425(.4429)	.4411(.4440)	.4414(.4419)	.7971(.2441)	.8911(.0215)	**.8953(.0176)**
VIOLIN	.9738(.0058)	.9823(.0034)	.9815(.0024)	.9863(.0022)	.8960(.2765)	.9875(.0020)	**.9886(.0014)**
WHISTLE	.9167(.0215)	.9341(.0149)	**.9427(.0141)**	.9388(.0213)	.8124(.2490)	.9155(.0064)	.9179(.0046)
GLOBAL	.8871(.0240)	.8175(.0106)	.8387(.0113)	.8528(.0101)	.8126(.2487)	**.9154(.0033)**	.9149(.0027)

catch-all class to which some conservative annotators resort too often. The case of "Other" is also related to the interest of astrophysicists to study it separately in the experiment of previous section 4.2.

It is also important to highlight that all these methods are scalable enough so as to cope with the full LIGO data set. More specifically, figure 12 shows the elapsed time at training and testing for the compared methods. In general, the proposed SVGPCR is competitive with DL-based methods in these aspects. At training, SVGPCR is significantly faster than AggNet due to the heavy iterative EM scheme of the latter, and is slower than CL-MW[6]. Nonetheless, less than one hour of training is a competitive result for a data set with 173565 instances (recall section 3). At testing, SVGPCR is the fastest approach, which is convenient for real-time applications the system might be used for.

[6]Results of CL-VW/CL-VWB being worse than CL-MW in figure 12 might be attributed to implementation inefficiency, since the former include for loops whereas matrix multiplication is used in the latter.

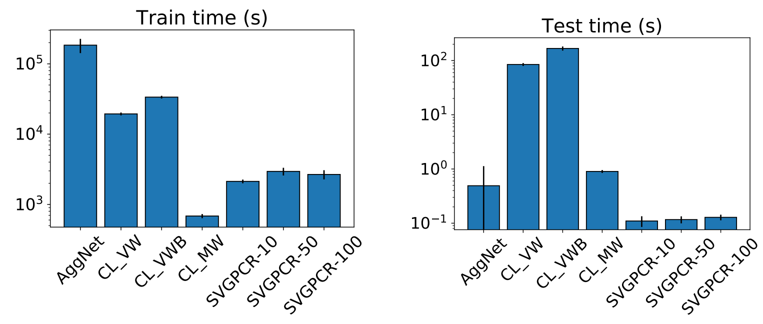


Figure 12: Train and test times (mean and standard deviation over ten independent runs) for the compared methods in the LIGO experiment. Notice the logarithmic scale in y-axis. The proposed SVGPCR is competitive with all the DL-based methods in terms of computational cost at both training and testing.

As already pointed out in Table 5, the underlying GP model of SVGPCR implies an advantage over DL-based methods in terms of uncertainty quantification. The test likelihood metric is a global measure of the quality of the predictive distribution obtained for each individual test instance. To clearly understand the benefits of the GP modelling, figure 13 shows the predictive distributions for some test instances which are behind the better global performance of SVGPCR. Only the best method (in terms of test likelihood) of each type (i.e. CL-based ones, SVGPCR ones, and AggNet) is considered, which yields the three columns in figure 13. Each row represents a different test instance.

Interestingly, we observe that the three approaches correctly classify the four instances, that is, they assign the highest probability to the correct class (which is highlighted in red). In particular, this means that these four instances contribute equally to the test accuracy of the three methods. However, notice that the quality of the predictive distribution worsens from left to right (i.e., from better to worse uncertainty quantification theoretical properties), since the methods become less certain about the correct answer and assign more probability to wrong ones. This is precisely what is accounted for in the test likelihood metric. From a practical perspective, this better quality of the predictive distributions has been particularly appreciated by astrophysicists at GravitySpy, in addition to the improvement in test accuracy (recall table 4). We stress that test instances in figure 13 are among those that most contribute to the outperformance in test likelihood. In general, when it is not certain about the outcome, SVGPCR provides uncertainty for its estimations. This is illustrated in the third section of the supplementary material. The figure there is analogous to this one, but using randomly selected test points.

Finally, a key aspect of crowdsourcing methods is the identification of the different annotators behavior. Unlike in section 4.1, where we had simulated annotators to check the good estimations of SVGPCR, in this real experiment we do not have available a ground-truth. Nonetheless, let us compare the predictions obtained by the different methods. We will see that they capture similar patterns, some of which can be explained from the experience of astrophysicists. Figure 14 shows the confusion matrices predicted by the compared methods for five different annotators. In the CL-based family we only consider CL-MW, as it is the best in test likelihood and the only one which provides a confusion matrix.

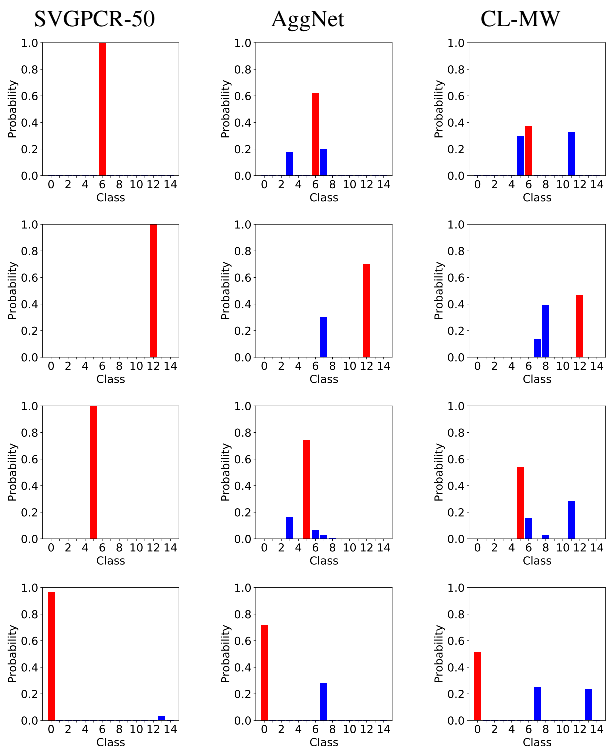


Figure 13: Test predictive distribution obtained by the compared methods for four different test instances in the LIGO experiment. All the methods assign the highest probability to the correct class (which is highlighted in red). However, from left to right, the quality of the predictive distribution decreases, as greater probability is assigned to wrong classes. This is related to the uncertainty quantification capabilities of each method, an aspect at which the GP modelling of SVGPCR stands out. Moreover, these differences in the predictive distributions are behind the superiority of SVGPCR in test likelihood (table 5).

One of the most distinctive features for all instances and methods is the predominance of high values in the diagonal. This was considered as a positive feedback by astrophysicists, as it means that annotators have been generally well instructed to distinguish among glitches. Additionally, other patterns out of the diagonal are worth an analysis. For the first column (first volunteer), SVGPCR and AggNet detect that glitches of type 1 (i.e. "1400Ripple", recall figure 3) are classified as class 13 ("Violin Mode Harmonic"). This is a very frequent mistake according to experts, since the general appearance of both glitches is similar. We also observe that CL-MW does not agree on this prediction. This discrepancy of CL-MW for some particular patterns is recurrent across different annotators, and can be attributed to the different modelling of the annotators noise (non-probabilistic one, but through weights in the DNN). The second column shows a typical conservative annotator, who resorts too frequently to the catch-all "Other" class. This is reflected in the persistent high values of the row number 8 in the matrices, regardless of the column (the real class). For the third column, the three methods identify the confusion from "Violin Mode Harmonic" to "1400Ripple". Notice that this is the opposite to the first annotator, where the confusion was the other way round. In the fourth annotator, SVGPCR detects the same issue with "Violin Mode Harmonic" and "1400Ripple", whereas the others are less certain about this. Moreover, AggNet

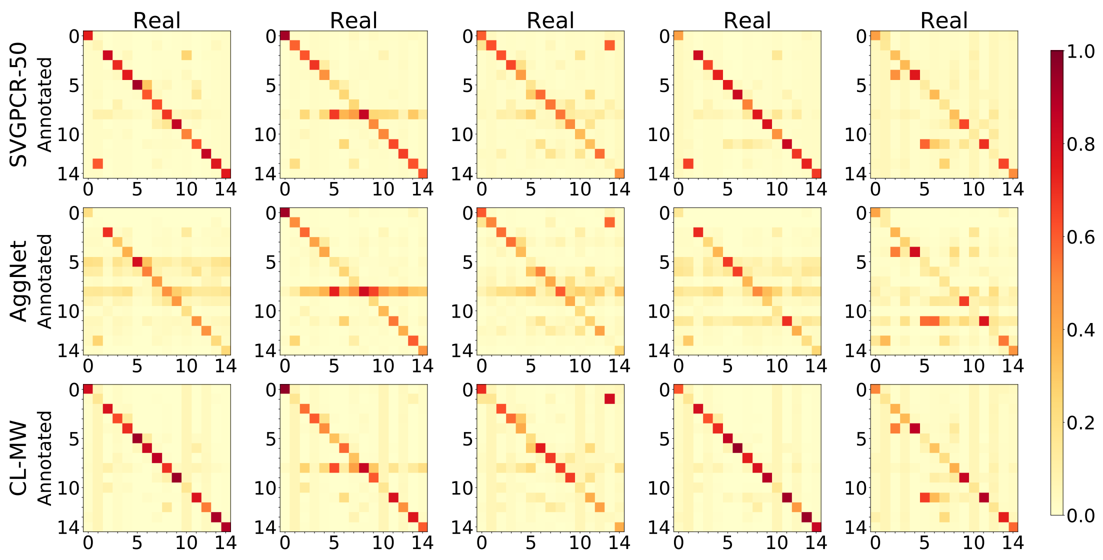


Figure 14: Confusion matrices estimated by the three families of algorithms (rows) and for five different annotators (columns) in the LIGO experiment. In each matrix, the real class is represented in the x-axis, and the annotated one in the y-axis (that is, the matrices add up to one by columns). For every annotator, the general structure of the three estimated matrices is similar. In particular, the highest values are located in the diagonal, which means that annotators have been correctly instructed in general. Moreover, recall that SVGPCR provides a full probability distribution (in particular uncertainties) for these predictions.

exhibits a noisy behavior compared to SVGPCR and CL-MW. Although perhaps less explicitly, this can be also observed across different annotators, and might be due to the iterative nature of AggNet, which does not allow for an end-to-end learning and leaves some extra noise after training. In the fifth annotator, the three methods identify a very common confusion, which is labelling instances whose real class is “Blip” as “Koifish” (classes 2 and 4, respectively). Although these glitches seem pretty different in the paradigmatic examples shown in figure 3, wider “Blip” and narrower “Koifish” are frequent in the data set, and might mislead a non-expert volunteer.

Most importantly, the identification of all these wrong behaviors allows crowdsourcing methods to take full advantage of the noisy annotations. It is also worth noticing that the Bayesian nature of SVGPCR provides uncertainties for the confusion matrices obtained here (recall the full posterior Dirichlet distributions in eq. (14)), which is not available for the DL-based methods.

5 Exploring recent inference techniques

Once SVGPCR has been successfully developed, this section explores modern GP inference approaches beyond the standard SVGP. As motivated in the introduction, Section 5.1 uses Normalizing Flows [Rezende and Mohamed, 2015] to represent more complex posterior distributions for SVGPCR. Then, Section 5.2 adapts the GP-Net model [Shi et al., 2019] to crowdsourcing.

Table 6: Per-class and global test accuracy for recent inference methods on GPs. Mean and standard deviation over ten runs are shown. The best performing SVGPCR method is shown for comparison. Both NF and GP-Net slightly outperform SVGPCR. Moreover, unlike inducing points based ones, GP-Net achieves very competitive results for $M = 10$, as theoretically expected.

	SVGPCR (M=50)	SVGPCR-NF (M=10)	SVGPCR-NF (M=50)	SVGPCR-NF (M=100)	GPNETCR (M=10)	GPNETCR (M=50)	GPNETCR (M=100)
1080LINE	.9720(.0076)	**.9756(.0098)**	.9724(.0072)	.9719(.0067)	.9740(.0045)	.9732(.0037)	.9727(.0042)
1400RIPPLE	.8528(.0203)	.2374(.3627)	.8463(.0216)	**.8537(.0209)**	.8398(.0247)	.8195(.0272)	.8398(.0499)
BLIP	.9643(.0044)	.8718(.2906)	.9632(.0042)	.9638(.0043)	.9665(.0089)	**.9679(.0038)**	.9645(.0066)
EXTR.LOUD	.7261(.0398)	.5727(.2887)	.7091(.0322)	.7080(.0469)	.7455(.0629)	.7420(.0653)	**.7523(.0794)**
KOIFISH	.8844(.0129)	.7996(.2669)	.8845(.0144)	.8841(.0139)	.8825(.0141)	.8845(.0206)	**.8912(.0130)**
L.F.BURST	.8864(.0087)	.7974(.2660)	.8890(.0097)	.8832(.0090)	.8896(.0162)	**.8950(.0136)**	.8864(.0108)
L.F.LINE	.9150(.0105)	.8296(.2766)	.9159(.0097)	**.9231(.0119)**	.9096(.0184)	.9077(.0178)	.9116(.0105)
NOGLITCH	.7941(.0223)	.7123(.2380)	.7919(.0171)	**.7952(.0146)**	.7739(.0285)	.7797(.0136)	.7800(.0240)
OTHER	.3977(.0083)	.3630(.1212)	.3996(.0102)	.4004(.0079)	.4061(.0253)	.4011(.0157)	**.4069(.0152)**
P.L.60HZ	.8438(.0102)	.7468(.2495)	.8472(.0100)	.8443(.0146)	.8574(.0096)	**.8634(.0055)**	.8579(.0127)
REP.BLIPS	.6649(.0215)	.0000(.0000)	.6598(.0198)	**.6726(.0284)**	.5812(.0608)	.6043(.0465)	.6068(.0606)
SCATT.LIGHT	.9600(.0050)	.8662(.2888)	.9567(.0049)	.9584(.0033)	**.9630(.0088)**	.9606(.0067)	.9622(.0068)
SCRATCHY	.8953(.0206)	.8067(.2694)	**.9033(.0194)**	.8973(.0164)	.9027(.0191)	.8880(.0183)	.8947(.0217)
VIOLIN	.9899(.0027)	.8924(.2975)	.9903(.0026)	.9906(.0023)	.9924(.0028)	**.9935(.0012)**	.9921(.0020)
WHISTLE	.9166(.0070)	.8184(.2732)	.9219(.0047)	.9237(.0040)	**.9281(.0156)**	.9272(.0096)	.9254(.0106)
GLOBAL	.9184(.0031)	.8400(.1935)	.9185(.0028)	**.9192(.0020)**	.9185(.0021)	.9186(.0018)	.9184(.0021)

Table 7: Per-class and global test likelihood for recent inference methods on GPs. Mean and standard deviation over ten runs are shown. The best performing SVGPCR method is shown for comparison. Both NF and GP-Net slightly outperform SVGPCR. Moreover, unlike inducing points based ones, GP-Net achieves very competitive results for $M = 10$, as theoretically expected.

	SVGPCR (M=50)	SVGPCR-NF (M=10)	SVGPCR-NF (M=50)	SVGPCR-NF (M=100)	GPNETCR (M=10)	GPNETCR (M=50)	GPNETCR (M=100)
1080LINE	.9689(.0082)	.8789(.2708)	.9693(.0074)	.9685(.0066)	**.9712(.0043)**	.9707(.0037)	.9704(.0043)
1400RIPPLE	**.8475(.0182)**	.2446(.3597)	.8423(.0234)	.8451(.0209)	.8373(.0233)	.8167(.0261)	.8331(.0490)
BLIP	.9606(.0044)	.8749(.2695)	.9598(.0044)	.9592(.0044)	.9632(.0089)	**.9645(.0041)**	.9615(.0060)
EXTR.LOUD	.7266(.0435)	.5724(.2712)	.7016(.0382)	.7069(.0487)	.7385(.0659)	.7342(.0611)	**.7447(.0740)**
KOIFISH	.8788(.0109)	.8030(.2458)	.8771(.0130)	.8775(.0116)	.8806(.0145)	.8813(.0192)	**.8839(.0130)**
L.F.BURST	.8851(.0083)	.8027(.2456)	.8862(.0096)	.8803(.0095)	.8877(.0162)	**.8919(.0136)**	.8835(.0107)
L.F.LINE	.9125(.0094)	.8329(.2556)	.9122(.0104)	**.9179(.0116)**	.9065(.0182)	.9045(.0181)	.9083(.0104)
NOGLITCH	.7919(.0238)	.7127(.2160)	.7883(.0170)	**.7929(.0143)**	.7688(.0283)	.7776(.0131)	.7799(.0220)
OTHER	.3959(.0095)	.3690(.1012)	.3963(.0089)	.3958(.0078)	**.4056(.0259)**	.3999(.0153)	.4041(.0145)
P.L.60HZ	.8384(.0107)	.7506(.2286)	.8421(.0100)	.8393(.0134)	.8546(.0115)	**.8595(.0057)**	.8542(.0134)
REP.BLIPS	.6581(.0191)	.0067(.0199)	.6537(.0221)	**.6641(.0259)**	.5834(.0663)	.6031(.0412)	.6043(.0596)
SCATT.LIGHT	.9558(.0047)	.8700(.2679)	.9523(.0045)	.9537(.0033)	**.9597(.0080)**	.9578(.0061)	.9593(.0066)
SCRATCHY	.8911(.0215)	.8029(.2463)	.8959(.0203)	.8901(.0168)	**.8985(.0201)**	.8826(.0163)	.8904(.0245)
VIOLIN	.9875(.0020)	.8967(.2767)	.9876(.0021)	.9879(.0018)	.9898(.0026)	**.9909(.0011)**	.9900(.0018)
WHISTLE	.9155(.0064)	.8163(.2502)	.9180(.0062)	.9185(.0052)	**.9297(.0123)**	.9253(.0076)	.9224(.0092)
GLOBAL	.9154(.0033)	.8176(.2504)	.9148(.0029)	.9151(.0018)	.9157(.0020)	**.9158(.0019)**	.9155(.0023)

5.1 Normalizing Flows

Normalizing Flows (NF), originally introduced in [Rezende and Mohamed, 2015], see also the recent review [Papamakarios et al., 2019], have become a very popular technique to represent complex distributions that can be used as approximate posteriors in VI. The idea is to transform a simple base distribution (typically a Gaussian) through a sequence of invertible transformations. Sampling from such a distribution is straightforward, and its density depends on the determinant of the transformations Jacobian, which must be cheap to compute. Different transformations yield different NFs, such as planar and radial [Rezende and Mohamed, 2015], inverse autoregressive flow [Kingma et al., 2016], or masked autoregressive flow [Papamakarios et al., 2017].

In our model of SVGPCR, we used an approximate Gaussian posterior $\mathrm{q}(\mathbf{U})$ for the inducing points, recall eq. (13). This yields a tractable KL term in the ELBO between two Gaussians (prior and posterior), recall eq. (15). However, the true posterior might not be Gaussian, since we are using a non-conjugate likelihood for $\mathbf{z}|\mathbf{F}$ (in addition to the crowdsourcing likelihood for $\mathbf{Y}|\mathbf{z}$).

In this section, we consider an approximate posterior for $\mathbf{U}$ based on a planar NF with Leaky ReLu (LReLu) non-linearities. More specifically, each $\mathbf{u}_k$ is defined through $\mathbf{u}_k^0 \to \cdots \to \mathbf{u}_k^L = \mathbf{u}_k$, where $\mathbf{u}_k^0 \sim \mathcal{N}(\mathbf{m}_k^0, \mathbf{S}_k^0)$ and $\mathbf{u}_k^l = f_k^l(\mathbf{u}_k^{l-1})$ with $f_k^l(\mathbf{a}) = \mathbf{a} + \mathbf{v} \cdot \mathrm{LReLu}(\mathbf{w}^\intercal \mathbf{a} + b)$ (logically, each $\mathbf{v}$, $\mathbf{w}$ and b depend on k and l, but this is omitted to lighten the notation). As explained in [Rezende and Mohamed, 2015, Section 4.1], the determinant of the Jacobian matrix for this transformation, $\partial f_k^l / \partial \mathbf{a}$, can be computed in $\mathcal{O}(M)$, where M is the dimensionality of $\mathbf{u}_k^l$, i.e. the number of inducing points. Therefore, the new variational parameters are $\mathbf{V} = \{\mathbf{m}_k^0, \mathbf{S}_k^0, \mathbf{v}_k^l, \mathbf{w}_k^l, b_k^l\}_{k,l}$. This approach will be called SVGPCR-NF. In our experiments, we will use a flow length of $L = 6$.

Following the approach in [Rezende and Mohamed, 2015], the new KL term in the ELBO is as follows:

$$\mathrm{KL}(\mathrm{q}(\mathbf{u}_k)||\mathrm{p}(\mathbf{u}_k)) = \mathbb{E}_{\mathrm{q}(\mathbf{u}_k^0)} \log \mathrm{q}(\mathbf{u}_k^0) - \sum_{l=1}^{L} \mathbb{E}_{\mathrm{q}(\mathbf{u}_k^0)} \log \left| \det \frac{\partial f_k^l}{\partial \mathbf{u}_k^{l-1}} \right| - \mathbb{E}_{\mathrm{q}(\mathbf{u}_k^0)} \log \mathrm{p}(\mathbf{u}_k^L). \tag{17}$$

The first term can be computed in closed form (the entropy of a Gaussian is well-known), the second term can be stochastically approximated in $\mathcal{O}(LM)$ by sampling from $\mathrm{q}(\mathbf{u}_k^0)$ and evaluating the known Jacobian, recall [Rezende and Mohamed, 2015, Section 4.1], and the same applies for the third one. The rest of the ELBO is as in eq. (15), where $\mathrm{q}(\mathbf{f}_{n,:})$ does not have a closed-form expression anymore, but it is approximated by sampling S values from the flow and averaging the S conditional Gaussians of $\mathbf{f}$ given $\mathbf{u}$. The same applies for predicting. Notice that the computational complexity is still dominated by the SVGPCR operations.

We apply SVGPCR-NF on the full LIGO dataset with $M = 10, 50, 100$, as was done for the standard SVGPCR. The results in terms of test accuracy and likelihood are shown in Tables 6 and 7, respectively. The performance is very similar to SVGPCR, exhibiting a slight improvement. This suggests that, in this particular case, the true posterior distribution might be unimodal and a Gaussian could be enough to represent it. Interesting properties of SVGPCR, such as the regularity across classes and avoiding dramatic failures in difficult ones, are also maintained. Moreover, just like for SVGPCR, results are clearly improved when growing from $M = 10$ to $M = 50$, but stay similar for $M = 100$. This is to be expected, since the underlying SVGP model is still the same, and it is well-known that few inducing points limit the expressiveness of the predictive distribution. Next we explore a recent model that adopts a different approach.

5.2 Inference Networks for GPs

In this section we adapt GP-Net [Shi et al., 2019] to the crowdsourcing scenario. Since the expressiveness of inducing points based sparse GP approaches is limited by the amount of such points, GP-Net explores the use of inference networks to approximate the posterior distribution of GPs. Inference is performed in the function space directly, see also [Sun et al., 2019], and the posterior stochastic process $\mathrm{q}(f)$ is modelled with a parametric inference network $\mathrm{q}_\gamma(f)$. The joint distribution must be Gaussian for a finite set of points, see [Shi et al., 2019, Section 4] for different choices, such as the random feature expansions that will be used here. To learn the parameters γ, a stochastic functional mirror-descent algorithm is tracked, which iteratively adapts the true stochastic posterior [Shi et al., 2019, Section 3.1]. In each step, the inference network is matched to such posterior on a measurement set $\mathbf{X}_M$ of size M, which allows for learning meaningful correlations [Shi et al., 2019, Section 3.2]. Importantly, notice that this M does not limit the expressiveness of the inference network q_γ.

In the crowdsourcing scenario, recall that eq. (7) defines the full probabilistic model before inducing points are introduced. Now, we consider an approximate posterior $\mathrm{q}(\mathbf{F}, \mathbf{Z}, \mathbf{R}) = \mathrm{q}(\mathbf{F})\mathrm{q}(\mathbf{Z})\mathrm{q}(\mathbf{R})$ where $\mathrm{q}(\mathbf{F})$ is given by K inference networks $\mathrm{q}_{\gamma_1}(f_1), \ldots, \mathrm{q}_{\gamma_K}(f_K)$, one for each class. We use random feature expansions with 100 hidden units, recall [Shi et al., 2019, Section 4]. Then, VI can be applied to minimize $\mathrm{KL}(\mathrm{q}(\mathbf{F}, \mathbf{Z}, \mathbf{R})||\mathrm{p}(\mathbf{F}, \mathbf{Z}, \mathbf{R}|\mathbf{Y}, \boldsymbol{\Theta}))$ in turns. Fixing $\mathrm{q}(\mathbf{F})$, and following the derivation of eq. (15), a gradient step must be given on $\{q_{nk}\}$ and $\{\tilde{\alpha}^a_{ij}\}$ (the variational parameters of $\mathrm{q}(\mathbf{Z})$ and $\mathrm{q}(\mathbf{R})$ respectively) to maximize

$$\begin{aligned}\mathcal{F}(\{q_{nk}\}, \{\tilde{\alpha}^a_{ij}\}) = &\sum_{n=1}^{N}\sum_{k=1}^{K} q_{nk}\mathbb{E}_{\mathrm{q}_\gamma(\mathbf{f}_{n,:})}\left[\log \mathrm{p}(\mathbf{e}_k|\mathbf{f}_{n,:})\right] + \\ &+\sum_{n=1}^{N}\sum_{a\in A_n}\sum_{\mathbf{y}\in\mathbf{Y}^a_n}\sum_{k=1}^{K} q_{nk}\mathbb{E}_{\mathrm{q}(\mathbf{r}^a_k)}\left[\log \mathrm{p}(\mathbf{y}|\mathbf{e}_k, \mathbf{r}^a_k)\right] - \\ &-\sum_{n=1}^{N}\sum_{k=1}^{K} q_{nk}\log q_{nk} - \sum_{a=1}^{A}\sum_{k=1}^{K}\mathrm{KL}(\mathrm{q}(\mathbf{r}^a_k)||\mathrm{p}(\mathbf{r}^a_k)).\end{aligned} \tag{18}$$

This is analogous to eq. (15), but the inducing points term is not present anymore, and $\mathrm{q}_\gamma(\mathbf{f}_{n,:})$ is given by the inference networks (fixed in this step; γ denotes $\gamma_1, \ldots, \gamma_K$ jointly). Then, fixing $\mathrm{q}(\mathbf{U})$ and $\mathrm{q}(\mathbf{R})$, and following the derivations in [Shi et al., 2019, Section 3.2] (non-conjugate likelihood case), the inference networks parameters γ must be updated by maximizing

$$\begin{aligned}\mathcal{L}(\gamma) = \mathbb{E}_{\mathrm{q}_\gamma(\mathbf{F}_M,\mathbf{F})}\Bigg[&\beta_t\sum_{n=1}^{N}\sum_{k=1}^{K} q_{nk}\log \mathrm{p}(\mathbf{e}_k|\mathbf{f}_{n,:}) + \\ &+\beta_t\log \mathrm{p}(\mathbf{F}_M, \mathbf{F}) + (1-\beta_t)\log \mathrm{q}_{\gamma_t}(\mathbf{F}_M, \mathbf{F}) - \log \mathrm{q}_\gamma(\mathbf{F}_M, \mathbf{F})\Bigg].\end{aligned}$$

Here, q_{γ_t} refers to the previous value of the inference network, $\mathbf{F}_M$ denotes the evaluation of the K inference networks on the measurement set $\mathbf{X}_M$ (which is randomly sampled from the training set as in [Shi et al., 2019]), and β_t is the learning rate of the functional mirror-descent algorithm (which is set to $\beta_t = \beta_0(1 + \xi\sqrt{t})^{-1}$ as in [Shi et al., 2019]). In practice, the first term is approximated with a mini-batch, as in eqs. (15) and (18). The GP hyperparameters $\boldsymbol{\Theta}$ are optimized as in [Shi et al., 2019], by minimizing the KL divergence between q_{γ_t} and the GP prior on the current mini-batch. This method is called GPNETCR.

As with SVGPCR and SVGPCR-NF, we apply GPNETCR on the full LIGO dataset with $M = 10, 50, 100$ (here M is the measurement set size). The results in terms of test accuracy and likelihood are shown in Table 6 and 7, respectively. The performance for $M = 50, 100$ is very similar to SVGPCR-NF, i.e. just slightly better than SVGPCR. This confirms the robustness and convenience of the GP-based crowdsourcing formulations, which show good results across classes and avoid noticeable failures in difficult ones. However, the results for $M = 10$ are very different. Whereas 10 inducing points do not yield a sufficiently expressive posterior in SVGPCR(-NF), the GPNETCR inference network capacity is not constrained by M. In fact, the GPNETCR with $M = 10$ performs better than SVGPCR for any amount of M. This confirms that the properties of GP-Net can be extended to crowdsourcing.

6 Conclusions and future work

In this work we have first introduced SVGPCR, an extension of SVGP to crowdsourcing that can scale up to very large data sets through its mini-batch training scheme. The motivation is the problem of glitch classification in the laureate LIGO project, which is addressed with crowdsourcing techniques in the GravitySpy sub-project. To that end, and in order to obtain accurate predictive distributions, astrophysicists were interested in combining the excellent uncertainty quantification of GP-based crowdsourcing methods with the scalability of those based on deep learning (DL). The proposed SVGPCR brings back GP-based methods to the state of the art in crowdsourcing.

SVGPCR is competitive with DL-based approaches in terms of test accuracy and computational cost, and stands out in terms of predictive distribution quality. Moreover, its behavior naturally follows its theoretical formulation: it provides very accurate estimations for the annotators expertise degree, and the inducing points influence the test performance and the computational cost as expected. We further leveraged recent inference techniques to propose SVGPCR-NF and GP-NETCR.

In the LIGO problem, the glitches were given by relevant features extracted by astrophysicists. However, in the case of more complex data such as images, audio or natural language, DL-based methods can benefit from convolutional layers in the deep neural network. From a probabilistic perspective, this could be addresed through Deep Gaussian Processes [Salimbeni and Deisenroth, 2017] and the very recent attempts to introduce convolutional structure in GPs [Van der Wilk et al., 2017, Blomqvist et al., 2018]. Moreover, recent successful models combining the benefits of GPs and Neural Networks, such as (Conditional) Neural Processes [Garnelo et al., 2018a, Garnelo et al., 2018b], could be extended to crowdsourcing. In fact, these can also be endowed with the aforementioned convolutional structure [Gordon et al., 2019].

A Supplementary Material

A.1 Derivation of the ELBO

Let us derive step by step the eq. (15) in the paper:

$$
\begin{aligned}
\mathrm{ELBO}(\mathbf{V},\boldsymbol{\Theta},\tilde{\mathbf{X}}) &= \mathbb{E}_{\mathrm{q}(\mathbf{Z})\mathrm{p}(\mathbf{F}|\mathbf{U})\mathrm{q}(\mathbf{U})\mathrm{q}(\mathbf{R})}\left(\log\frac{\mathrm{p}(\mathbf{Y}|\mathbf{Z},\mathbf{R})\mathrm{p}(\mathbf{Z}|\mathbf{F})\cancel{\mathrm{p}(\mathbf{F}|\mathbf{U})}\mathrm{p}(\mathbf{U})\mathrm{p}(\mathbf{R})}{\mathrm{q}(\mathbf{Z})\cancel{\mathrm{p}(\mathbf{F}|\mathbf{U})}\mathrm{q}(\mathbf{U})\mathrm{q}(\mathbf{R})}\right)\\
&\qquad\textit{[Grouping and integrating straightforward terms]}\\
&= \mathbb{E}_{\mathrm{q}(\mathbf{Z})\mathrm{q}(\mathbf{R})}\log\mathrm{p}(\mathbf{Y}|\mathbf{Z},\mathbf{R}) + \mathbb{E}_{\mathrm{q}(\mathbf{Z})\mathrm{p}(\mathbf{F}|\mathbf{U})\mathrm{q}(\mathbf{U})}\log\mathrm{p}(\mathbf{Z}|\mathbf{F}) - \mathbb{E}_{\mathrm{q}(\mathbf{Z})}\log\mathrm{q}(\mathbf{Z})\\
&+ \mathbb{E}_{\mathrm{q}(\mathbf{U})}\left(\log\frac{\mathrm{p}(\mathbf{U})}{\mathrm{q}(\mathbf{U})}\right) + \mathbb{E}_{\mathrm{q}(\mathbf{R})}\left(\log\frac{\mathrm{p}(\mathbf{R})}{\mathrm{q}(\mathbf{R})}\right)\\
&\qquad\textit{[Using the different terms expressions and KL div. definition]}\\
&= \sum_{n=1}^{N}\sum_{a\in A_n}\sum_{\mathbf{y}\in\mathbf{Y}_n^a}\mathbb{E}_{\mathrm{q}(\mathbf{z}_n)\mathrm{q}(\mathbf{R}^a)}\log\mathrm{p}(\mathbf{y}|\mathbf{z}_n,\mathbf{R}^a) + \sum_{n=1}^{N}\mathbb{E}_{\mathrm{q}(\mathbf{z}_n)\mathrm{q}(\mathbf{f}_{n,:})}\log\mathrm{p}(\mathbf{z}_n|\mathbf{f}_{n,:})\\
&- \sum_{n=1}^{N}\mathbb{E}_{\mathrm{q}(\mathbf{z}_n)}\log\mathrm{q}(\mathbf{z}_n) - \sum_{k=1}^{K}\mathrm{KL}(\mathrm{q}(\mathbf{u}_k)||\mathrm{p}(\mathbf{u}_k)) - \sum_{a=1}^{A}\sum_{k=1}^{K}\mathrm{KL}(\mathrm{q}(\mathbf{r}_k^a)||\mathrm{p}(\mathbf{r}_k^a))\\
&\qquad\textit{[Doing the expectation over the discrete distribution } \mathrm{q}(\mathbf{z}_n)\textit{]}\\
&= \sum_{n=1}^{N}\sum_{a\in A_n}\sum_{\mathbf{y}\in\mathbf{Y}_n^a}\sum_{k=1}^{K}q_{nk}\mathbb{E}_{\mathrm{q}(\mathbf{r}_k^a)}\log\mathrm{p}(\mathbf{y}|\mathbf{e}_k,\mathbf{r}_k^a) + \sum_{n=1}^{N}\sum_{k=1}^{K}q_{nk}\mathbb{E}_{\mathrm{q}(\mathbf{f}_{n,:})}\log\mathrm{p}(\mathbf{e}_k|\mathbf{f}_{n,:})\\
&- \sum_{n=1}^{N}\sum_{k=1}^{K}q_{nk}\log q_{nk} - \sum_{k=1}^{K}\mathrm{KL}(\mathrm{q}(\mathbf{u}_k)||\mathrm{p}(\mathbf{u}_k)) - \sum_{a=1}^{A}\sum_{k=1}^{K}\mathrm{KL}(\mathrm{q}(\mathbf{r}_k^a)||\mathrm{p}(\mathbf{r}_k^a)).
\end{aligned}
\tag{19}
$$

Indeed, this final expression is the same formula as that in eq. (15) in the article. The posterior over $\mathbf{F}$, $\mathrm{q}(\mathbf{F})$, whose marginal $\mathrm{q}(\mathbf{f}_{n,:})$ is used in the second term, can be easily obtained in closed form, since both $\mathrm{q}(\mathbf{F}|\mathbf{U}) = \mathrm{p}(\mathbf{F}|\mathbf{U})$ and $\mathrm{q}(\mathbf{U})$ are Gaussians:

$$
\mathrm{q}(\mathbf{F}) = \prod_{k=1}^{K}\mathrm{q}(\mathbf{f}_k), \quad \text{with} \quad \mathrm{q}(\mathbf{f}_k) = \mathcal{N}\left(\mathbf{f}_k|\mathbf{B}\cdot\mathbf{m}_k, \mathbf{K}_{\mathbf{XX}} + \mathbf{B}(\mathbf{S}_k - \mathbf{K}_{\tilde{\mathbf{X}}\tilde{\mathbf{X}}})\mathbf{B}^{\intercal}\right), \quad \mathbf{B} = \mathbf{K}_{\mathbf{X}\tilde{\mathbf{X}}}\mathbf{K}_{\tilde{\mathbf{X}}\tilde{\mathbf{X}}}^{-1}.
$$

A.2 Explicit expressions for the ELBO terms

In order to maximize the ELBO w.r.t. the variational paremeters $\mathbf{V}$, the kernel parameters $\boldsymbol{\Theta}$, and the inducing locations $\tilde{\mathbf{X}}$, we need explicit expressions on these variables. All the five terms in eq. (19) but the second one can be analytically expressed in terms of $\mathbf{V}$, $\boldsymbol{\Theta}$, $\tilde{\mathbf{X}}$. Moreover, the second one admits an approximated expression through Gaussian-Hermite quadrature [Olver et al., 2010b].

For the first term, we use the expectation of the logarithm of a component of a Dirichlet distribution (which is in the exponential family): if $(X_1,\ldots,X_S) \to \mathrm{Dir}(\gamma_1,\ldots,\gamma_S)$, then $\mathbb{E}(\log X_i) =$

$\psi(\gamma_i) - \psi(\sum_{i=1}^{S} \gamma_i)$ [Bishop, 2006]. Here, ψ is the digamma function, which is the logarithmic derivative of the gamma function and is implemented in TensorFlow as `tf.math.digamma`. With this, we have:

$$q_{nk} \cdot \mathbb{E}_{\mathrm{q}(\mathbf{r}_k^a)} \log \mathrm{p}(\mathbf{y}|\mathbf{e}_k, \mathbf{r}_k^a) = q_{nk} \cdot \prod_{i=1}^{K} \left(\psi(\tilde{\alpha}_{ik}^a) - \psi\left(\sum_{c=1}^{K} \tilde{\alpha}_{ck}^a\right)\right)^{y_i},$$

which depends on $\{\tilde{\alpha}_{ij}^a\}$ and $\{q_{nk}\}$, recall table 1 in the article.

The second term involves the expectation of the log-likelihood over a Gaussian. If the likelihood is also defined through a Gaussian (like in most regression problems) then this can be obtained in closed-form. However, this is not possible for the classification likelihoods that are used in crowdsourcing. In this case, Gaussian-Hermite quadrature is applied in order to approximate the result [Olver et al., 2010b]. For the particular case of our RobustMax likelihood, the approximation is already integrated within the GPflow library, see the `variational_expectations` method of the `Multiclass` likelihood class [Matthews et al., 2017]. As summarized in table 1 in the main paper, this second term depends on $\{q_{nk}\}$, $\{\mathbf{m}_k, \mathbf{S}_k\}$, $\boldsymbol{\Theta}$, and $\tilde{\mathbf{X}}$. The third term is already expressed in terms of $\{q_{nk}\}$ in eq. (19).

For the fourth term, only the well-known expression of the KL divergence between (multivariate) Gaussians is needed [Rasmussen and Williams, 2006]:

$$\mathrm{KL}(\mathrm{q}(\mathbf{u}_k)||\mathrm{p}(\mathbf{u}_k)) = \mathrm{KL}\left(\mathcal{N}(\boldsymbol{\mu}_k, \mathbf{S}_k)||\mathcal{N}(\mathbf{0}, \mathbf{K})\right) = \frac{1}{2}\Big(\mathrm{tr}\left(\mathbf{K}^{-1}\mathbf{S}_k\right) + \\ + \boldsymbol{\mu}_k^{\intercal}\mathbf{K}^{-1}\boldsymbol{\mu}_k - M + \log\left(\frac{\det \mathbf{K}}{\det \mathbf{S}_k}\right)\Big),$$

where $\mathbf{K} = \mathbf{K}_{\tilde{\mathbf{X}}\tilde{\mathbf{X}}}$ is the prior covariance matrix for the inducing points, and M is the dimension of the variable $\mathbf{u}_k$ (i.e. the amount of inducing points). This expression depends on $\{\mathbf{m}_k, \mathbf{S}_k\}$, $\boldsymbol{\Theta}$, and $\tilde{\mathbf{X}}$ (the last two through $\mathbf{K}$).

In the fifth term, it is straightforward to obtain a closed-form expression for the KL divergence between Dirichlet distributions:

$$\mathrm{KL}(\mathrm{q}(\mathbf{r}_k^a)||\mathrm{p}(\mathbf{r}_k^a)) = \log B(\boldsymbol{\alpha}_k^a) - \log B(\tilde{\boldsymbol{\alpha}}_k^a) + \sum_{i=1}^{K} (\tilde{\alpha}_{ik}^a - \alpha_{ik}^a) \cdot \left(\psi(\tilde{\alpha}_{ik}^a) - \psi\left(\sum_{j=1}^{K} \tilde{\alpha}_{jk}^a\right)\right).$$

Here, ψ is the aforementioned digamma function, and B is the beta function, which is also implemented in TensorFlow and is defined as $B(\boldsymbol{\alpha}) = B(\alpha_1, \ldots, \alpha_K) = (\prod_i \Gamma(\alpha_i))/\Gamma(\sum_i \alpha_i)$, with Γ the Gamma function. This term only depends on $\{\tilde{\alpha}_{ij}^a\}$.

A.3 Predictive distribution for SVGPCR

In the full LIGO experiment, Section 4.3 of the paper, Figure 13 shows the predictive distribution for the test instances that most contribute to the outperformance in test likelihood. The goal is to illustrate this global metric through some specific examples, showing that SVGPCR can be very confident about straightforward decisions. However, it may lead one to think that SVGPCR cannot provide uncertainty if needed. Figure 15 shows that this is not the case, and SVGPCR also provides uncertainty when it is not sure about the decision.

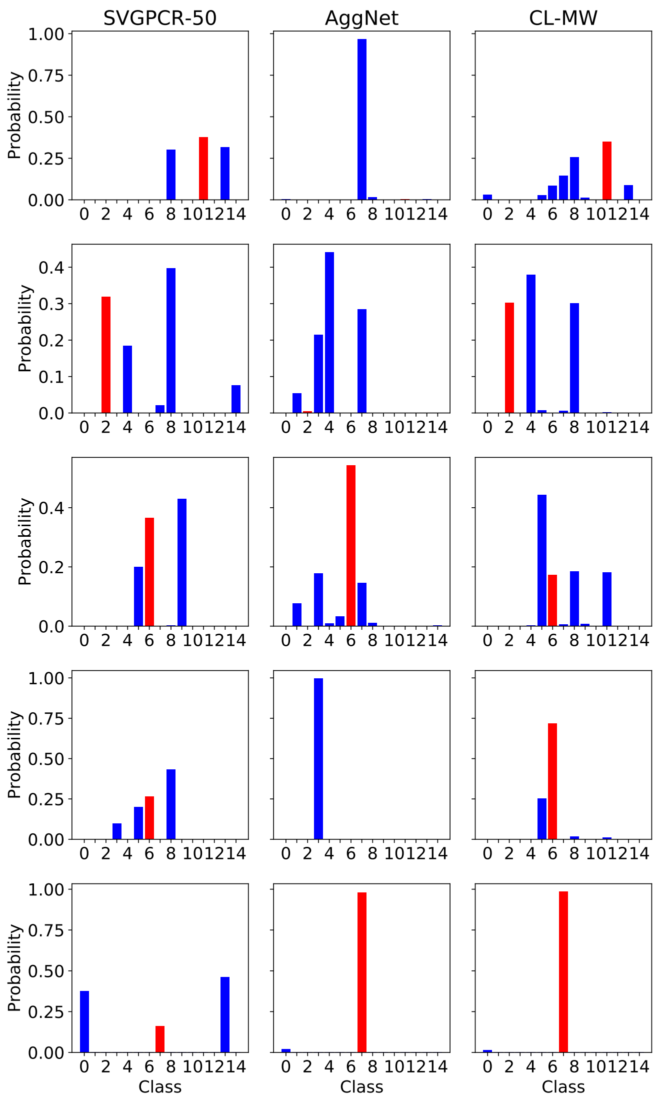


Figure 15: Test predictive distribution obtained by the compared methods for five different test instances in the LIGO experiment. As a Bayesian method, SVGPCR provides uncertainty in the predictions when it is not confident about the outcome.

References

[Abbott et al., 2016] Abbott, B. P., Abbott, R., Abbott, T., Abernathy, M., Acernese, F., Ackley, K., Adamo, M., Adams, C., Adams, T., Addesso, P., et al. (2016). Characterization of transient noise in advanced ligo relevant to gravitational wave signal gw150914. *Classical and Quantum Gravity*, 33(13):134001.

[Abramovici et al., 1992] Abramovici, A., Althouse, W. E., Drever, R. W. P., Gürsel, Y., Kawamura, S., Raab, F. J., Shoemaker, D., Sievers, L., Spero, R. E., Thorne, K. S., Vogt, R. E., Weiss, R., Whitcomb, S. E., and Zucker, M. E. (1992). Ligo: The laser interferometer gravitational-wave observatory. *Science*, 256(5055):325–333.

[Albarqouni et al., 2016] Albarqouni, S., Baur, C., Achilles, F., Belagiannis, V., Demirci, S., and Navab, N. (2016). AggNet: Deep Learning From Crowds for Mitosis Detection in Breast Cancer Histology Images. *IEEE Transactions on Medical Imaging*, 35(5):1313–1321.

[B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration), 2016] B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration) (2016). Observation of gravitational waves from a binary black hole merger. *Physical Review Letters*, 116:061102.

[Bahaadini et al., 2018] Bahaadini, S., Noroozi, V., Rohani, N., Coughlin, S., Zevin, M., Smith, J., Kalogera, V., and Katsaggelos, A. (2018). Machine learning for gravity spy: Glitch classification and dataset. *Information Sciences*, 444:172–186.

[Bauer et al., 2016] Bauer, M., van der Wilk, M., and Rasmussen, C. E. (2016). Understanding probabilistic sparse gaussian process approximations. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1533–1541.

[Bishop, 2006] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Blei et al., 2017] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

[Blomqvist et al., 2018] Blomqvist, K., Kaski, S., and Heinonen, M. (2018). Deep convolutional gaussian processes. *arXiv preprint arXiv:1810.03052*.

[Buhrmester et al., 2011] Buhrmester, M., Kwang, T., and Gosling, S. D. (2011). Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5.

[Burt et al., 2019] Burt, D., Rasmussen, C. E., and Van Der Wilk, M. (2019). Rates of convergence for sparse variational gaussian process regression. In *International Conference on Machine Learning*, pages 862–871.

[Casale et al., 2018] Casale, F. P., Dalca, A., Saglietti, L., Listgarten, J., and Fusi, N. (2018). Gaussian process prior variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 10369–10380.

[Castelvecchi and Witze, 2016] Castelvecchi, D. and Witze, A. (2016). Einstein's gravitational waves found at last. *Nature news*.

[Dawid and Skene, 1979] Dawid, A. P. and Skene, A. M. (1979). Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm. *Journal of the Real Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

[Donmez and Carbonell, 2008] Donmez, P. and Carbonell, J. G. (2008). Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *ACM Conference on Information and Knowledge Management*, pages 619–628.

[Fritz et al., 2017] Fritz, S., See, L., Perger, C., McCallum, I., Schill, C., Schepaschenko, D., Duerauer, M., Karner, M., Dresel, C., Laso-Bayas, J.-C., et al. (2017). A global dataset of crowdsourced land cover and land use reference data. *Scientific data*, 4:170075.

[Garnelo et al., 2018a] Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. A. (2018a). Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713.

[Garnelo et al., 2018b] Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. (2018b). Neural processes. *arXiv preprint arXiv:1807.01622*.

[Gordon et al., 2019] Gordon, J., Bruinsma, W. P., Foong, A. Y., Requeima, J., Dubois, Y., and Turner, R. E. (2019). Convolutional conditional neural processes. *arXiv preprint arXiv:1910.13556*.

[Guan et al., 2018] Guan, M. Y., Gulshan, V., Dai, A. M., and Hinton, G. E. (2018). Who said what: Modeling individual labelers improves classification. In *Conference on Artificial Intelligence (AAAI)*, pages 3109–3118.

[Guerrini et al., 2018] Guerrini, C. J., Majumder, M. A., Lewellyn, M. J., and McGuire, A. L. (2018). Citizen science, public policy. *Science*, 361(6398):134–136.

[Heim et al., 2018] Heim, E., Seitel, A., Andrulis, J., Isensee, F., Stock, C., Ross, T., and Maier-Hein, L. (2018). Clickstream analysis for crowd-based object segmentation with confidence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2814–2826.

[Hensman et al., 2013] Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 282–290.

[Hensman et al., 2015a] Hensman, J., Matthews, A., and Ghahramani, Z. (2015a). Scalable Variational Gaussian Process Classification. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 38, pages 351–360.

[Hensman et al., 2015b] Hensman, J., Matthews, A. G., Filippone, M., and Ghahramani, Z. (2015b). Mcmc for variationally sparse gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1648–1656.

[Hernández-Lobato et al., 2011] Hernández-Lobato, D., Hernández-Lobato, J. M., and Dupont, P. (2011). Robust multi-class gaussian process classification. In *Advances in Neural Information Processing Systems (NIPS)*, pages 280–288.

[Hoffman et al., 2013] Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

[Ipeirotis et al., 2010] Ipeirotis, P. G., Provost, F., and Wang, J. (2010). Quality Management on Amazon Mechanical Turk. In *ACM SIGKDD Workshop on Human Computation (HCOMP'10)*, pages 64–67.

[Irwin, 2018] Irwin, A. (2018). No phds needed: how citizen science is transforming research. *Nature*, 562:480–482.

[Kennefick, 2016] Kennefick, D. J. (2016). *Traveling at the speed of thought: Einstein and the quest for gravitational waves*. Princeton university press.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference for Learning Representations (ICLR)*.

[Kingma et al., 2016] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751.

[LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Matthews et al., 2017] Matthews, D. G., Alexander, G., Van Der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. (2017). Gpflow: A gaussian process library using tensorflow. *The Journal of Machine Learning Research*, 18(1):1299–1304.

[Minka, 2001] Minka, T. P. (2001). *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, University of Cambridge.

[Morales-Álvarez et al., 2018] Morales-Álvarez, P., Pérez-Suay, A., Molina, R., and Camps-Valls, G. (2018). Remote sensing image classification with large-scale gaussian processes. *IEEE Transactions on Geoscience and Remote Sensing*, 56(2):1103–1114.

[Morales-Álvarez et al., 2019] Morales-Álvarez, P., Ruiz, P., Santos-Rodríguez, R., Molina, R., and Katsaggelos, A. K. (2019). Scalable and efficient learning from crowds with gaussian processes. *Information Fusion*, 52:110 – 127.

[Murphy, 2012] Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective*. MIT.

[Nuttall et al., 2015] Nuttall, L. K., Massinger, T. J., Areeda, J., Betzwieser, J., Dwyer, S., Effler, A., Fisher, R. P., Fritschel, P., Kissel, J. S., Lundgren, A. P., et al. (2015). Improving the data quality of advanced ligo based on early engineering run results. *Classical and Quantum Gravity*, 32(24):245005.

[Olver et al., 2010a] Olver, F. W., Lozier, D. W., Boisvert, R. F., and Clark, C. W. (2010a). *NIST handbook of mathematical functions*. Cambridge university press.

[Olver et al., 2010b] Olver, F. W., Lozier, D. W., Boisvert, R. F., and Clark, C. W. (2010b). *NIST Handbook of Mathematical Functions*. Cambridge University Press, New York, NY, USA, 1st edition.

[Papamakarios et al., 2019] Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2019). Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*.

[Papamakarios et al., 2017] Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT.

[Raykar et al., 2010] Raykar, V., Yu, S., Zhao, L., Hermosillo Valadez, G., Florin, C., Bogoni, L., and Moy, L. (2010). Learning from crowds. *The Journal of Machine Learning Research*, 11(Apr):1297–1322.

[Rezende and Mohamed, 2015] Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538.

[Rodrigues et al., 2017] Rodrigues, F., Lourenco, M., Ribeiro, B., and Pereira, F. (2017). Learning supervised topic models for classification and regression from crowds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2409–2422.

[Rodrigues and Pereira, 2018] Rodrigues, F. and Pereira, F. (2018). Deep learning from crowds. In *Conference on Artificial Intelligence (AAAI)*, pages 1611–1618.

[Rodrigues et al., 2014] Rodrigues, F., Pereira, F., and Ribeiro, B. (2014). Gaussian process classification and active learning with multiple annotators. In *International Conference on Machine Learning (ICML)*, pages 433–441.

[Ruiz et al., 2019] Ruiz, P., Morales-Álvarez, P., Molina, R., and Katsaggelos, A. K. (2019). Learning from crowds with variational gaussian processes. *Pattern Recognition*, 88:298 – 311.

[Saez-Rodriguez et al., 2016] Saez-Rodriguez, J., Costello, J. C., Friend, S. H., Kellen, M. R., Mangravite, L., Meyer, P., Norman, T., and Stolovitzky, G. (2016). Crowdsourcing biomedical research: leveraging communities as innovation engines. *Nature Reviews Genetics*, 17(8):470–486.

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

[Sheng et al., 2008] Sheng, V. S., Provost, F., and Ipeirotis, P. G. (2008). Get another label? improving data quality and data mining using multiple, noisy labelers. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622.

[Shi et al., 2019] Shi, J., Khan, M. E., and Zhu, J. (2019). Scalable training of inference networks for gaussian-process models. In *International Conference on Machine Learning*, pages 5758–5768.

[Snelson and Ghahramani, 2006] Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264.

[Snow et al., 2008] Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Y. (2008). Cheap and fast, but is it good?: evaluating non-expert annotations for natural language tasks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263.

[Sun et al., 2019] Sun, S., Zhang, G., Shi, J., and Grosse, R. (2019). Functional variational bayesian neural networks. In *International Conference on Learning Representations*.

[Tegner et al., 2018] Tegner, M., Bloem-Reddy, B., and Roberts, S. (2018). Sequential sampling of gaussian process latent variable models. *arXiv preprint arXiv:1807.04932*.

[Titsias, 2009] Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 5, pages 567–574.

[Van der Wilk et al., 2017] Van der Wilk, M., Rasmussen, C. E., and Hensman, J. (2017). Convolutional gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2849–2858.

[Whitehill et al., 2009] Whitehill, J.and Wu, T.-F., Bergsma, J., Movellan, J. R., and Ruvolo, P. L. (2009). Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2035–2043.

[Yan et al., 2014] Yan, Y., Rosales, R., Fung, G., Subramanian, R., and Dy, J. (2014). Learning from multiple annotators with varying expertise. *Machine Learning*, 95(3):291–327.

[Zevin et al., 2017] Zevin, M., Coughlin, S., Bahaadini, S., Besler, E., Rohani, N., Allen, S., et al. (2017). Gravity spy: integrating advanced ligo detector characterization, machine learning, and citizen science. *Classical and Quantum Gravity*, 34(6):064003.

CHAPTER 7

Integrating expert knowledge in Gaussian Processes based crowdsourcing

7.1 Publication details

Authors: Pablo Ruiz, Pablo Morales-Álvarez, Scott Coughlin, Rafael Molina, Aggelos K. Katsaggelos
Title: Probabilistic fusion of crowds and experts for the search of gravitational waves
Reference: Under review at the International Conference on Pattern Recognition (ICPR) 2020
Status: Under review
Quality indices:

- GGS Rating: A-
- GGS Class: 2
- Core: B

7.2 Main contributions

- In this work we propose SVGPCR-Mix, a novel methodology to integrate expert labels into the GP-based crowdsourcing model presented in Chapter 6 (the one based on inducing points). The resulting training objective is a natural combination of crowdsourcing and standard classification ones. Desirable theoretical properties of crowdsourcing, such as scalability and sound estimation of annotator behavior and ground truth, translate straightforwardly into this setting. Moreover, the expert labels play the role of anchor points to disclose the underlying truth and the annotators' behavior.

- The proposed approach is evaluated on two types of datasets. First, a synthetic experiment provides a controlled setting to analyze the behavior of the method. Second, it is applied on the LIGO dataset (recall Chapter 6), which contains a golden set of expert labels that motivated the development of this new method.

We show that the combination of both sources of information yields better results than when used separately.

PROBABILISTIC FUSION OF CROWDS AND EXPERTS FOR THE SEARCH OF GRAVITATIONAL WAVES

Pablo Ruiz
Dept. of Electrical Engineering and Computer Science
Northwestern University, USA

Pablo Morales-Álvarez
Dept. of Computer Science and AI
University of Granada, Spain

Scott Coughlin
Center for Interdisciplinary Exploration and Research in Astrophysics
Northwestern University, USA

Rafael Molina
Dept. of Computer Science and AI
University of Granada, Spain

Aggelos K. Katsaggelos
Dept. of Electrical Engineering and Computer Science
Northwestern University, USA

ABSTRACT

The acquisition of labels provided by experts (expert labels) in real-world classification problems is expensive. In the last years, crowdsourcing has emerged as a popular alternative to label a training set. However, in many real applications, a limited number of expert labels can still be collected to complement the crowdsourced ones. This is precisely the setting in the GravitySpy project, which searches for gravitational waves using millions of crowdsourcing annotations and only a few expert labels. In this work, we extend a state-of-the-art probabilistic crowdsourcing model to allow for the integration of expert labels. The resulting objective (the Evidence Lower Bound) is a natural fusion of crowdsourcing and standard classification ones. Desirable theoretical properties of crowdsourcing, such as scalability and sound estimation of annotator behavior and ground truth, translate straightforwardly into this setting. Moreover, the expert labels play the role of anchor points to disclose the underlying truth and the annotators' behavior. A controlled experiment illustrates the properties and behavior of the method. The application to GravitySpy shows that this fusion also benefits a challenging real-world astrophysics problem.

1 Introduction

Crowdsourcing, also known as citizen science, has become a popular approach to label real-world data sets [1, 2]. Instead of relying on a single expert, it shares the labeling effort among a large number of annotators with different degrees of expertise. Many crowdsourcing algorithms have been developed to extract knowledge from such a heterogeneous scenario [3, 4, 5].

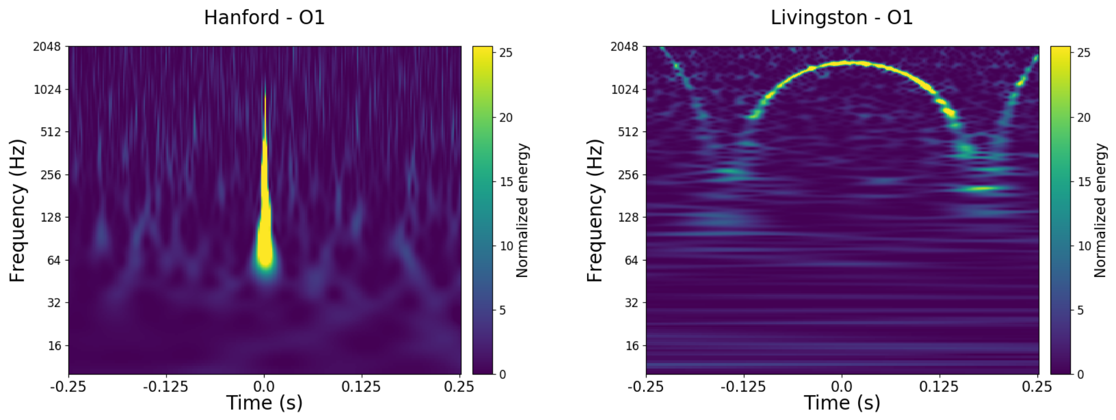


Figure 1: Two examples of glitches observed by the LIGO detector. More details about the problem are provided in Section 4.2.

However, both paradigms (crowdsourcing vs expert labels) should not be regarded as mutually exclusive, and some expert labels can still be collected to complement the crowdsourcing annotations. In fact, one of the main limitations of crowdsourcing methods, their *identifiability*, can be alleviated by adding some expert labels. As already stressed in the founding work [6], a majority of unreliable annotators would make crowdsourcing methods learn an incorrect concept, and in practice it is assumed that most annotators are reliable. Intuitively, the addition of some expert labels is expected to guide crowdsourcing algorithms to identify the underlying truth.

Interestingly, this is the setting available within the GravitySpy project [7]. GravitySpy aims at classifying glitches produced in the laureate Laser Interferometer Gravitational-Waves Observatory (LIGO), see Fig. 1. Whereas the labeling process has been crowdsourced, astrophysicists have also labeled a (smaller) training set. The best results with the expert labels were obtained by Convolutional Neural Networks [8]. Then, crowdsourcing methods leveraged the larger crowdsourcing set to establish a new state-of-the-art [5]. We will show here that a probabilistic fusion of both settings outperforms the results achieved separately.

The proposed model, which is named *SVGPCR-Mix*, extends the probabilistic crowdsourcing method *SVGPCR* in [5] to integrate expert labels. It is based on (sparse) Gaussian Processes (GPs) [9], which were shown to outperform deep learning crowdsourcing approaches in the GravitySpy data, see [5]. The expertise of annotators to label the different classes is modeled through confusion matrices, which are estimated, along with the rest of model parameters, following a variational inference scheme [10]. The derived variational objective (the Evidence Lower Bound, *ELBO*) is a natural fusion of those in [9] (i.e. if sparse GPs were applied on the expert labels only) and [5] (i.e. if the crowdsourcing labels were used only).

Synthetic and real data are used to analyse the proposed approach in depth. First, a controlled experiment illustrates the behavior of the method as the amount of expert labels increase. Then, the probabilistic fusion of crowdsourcing and expert labels is shown to establish a new state-of-the-art in the challenging real-world astrophysics application of GravitySpy. Different properties of the model will be demonstrated along the experiments. Many of them generalize straightforwardly from *SVGPCR*, such as the scalability to large datasets, the estimation of annotators confusion matrices (i.e. their degree of expertise), and the estimation of the ground truth for the crowdsourced samples. Some others are specific to *SVGPCR-Mix*, such as the role of anchor points for the expert labels, the relevance of the coupling term in the ELBO, and the role of the initialization.

The proposed model and inference are included in sections 2 and 3, respectively. Sections 4.1 and 4.2 contain the controlled experiment and the application to LIGO data, respectively. Section 5 concludes the paper.

2 Probabilistic Modeling

Let $\mathbf{X} \in \mathbb{R}^{N\times D}$ be an (observed) training set of N D-dimensional samples. The (mostly non-observed) true labels are denoted as $\mathbf{Z} \in \{0,1\}^{N\times K}$, where each K-class label is expressed with one-hot encoding. Two types of information are available: a few expert (i.e. true) labels and crowdsourcing ones. For the expert labels, let $\mathcal{O} \subseteq \{1,\dots,N\}$ be the samples having expert label (and let $\mathcal{U} = \{1,\dots,N\}\setminus\mathcal{O}$ refer to the rest). We split $\mathbf{Z}$ into $\mathbf{Z}_{\mathcal{O}}$ (the observed true labels) and $\mathbf{Z}_{\mathcal{U}}$ (the non-observed, most of them). For the crowdsourcing ones, let A be the number of annotators, $\mathcal{A}_n \subseteq \{1,\dots,A\}$ the subset of annotators who labeled the n-th sample, and $\mathbf{Y}_n^a$ the set of labels provided by the a-th annotator for that sample[1]. All crowdsourcing labels (for all samples and annotators) are jointly denoted as $\mathbf{Y}$. Notice that $\mathbf{Y}_n^a$ may be available for $n \in \mathbf{Z}_{\mathcal{O}}$ and $n \in \mathbf{Z}_{\mathcal{U}}$.

Each annotator, a, is modeled using a confusion matrix $\mathbf{R}^a = (r_{ij}^a)_{1\le i,j\le K}$, where r_{ij}^a is the probability that annotator a provides the label i for a sample whose real class is j. This is mathematically expressed as $\mathrm{p}(\mathbf{y}|\mathbf{z},\mathbf{R}^a) = \mathbf{y}^\intercal\mathbf{R}^a\mathbf{z}$. Assuming that annotators label samples independently, we have

$$\mathrm{p}(\mathbf{Y}|\mathbf{Z},\mathbf{R}) = \prod_{n=1}^{N}\prod_{a\in\mathcal{A}_n}\prod_{\mathbf{y}\in\mathbf{Y}_n^a}\mathrm{p}(\mathbf{y}|\mathbf{z}_n,\mathbf{R}^a). \tag{1}$$

Prior knowledge on annotators is modeled with a (conjugate) Dirichlet distribution:

$$\mathrm{p}(\mathbf{R}) = \prod_{a=1}^{A}\prod_{k=1}^{K}\mathrm{p}(\mathbf{r}_k^a) = \prod_{a=1}^{A}\prod_{k=1}^{K}\mathrm{Dir}(\mathbf{r}_k^a|\alpha_{1k}^a,\dots,\alpha_{Kk}^a), \tag{2}$$

where $\mathbf{r}_k^a = (r_{1k}^a,\dots,r_{Kk}^a)^\intercal$ is the k-th column of $\mathbf{R}^a$ (this column will also be denoted by $\mathbf{r}_{\mathbf{e}_k}^a$, where $\mathbf{e}_k$ is the k-th K-dimensional one-hot encoding vector), and $\boldsymbol{\alpha} = \{\alpha_{ij}^a : i,j = 1,\dots,K, a = 1,\dots,A\}$ are hyperparameters. If there is no prior information on annotator a, we set $\alpha_{ij}^a = 1, i,j = 1,\dots,K$, which produces a uniform prior.

To connect true labels $\mathbf{Z}$ with observed features $\mathbf{X}$, we resort to GPs [11]. We introduce latent variables $\mathbf{F} = [\mathbf{f}_1,\dots,\mathbf{f}_K] \in \mathbb{R}^{N\times K}$, for which the following prior is considered:

$$\mathrm{p}(\mathbf{F}|\mathbf{X},\boldsymbol{\Omega}) = \prod_{k=1}^{K}\mathrm{p}(\mathbf{f}_k|\mathbf{X},\boldsymbol{\omega}_k) = \prod_{k=1}^{K}\mathcal{N}(\mathbf{f}_k|\mathbf{0},\mathbf{K}_{\boldsymbol{\omega}_k}(\mathbf{X})). \tag{3}$$

A standard RBF kernel is used for the GP, whose parameters $\boldsymbol{\Omega}$ are estimated during training (see next section).

[1]Although $\mathbf{Y}_n^a$ typically contains only one label, it is straightforward to model the case when an annotator provides more than one label for the same sample, which happens in the GravitySpy data.

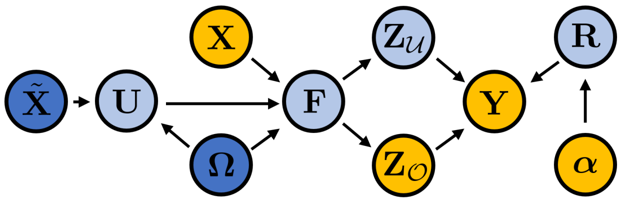


Figure 2: Probabilistic graphical model for *SVGPCR-Mix*. Yellow nodes are observed, light blue nodes are estimated through a posterior distribution, and dark blue nodes are inferred with point estimates.

The relationship between true labels $\mathbf{Z}$ and latent variables $\mathbf{F}$ is modeled through the Robust-Max likelihood [12]. We have

$$\mathrm{p}(\mathbf{Z}|\mathbf{F}) = \prod_{n=1}^{N} \mathrm{p}(\mathbf{z}_n|\mathbf{f}_{n,:}) = \mathrm{p}(\mathbf{Z}_{\mathcal{O}}|\mathbf{F})\mathrm{p}(\mathbf{Z}_{\mathcal{U}}|\mathbf{F}). \tag{4}$$

Here we explicitly split $\mathbf{Z}$ both terms are conceptually different: whereas $\mathbf{Z}_{\mathcal{O}}$ is observed (along with $\mathbf{Y}$), $\mathbf{Z}_{\mathcal{U}}$ is unknown and will be estimated during inference (next section).

One of the main limitations of GPs is their scalability [13]. Their training cost is $\mathcal{O}(N^3)$, which hampers their application beyond a few thousand samples (typically 10K). Since the GravitySpy set is much larger, we sparsify our GP based on standard inducing points approaches, e.g. [9]. Namely, latent variables $\mathbf{F}$ are extended with $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K] \in \mathbb{R}^{M\times K}$ with $M << N$. These variables are called inducing points, and represent the values of the GP at different M inducing locations $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_M]^T \in \mathbb{R}^{M\times D}$.

To sum up, the full model (with inducing points) is

$$\mathrm{p}(\mathbf{Y}, \mathbf{Z}_{\mathcal{O}}, \mathbf{Z}_{\mathcal{U}}, \mathbf{F}, \mathbf{U}, \mathbf{R}|\boldsymbol{\Omega}) = \mathrm{p}(\mathbf{Y}|\mathbf{Z}, \mathbf{R})\mathrm{p}(\mathbf{Z}_{\mathcal{O}}|\mathbf{F})\mathrm{p}(\mathbf{Z}_{\mathcal{U}}|\mathbf{F})\mathrm{p}(\mathbf{F}|\mathbf{U}, \boldsymbol{\Omega})\mathrm{p}(\mathbf{U}|\boldsymbol{\Omega})\mathrm{p}(\mathbf{R}). \tag{5}$$

The probabilistic graphical model is shown in Fig. 2.

3 Variational Inference

The proposed probabilistic model, eq. (5), is not mathematically tractable. We resort to the variational approach [14] to cast inference as an optimization problem. Specifically, the following posterior is proposed:

$$\mathrm{q}(\mathbf{Z}_{\mathcal{U}}, \mathbf{F}, \mathbf{U}, \mathbf{R}) = \mathrm{q}(\mathbf{Z}_{\mathcal{U}})\mathrm{q}(\mathbf{F}|\mathbf{U}, \boldsymbol{\Omega})\mathrm{q}(\mathbf{U})\mathrm{q}(\mathbf{R}), \tag{6}$$

with

$$\begin{aligned}
&\mathrm{q}(\mathbf{Z}_{\mathcal{U}}) = \prod_{n\in\mathcal{U}} \mathrm{q}(\mathbf{z}_n) = \prod_{n\in\mathcal{U}} \mathbf{z}_n^{\intercal}\mathbf{q}_n,\\
&\mathrm{q}(\mathbf{F}|\mathbf{U}, \boldsymbol{\Omega}) = \mathrm{p}(\mathbf{F}|\mathbf{U}, \boldsymbol{\Omega}),\\
&\mathrm{q}(\mathbf{U}) = \prod_{k=1}^{K} \mathrm{q}(\mathbf{u}_k) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{u}_k|\mathbf{m}_k, \mathbf{S}_k),\\
&\mathrm{q}(\mathbf{R}) = \prod_{a=1}^{A}\prod_{k=1}^{K} \mathrm{q}(\mathbf{r}_k^a) = \prod_{a=1}^{A}\prod_{k=1}^{K} \mathrm{Dir}(\mathbf{r}_k^a|\tilde{\alpha}_{1k}^a, \ldots, \tilde{\alpha}_{Kk}^a).
\end{aligned}$$

The variational parameters of this posterior, which will be jointly denoted as $\mathbf{\Theta}$, are: 1) the ground truth estimation for $\mathbf{Z}_{\mathcal{U}}$, i.e. $\mathbf{q}_n = (q_{n1}, \dots, q_{nK}), q_{nk} \geq 0, \sum_k q_{nk} = 1, n \in \mathcal{U}$; 2) the means and covariances in the inducing points, i.e. $\{\mathbf{m}_k, \mathbf{S}_k : k = 1, \dots, K\}$; 3) the posterior Dirichlet parameters, i.e. $\{\tilde{\alpha}^a_{ij} > 0, \ i,j = 1, \dots, K, \ a = 1, \dots, A\}$. Notice that the posterior of $\mathbf{F}|\mathbf{U}$ is equal to the prior, which is a common assumption to achieve tractability when using variational inference with GPs [15].

With this posterior, the Evidence Lower Bound (ELBO) is:

$$\begin{aligned}
&\mathrm{ELBO}(\mathbf{\Omega}, \tilde{\mathbf{X}}, \mathbf{\Theta}) = \sum_{n\in\mathcal{U}} \sum_{a\in\mathcal{A}_n} \sum_{\mathbf{y}\in\mathbf{Y}^a_n} \sum_{k=1}^{K} q_{nk} \mathbb{E}_{\mathrm{q}(\mathbf{r}^a_k)} \log \mathrm{p}(\mathbf{y}|\mathbf{e}_k, \mathbf{r}^a_k) + \\
&\sum_{n\in\mathcal{U}} \sum_{k=1}^{K} q_{nk} \mathbb{E}_{\mathrm{q}(\mathbf{f}_{n,:})} \log \mathrm{p}(\mathbf{e}_k|\mathbf{f}_{n,:}) - \sum_{a=1}^{A} \sum_{k=1}^{K} \mathrm{KL}(\mathrm{q}(\mathbf{r}^a_k) \| \mathrm{p}(\mathbf{r}^a_k)) - \\
&\sum_{n\in\mathcal{U}} \sum_{k=1}^{K} q_{nk} \log q_{nk} - \sum_{k=1}^{K} \mathrm{KL}(\mathrm{q}(\mathbf{u}_k) \| \mathrm{p}(\mathbf{u}_k)) + \qquad (7) \\
&\sum_{n\in\mathcal{O}} \mathbb{E}_{\mathrm{q}(\mathbf{f}_{n,:})} \log \mathrm{p}(\mathbf{z}_n|\mathbf{f}_{n,:}) + \sum_{n\in\mathcal{O}} \sum_{a\in\mathcal{A}_n} \sum_{\mathbf{y}\in\mathbf{Y}^a_n} \mathbb{E}_{\mathrm{q}(\mathbf{r}^a_{\mathbf{z}_n})} \log \mathrm{p}(\mathbf{y}|\mathbf{z}_n, \mathbf{r}^a_{\mathbf{z}_n}).
\end{aligned}$$

This objective function is maximized w.r.t. the variational parameters $\mathbf{\Theta}$ and the model parameters that are inferred through point estimate, i.e. $\tilde{\mathbf{X}}$, $\mathbf{\Omega}$. As optimizer we use Adam with default settings [16].

Interestingly, the first five terms of the ELBO are those obtained in *SVGPCR* [5], i.e. when only crowdsourcing labels are available. The fifth and sixth terms coincide with the ELBO of *SVGP* [9], i.e. if a sparse GP was used only on the true labels. The seventh term is new, and couples both parts in the presence of samples that have both expert and crowdsourcing labels. It contributes to learn the behavior of annotators by comparing both types of labels, and its role will be analyzed in the experiments.

As in the case of *SVGPCR* and *SVGP*, the ELBO in eq. (7) allows for training in mini-batches (the seventh term also factorizes across data points). The computational cost is the same as in *SVGPCR*, i.e. $\mathcal{O}(N_b(M^2 + A_b K))$, where N_b is the number of samples in the minibatch and A_b is the average number of annotations per instance (in the minibatch).

Let us discuss the initialization of the model. In crowdsourcing methods, the ground truth $\mathrm{q}(\mathbf{Z})$ is typically initialized relying on the majority opinion from the crowd (this will be referred to as MO initialization). This implicitly assumes a majority of reliable annotators, which circumvents the well-known identifiability issues of crowdsourcing [6, section 3]. However we will see in the experiments that, if this hypothesis is not satisfied in practice, crowdsourcing methods fail catastrophically. In our setting, an alternative initialization is provided by the true labels (which is referred to as TL). Namely, these can be used to estimate the initial ground truth (e.g. by training a standard *SVGP*). Therefore, this initialization does not rely on the quality of annotators, and will be used by default in the experiments (unless otherwise stated).

Once the ELBO is maximized, the estimated values for $\mathbf{\Omega}$, $\tilde{\mathbf{X}}$ and $\mathbf{\Theta}$ are plugged in eq. (6) to fully determine the approximating posterior $\mathrm{q}(\mathbf{Z}_{\mathcal{U}}, \mathbf{F}, \mathbf{U}, \mathbf{R})$. This distribution summarizes all the information extracted from the observed data $\{\mathbf{Y}, \mathbf{Z}_{\mathcal{O}}, \mathbf{X}\}$. Specifically, the ground truth for the samples without true labels is contained in $\mathrm{q}(\mathbf{Z}_{\mathcal{U}})$. Annotators behavior is encoded in $\mathrm{q}(\mathbf{R})$. Finally, $\mathrm{q}(\mathbf{U})$ allows for predicting on new samples $\mathbf{x}^*$ by conditioning on the inducing points [9, 5].

	Sensitivities					Specificities				
Annotators	1	2	3	4	5	1	2	3	4	5
good	0.9	0.7	0.8	0.9	0.1	0.6	0.8	0.5	0.8	0.2
adversarial	0.1	0.3	0.2	0.1	0.9	0.4	0.2	0.5	0.2	0.8
spammer	0.5	0.57	0.48	0.49	0.9	0.45	0.51	0.5	0.51	0.8

Table 1: Sensitivity and specificity values in the three different scenarios: majority of *good*, *adversarial* or *spammer* annotators.

4 Experimental Results

Here we first include a controlled experiment to illustrate the behavior and properties of the proposed method. Then, we show it also benefits a challenging real-world astrophysics problem: glitch classification in signals acquired by the Laser Interferometer Gravitational-wave Observatory (LIGO).

4.1 Controlled experiment

A two-class synthetic data set is considered on $(-\pi, \pi)$. For each $x \in (-\pi, \pi)$, its class is given by the sign of $\cos(3x)$, i.e., $x \in \mathcal{C}_1$ if $\cos(3x) > 0$ and $x \in \mathcal{C}_0$ otherwise. The ground truth can be seen in Fig. 4) (GT curve). Notice that both classes are not linearly separable.

Crowdsourcing annotations are simulated on 100 randomly distributed samples on $(-\pi, \pi)$. Specifically, to analyze identifiability issues, three different scenarios are considered with majority of *good*, *adversarial*, or *spammers* annotators. In each scenario, five annotators are simulated. Annotators are modeled by their sensitivity and specificity (i.e. the entries r_{11} and r_{00} of their confusion matrix, respectively)[2]. The used values are reported in table 1. Notice that each annotator labels the 100 samples.

First, we illustrate the performance of *SVGPCR-Mix* as the amount of expert labels grows from 2% (so that there is at least one sample of each class) to 100%[3]. It is compared to three closely related methods. The first two separately rely on the two sources of available information: *SVGPCR* (if the crowdsourcing labels were available only), and *GPSubset* (if a GP was applied on the true labels only). The third, *GPFull*, represents the ideal case when expert labels are available for all the training points and a GP is trained on them (this must be understood as a golden reference). Results are shown in Fig. 3, for the three different scenarios, and averaged over 10 independent runs (a test set of size 1000 is used).

Several aspects are highlighted. First, *SVGPCR-Mix* performance improves with the amount of expert labels, approaching the golden reference *GPFull*. Moreover, the curves saturate quickly (earlier than 20%), which supports the idea that just a few expert labels are needed to complement the crowdsourcing ones. Second, notice that *SVGPCR* exhibits difficulties when annotators become less reliable (due to the identifiability issues of crowdsourcing methods). Interestingly, *SVGPCR-Mix* requires just a small percentage of expert labels to fix this. Third, as theoretically

[2]*Good* annotators' values of sensitivity and specificity are high, *adversarial*'s are low (i.e. they learned the wrong concept), and *spammers*' are around 0.5 (i.e. they provide a random label).

[3]Every two expert labels, one is obtained for a sample that also has crowdsourcing labels and the other for a new sample. The relevance of this is analyzed in Fig. 6.

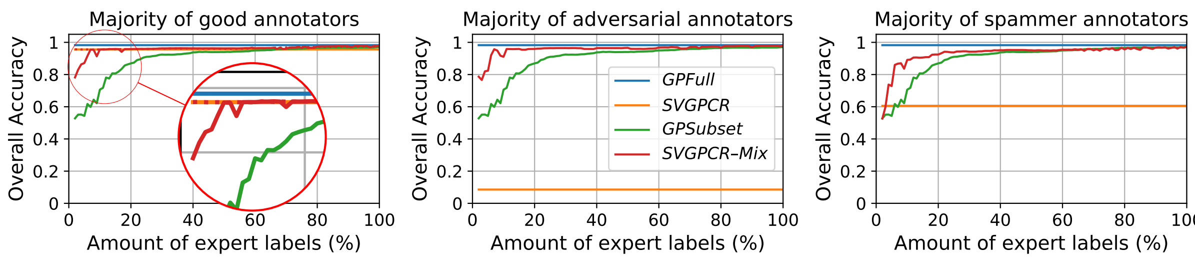


Figure 3: Performance of *SVGPCR-Mix* and related methods as the amount of expert labels increases in the three different scenarios considered. The proposed fusion of expert labels improves the results, and this is more significant as the annotators are less reliable. The leftmost plot includes a dotted red line which corresponds to *SVGPCR-Mix* with MO initialization (more details in the text).

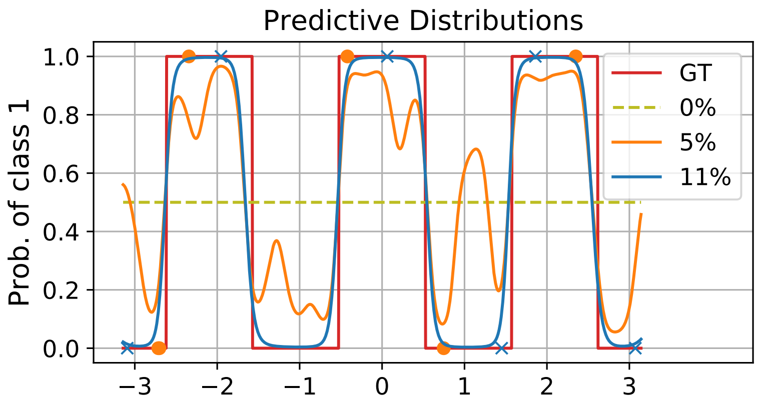


Figure 4: Predictive distribution of *SVGPCR-Mix* as the amount of expert labels increases. These play the role of anchor points to unravel the ground truth.

expected, *SVGPCR-Mix* stays above *SVGPCR* and *GPSubset* (which use only one of the sources of information)[4].

The second experiment analyzes how the expert labels improve the performance of *SVGPCR-Mix*. Fig. 4 shows the predictive distribution of the model as the amount of expert labels grows (a majority of *spammers* scenario is considered). If crowdsourcing labels are used only (0% curve), the information is so noisy that kernel parameters converge to zero and the predictive distribution is constant (recall we are in the majority-of-*spammers* scenario). The first significant change happens when 5% of expert labels are added (these labels are depicted as orange dots). The accuracy (threshold=0.5) is close to the one provided by the GT model, but the actual posterior probability values are not. The second change occurs with 11% (blue crosses). Now, the predictive distribution approximates the ground truth very accurately. Interestingly, notice that no true labels were added in the connected component containing -1; however, the model learned the connection between true and crowdsourcing labels in other regions, and used it to its benefit also here.

The third experiment studies how *SVGPCR-Mix* exploits the expert labels to learn the annotators behavior. Fig. 5 shows the sensitivity and specificity estimations as the amount of expert labels increases (in the majority of *spammers* scenario). For annotators 1-4, the estimations keep close to

[4]The only exception is at the beginning of the *good* annotators case, where *SVGPCR* outperforms *SVGPCR-Mix*. This is a consequence of the TL initialization being used with very few expert labels (notice that the MO initialization bypasses the issue), and is not representative for larger real sets.

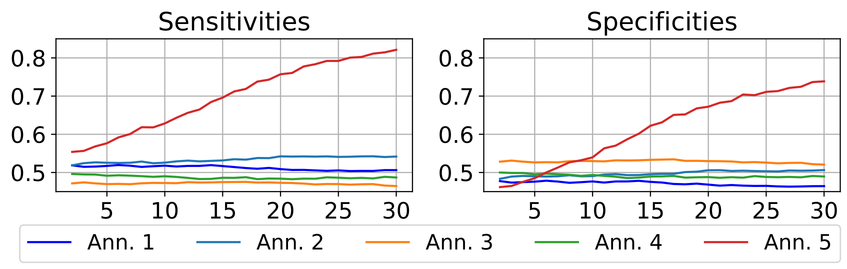


Figure 5: sensitivities and specificities estimated by the proposed model in the majority of *spammers* scenario. *SVGPCR-Mix* leverages expert labels to learn the behavior of annotators.

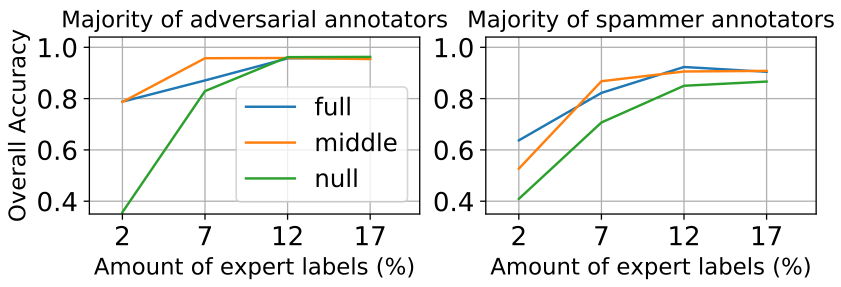


Figure 6: Performance of *SVGPCR-Mix* as expert labels are added following three different schemes (more details on the text). The best results are obtained when the ELBO seventh term is considered.

0.5 (recall from table 1 that all of them are *spammers*). For annotator 5, who has high sensitivity and specificity, the estimation evolves. In the beginning, *SVGPCR-Mix* cannot distinguish this annotator from the *spammers*. However, as the percentage of expert labels increases, it is able to make a better estimation.

The last experiment studies the influence of the ELBO seventh term (the new one introduced by this model). Such term appears if there are samples with both expert and crowdsourcing labels. Therefore, Fig. 6 shows the performance of *SVGPCR-Mix* as expert labels are added following three different schemes: *full* (they are added on samples that also have crowdsourcing labels), *null* (they are added on new samples), and *middle* (every two, one is of the *full*-type and the other of the *null*-type). The results are significantly better for the *full* and *middle* cases, that is, when the seventh term of the ELBO is being actually used.

4.2 Glitch detection in LIGO

In this section, we evaluate the proposed model on the real problem that motivated its development: glitch detection in signals acquired by LIGO.

LIGO is a large-scale physics experiment to detect gravitational waves (GWs) [17]. GWs are ripples in the space-time produced by massive astronomical events (such as binary black holes or neutron stars mergers). Although their existence is a theoretical consequence of General Relativity, their first direct observation was made on 2015 by LIGO. The discovery had a tremendous impact in the scientific community, and was awarded the 2017 Physics Nobel.

To identify GWs, LIGO deploys cutting-edge technology that is sensitive to different sources of noise. This contamination appears as *glitches* in the spectrograms that astrophysicists analyze to search for GWs (see Fig. 1). The goal of the GravitySpy project[5] is to classify different types of glitches. Since LIGO produces a constant stream of data, GravitySpy leverages the Zooniverse platform[6] to obtain crowdsourcing labels. Moreover, to complement these, some expert labels have been provided by astrophysicists.

Namely, our training set contains 173565 samples (glitches) and 1828981 crowdsourcing annotations (i.e., a mean value of more than 10 labels per sample), which have been provided by 3443 collaborators through Zooniverse. For each glitch, we use 256 relevant features extracted in [8]. The glitches have been classified into 15 different classes proposed by astrophysicists (they all are shown in [5, Figure 3]). Moreover, there are 7901 samples with expert labels (2593 of them also have crowdsourcing annotations; this ensures that the seventh term of the ELBO is used, recall Fig. 6). GravitySpy test set is made up of 9997 samples.

Two methods have addressed this problem so far. The first one, which will be referred to as *DL*, uses the expert labels to train a Convolutional Neural Network [8]. The second one is *SVGPCR* [5], which uses a GP-based crowdsourcing model to train with all the crowdsourcing labels. Recall that each of these method leverages one type of labels, whereas the proposed *SVGPCR-Mix* is able to train with both. Since *SVGPCR-Mix* is a generalization of both *SVGPCR* and *SVGP* [9] (recall section 3), we also include the later in the comparison for completeness.

Table 2 shows the overall accuracy (OA) and test likelihood (TL) for the four compared methods across the different classes. Whereas the former considers just the predictive mode, the latter also takes into account the quality of the uncertainties. *SVGPCR-Mix* consistently obtains the best results in both metrics, which justifies the proposed fusion of expert and crowdsourcing labels. Notice also that the samples with expert labels are only 5% of the samples with crowdsourcing labels. This supports the idea illustrated in the synthetic experiment that just a few expert labels are enough to complement the crowdsourcing ones.

Let us analyze several aspects more in-depth. The good performance of *SVGP* and *DL* (which only use 7901 samples) is due to 1) the quality of the expert labels and 2) (for *DL*) the representation power of Convolutional Neural Nets for the spectrograms (images). Notice also that the results of *SVGPCR* are not far from those of *SVGPCR-Mix*. This implies that there exists a majority of reliable annotators (otherwise, the identifiability issues would severely harm the performance of *SVGPCR*, recall Fig. 3). This is interesting for astrophysicists, since it validates the training system designed for the volunteers. We further verify this by empirically estimating the overall accuracy of annotators. We do it based on the 2593 samples that have both expert and crowdsourcing labels. Indeed, results in Fig. 7 show an estimated OA greater than 0.9 for almost all the annotators. Finally, we also stress the scalability of *SVGPCR-Mix*, which is able to cope with 173565 training samples and 1828981 crowdsourcing labels (far beyond the standard GPs limit).

Finally, let us illustrate the ability of *SVGPCR-Mix* to estimate the annotators confusion matrices. We consider annotator #80, which has annotated 927 samples that also have expert labels. This allows us to empirically calculate its confusion matrix through a frequentist analysis of its annotations. Additionally, we only consider the classes for which the selected annotator provided more than 100 annotations. The confusion matrices estimated by *SVGPCR-Mix* (and also by *SVGPCR*) are shown at the top of Fig. 8. Both matrices are very similar, and have values close to the empirical

[5]https://ciera.northwestern.edu/programs/gravityspy/
[6]https://www.zooniverse.org/projects/zooniverse/gravity-spy

	Overall Accuracies				Test likelihood			
Classes	*DL*	*SVGP*	*SVGPCR*	*SVGPCR-Mix*	*DL*	*SVGP*	*SVGPCR*	*SVGPCR-Mix*
1080LINE	.9759 (.0025)	.9697 (.0064)	.9720 (.0069)	**.9883 (.0023)**	.9727 (.0023)	.9209 (.0088)	.9688 (.0075)	**.9853 (.0021)**
1400RIPPLE	.7569 (.0106)	.6642 (.0416)	**.8577 (.0171)**	.7967 (.0267)	.7541 (.0064)	.5884 (.0363)	**.8509 (.0156)**	.7975 (.0275)
BLIP	.9603 (.0018)	.9592 (.0032)	.9622 (.0052)	**.9715 (.0028)**	.9587 (.0012)	.9481 (.0057)	.9587 (.0055)	**.9685 (.0024)**
EXTR.LOUD	.8136 (.0185)	**.8784 (.0283)**	.7295 (.0427)	.8273 (.0422)	.7993 (.0156)	.7835 (.0225)	.7242 (.0408)	**.8146 (.0435)**
KOIFISH	.7797 (.0132)	.7992 (.0125)	**.8828 (.0115)**	.8522 (.0127)	.7711 (.0112)	.7788 (.0130)	**.8784 (.0117)**	.8484 (.0138)
L.F.BURST	**.8996 (.0052)**	.8904 (.0115)	.8861 (.0105)	.8983 (.0057)	**.8988 (.0056)**	.8787 (.0128)	.8838 (.0098)	.8959 (.0053)
L.F.LINE	.8490 (.0152)	.8693 (.0304)	**.9156 (.0111)**	.8785 (.0132)	.8403 (.0144)	.8304 (.0326)	**.9118 (.0103)**	.8752 (.0135)
NOGLITCH	.9290 (.0025)	.9400 (.0068)	.7951 (.0162)	**.9506 (.0071)**	.9272 (.0019)	.8791 (.0139)	.7932 (.0146)	**.9461 (.0055)**
OTHER	.4859 (.0141)	**.4954 (.0212)**	.4011 (.0091)	.3870 (.0132)	**.4800 (.0119)**	.4571 (.0137)	.3999 (.0091)	.3854 (.0155)
P.L.60HZ	.7983 (.0165)	.9264 (.0076)	.8425 (.0127)	**.9396 (.0037)**	.7937 (.0181)	.8720 (.0135)	.8380 (.0107)	**.9374 (.0055)**
REP.BLIPS	.5197 (.0137)	.5581 (.0509)	**.6700 (.0210)**	.6641 (.0289)	.5227 (.0087)	.5094 (.0432)	**.6651 (.0198)**	.6493 (.0223)
SCATT.LIGHT	.9585 (.0016)	.9580 (.0071)	.9562 (.0056)	**.9667 (.0024)**	.9581 (.0011)	.9302 (.0107)	.9520 (.0057)	**.9640 (.0024)**
SCRATCHY	**.9220 (.0060)**	.9013 (.0148)	.9000 (.0165)	.8847 (.0166)	**.9194 (.0055)**	.8419 (.0107)	.8953 (.0176)	.8819 (.0204)
VIOLIN	.9769 (.0013)	.9700 (.0032)	**.9914 (.0017)**	.9758 (.0016)	.9764 (.0006)	.9574 (.0045)	**.9886 (.0014)**	.9738 (.0011)
WHISTLE	.9535 (.0069)	**.9649 (.0000)**	.9201 (.0047)	.9535 (.0111)	**.9528 (.0019)**	.9370 (.0037)	.9179 (.0046)	.9483 (.0060)
GLOBAL	.9113 (.0020)	.9145 (.0043)	.9183 (.0027)	**.9258 (.0019)**	.9081 (.0018)	.8813 (.0045)	.9149 (.0027)	**.9227 (.0073)**

Table 2: Performance (accuracy and test likelihood) for the four compared methods in the LIGO problem. *DL* is the Convolutional Neural Network introduced in [8], *SVGP* refers to the Scalable Variational GP introduced in [9], and *SVGPCR* denotes the recent crowdsourcing method in [5]. The proposed *SVGPCR-Mix* obtains the best global results in terms of OA and TL.

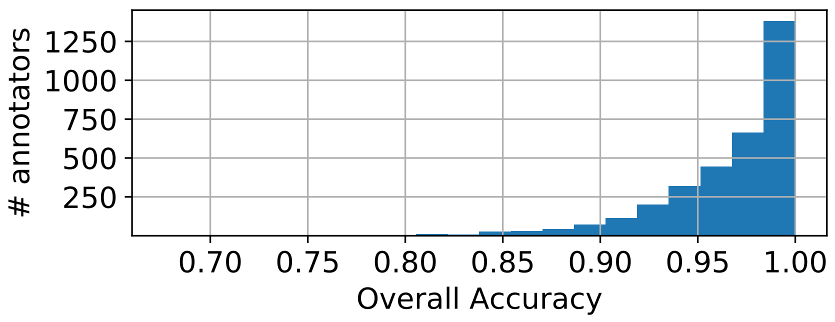


Figure 7: Histogram of the annotators according to their Overall Accuracy evaluated on $\mathbf{Z}_{\mathcal{O}}$. In this problem there exists a majority of good annotators.

estimation. Notice also that the annotator is a reliable one (matrices do not look diagonal because only classes with at least 100 annotations are shown). Recall from section 3 that annotator confusion matrices are estimated through a posterior Dirichlet distribution $\mathrm{q}(\mathbf{R})$. The value reported here is the expectation of this distribution.

In the second row of Fig. 8, we compare the confusion vectors for classes 3 (BLIP) and 12 (SCATTERED LIGHT). By confusion vector we refer to a column of the confusion matrix, i.e. the probabilities assigned by the annotator for a certain class. Here we chose these two classes for being those where *SVGPCR-Mix* and *Empirical* confusion vectors are most similar and different (square error sense), respectively. However, in both cases we observe that *SVGPCR* and *SVGPCR-Mix* almost match the empirical value, which confirms the accuracy of their estimations. In addition to the discussed case of annotator $\#80$, the global examination of *SVGPCR* and *SVGPCR-Mix* confusion matrices yields very similar results. This makes us conclude that the improvement in performance is due to the proposed fusion of expert labels, which obtains a better underlying classifier.

5 Conclusions

In this work we have proposed a new probabilistic model for detecting glitches in signals acquired by LIGO. The data set collected by GravitySpy project motivated the development, by combining the quality of labels provided by experts with the ability of the crowds to label huge data sets. The

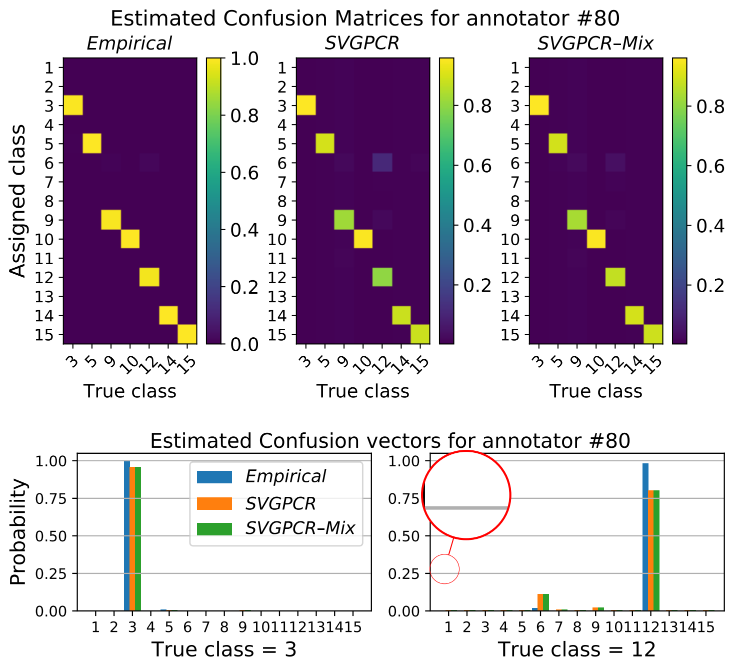


Figure 8: First row, from left to right: for certain annotator, *Empirical* confusion matrix and those estimated by *SVGPCR* and *SVGPCR-Mix*, respectively. Second row: for the same annotator, detail of the assigned classes for two different true classes (BLIP and SCATTERED LIGHT). These are commonly referred to as confusion vectors. The estimations by both *SVGPCR* and *SVGPCR-Mix* are very similar to the empirical values.

proposed method is a natural generalization of *SVGPCR* and *SVGP*. We have demonstrated that the use of true labels makes our method robust in scenarios where the majority of annotators are not reliable, whereas previous crowdsourcing methods in the literature catastrophically fail in that case. Furthermore, we have seen that only a small percentage of samples with true labels suffices for *SVGPCR-Mix* to recognize the behavior of annotators and extract all the useful knowledge from the crowdsourcing data. Then, we have applied *SVGPCR-Mix* to the GravitySpy data, outperforming all the previous results in the literature. Finally, we have illustrated the differences between *SVGPCR* and *SVGPCR-Mix* when estimating the confusion matrices of GravitySpy annotators.

References

[1] A. Irwin, “No phds needed: how citizen science is transforming research,” *Nature*, vol. 562, pp. 480–482, 2018.

[2] C. Guerrini, M. Majumder, M. Lewellyn, and A. McGuire, “Citizen science, public policy,” *Science*, vol. 361, no. 6398, pp. 134–136, 2018.

[3] S. Albarqouni, C. Baur, F. Achilles, V. Belagiannis, S. Demirci, and N. Navab, “AggNet: Deep Learning From Crowds for Mitosis Detection in Breast Cancer Histology Images,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1313–1321, 2016.

[4] F. Rodrigues and F. Pereira, “Deep learning from crowds,” in *Conference on Artificial Intelligence (AAAI)*, 2018, pp. 1611–1618.

[5] P. Morales-Álvarez, P. Ruiz, S. Coughlin, R. Molina, and A. Katsaggelos, "Scalable Variational Gaussian Processes for Crowdsourcing:Glitch Detection in LIGO," *arXiv preprint arXiv:1911.01915*, 2019.

[6] A. Dawid and A. Skene, "Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm," *Journal of the Real Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 20–28, 1979.

[7] M. Zevin, S. Coughlin, S. Bahaadini, N. Besler, E.and Rohani, S. Allen *et al.*, "Gravity Spy: integrating advanced LIGO detector characterization, Machine Learning, and citizen science," *Classical and Quantum Gravity*, vol. 34, no. 6, p. 064003, 2017.

[8] S. Bahaadini, V. Noroozi, N. Rohani, S. Coughlin, M. Zevin, J. Smith, V. Kalogera, and A. Katsaggelos, "Machine learning for gravity spy: Glitch classification and dataset," *Information Sciences*, vol. 444, pp. 172–186, 2018.

[9] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable Variational Gaussian Process Classification," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 38, 2015, pp. 351–360.

[10] P. Ruiz, P. Morales-Álvarez, R. Molina, and A. Katsaggelos, "Learning from crowds with variational gaussian processes," *Pattern Recognition*, vol. 88, pp. 298 – 311, 2019.

[11] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT, 2006.

[12] D. Hernández-Lobato, J. Hernández-Lobato, and P. Dupont, "Robust multi-class gaussian process classification," in *Advances in Neural Information Processing Systems (NIPS)*, 2011, pp. 280–288.

[13] P. Morales-Álvarez, P. Ruiz, R. Santos-Rodríguez, R. Molina, and A. Katsaggelos, "Scalable and efficient learning from crowds with gaussian processes," *Information Fusion*, vol. 52, pp. 110 – 127, 2019.

[14] D. Blei, A. Kucukelbir, and J. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[15] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 5, 2009, pp. 567–574.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference for Learning Representations (ICLR)*, 2015.

[17] A. Abramovici, W. Althouse, R. P. Drever, Y. Gürsel, S. Kawamura, F. Raab, D. Shoemaker, L. Sievers, R. Spero, K. Thorne, R. Vogt, R. Weiss, S. Whitcomb, and M. Zucker, "Ligo: The laser interferometer gravitational-wave observatory," *Science*, vol. 256, no. 5055, pp. 325–333, 1992.

CHAPTER 8

Activation-level uncertainty in deep neural networks

8.1 Publication details

Authors: Pablo Morales-Álvarez, Daniel Hernández-Lobato, Rafael Molina, José Miguel Hernández-Lobato
Title: Activation-level uncertainty in deep neural networks
Reference: Under review at Advances in Neural Information Processing Systems (NeurIPS) 2020
Status: Under review in the regular review process (13% of submissions were rejected on a previous phase).
Quality indices:

- GGS Rating: A++
- GGS Class: 1
- Core: A++

8.2 Main contributions

- In this work we introduce a new approach to model uncertainty in Deep Neural Networks. It is based on deterministic weights and simple stochastic non-linearities (as opposed to the more classical formulation based on stochastic weights).
- We propose the use of non-parametric one-dimensional GPs as the prior for the activation functions, including the triangular kernel inspired by the Rectified Linear Unit (ReLu).
- Our algorithm addresses well-known limitations of Bayesian Neural Networks (BNN) and functional BNNs. These are the uncertainty underestimation for in-between data (i.e. data that lies in-between two cluster of training points), and the extrapolation to out-of-distribution data. Moreover, it is competitive or superior in standard prediction benchmarks.

- An interesting connection with deep GPs is also established: our approach requires fewer inducing points and is better suited for deep architectures, achieving superior performance.

Activation-level Uncertainty in Deep Neural Networks

Pablo Morales-Álvarez
Dept. of Computer Science and AI
University of Granada, Spain

Daniel Hernández-Lobato
Dept. of Computer Science
Autonomous University of Madrid, Spain

Rafael Molina
Dept. of Computer Science and AI
University of Granada, Spain

José Miguel Hernández-Lobato
Dept. of Engineering
University of Cambridge, UK

Abstract

Current approaches for uncertainty estimation in deep learning often produce too confident results. Bayesian Neural Networks (BNNs) model uncertainty in the space of weights, which is usually high-dimensional and limits the quality of variational approximations. The more recent functional BNNs (fBNNs) address this only partially because, although the prior is specified in the space of functions, the posterior approximation is still defined in terms of stochastic weights. In this work we propose to move uncertainty from the weights (which are deterministic) to the activation function. Specifically, the activations are modelled with simple 1D Gaussian Processes (GP), for which a triangular kernel inspired by the ReLu non-linearity is explored. Our experiments show that activation-level stochasticity provides more reliable uncertainty estimates than BNN and fBNN, whereas it performs competitively or favorably in standard prediction tasks. We also study the connection with deep GPs, both theoretically and empirically. More precisely, we show that activation-level uncertainty requires fewer inducing points and is better suited for deep architectures.

1 Introduction

Deep Neural Networks (DNNs) have achieved state-of-the-art performance in many different tasks, such as speech recognition [Hinton et al., 2012], natural language processing [Mikolov et al., 2013] or computer vision [Krizhevsky et al., 2012]. In spite of their predictive power, DNNs are limited in terms of calibration and uncertainty estimation [Wenzel et al., 2020, Lakshminarayanan et al., 2017, Guo et al., 2017]. This ability to "know what is not known" is essential for critical applications such as medical diagnosis [Esteva et al., 2017, Filos et al., 2019, Mobiny et al., 2019] or autonomous driving [Kendall and Gal, 2017, Gal, 2016]. This has led to a growing interest in uncertainty estimation for deep learning [Blundell et al., 2015, Gal and Ghahramani, 2016, Sun et al., 2019, Foong et al., 2019b].

Bayesian Neural Networks (BNNs) address this problem through a Bayesian treatment in the network weights[1] [MacKay, 1992, Neal, 1995]. This will be also refered to as weight-space stochasticity. However, dealing with uncertainty in weight space is challenging, since it contains many symmetries and is highly dimensional [Wenzel et al., 2020, Sun et al., 2019, Snoek et al., 2019, Fort et al., 2019]. Here we focus on two specific limitations. First, it has been recently shown that BNNs with well-established inference methods such as Bayes by Backprop (BBP) [Blundell et al., 2015] and MC-Dropout [Gal and Ghahramani, 2016] underestimate the predictive uncertainty for instances located in-between two clusters of training points [Foong et al., 2019a, Foong et al., 2019b]. Second, BNNs do not extrapolate sensibly to out-of-distribution (OOD) data [Sun et al., 2019, Nguyen et al., 2015, Ren et al., 2019]. Both issues will be analyzed in Fig. 3.

As an alternative, Functional Bayesian Neural Nets (fBNN) specify the prior in the space of functions [Sun et al., 2019]. This provides a mechanism to guide the extrapolation in OOD data, e.g. predictions can be encouraged to reverse to the prior in regions of no observed data. However, the posterior stochastic process is still defined by a factorized Gaussian on the network weights (i.e. as in BBP), see [Sun et al., 2019, Sec. 3.1]. This makes fBNN inherit the underestimation of predictive uncertainty for in-between data.

In this work, we adopt a different approach by moving stochasticity from the weights to the activation function, see Fig. 1. This will be referred to as auNN (activation-level uncertainty for Neural Networks). The activation functions are modelled with (one-dimensional) GP priors, for which a triangular kernel inspired by the ReLu non-linearity [Nair and Hinton, 2010, Glorot et al., 2011] is used. Since non-linearities are typically simple functions (e.g. ReLu, sigmoid, tanh), our GPs are sparsified with few inducing points. The network weights are deterministic parameters which are estimated to maximize the likelihood of the observed data. The motivation behind auNN is to avoid inference in the complex space of weights. We hypothesise that it could be enough to introduce stochasticity in the activation functions that follow the linear projections. Importantly, we show that auNN obtains well-calibrated estimations for in-between data, and its prior allows for sensible extrapolation to OOD data by reversing to the empirical mean. This will be visualized in a simple 1D example (see Fig. 3 and Tab. 1). Moreover, auNN obtains competitive or superior predictive performance in standard benchmarks, is scalable (datasets of up to ten millions training points are used), and can be readily used for classification.

The use of GPs for the activations establishes an interesting connection with deep GPs (DGPs) [Damianou and Lawrence, 2013, Salimbeni and Deisenroth, 2017]. The main difference is the linear projection before the GP, recall Fig. 1(c-d). This allows auNN units to model simpler mappings between layers, which are defined along one direction of the input space, similarly to neural networks. However, DGP units model more complex mappings defined on the whole input space, see also Fig. 2a. We will show that auNN units require fewer inducing points and are better suited for deep architectures, achieving superior performance.

In summary, the main contributions of this paper are: (1) a new approach to model uncertainty in DNNs, based on deterministic weights and simple stochastic non-linearities (in principle, not necessarily modelled by GPs); (2) the specific use of non-parametric GPs as a prior, including the triangular kernel inspired by the ReLu; (3) auNN addresses well-known limitations of BNNs and

[1]The bias term will be absorbed within the weights throughout the work.

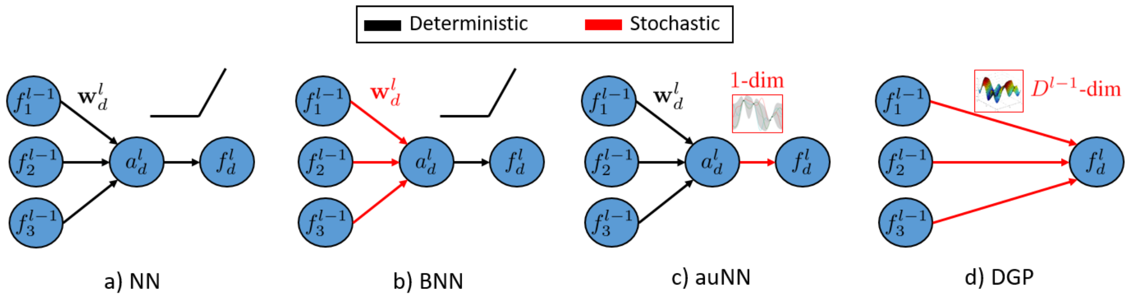


Figure 1: Graphical representation of the artificial neurons for closely related methods. **(a)** In standard Neural Networks (NN), both the weights and the activation function are deterministic. **(b)** In Bayesian NNs, weights are stochastic and the activation is deterministic. **(c)** In auNN (this work), weights are deterministic and the activation is stochastic. **(d)** Deep GPs do not have a linear projection through weights, and the output is modelled directly with a GP defined on the D^{l-1}-dimensional input space.

fBNNs (uncertainty underestimation for in-between data and extrapolation to OOD data), and is competitive or superior in standard prediction tasks; (4) auNN units require fewer inducing points and are better suited for deep architectures than DGP ones, achieving superior performance.

2 Probabilistic Model and Inference

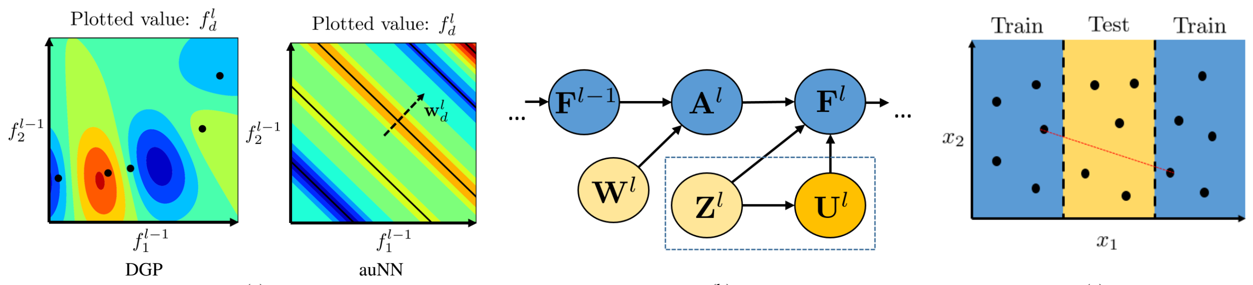
We focus on a supervised task (e.g. regression or classification) with training data[2] $\{\mathbf{x}_{n,:}, \mathbf{y}_{n,:}\}_{n=1}^N$. The graphical model in Fig. 2b will be useful throughout this section. We assume a model of L layers, each one with D^l units as in Fig. 1c. Each activation is modelled with a (1D) GP prior, i.e. $f_d^l(a_d^l) \sim \mathcal{GP}(\mu_d^l, k_d^l)$, with $\mu_d^l : \mathbb{R} \to \mathbb{R}$ and $k_d^l : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ the mean and covariance functions, respectively. The GP hyperparameters $\boldsymbol{\theta}_d^l$ will be omitted for clarity (for the kernels considered here, $\boldsymbol{\theta}_d^l$ includes the amplitude and the lengthscale). Assuming independence between units, each layer depends on the previous one as:

$$\mathrm{p}(\mathbf{F}^l|\mathbf{F}^{l-1}, \mathbf{W}^l) = \mathrm{p}(\mathbf{F}^l|\mathbf{A}^l) = \prod_{d=1}^{D^l} \mathrm{p}(\mathbf{f}_d^l|\mathbf{a}_d^l), \tag{1}$$

where $\mathbf{F}^l$ is the $N \times D^l$ matrix of outputs of the l-th layer for N inputs, $\mathbf{W}^l$ is the $D^{l-1} \times D^l$ matrix of weights in that layer, and $\mathbf{A}^l$ is the $N \times D^l$ matrix of pre-activations, i.e. $\mathbf{A}^l = \mathbf{F}^{l-1} \cdot \mathbf{W}^l$. As usual, the columns and rows of $\mathbf{F}^l$ are denoted as $\mathbf{f}_d^l$ and $\mathbf{f}_{n,:}^l$, respectively (and analogously for the other matrices). Since it is defined by a GP, we have $p(\mathbf{f}_d^l|\mathbf{a}_d^l) = \mathcal{N}(\mathbf{f}_d^l|\boldsymbol{\mu}_d^l, \mathbf{K}_d^l)$, with $\boldsymbol{\mu}_d^l$ (resp. $\mathbf{K}_d^l$) the result of evaluating μ_d^l (resp. k_d^l) on $\mathbf{a}_d^l$. To fully specify the model, the output $\mathbf{Y}$ is defined from the last layer with a likelihood that factorizes across data points, i.e. $\mathrm{p}(\mathbf{Y}|\mathbf{F}^L) = \prod_{n=1}^N \mathrm{p}(\mathbf{y}_{n,:}|\mathbf{f}_{n,:}^L)$. This formulation resembles that of DGPs [Damianou and Lawrence, 2013, Salimbeni and Deisenroth, 2017]. The main difference is that we model $\mathbf{F}^l|\mathbf{F}^{l-1}$ through D^l 1D GPs evaluated on the pre-activations $\mathbf{A}^l$ (i.e. the projections of $\mathbf{F}^{l-1}$ through $\mathbf{W}^l$), whereas DGPs use D^l GPs of dimension D^{l-1} evaluated directly on $\mathbf{F}^{l-1}$, recall Fig. 1(c-d).

Inference in the proposed model is intractable. To address this, we follow standard sparse variational GP approaches [Titsias, 2009, Hensman et al., 2013, Hensman et al., 2015], similarly to the Doubly Stochastic Variational Inference (DSVI) for DGPs [Salimbeni and Deisenroth, 2017]. Specifically, in each unit of each layer we introduce M^l inducing values $\mathbf{u}_d^l$, which are the result of evaluating the

[2]The output is represented as a vector since all the derivations apply for the multi-output case.



(a) (b) (c)

Figure 2: (a) Type of mappings modelled by DGP and auNN units (colours represent different values). Whereas DGP units describe complex functions defined on the whole D^{l-1} dimensional input space, the linear projection through $\mathbf{w}_d^l$ in auNN yields simpler functions defined on just one direction. This is closer in spirit to NNs, requires fewer inducing points, and is better suited for deep architectures. The inducing points are shown in black (for auNN, these correspond to (hyper)planes in the input space before the projection). **(b)** Probabilistic graphical model for an auNN layer. Yellow variables are to be estimated (light ones through point estimates and the dark one through a posterior distribution). The box highlights the auxiliary variables (inducing points and their values). **(c)** Graphical representation of the UCI in-between splits. In red, a segment that crosses the gap joining two training points from different components, which will be used in the experiments.

GP on the one-dimensional inducing points $\mathbf{z}_d^l$. We naturally write $\mathbf{U}^l$ and $\mathbf{Z}^l$ for the corresponding $M^l \times D^l$ matrices. Following eq. (1), the augmented model for one layer is

$$\mathrm{p}(\mathbf{F}^l, \mathbf{U}^l | \mathbf{F}^{l-1}, \mathbf{W}^l, \mathbf{Z}^l) = \mathrm{p}(\mathbf{F}^l | \mathbf{U}^l, \mathbf{A}^l, \mathbf{Z}^l)\mathrm{p}(\mathbf{U}^l | \mathbf{Z}^l) = \prod_{d=1}^{D^l} \mathrm{p}(\mathbf{f}_d^l | \mathbf{u}_d^l, \mathbf{a}_d^l, \mathbf{z}_d^l)\mathrm{p}(\mathbf{u}_d^l | \mathbf{z}_d^l). \tag{2}$$

Variational inference (VI) involves the approximation of the true posterior $\mathrm{p}(\{\mathbf{F}^l, \mathbf{U}^l\}_l | \mathbf{Y})$. Following [Hensman et al., 2013, Salimbeni and Deisenroth, 2017], we propose a posterior given by $\mathrm{p}(\mathbf{F}|\mathbf{U})$ and a parametric Gaussian on $\mathbf{U}$:

$$\mathrm{q}(\{\mathbf{F}^l, \mathbf{U}^l\}_l) = \prod_{l=1}^{L} \mathrm{p}(\mathbf{F}^l | \mathbf{U}^l, \mathbf{A}^l, \mathbf{Z}^l)\mathrm{q}(\mathbf{U}^l) = \prod_{l=1}^{L}\prod_{d=1}^{D^l} \mathrm{p}(\mathbf{f}_d^l | \mathbf{u}_d^l, \mathbf{a}_d^l, \mathbf{z}_d^l)\mathrm{q}(\mathbf{u}_d^l), \tag{3}$$

where $\mathrm{q}(\mathbf{u}_d^l) = \mathcal{N}(\mathbf{u}_d^l | \mathbf{m}_d^l, \mathbf{S}_d^l)$, with $\mathbf{m}_d^l \in \mathbb{R}^{M^l}$ and $\mathbf{S}_d^l \in \mathbb{R}^{M^l \times M^l}$ variational parameters to be estimated. Minimizing the KL divergence between $\mathrm{q}(\{\mathbf{F}^l, \mathbf{U}^l\}_l)$ and the true posterior is equivalent to maximizing the following evidence lower bound (ELBO):

$$\log \mathrm{p}(\mathbf{Y} | \{\mathbf{W}^l, \mathbf{Z}^l\}_l) \geq \mathrm{ELBO} = \sum_{n=1}^{N} \mathbb{E}_{\mathrm{q}(\mathbf{f}_{n,:}^L)}\left[\log \mathrm{p}(\mathbf{y}_{n,:} | \mathbf{f}_{n,:}^L)\right] - \sum_{l=1}^{L}\sum_{d=1}^{D^l} \mathrm{KL}\left(\mathrm{q}(\mathbf{u}_d^l) || \mathrm{p}(\mathbf{u}_d^l)\right). \tag{4}$$

In the ELBO, the KL term can be computed in closed-form, as both $\mathrm{q}(\mathbf{u}_d^l)$ and $\mathrm{p}(\mathbf{u}_d^l)$ are Gaussians. The log likelihood term can be approximated by sampling from the marginal posterior $\mathrm{q}(\mathbf{f}_{n,:}^L)$, which can be done efficiently through univariate Gaussians as in [Salimbeni and Deisenroth, 2017]. Specifically, $\mathbf{U}^l$ can be analytically marginalized in eq. (3), which yields $\mathrm{q}(\{\mathbf{F}^l\}^l) = \prod_l \mathrm{q}(\mathbf{F}^l | \mathbf{F}^{l-1}, \mathbf{W}^l) = \prod_{l,d} \mathcal{N}(\mathbf{f}_d^l | \tilde{\boldsymbol{\mu}}_d^l, \tilde{\boldsymbol{\Sigma}}_d^l)$, with:

$$[\tilde{\boldsymbol{\mu}}_d^l]_i = \mu_d^l(a_{id}^l) + \boldsymbol{\alpha}_d^l(a_{id}^l)^\intercal(\mathbf{m}_d^l - \mu_d^l(\mathbf{z}_d^l)), \tag{5}$$

$$[\tilde{\boldsymbol{\Sigma}}_d^l]_{ij} = k_d^l(a_{id}^l, a_{jd}^l) - \boldsymbol{\alpha}_d^l(a_{id}^l)^\intercal(k_d^l(\mathbf{z}_d^l) - \mathbf{S}_d^l)\boldsymbol{\alpha}_d^l(a_{jd}^l), \tag{6}$$

where $\boldsymbol{\alpha}_d^l(x) = k_d^l(x, \mathbf{z}_d^l)[k_d^l(\mathbf{z}_d^l)]^{-1}$ and $\mathbf{a}_{n,:}^l = \mathbf{W}^l \mathbf{f}_{n,:}^{l-1}$. Importantly, the marginal posterior $\mathrm{q}(\mathbf{f}_{n,:}^l)$ is a Gaussian that depends only on $\mathbf{a}_{n,:}^l$, which in turn only depends on $\mathrm{q}(\mathbf{f}_{n,:}^{l-1})$. Therefore, sampling from $\mathbf{f}^l_{n,:}$ is straightforward using the reparametrization trick [Kingma and Welling, 2013, Rezende et al., 2014]:

$$f^l_{nd} = [\tilde{\boldsymbol{\mu}}^l_d]_n + \varepsilon \cdot [\tilde{\boldsymbol{\Sigma}}^l_d]^{1/2}_{nn}, \quad \text{with } \varepsilon \sim \mathcal{N}(0,1), \quad \text{and } \mathbf{f}^0_{n,:} = \mathbf{x}_{n,:}. \tag{7}$$

Training consists on maximizing the ELBO, eq. (4), w.r.t. the variational parameters $\{\mathbf{m}^l_d, \mathbf{S}^l_d\}$, the inducing points $\{\mathbf{z}^l_d\}$, and the model parameters (i.e. weights $\{\mathbf{w}^l_d\}$ and kernel parameters $\{\boldsymbol{\theta}^l_d\}$). This can be done through mini-batches, which allows for scalability to very large datasets. The complexity to evaluate the ELBO is $\mathcal{O}(NM^2(D^1 + \cdots + D^L))$, the same as DGPs with DSVI [Salimbeni and Deisenroth, 2017].[3]

Predictions. Given a new $\mathbf{x}_{*,:}$, we want to compute[4] $\mathrm{p}(\mathbf{f}^L_{*,:}|\mathbf{X},\mathbf{Y}) \approx \mathbb{E}_{\mathrm{q}(\{\mathbf{U}^l\})}\left[\mathrm{p}(\mathbf{f}^L_{*,:}|\{\mathbf{U}^l\})\right]$. As in [Salimbeni and Deisenroth, 2017], this can be approximated by sampling S values up to the $(L-1)$-th layer with the same eq. (7), but starting with $\mathbf{x}_{*,:}$. Then, $\mathrm{p}(\mathbf{f}^L_{*,:}|\mathbf{X},\mathbf{Y})$ is given by the mixture of the S Gaussians distributions obtained with eqs. (5)-(6).

Triangular kernel. One of the most popular kernels in GPs is the RBF [Williams and Rasmussen, 2006], which produces very smooth functions. However, the ReLu non-linearity led to a general boost in performance in DNNs [Nair and Hinton, 2010, Glorot et al., 2011], and we aim to model similar activations. Therefore, we introduce the use of the *triangular* (TRI) kernel. Just like RBF, TRI is an isotropic kernel, i.e. it depends on the distance between the inputs, $k(x,y) = \gamma \cdot g(|x-y|/\ell)$, with γ and ℓ the amplitude and lengthscale. For RBF, $g(t) = e^{-t^2/2}$. For TRI, $g(t) = \max(1-t, 0)$. This is a valid kernel, see [Williams and Rasmussen, 2006, Sec. 4.2.1]. Similarly to the ReLu, the functions modelled by TRI are piecewise linear, see Fig. 6a, and Fig. 2 in the Appendix.

Comparison with DGP. The difference between auNN and DGP units is graphically illustrated in Fig. 2a. Whereas DGP mappings from one layer to the next are complex functions defined on D^{l-1} dimensions ($D^{l-1} = 2$ in the figure), auNN mappings are defined just along one direction via the weight projection. This is closer in spirit to NNs, whose mappings are also simpler and better suited for feature extraction and learning more abstract concepts. Moreover, since the GP is defined on a 1D space, auNN requires fewer inducing points than DGP (which, intuitively, can be regarded as inducing (hyper)planes in the D^{l-1}-dimensional space before the projection). The benefits of auNN units will be experimentally assessed in the next section.

3 Experiments

In this section, auNN is compared to BNN, fBNN [Sun et al., 2019] and DSVI DGP [Salimbeni and Deisenroth, 2017]. BNNs are trained with BBP [Blundell et al., 2015], since auNN also leverages a simple VI-based inference approach. In each section we will highlight the most relevant experimental aspects, and all the details can be found at the Appendix. In the sequel, NLL stands for Negative Log Likelihood. Anonymized code for auNN is provided in the supplementary material, along with a script to run it for the 1D illustrative example of Sec. 3.1.

[3]As in [Salimbeni and Deisenroth, 2017], there exists also a cubic term $\mathcal{O}(M^3(D^1 + \cdots + D^L))$ that is assumed to be dominated by the former (since the mini-batch size N is typically larger than M). Moreover, in auNN we have the multiplication by weights, with complexity $\mathcal{O}(ND^{l-1}D^l)$ for each layer. This is also typically dominated by the former.

[4]The distribution $\mathrm{p}(\mathbf{y}^L_{*,:}|\mathbf{X},\mathbf{Y})$ is obtained as the expectation of the likelihood over $\mathrm{p}(\mathbf{f}^L_{*,:}|\mathbf{X},\mathbf{Y})$. A Gaussian likelihood will be used for regression, whereas the Robust-Max [Hernández-Lobato et al., 2011] will be considered for classification.

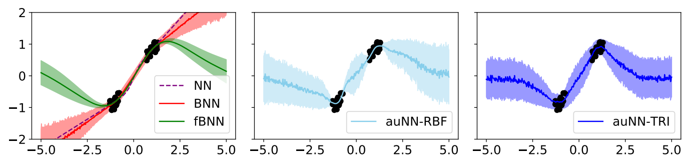


Figure 3: Predictive distribution (mean and one standard deviation) after training on a 1D dataset with two clusters of points. This simple example illustrates the main limitations of NN, BNN and fBNN, which are overcome by the novel auNN. See Tab. 1 for a summary and the text for details.

Table 1: Visual overview of conclusions from the 1D experiment in Fig. 3. This shows that NN, BNN, fBNN and the novel auNN increasingly expand their capabilities.

	Epistemic uncertainty	Reverses to mean	In-between uncertainty	Reference
NN	✗	✗	✗	-
BNN	✓	✗	✗	[Blundell et al., 2015]
fBNN	✓	✓	✗	[Sun et al., 2019]
auNN	✓	✓	✓	This work

3.1 An illustrative example

Here we illustrate the two issues that were highlighted in the introduction: the extrapolation to OOD data and the underestimation of predictive uncertainty for instances located in-between two clusters of training points. Fig. 3 shows the predictive distribution of NN, BNN, fBNN and auNN (with RBF and TRI kernels) after training on a simple 1D dataset with two clusters of points. All the methods have one hidden layer with 25 units, and 5 inducing points are used for auNN.

In Fig. 3, the deterministic nature of NNs prevents them from providing epistemic uncertainty (i.e. the one originating from the model [Kendall and Gal, 2017]). Moreover, there is no prior to guide the extrapolation to OOD data. For example, in absence of additional information, it could be desirable that the predictions reverse to the empirical mean of the observed data as $x \to \pm\infty$ (however, the predictions of NN diverge). BNNs provide epistemic uncertainty. However, the prior in the complex space of weights is not expressive enough to produce sensible extrapolations. Moreover, note that BNNs underestimate the predictive uncertainty in the region between the two clusters, where there is no observed data (this region is usually called the *gap*). More specifically, as shown in [Foong et al., 2019b], the predictive uncertainty for data points in the gap is limited by that on the extremes. By specifying the prior in function space, fBNN can induce properties in the output, such as reversing to the empirical mean for OOD data through a zero-mean GP prior. However, the underestimation of in-between uncertainty persists, since the posterior stochastic process for fBNN is based on a weight-space factorized Gaussian (as BNN with BBP), recall [Sun et al., 2019, Sec. 3.1]. Finally, auNN (either with RBF or TRI kernel) bypasses both issues through the novel activation-level modelling of uncertainty, which utilizes a zero-mean GP prior for the activations. Tab. 1 summarizes the main characteristics of each method. Next, a more comprehensive experiment with deeper architectures and a wider variety of datasets is provided.

3.2 UCI regression datasets with in-between splits

Standard splits are not appropriate to evaluate the quality of uncertainty estimates for in-between data, since both train and test sets may cover the space equally. This motivated the introduction of in-between splits [Foong et al., 2019a]. Namely, a set with D dimensions admits D such train-test partitions by considering each dimension, sorting the points according to its value, and selecting the middle 1/3 for test (and the outer 2/3 for training), see Fig. 2c. With these partitions, overconfident predictions for data points in the gap manifest as very high values of test negative log likelihood.

We train BNN, fBNN, auNN-RBF and auNN-TRI for $L = 2, 3$ layers (i.e. one and two hidden layers, respectively) on six UCI datasets using in-between splits (namely, Boston, Concrete, Energy,

Table 2: Ranks in terms of NLL and RMSE for the UCI in-between splits. The lower the better. Per-group average rank is provided too (weight vs activation stochasticity). Activation stochasticity achieves superior performance in both metrics. The reported values are the mean and standard error over all the datasets and splits.

		BNN-2	BNN-3	fBNN-2	fBNN-3	auNN-RBF-2	auNN-RBF-3	auNN-TRI-2	auNN-TRI-3
NLL	Rank	3.92±0.79	4.98±0.70	5.04±0.36	5.36±0.50	4.69±0.61	5.29±0.89	**3.25±0.57**	3.47±0.80
	Average	4.83±0.32				**4.17±0.40**			
RMSE	Rank	4.09±0.67	6.15±0.91	4.70±0.54	4.71±0.53	4.32±0.47	4.27±0.55	3.90±0.33	**3.85±0.46**
	Average	4.91±0.37				**4.09±0.23**			

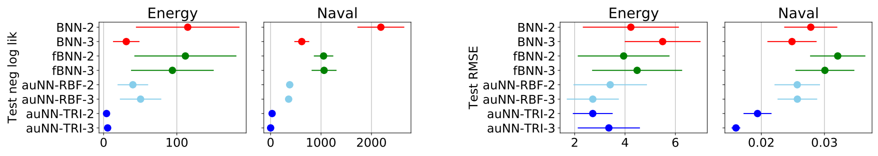


Figure 4: Test NLL and RMSE for the in-between splits in Energy and Naval datasets (mean and one standard error, the lower the better). Activation-level uncertainty, specially through the triangular kernel, avoids the dramatic failure of BNN and fBNN in terms of NLL (see the scale). The similar values in RMSE reveal that this failure actually comes from an extremely overconfident estimation by BNN and fBNN, see also Fig. 5.

Naval, Wine and Yacht). In all cases, $D = 50$ hidden units are used, and auNN uses $M = 5$ inducing points. Two groups of models are distinguished in terms of posterior stochasticity: those based on stochastic weights (BNN and fBNN), and those based on stochastic activations (auNN-RBF and auNN-TRI).

As a summary, Tab. 2 shows the individual and per-group ranks for test NLL and RMSE, which are the average over all the datasets and splits. The full results are provided in Tab. 1 and 2 in the Appendix. It can be observed that activation-level uncertainty obtains superior performance in both metrics. Moreover, it is important to analyze the different behavior in some datasets. Namely, Fig. 4 shows the complete results for Energy and Naval (Fig. 1 in the Appendix provides analogous plots for the rest of datasets). Interestingly, we observe that BNN and fBNN severely underestimate the predictive uncertainty. Namely, whereas very high NLL is obtained (see the scale), the RMSE is closer to that of the best method. Therefore, the problem does not come from the predictive mean, but from an extremely overconfident prediction. Notice that this behavior is improved by auNN, specially with the triangular kernel, which achieves top calibration for in-between data.

To further understand the different results, Fig. 5 shows the predictive distribution over a segment that crosses the gap, recall Fig. 2c. We observe that activation-level approaches obtain more sensitive (less confident) uncertainties in the gap, where there is no observed data. In particular, BNN and fBNN predictions in Naval are unjustifiably overconfident, since the output in that dataset ranges from 0.95 to 1. Finally, to illustrate the internal mechanism of auNN, Fig. 6a shows one example of the activations learned when using each kernel. Although it is just one example, it allows for visualising the different nature: smoother for RBF and piecewise linear for TRI. All the activation functions for a particular network and for both kernels are shown in Fig. 2 in the Appendix.

3.3 UCI regression datasets with standard splits

Here, the previous section is complemented with results for standard (random) train-test splits on the same UCI datasets. This shows that, in addition to the enhanced uncertainty estimation, auNN is a competitive alternative in general practice. As a summary, Tab. 3 shows the individual and

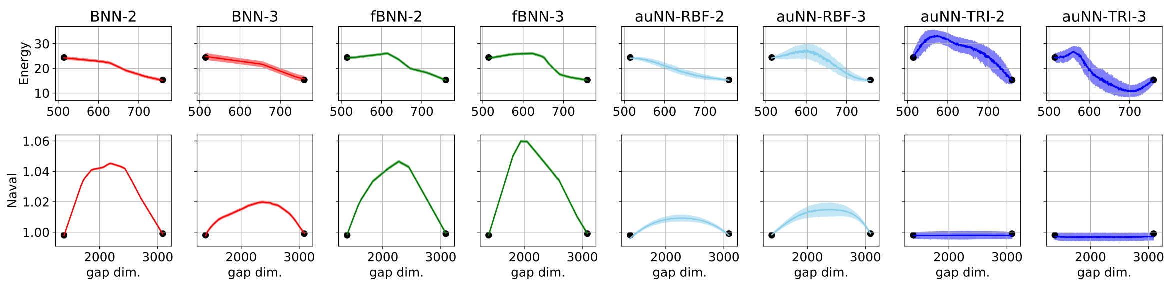


Figure 5: Predictive distribution (mean and one standard deviation) over a segment that crosses the gap, joining two training points from different connected components. The two most challenging datasets in terms of uncertainty estimation for in-between data are shown (Energy, upper row, and Naval, lower row; recall Fig. 4). auNN achieves better calibrated uncertainty in the gap, where there is no observed data.

Table 3: Ranks in terms of NLL and RMSE for the UCI standard splits. The lower the better. Per-group average rank is provided too (weight vs activation stochasticity). Activation stochasticity achieves competitive or slightly superior performance. The reported values are the mean and standard error over all the datasets and splits.

		BNN-2	BNN-3	fBNN-2	fBNN-3	auNN-RBF-2	auNN-RBF-3	auNN-TRI-2	auNN-TRI-3
NLL	Rank	**3.78±0.41**	6.25±0.70	4.13±0.57	3.83±0.85	3.97±0.60	4.42±0.85	4.78±0.92	4.83±1.01
	Average	**4.50±0.39**				**4.50±0.43**			
RMSE	Rank	4.70±0.48	6.50±0.64	3.70±0.61	3.45±0.88	4.25±0.35	**3.35±0.77**	5.40±0.80	4.65±1.00
	Average	4.59±0.41				**4.41±0.41**			

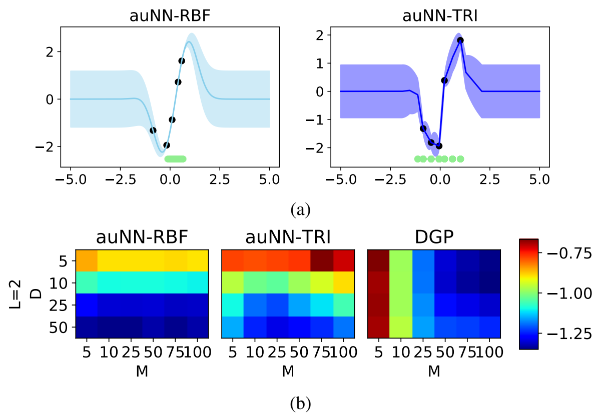
per-group (activation vs weight uncertainty) ranks in terms of NLL and RMSE, when using the same datasets but with 10 random 90%-10% train-test splits. The same experimental setup is used too. We observe that activation-level approaches are competitive or slightly superior in both metrics. The reported values are the mean over all the datasets and splits. Full results are in Tab. 3 and 4 in the Appendix.

3.4 Comparison with DGPs

As explained in Sec. 2, the choice of a GP prior for activation stochasticity establishes a strong connection with DGPs. The main difference is that auNN performs a linear projection from D^{l-1} to D^l dimensions before applying D^l 1D GPs, whereas DGPs define D^l GPs directly on the D^{l-1} dimensional space. This means that auNN units are simpler than those of DGP, recall Fig. 2a. Here we show two practical implications of this.

First, it is reasonable to hypothesise that DGP units may require a higher number of inducing points M than auNN, since they need to cover a multi-dimensional input space. By contrast, auNN may require a higher number of hidden units D, since these are simpler. Importantly, the computational cost is not symmetric in M and D, but significantly cheaper on D, recall Sec. 2. Fig. 6b shows the performance of auNN and DGP for different values of M and D on the UCI Kin8 set (with one hidden layer; depth will be analyzed next). As expected, note the different influence by M and D: whereas auNN improves "horizontally" (i.e. as D grows), DGP does it "vertically" (i.e. as M grows)[5]. In the next section, we will see that this makes auNN faster than DGP in practice. An analogous figure for RMSE and full numeric results are in Fig. 3 and Tabs. 6-7 in the Appendix.

[5]Interestingly, the fact that DGP is not greatly influenced by D could be appreciated in the recommended value in [Salimbeni and Deisenroth, 2017]. They set $D = \min(30, D^0)$, where D^0 is the input dimension.



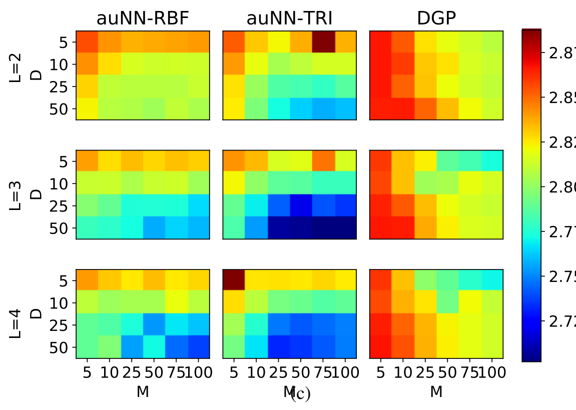


Figure 6: **(a)** One example of activation function (mean and standard deviation) learned by auNN with each kernel. RBF's one is smoother, whereas TRI's is piecewise linear, inspired by ReLu. Black dots represent (the mean of) the inducing point values. Green dots are the locations of input data when propagated to the corresponding unit. **(b)** Test NLL of auNN and DGP for different values of M (number of inducing points) and D (number of hidden units). The lower the better. The results are the average over five independent runs with different splits. Whereas DGP improves vertically (i.e. with M), auNN does it horizontally (i.e. with D). This is as hypothesized, and is convenient from a scalability viewpoint. **(c)** Test NLL with increasing depth ($L = 2, 3, 4$). This supports that auNN might benefit more than DGP from deeper networks. Moreover, the aforementioned different influence of M and D on DGP and auNN is confirmed here.

Second, auNN simpler units might be better suited for deeper architectures. Fig. 6c shows the performance on the UCI Power dataset when depth is additionally considered. It can be observed that auNN is able to take greater advantage of depth, which translates into significantly better overall performance. Moreover, the aforementioned different influence of D and M on DGP and auNN is also confirmed here. The results on RMSE are similar, see Fig. 4 and Tabs. 8-9 in Appendix.

3.5 Large scale experiments

So far, we experimented with small to medium regression datasets. However, as explained in Sec. 2, the computational cost of auNN is similar to DSVI DGP, which is scalable to millions of instances [Salimbeni and Deisenroth, 2017]. Here we experiment with datasets of that size (up to $N = 10^7$), achieving superior performance to DGP. Moreover, we also demonstrate that auNN can be readily used for classification. [6]

Classification. We use the well-known particle physics binary classification sets HIGGS (N = 11M, D = 28) and SUSY (N = 5M, D = 18) [Baldi et al., 2014]. The Robust-Max likelihood is used for classification [Hernández-Lobato et al., 2011]. We compare the performance of auNN and DGP, which obtained state-of-the-art results on these problems [Salimbeni and Deisenroth, 2017]. Tab. 4 shows the results for different depths in terms of AUC (the metric used in both [Salimbeni and Deisenroth, 2017] and the original work [Baldi et al., 2014]). The novel auNN clearly outperforms DGP[7]. Moreover, we observe that auNN results consistently improve as depth increases, which is not always the case for DGP (see DGP-4). Regarding both kernels (RBF and TRI), differences are not significant.

[6]In this section, both DGP and auNN are trained with one hidden layer and their optimal configuration according to the previous experiment: large M for DGP ($M = 100$, D is set as recommended by the authors, i.e. $D = \min(30, D^0)$), and large D for auNN ($D = 50$, M is set to an intermediate value of $M = 25$).

[7]DGP results reported in [Salimbeni and Deisenroth, 2017] are better than here (but still worse than those obtained by auNN). Since we have used the same code, training and initializations, the difference must be due to the different train-test splits.

Table 4: Performance of auNN and DGP in two large scale classification datasets. AUC is shown (the higher the better). The proposed method outperforms DGP for both kernels, and the results steadily improve with depth. The standard deviation (on three splits) is close to zero in all cases, see Tab. 5 in the Appendix.

			auNN						DGP		
	N	D	RBF-2	RBF-3	RBF-4	TRI-2	TRI-3	TRI-4	DGP-2	DGP-3	DGP-4
HIGGS	11M	28	0.8294	0.8494	**0.8551**	0.8288	0.8483	0.8533	0.8159	0.8176	0.8031
SUSY	5M	18	0.8779	0.8785	**0.8787**	0.8776	0.8783	0.8786	0.8237	0.8316	0.8263

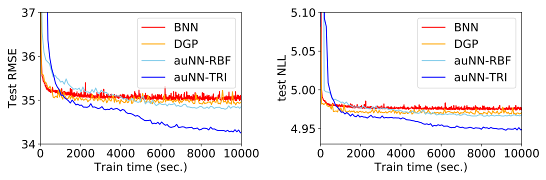


Figure 7: Performance in the extended airline dataset (2M training points) as training time goes on. Although a bit slower in the beginning, the proposed method achieves superior performance, specially with TRI kernel.

Table 5: Elapsed time to predict on airline test set (127068 instances). All the methods use 100 test samples.

Algorithm	Test time
BNN	**0.20±0.01**
DGP	7.35±0.06
auNN-RBF	3.78±0.06
auNN-TRI	3.15±0.06

Regression. Finally, we use an extended version of airline dataset as an example of large-scale regression [Hernández-Lobato and Hernández-Lobato, 2016]. After removing missing values, $N = 2127068$ instances remain. From these, 2M are used for train, and the rest for test. Fig. 7 shows the test performance (NLL and RMSE) as the training time goes on, and Tab. 5 shows the time required to predict on the whole test set. Although it takes a bit longer to converge to their optimal configuration, the proposed method, specially with the novel TRI kernel, achieves significantly better results. Test time results are as theoretically expected: auNN is faster than DGP (because lower values of M are required, recall footnote 6), but both are clearly slower than BNN (which does not involve GPs). Faster inference would be possible in activation-level uncertainty if the GPs were substituted by a different method (e.g. a Bayesian parametric model).

4 Related Work

Activation-level uncertainty is introduced here as an alternative to weight-space stochasticity. The expressiveness of the latter has been recently analyzed in [Wenzel et al., 2020], where the authors advocate a modified BNN objective that has proven successful in the context of MCMC [Zhang et al., 2020, Heek and Kalchbrenner, 2019, Leimkuhler et al., 2019] and VI [Ashukha et al., 2020, Osawa et al., 2019, Zhang et al., 2018]. Likewise, different prior specifications are studied in [Hafner et al., 2019, Pearce et al., 2019, Flam-Shepherd et al., 2017], in addition to the fBNN discussed here [Sun et al., 2019]. However, none of these works consider stochasticity on the activations.

Since we present a straightforward use of VI for auNN, in this work we have compared empirically with the well-known VI-based BBP for BNNs. Yet, we expect auNN to benefit from independent inference refinements like those proposed over the last years for BNNs. For instance, natural-gradient VI allows for leveraging techniques such as BatchNorm or data augmentation [Osawa et al., 2019], the information contained in the SGD trajectory can be used [Maddox et al., 2019], and MCMC approaches can benefit from a cyclical time stepping as well as Langevin dynamics defined by the corresponding SDE [Zhang et al., 2020, Särkkä and Solin, 2019].

A key aspect of auNN is the modelling of the activation function. This element of neural nets has been analyzed before. For instance, self-normalizing neural nets [Klambauer et al., 2017] induce the normalization that is explicitly performed in related approaches such as BatchNorm [Ioffe and Szegedy, 2015], and weight and layer normalization [Salimans and Kingma, 2016, Ba et al., 2016]. Learnable deterministic activations have been explored too, e.g. [He et al., 2015, Agostinelli et al., 2014]. However, as opposed to auNN, in all these cases the activations are deterministic.

A very preliminary study on GP-based activation functions is proposed in [Urban and van der Smagt, 2018]. However, the method is not empirically evaluated, no connection with deep GPs is provided, and the inference approach is limited. Namely, the output of each unit is approximated with a Gaussian whose mean and covariance are computed in closed-form, as was done in [Bui et al., 2016] for DGPs. However, this is only tractable for the RBF kernel (in particular, it cannot leverage the more convenient TRI kernel studied here), and the Gaussian approximation typically yields worse results than Monte Carlo approximations to the ELBO as used here (indeed, DSVI [Salimbeni and Deisenroth, 2017] substantially improved the results for DGPs compared to [Bui et al., 2016]).

5 Conclusions and Future Work

We proposed a novel approach for uncertainty estimation in neural network architectures. Whereas previous methods are mostly based on a Bayesian treatment of the weights, here we move the stochasticity to the activation functions, which are modelled with a simple 1D GP and a triangular kernel inspired by the ReLu. Our experiments show that the proposed method obtains better calibrated uncertainty estimates, and is competitive or superior in standard prediction tasks. Moreover, the connection with deep GPs is analyzed. Namely, our approach requires fewer inducing points and is better suited for deep architectures, achieving superior performance.

We hope this work raises interest on alternative approaches to model uncertainty in neural networks. For instance, activation-level uncertainty introduces the prior in the space of features or representations. In addition to the extrapolation to OOD data, it would be interesting to study how this prior influences other properties of the output functions. Also, the GP-based activation model could be substituted by a simpler Bayesian parametric one. Finally, since only the activation function is modified, important deep learning elements such as convolutional layers can be still incorporated.

A Statement of Broader Impact

Uncertainty estimation in deep learning is an important challenge that is typically motivated by critical applications of AI, such as autonomous driving and medical diagnosis. If better uncertainty estimates were available, expert advice could be required for challenging situations in the medical case. Likewise, additional data could be collected for autonomous driving scenarios in which the uncertainty is high. Although the positive impact for society can be enormous, technology must be analyzed from a critical viewpoint in order to disclose potentially negative outcomes.

For instance, we have studied that activation-level uncertainty allows for specifying properties of the output function (such us the extrapolation to OOD data). However, activation-level uncertainty is far from being fully interpretable. Indeed, although meaningful, the space of features (or representations) is difficult to visualize and understand in a systematic way. We believe that interpretability is a very important property for the technology to be deployed in real-world systems in a fair, transparent and safe way. Therefore, it would be extremely interesting to further study the global effects of uncertainty in the space of features or representations.

B Supplementary Material

B.1 Practical specifications for auNN

Whitening transformation for $\mathrm{q}(\mathbf{u}_d^l)$. The proposed parametric posterior for each unit is given by the Gaussian $\mathrm{q}(\mathbf{u}_d^l) = \mathcal{N}(\mathbf{u}_d^l|\mathbf{m}_d^l, \mathbf{S}_d^l)$. The GP prior on $\mathbf{u}_d^l$ is $\mathrm{p}(\mathbf{u}_d^l) = \mathcal{N}(\mathbf{u}_d^l|\boldsymbol{\mu}_d^l, \mathbf{K}_d^l)$, with $\boldsymbol{\mu}_d^l = \mu_d^l(\mathbf{z}_d^l)$ and $\mathbf{K}_d^l = k_d^l(\mathbf{z}_d^l, \mathbf{z}_d^l)$. For numerical stability and to reduce the amount of operations, we use a white representation for $\mathrm{q}(\mathbf{u}_d^l)$, as is common practice in (D)GPs [De G. Matthews et al., 2017, Salimbeni and Deisenroth, 2017]. That is, we consider the variable $\mathbf{v}_d^l \sim \mathcal{N}(\tilde{\mathbf{m}}_d^l, \tilde{\mathbf{S}}_d^l)$, with $\mathbf{u}_d^l = \boldsymbol{\mu}_d^l + (\mathbf{K}_d^l)^{1/2}\mathbf{v}_d^l$. Specifically, in the code the variable $\tilde{\mathbf{m}}_d^l$ is denoted as `q_mu`, and $\tilde{\mathbf{S}}_d^l$ is represented through its Cholesky factorization $(\tilde{\mathbf{S}}_d^l)^{1/2}$, which is named `q_sqrt`.

Initialization of the variational parameters $\{\mathbf{m}_d^l\}$. These are the mean of the posterior distribution on the inducing points. Therefore, their value determines the *initialization of the activation function*. If the RBF kernel is used, $\{\mathbf{m}_d^l\}$ are initialized to the prior $\boldsymbol{\mu}_d^l = \mu_d^l(\mathbf{z}_d^l)$ (since we are using the aforementioned white representation, `q_mu` is initialized to zero). This is the most standard initialization in GP literature. For the TRI kernel, $\{\mathbf{m}_d^l\}$ are initialized according to the ReLu which TRI is inspired by, i.e. $\mathbf{m}_d^l = \text{ReLu}(\mathbf{z}_d^l)$.

Initialization of the variational parameters $\{\mathbf{S}_d^l\}$. The posterior distribution covariance matrices are initialized to the prior $\mathbf{K}_d^l = k_d^l(\mathbf{z}_d^l, \mathbf{z}_d^l)$ (that is, `q_sqrt` is initialized to the identity matrix). Following common practise for DGPs [Salimbeni and Deisenroth, 2017], the covariance matrices of inner layers are scaled by 10^{-5}.

Initialization of the weights. The Glorot uniform initializer [Glorot and Bengio, 2010], also called Xavier uniform initializer, is used for the weights. The biases are initialized to zero.

Initialization of the kernel hyperparameters. The kernels used (RBF and TRI) have two hyperparameters: the variance γ and the lengthscale ℓ. Both are always initialized to 1 (except for the lengthscale in the 1D example, where ℓ is initialized to 0.1).

Initialization of the inducing points. In order to initialize $\mathbf{z}_d^l$, the N input data points are propagated through the network with the aforementioned initial weights, biases, and activation function. Then, in each layer and unit, $\mathbf{z}_d^l$ is initialized with a `linspace` between the minimum and maximum of the N values there (the minimum (resp. the maximum) is decreased (resp. increased) by 0.1 to strictly contain the interval of interest).

Initialization of the regression likelihood noise. In the regression problems, we use a Gaussian likelihood $\mathrm{p}(y|f) = \mathcal{N}(y|f, \sigma^2)$. The standard deviation of the noise is initialized to $\sigma = 0.1$.

Mean function. We always use a zero mean function. Since data is normalized to have zero mean (and standard deviation equal to one), a zero mean function allows for sensible extrapolation to OOD data, as explained in the paper.

Optimizer and learning rate. Throughout the work, we use the Adam Optimizer [Kingma and Ba, 2014] with default parameters and learning rate of 0.001.

B.2 Experimental details for the experiments

All the experiments were run on a NVIDIA Tesla P100. In order to predict, all the methods utilize 100 test samples in all the experiments. Details for each section are provided below.

An illustrative example (Sec. 3.1 in the paper). All the methods use two layers (i.e. one hidden layer). The hidden layer has $D = 25$ units in all cases. BNN and fBNN use ReLu activations. The auNN methods use $M = 5$ inducing points in each unit (the rest of methods do not have such inducing points). The methods are trained during 5000 epochs with the whole dataset (no mini-batches). The dataset is synthetically generated to have two clusters of points around $x = \pm 1$. More specifically, 30 points are sampled uniformly in each interval $(x - 0.3, x + 0.3)$ for $x = \pm 1$, and the output is given by the $\sin$ function plus a Gaussian noise of standard deviation 0.1.

UCI regression datasets with in-between splits (Sec. 3.2 in the paper). The methods use $L = 2, 3$ layers. In all cases, the hidden layers have $D = 50$ units. BNN and fBNN use ReLu activations. The methods are trained during 10000 epochs, with a mini-batch size that depends on the size of the dataset. For those with fewer than 5000 instances (i.e. Boston, Concrete, Energy, Wine and Yacht), the mini-batch size is 500. For those with more than 5000 (i.e. Naval), the mini-batch size is 5000. Recall from the paper that each dataset has as many in-between splits as dimensionality, with 2/3 for train and 1/3 for test. Regarding the segment used in the last experiment, each extreme of the segment is a point from a different connected component of the training set. These are chosen so that the function is well-known in the extremes (but not along the segment, which crosses the gap). Namely, the extremes are chosen as the training points who have minimum average distance to the closest five points in its connected component.

UCI regression datasets with standard splits (Sec. 3.3 in the paper). The details are exactly the same as in the previous experiment. The only difference is in the splits. Here, each dataset has 10 random 90%-10% train-test splits.

Comparison with DGPs (Sec. 3.4). Here, different values of depth L, number of inducing points M and number of hidden layers D are studied (see the paper). auNN is trained during 5000 epochs, with a mini-batch size of 5000 (20000 epochs are used for DGP, as proposed by the authors [Salimbeni and Deisenroth, 2017]). Each experiment is repeated on five random 90%-10% train-test splits.

Large scale experiments (Sec. 3.5). In the classification datasets, a RobustMax likelihood is used in all cases [Hernández-Lobato et al., 2011]. The values of D and M are chosen following the conclusions from Sec. 3.4. That is, DGP needs large M (the largest $M = 100$ is used), but is less influenced by D (this is chosen as recommended by the authors [Salimbeni and Deisenroth, 2017]: $D = \min(30, D^0)$, with D^0 the dimensionality of the input data). auNN needs large D (the largest $D = 50$ is used), but is less influenced by M (an intermediate value $M = 25$ is chosen). All the methods are trained during 100 epochs, with a mini-batch size of 5000. Three random train-test splits are used. In both datasets, 500000 instances are used for test (which leaves 10.5M and 4.5M training instances for HIGGS and SUSY, respectively).

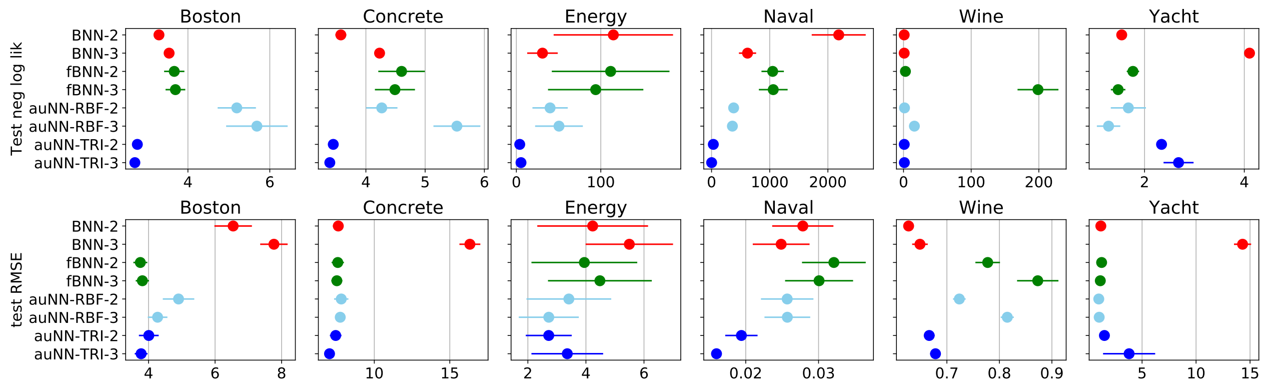


Figure 8: Performance of the compared methods in the UCI in-between splits for the six datasets. Mean and one standard error of NLL (upper row) and RMSE (lower row) are shown, the lower the better.

Table 6: Test NLL for the in-between splits of the UCI datasets (mean and one standard error, the lower the better). Last column is the per-group (weight-space stochasticity vs activation-level stochasticity) average rank.

	Boston	Concrete	Energy	Naval	Wine	Yacht	Rank	Rank (group)
BNN-2	3.29±0.10	3.58±0.09	114.84±70.69	2186.30±464.32	**0.96±0.01**	1.54±0.09	3.92±0.79	4.83±0.32
BNN-3	3.54±0.03	4.23±0.04	30.91±19.97	618.44±147.99	0.98±0.02	4.10±0.03	4.98±0.70	
fBNN-2	3.67±0.25	4.60±0.39	111.65±69.68	1050.65±192.61	2.80±0.31	1.77±0.12	5.04±0.36	
fBNN-3	3.69±0.24	4.49±0.34	93.92±56.45	1060.54±247.21	198.76±30.24	1.47±0.15	5.36±0.50	
auNN-RBF-2	5.19±0.47	4.27±0.26	39.93±20.89	379.55±67.74	1.44±0.05	1.68±0.35	4.69±0.61	**4.17±0.40**
auNN-RBF-3	5.68±0.75	5.54±0.40	50.48±28.26	352.94±72.13	16.05±1.13	**1.28±0.23**	5.29±0.89	
auNN-TRI-2	2.77±0.06	3.45±0.06	**3.99±1.14**	30.47±5.54	1.06±0.03	2.34±0.03	**3.25±0.57**	
auNN-TRI-3	**2.70±0.04**	**3.39±0.06**	5.50±2.45	**2.38±3.23**	1.23±0.04	2.68±0.30	3.47±0.80	

In the regression experiment, the depth for all methods is also $L = 2$, and the values of M and D for auNN and DGP are as in the classification experiment. BNN also uses $D = 50$ hidden units, and ReLu activations. All the methods are trained during 10000 seconds with a mini-batch size of 500. One single train-test split is considered.

B.3 Additional figures and tables

Finally, additional material is provided here. Every figure and table is referred from the paper.

Table 7: Test RMSE for the in-between splits of the UCI datasets (mean and one standard error, the lower the better). Last column is the per-group (weight-space stochasticity vs activation-level stochasticity) average rank.

	Boston	Concrete	Energy	Naval	Wine	Yacht	Rank	Rank (group)
BNN-2	6.54±0.56	7.62±0.35	4.23±1.91	0.03±0.00	**0.63±0.01**	1.18±0.11	4.09±0.67	4.91±0.37
BNN-3	7.77±0.40	16.33±0.67	5.27±1.41	0.02±0.00	0.64±0.01	14.31±0.76	6.15±0.91	
fBNN-2	**3.75±0.21**	7.58±0.41	3.95±1.82	0.03±0.00	0.78±0.02	1.25±0.08	4.70±0.54	
fBNN-3	3.81±0.20	7.52±0.36	4.48±1.79	0.03±0.00	0.87±0.04	1.13±0.12	4.71±0.53	
auNN-RBF-2	4.90±0.47	7.81±0.47	3.41±1.46	0.03±0.00	0.72±0.01	**0.99±0.18**	4.32±0.47	**4.09±0.23**
auNN-RBF-3	4.27±0.29	7.74±0.21	2.72±1.03	0.03±0.00	0.82±0.01	1.03±0.14	4.27±0.55	
auNN-TRI-2	4.01±0.30	7.44±0.38	**2.72±0.79**	0.02±0.00	0.67±0.01	1.51±0.20	3.90±0.33	
auNN-TRI-3	3.78±0.19	**7.03±0.23**	3.36±1.23	**0.02±0.00**	0.68±0.01	3.80±2.41	**3.85±0.46**	

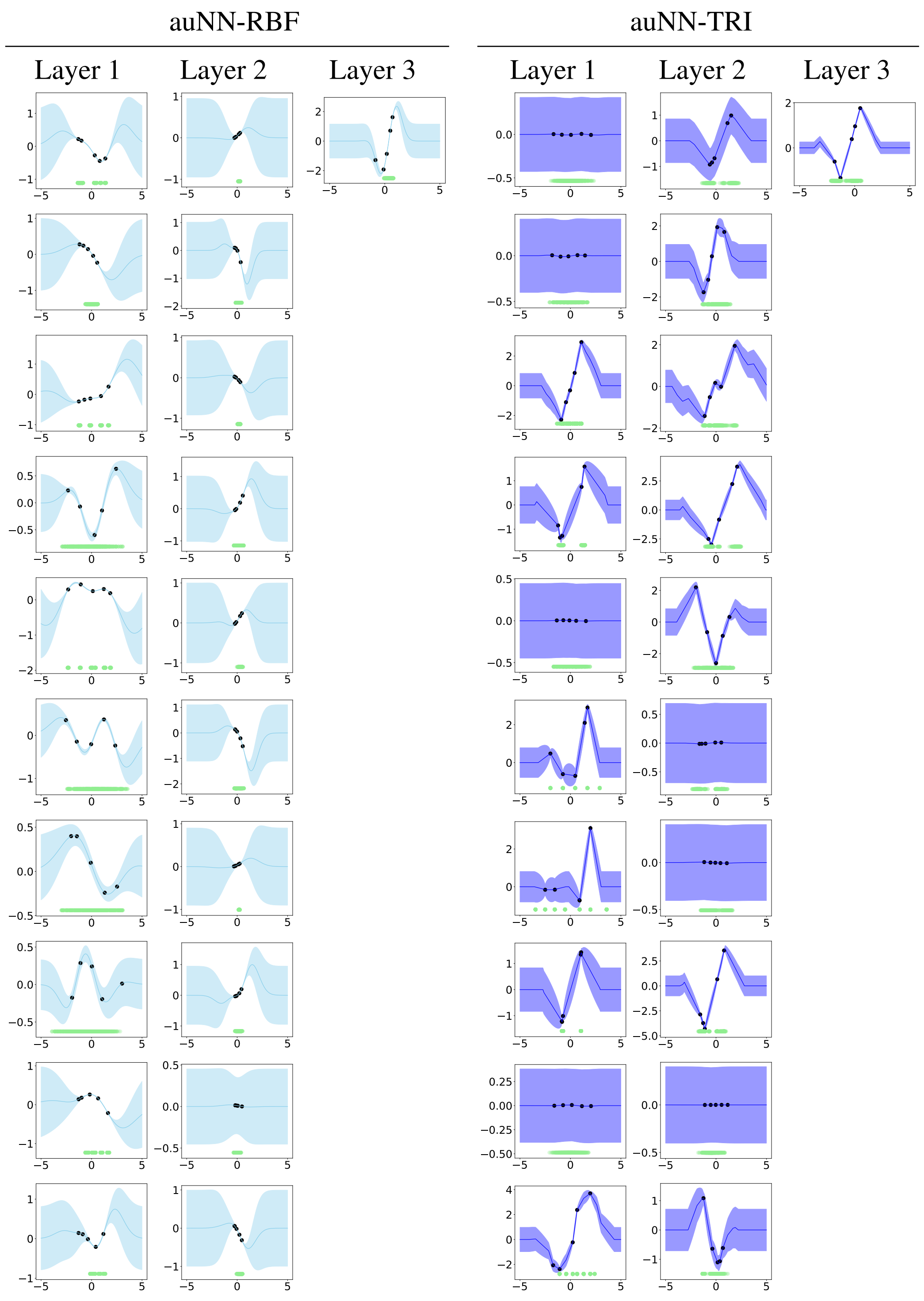


Figure 9: A complete example of the activation functions learned by auNN with RBF and TRI kernels. These were obtained for the Energy dataset with the first in-between split, using three layers, 10 hidden units per (hidden) layer, and 5 inducing points in each unit. Whereas auNN-RBF learns smoother activations, auNN-TRI ones are piece-wise linear, inspired by the ReLu. Notice that auNN allows units to switch off if they are not required. Black dots represent the five inducing points in each unit. Green points are the locations of the input data when propagated to the corresponding unit.

Table 8: Test NLL for the standard splits of the UCI datasets (mean and one standard error, the lower the better). Last column is the per-group (weight-space stochasticity vs activation-level stochasticity) average rank.

test NLL	Boston	Concrete	Energy	Naval	Wine	Yacht	Rank	Rank (group)
BNN-2	2.71±0.07	3.12±0.02	0.65±0.04	-5.38±0.59	0.99±0.02	1.01±0.07	**3.78±0.41**	**4.5±0.39**
BNN-3	3.62±0.05	4.24±0.01	0.80±0.03	-5.02±0.33	1.01±0.02	4.06±0.05	6.25±0.70	
fBNN-2	2.83±0.20	3.20±0.04	0.67±0.04	-6.17±0.02	1.55±0.08	0.77±0.02	4.13±0.57	
fBNN-3	2.75±0.14	3.13±0.05	0.65±0.03	**-6.26±0.00**	207.43±9.12	0.79±0.02	3.83±0.85	
auNN-RBF-2	3.38±0.30	3.14±0.05	0.63±0.03	-5.40±0.08	1.16±0.06	**0.52±0.04**	3.97±0.60	**4.5±0.43**
auNN-RBF-3	3.89±0.47	3.25±0.13	**0.53±0.07**	-5.69±0.03	8.98±1.51	0.54±0.03	4.42±0.85	
auNN-TRI-2	2.56±0.05	3.08±0.02	1.47±0.04	-4.81±0.07	**0.96±0.03**	2.25±0.02	4.78±0.92	
auNN-TRI-3	**2.50±0.02**	**2.98±0.02**	1.42±0.02	-3.43±0.32	1.10±0.07	2.26±0.01	4.83±1.01	

Table 9: Test RMSE for the standard splits of the UCI datasets (mean and one standard error, the lower the better). Last column is the per-group (weight-space stochasticity vs activation-level stochasticity) average rank.

test RMSE	Boston	Concrete	Energy	Naval	Wine	Yacht	Rank	Rank (group)
BNN-2	3.47±0.34	5.49±0.13	0.45±0.02	0.00±0.00	0.65±0.01	0.68±0.08	4.70±0.48	4.59±0.41
BNN-3	8.89±0.45	16.71±0.20	0.51±0.02	0.00±0.00	0.67±0.02	13.49±0.94	6.50±0.64	
fBNN-2	2.80±0.21	5.34±0.13	0.47±0.02	0.00±0.00	0.70±0.02	**0.33±0.04**	3.70±0.61	
fBNN-3	**2.74±0.16**	5.07±0.12	0.46±0.02	**0.00±0.00**	0.83±0.02	0.36±0.04	3.45±0.88	
auNN-RBF-2	3.16±0.23	5.13±0.16	0.45±0.02	0.00±0.00	0.67±0.02	0.41±0.04	4.25±0.35	**4.41±0.41**
auNN-RBF-3	3.01±0.25	**4.51±0.18**	**0.41±0.03**	0.00±0.00	0.76±0.02	0.38±0.03	**3.35±0.77**	
auNN-TRI-2	3.00±0.26	5.21±0.10	0.72±0.02	0.00±0.00	**0.62±0.02**	1.15±0.14	5.40±0.80	
auNN-TRI-3	2.81±0.17	4.67±0.15	0.65±0.03	0.01±0.00	0.62±0.02	1.16±0.15	4.65±1.00	

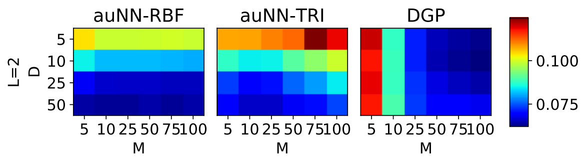


Figure 10: Test RMSE of auNN and DGP for different values of M (number of inducing points) and D (number of hidden units). Results are the average over 5 independent runs on the UCI Kin8 dataset. The lower the better. Whereas DGP improves vertically (i.e. with M), auNN does it horizontally (i.e. with D). This is as theoretically expected, and it is convenient from a scalability viewpoint.

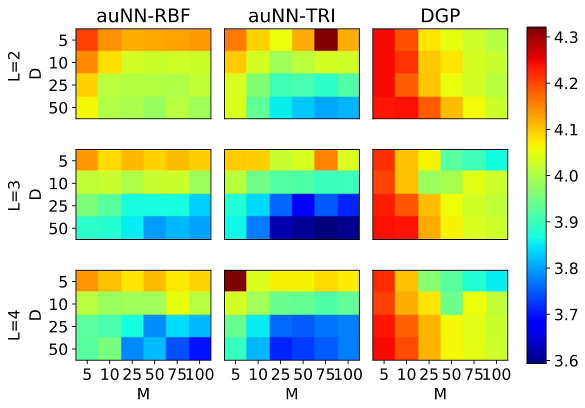


Figure 11: Test RMSE with increasing depth ($L = 2, 3, 4$). This supports that auNN might benefit more than DGP from deeper networks. Moreover, the aforementioned different influence of M and D on DGP and auNN is confirmed here.

Table 10: Standard error obtained by auNN and DGP in three splits of the large scale classification datasets.

			auNN						DGP		
	N	D	RBF-2	RBF-3	RBF-4	TRI-2	TRI-3	TRI-4	DGP-2	DGP-3	DGP-4
HIGGS	11M	28	0.0001	0.0006	0.0007	0.0003	0.0004	0.0008	0.0005	0.0009	0.0010
SUSY	5M	18	0.0004	0.0005	0.0005	0.0005	0.0005	0.0004	0.0005	0.0027	0.0035

Table 11: Test NLL of auNN and DGP for different values of M (number of inducing points) and D (number of hidden units). Mean and one standard error over 5 independent runs on the UCI Kin8 dataset are shown. The lower the better.

	auNN-RBF						auNN-TRI						DGP					
D M	5	10	25	50	75	100	5	10	25	50	75	100	5	10	25	50	75	100
5	-0.85±0.01	-0.89±0.01	-0.89±0.01	-0.89±0.01	-0.89±0.01	-0.90±0.01	-0.78±0.00	-0.79±0.03	-0.78±0.05	-0.77±0.04	-0.67±0.06	-0.71±0.04	-0.67±0.01	-0.98±0.00	-1.19±0.01	-1.30±0.01	-1.33±0.01	-1.34±0.01
10	-1.06±0.01	-1.09±0.01	-1.09±0.01	-1.09±0.01	-1.10±0.02	-1.10±0.01	-0.96±0.01	-1.02±0.01	-1.03±0.01	-0.98±0.03	-0.94±0.03	-0.89±0.03	-0.69±0.01	-0.98±0.00	-1.19±0.00	-1.30±0.01	-1.33±0.01	-1.35±0.01
25	-1.27±0.02	-1.30±0.02	-1.30±0.02	-1.30±0.02	-1.31±0.01	-1.31±0.02	-1.09±0.01	-1.19±0.01	-1.22±0.01	-1.15±0.02	-1.11±0.01	-1.06±0.03	-0.68±0.01	-0.98±0.00	-1.17±0.01	-1.26±0.01	-1.29±0.01	-1.30±0.01
50	-1.33±0.01	-1.34±0.01	-1.34±0.02	-1.33±0.01	-1.34±0.02	-1.32±0.03	-1.15±0.01	-1.24±0.01	-1.29±0.01	-1.26±0.01	-1.24±0.02	-1.19±0.02	-0.69±0.01	-0.96±0.01	-1.16±0.01	-1.21±0.01	-1.22±0.01	-1.24±0.01

Table 12: Test RMSE of auNN and DGP for different values of M (number of inducing points) and D (number of hidden units). Mean and one standard error over 5 independent runs on the UCI Kin8 dataset are shown. The lower the better.

	auNN-RBF						auNN-TRI						DGP					
D M	5	10	25	50	75	100	5	10	25	50	75	100	5	10	25	50	75	100
5	0.10±0.00	0.10±0.00	0.10±0.00	0.10±0.00	0.10±0.00	0.10±0.00	0.11±0.00	0.11±0.00	0.11±0.01	0.11±0.00	0.12±0.01	0.12±0.00	0.12±0.00	0.09±0.00	0.07±0.00	0.07±0.00	0.06±0.00	0.06±0.00
10	0.08±0.00	0.08±0.00	0.08±0.00	0.08±0.00	0.08±0.00	0.08±0.00	0.09±0.00	0.08±0.00	0.08±0.00	0.09±0.00	0.09±0.00	0.10±0.00	0.12±0.00	0.09±0.00	0.07±0.00	0.06±0.00	0.06±0.00	0.06±0.00
25	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.08±0.00	0.08±0.00	0.08±0.00	0.12±0.00	0.09±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.06±0.00
50	0.06±0.00	0.06±0.00	0.06±0.00	0.06±0.00	0.06±0.00	0.06±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.12±0.00	0.09±0.00	0.07±0.00	0.07±0.00	0.07±0.00	0.07±0.00

Table 13: Test NLL of auNN and DGP for different values of M (number of inducing points) and D (number of hidden units) as the depth increases from $L = 2$ to $L = 4$. Mean and one standard error over 5 independent runs on the UCI Power dataset are shown. The lower the better.

		auNN-RBF						auNN-TRI						DGP					
L	D M	5	10	25	50	75	100	5	10	25	50	75	100	5	10	25	50	75	100
2	5	2.86±0.02	2.84±0.02	2.84±0.02	2.84±0.02	2.84±0.02	2.84±0.02	2.85±0.02	2.83±0.02	2.82±0.02	2.84±0.02	2.89±0.03	2.84±0.02	2.87±0.02	2.85±0.02	2.83±0.02	2.82±0.02	2.81±0.02	2.81±0.02
	10	2.84±0.02	2.83±0.02	2.82±0.02	2.81±0.02	2.81±0.02	2.81±0.02	2.84±0.02	2.82±0.02	2.80±0.02	2.81±0.02	2.81±0.02	2.81±0.02	2.87±0.02	2.86±0.02	2.83±0.02	2.83±0.02	2.82±0.02	2.81±0.02
	25	2.83±0.02	2.81±0.02	2.81±0.02	2.81±0.02	2.81±0.02	2.81±0.02	2.83±0.02	2.80±0.02	2.78±0.02	2.78±0.02	2.78±0.02	2.79±0.02	2.87±0.02	2.85±0.02	2.83±0.02	2.82±0.02	2.81±0.02	2.81±0.02
	50	2.82±0.02	2.81±0.02	2.81±0.02	2.80±0.02	2.81±0.02	2.81±0.02	2.82±0.02	2.80±0.02	2.77±0.02	2.76±0.02	2.76±0.02	2.76±0.03	2.86±0.02	2.87±0.02	2.85±0.02	2.83±0.02	2.82±0.02	2.81±0.02
3	5	2.84±0.02	2.83±0.02	2.83±0.02	2.83±0.03	2.83±0.02	2.83±0.02	2.84±0.02	2.83±0.02	2.82±0.02	2.82±0.02	2.85±0.02	2.82±0.02	2.86±0.02	2.83±0.02	2.82±0.02	2.79±0.02	2.78±0.02	2.77±0.01
	10	2.81±0.02	2.81±0.02	2.81±0.02	2.81±0.02	2.81±0.02	2.80±0.02	2.82±0.02	2.80±0.02	2.79±0.02	2.79±0.02	2.78±0.02	2.78±0.02	2.86±0.02	2.83±0.02	2.80±0.02	2.81±0.02	2.82±0.02	2.81±0.02
	25	2.80±0.02	2.79±0.02	2.77±0.02	2.77±0.02	2.77±0.02	2.77±0.02	2.79±0.02	2.77±0.02	2.74±0.02	2.72±0.02	2.74±0.03	2.74±0.03	2.86±0.02	2.85±0.02	2.83±0.02	2.82±0.02	2.81±0.02	2.81±0.02
	50	2.78±0.02	2.78±0.02	2.77±0.02	2.76±0.02	2.76±0.02	2.76±0.03	2.78±0.02	2.75±0.02	2.71±0.02	2.71±0.03	2.70±0.03	2.70±0.02	2.87±0.02	2.87±0.02	2.84±0.02	2.82±0.02	2.82±0.02	2.81±0.02
4	5	2.84±0.02	2.83±0.02	2.82±0.02	2.83±0.02	2.82±0.01	2.83±0.02	3.69±0.35	2.83±0.01	2.83±0.02	2.83±0.02	2.83±0.02	2.82±0.02	2.86±0.02	2.83±0.02	2.80±0.02	2.79±0.02	2.78±0.02	2.77±0.02
	10	2.81±0.02	2.80±0.02	2.80±0.02	2.80±0.02	2.82±0.02	2.81±0.02	2.83±0.02	2.81±0.01	2.79±0.01	2.79±0.02	2.79±0.02	2.79±0.02	2.86±0.02	2.84±0.02	2.83±0.02	2.79±0.02	2.82±0.02	2.81±0.02
	25	2.79±0.02	2.78±0.02	2.77±0.02	2.75±0.02	2.77±0.02	2.76±0.02	2.80±0.01	2.78±0.02	2.75±0.02	2.74±0.02	2.75±0.03	2.75±0.02	2.86±0.02	2.85±0.02	2.83±0.02	2.82±0.02	2.82±0.02	2.81±0.02
	50	2.79±0.02	2.80±0.02	2.75±0.03	2.77±0.03	2.75±0.03	2.74±0.03	2.79±0.01	2.77±0.01	2.73±0.02	2.74±0.02	2.74±0.02	2.75±0.02	2.87±0.02	2.85±0.02	2.84±0.02	2.82±0.02	2.82±0.02	2.81±0.02

Table 14: Test RMSE of auNN and DGP for different values of M (number of inducing points) and D (number of hidden units) as the depth increases from $L = 2$ to $L = 4$. Mean and one standard error over 5 independent runs on the UCI Power dataset are shown. The lower the better.

			auNN-RBF						auNN-TRI						DGP					
L	D	M	5	10	25	50	75	100	5	10	25	50	75	100	5	10	25	50	75	100
2	5		4.20±0.09	4.14±0.08	4.12±0.08	4.13±0.08	4.13±0.07	4.14±0.09	4.16±0.08	4.09±0.08	4.06±0.09	4.12±0.09	4.32±0.13	4.12±0.09	4.24±0.10	4.19±0.09	4.08±0.09	4.05±0.08	4.03±0.07	4.01±0.07
	10		4.15±0.09	4.08±0.08	4.03±0.08	4.03±0.07	4.03±0.09	4.03±0.09	4.10±0.10	4.03±0.08	3.99±0.08	4.01±0.08	4.03±0.09	4.03±0.07	4.24±0.10	4.21±0.09	4.10±0.08	4.08±0.08	4.03±0.08	4.02±0.07
	25		4.09±0.08	4.01±0.08	4.01±0.08	4.01±0.08	4.00±0.09	4.02±0.08	4.04±0.08	3.96±0.07	3.90±0.08	3.91±0.08	3.89±0.08	3.92±0.07	4.24±0.09	4.18±0.09	4.10±0.09	4.06±0.08	4.03±0.08	4.01±0.08
	50		4.06±0.08	4.00±0.07	4.00±0.07	3.98±0.07	4.01±0.09	3.99±0.08	4.04±0.08	3.93±0.07	3.86±0.09	3.83±0.08	3.81±0.08	3.81±0.10	4.24±0.10	4.24±0.10	4.18±0.09	4.11±0.09	4.06±0.09	4.03±0.08
3	5		4.14±0.09	4.08±0.08	4.11±0.06	4.09±0.11	4.11±0.09	4.09±0.09	4.10±0.09	4.10±0.07	4.02±0.08	4.04±0.08	4.15±0.08	4.04±0.09	4.22±0.09	4.10±0.08	4.07±0.09	3.92±0.07	3.90±0.06	3.86±0.05
	10		4.02±0.08	4.02±0.08	4.00±0.07	4.02±0.07	4.02±0.07	3.99±0.07	4.01±0.08	3.95±0.07	3.92±0.07	3.92±0.08	3.90±0.07	3.90±0.08	4.20±0.09	4.10±0.08	3.98±0.08	3.99±0.07	4.05±0.08	4.03±0.08
	25		3.96±0.08	3.93±0.07	3.87±0.07	3.87±0.07	3.87±0.07	3.83±0.07	3.88±0.08	3.84±0.08	3.76±0.07	3.67±0.06	3.75±0.10	3.71±0.09	4.23±0.10	4.19±0.08	4.11±0.09	4.06±0.08	4.03±0.08	4.02±0.08
	50		3.89±0.08	3.88±0.07	3.85±0.06	3.80±0.09	3.82±0.07	3.80±0.08	3.86±0.08	3.77±0.09	3.62±0.06	3.61±0.08	3.59±0.09	3.60±0.07	4.24±0.10	4.24±0.10	4.12±0.09	4.07±0.08	4.04±0.08	4.03±0.08
4	5		4.14±0.10	4.10±0.08	4.08±0.08	4.11±0.09	4.07±0.05	4.09±0.09	12.00±3.26	4.04±0.07	4.06±0.09	4.07±0.07	4.09±0.08	4.07±0.08	4.22±0.09	4.10±0.08	3.97±0.07	3.93±0.07	3.88±0.07	3.85±0.07
	10		4.01±0.08	3.98±0.07	3.99±0.07	3.99±0.07	4.05±0.07	4.01±0.06	4.03±0.09	3.99±0.06	3.94±0.06	3.94±0.07	3.92±0.07	3.93±0.08	4.20±0.09	4.12±0.08	4.08±0.10	3.94±0.07	4.06±0.08	4.01±0.08
	25		3.93±0.09	3.91±0.08	3.87±0.07	3.78±0.06	3.84±0.07	3.82±0.07	3.94±0.07	3.85±0.08	3.76±0.08	3.75±0.08	3.77±0.10	3.78±0.09	4.24±0.09	4.18±0.09	4.11±0.09	4.06±0.08	4.04±0.08	4.03±0.08
	50		3.92±0.08	3.96±0.07	3.78±0.11	3.82±0.07	3.74±0.10	3.70±0.07	3.90±0.08	3.82±0.06	3.71±0.09	3.73±0.09	3.75±0.08	3.77±0.08	4.24±0.09	4.20±0.10	4.12±0.09	4.06±0.08	4.04±0.08	4.03±0.08

References

[Agostinelli et al., 2014] Agostinelli, F., Hoffman, M., Sadowski, P., and Baldi, P. (2014). Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*.

[Ashukha et al., 2020] Ashukha, A., Lyzhov, A., Molchanov, D., and Vetrov, D. (2020). Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *International Conference on Learning Representations*.

[Ba et al., 2016] Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[Baldi et al., 2014] Baldi, P., Sadowski, P., and Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5:4308.

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622.

[Bui et al., 2016] Bui, T., Hernández-Lobato, D., Hernandez-Lobato, J., Li, Y., and Turner, R. (2016). Deep Gaussian processes for regression using approximate expectation propagation. In *International conference on machine learning*, pages 1472–1481.

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

[De G. Matthews et al., 2017] De G. Matthews, A. G., Van Der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrá, P., Ghahramani, Z., and Hensman, J. (2017). Gpflow: A Gaussian process library using tensorflow. *The Journal of Machine Learning Research*, 18(1):1299–1304.

[Esteva et al., 2017] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118.

[Filos et al., 2019] Filos, A., Farquhar, S., Gomez, A. N., Rudner, T. G., Kenton, Z., Smith, L., Alizadeh, M., de Kroon, A., and Gal, Y. (2019). Benchmarking Bayesian deep learning with diabetic retinopathy diagnosis.

[Flam-Shepherd et al., 2017] Flam-Shepherd, D., Requeima, J., and Duvenaud, D. (2017). Mapping Gaussian process priors to Bayesian neural networks. In *NIPS Bayesian deep learning workshop*.

[Foong et al., 2019a] Foong, A. Y., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2019a). 'in-between'uncertainty in Bayesian neural networks. *ICML 2019 Workshop on Uncertainty and Robustness in Deep Learning*.

[Foong et al., 2019b] Foong, A. Y. K., Burt, D. R., Li, Y., and Turner, R. E. (2019b). On the expressiveness of approximate inference in Bayesian neural networks. *arXiv preprint arXiv:1909.00719v3*.

[Fort et al., 2019] Fort, S., Hu, H., and Lakshminarayanan, B. (2019). Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*.

[Gal, 2016] Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

[Gal and Ghahramani, 2016] Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

[Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

[Guo et al., 2017] Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org.

[Hafner et al., 2019] Hafner, D., Tran, D., Lillicrap, T. P., Irpan, A., and Davidson, J. (2019). Noise contrastive priors for functional uncertainty. In *UAI*, page 332. AUAI Press.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

[Heek and Kalchbrenner, 2019] Heek, J. and Kalchbrenner, N. (2019). Bayesian inference for large scale image classification. *arXiv preprint arXiv:1908.03491*.

[Hensman et al., 2013] Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 282–290.

[Hensman et al., 2015] Hensman, J., Matthews, A., and Ghahramani, Z. (2015). Scalable variational Gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360.

[Hernández-Lobato and Hernández-Lobato, 2016] Hernández-Lobato, D. and Hernández-Lobato, J. M. (2016). Scalable Gaussian process classification via expectation propagation. In *Artificial Intelligence and Statistics*, pages 168–176.

[Hernández-Lobato et al., 2011] Hernández-Lobato, D., Hernández-Lobato, J. M., and Dupont, P. (2011). Robust multi-class Gaussian process classification. In *Advances in neural information processing systems*, pages 280–288.

[Hinton et al., 2012] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

[Kendall and Gal, 2017] Kendall, A. and Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*

[Kingma and Welling, 2013] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114.*

[Klambauer et al., 2017] Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in neural information processing systems*, pages 971–980.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Lakshminarayanan et al., 2017] Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413.

[Leimkuhler et al., 2019] Leimkuhler, B., Matthews, C., and Vlaar, T. (2019). Partitioned integrators for thermodynamic parameterization of neural networks. *arXiv preprint arXiv:1908.11843.*

[MacKay, 1992] MacKay, D. J. (1992). A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.

[Maddox et al., 2019] Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2019). A simple baseline for Bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13132–13143.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

[Mobiny et al., 2019] Mobiny, A., Singh, A., and Van Nguyen, H. (2019). Risk-aware machine learning classifier for skin lesion diagnosis. *Journal of clinical medicine*, 8(8):1241.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

[Neal, 1995] Neal, R. M. (1995). *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto.

[Nguyen et al., 2015] Nguyen, A., Yosinski, J., and Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436.

[Osawa et al., 2019] Osawa, K., Swaroop, S., Khan, M. E. E., Jain, A., Eschenhagen, R., Turner, R. E., and Yokota, R. (2019). Practical deep learning with Bayesian principles. In *Advances in Neural Information Processing Systems*, pages 4289–4301.

[Pearce et al., 2019] Pearce, T., Tsuchida, R., Zaki, M., Brintrup, A., and Neely, A. (2019). Expressive priors in Bayesian neural networks: Kernel combinations and periodic functions. In *35th Conference on Uncertainty in Artificial Intelligence, UAI 2019.*

[Ren et al., 2019] Ren, J., Liu, P. J., Fertig, E., Snoek, J., Poplin, R., Depristo, M., Dillon, J., and Lakshminarayanan, B. (2019). Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, pages 14680–14691.

[Rezende et al., 2014] Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286.

[Salimans and Kingma, 2016] Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in neural information processing systems*, pages 901–909.

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

[Särkkä and Solin, 2019] Särkkä, S. and Solin, A. (2019). *Applied stochastic differential equations*, volume 10. Cambridge University Press.

[Snoek et al., 2019] Snoek, J., Ovadia, Y., Fertig, E., Lakshminarayanan, B., Nowozin, S., Sculley, D., Dillon, J., Ren, J., and Nado, Z. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13969–13980.

[Sun et al., 2019] Sun, S., Zhang, G., Shi, J., and Grosse, R. (2019). Functional variational Bayesian neural networks. In *International Conference on Learning Representations*.

[Titsias, 2009] Titsias, M. (2009). Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574.

[Urban and van der Smagt, 2018] Urban, S. and van der Smagt, P. (2018). Gaussian process neurons. `https://openreview.net/forum?id=By-IifZRW`. Accessed: 2020-05-15.

[Wenzel et al., 2020] Wenzel, F., Roth, K., Veeling, B. S., Świątkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R., and Nowozin, S. (2020). How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*.

[Williams and Rasmussen, 2006] Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

[Zhang et al., 2018] Zhang, G., Sun, S., Duvenaud, D., and Grosse, R. (2018). Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, pages 5852–5861.

[Zhang et al., 2020] Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. (2020). Cyclical stochastic gradient mcmc for Bayesian deep learning. In *International Conference on Learning Representations*.

CHAPTER 9

Concluding remarks

The main global conclusion of this PhD thesis is the versatility of Gaussian Processes to model different scenarios (regression, classification, crowdsourcing) and target various applications (remote sensing, security, astrophysics), either as the central algorithm to perform the task at hand (Chapters 2-7) or as an auxiliary tool to be integrated within a larger model (Chapter 8). This can be specified through six specific conclusions:

- Scalability in GP classification can be achieved through the use of Fourier features. These can be sampled and fixed from the beginning or learned during the training step. This provides an alternative to the inducing point based approach.
- Local variational methods outperform Expectation Propagation when doing inference in GP based crowdsourcing. The benefit is both in terms of accuracy and computational cost.
- Scalability in GP based crowdsourcing can be also achieved through the use of Fourier features. As in the classification case, higher efficiency can be obtained if Fourier features are sampled and fixed from the beginning, whereas higher accuracy can be reached if they are estimated during training.
- Popular techniques for GP classification such as inducing point based scalability, Normalizing Flows, and inference networks can be extended to the crowdsourcing scenario. The adapted algorithms inherit the main properties of their classification counterparts.
- Expert knowledge can be incorporated within the inducing point based method for GP crowdsourcing. This allows for obtaining state-of-the-art results for glitch detection in the search for gravitational waves in the LIGO project.
- GPs can be used to model the activation function of deep neural networks, leading to the novel notion of activation-level uncertainty.

REFERENCES

Alaa, A. M. and van der Schaar, M. (2017), Bayesian inference of individualized treatment effects using multi-task gaussian processes, *in* 'Advances in Neural Information Processing Systems', pp. 3424–3432.

Albarqouni, S., Baur, C., Achilles, F., Belagiannis, V., Demirci, S. and Navab, N. (2016), 'Aggnet: deep learning from crowds for mitosis detection in breast cancer histology images', *IEEE transactions on medical imaging* **35**(5), 1313–1321.

Bishop, C. M. (2006), *Pattern recognition and machine learning*, springer.

Buhrmester, M., Kwang, T. and Gosling, S. D. (2011), 'Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data?', *Perspectives on Psychological Science* **6**(1), 3–5.

Burt, D., Rasmussen, C. E. and Van Der Wilk, M. (2019), Rates of convergence for sparse variational gaussian process regression, *in* 'International Conference on Machine Learning', pp. 862–871.

Cheng, C. A. and Boots, B. (2016), Incremental variational sparse gaussian process regression, *in* 'Advances in Neural Information Processing Systems', pp. 4410–4418.

Dai, B., He, N., Dai, H. and Song, L. (2016), Provable bayesian inference via particle mirror descent, *in* 'Artificial Intelligence and Statistics', pp. 985–994.

Dai, J. and Krems, R. V. (2020), 'Interpolation and extrapolation of global potential energy surfaces for polyatomic systems by gaussian processes with composite kernels', *Journal of Chemical Theory and Computation* **16**(3), 1386–1395.

Desai, A., Warner, J., Kuderer, N., Thompson, M., Painter, C., Lyman, G. and Lopes, G. (2020), 'Crowdsourcing a crisis response for covid-19 in oncology', *Nature Cancer* **1**(5), 473–476.

Filos, A., Farquhar, S., Gomez, A. N., Rudner, T. G., Kenton, Z., Smith, L., Alizadeh, M., de Kroon, A. and Gal, Y. (2019), 'A systematic comparison of bayesian deep learning robustness in diabetic retinopathy tasks', *arXiv preprint arXiv:1912.10481* .

Fritz, S., See, L., Perger, C., McCallum, I., Schill, C., Schepaschenko, D., Duerauer, M., Karner, M., Dresel, C., Laso-Bayas, J.-C. et al. (2017), 'A global dataset of crowd-sourced land cover and land use reference data', *Scientific data* **4**, 170075.

Gal, Y. (2016), Uncertainty in Deep Learning, PhD thesis, University of Cambridge.

Heim, E., Seitel, A., Andrulis, J., Isensee, F., Stock, C., Ross, T. and Maier-Hein, L. (2017), 'Clickstream analysis for crowd-based object segmentation with confidence', *IEEE transactions on pattern analysis and machine intelligence* **40**(12), 2814–2826.

Hensman, J., Fusi, N. and Lawrence, N. D. (2013), Gaussian processes for big data, *in* 'Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence', pp. 282–290.

Hensman, J., Matthews, A. and Ghahramani, Z. (2015), Scalable variational Gaussian process classification, *in* 'Artificial Intelligence and Statistics', pp. 351–360.

Howe, J. (2006), 'The rise of crowdsourcing', *Wired magazine* **14**(6), 1–4.

Kendall, A. and Gal, Y. (2017), What uncertainties do we need in Bayesian deep learning for computer vision?, *in* 'Advances in neural information processing systems', pp. 5574–5584.

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E. and Figueiras-Vidal, A. R. (2010), 'Sparse spectrum gaussian process regression', *The Journal of Machine Learning Research* **11**, 1865–1881.

Manobel, B., Sehnke, F., Lazzús, J. A., Salfate, I., Felder, M. and Montecinos, S. (2018), 'Wind turbine power curve modeling based on gaussian processes and artificial neural networks', *Renewable Energy* **125**, 1015–1020.

Mobiny, A., Singh, A. and Van Nguyen, H. (2019), 'Risk-aware machine learning classifier for skin lesion diagnosis', *Journal of clinical medicine* **8**(8), 1241.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005), 'A unifying view of sparse approximate gaussian process regression', *Journal of Machine Learning Research* **6**(Dec), 1939–1959.

Rahimi, A. and Recht, B. (2008), Random features for large-scale kernel machines, *in* 'Advances in neural information processing systems', pp. 1177–1184.

Rodrigues, F., Lourenco, M., Ribeiro, B. and Pereira, F. C. (2017), 'Learning supervised topic models for classification and regression from crowds', *IEEE transactions on pattern analysis and machine intelligence* **39**(12), 2409–2422.

Rodrigues, F., Pereira, F. and Ribeiro, B. (2014), Gaussian process classification and active learning with multiple annotators, *in* 'International conference on machine learning', pp. 433–441.

Rudin, W. (1962), *Fourier analysis on groups*, Vol. 121967, Wiley Online Library.

Sáez-Rodríguez, J., Costello, J. C., Friend, S. H., Kellen, M. R., Mangravite, L., Meyer, P., Norman, T. and Stolovitzky, G. (2016), 'Crowdsourcing biomedical research: leveraging communities as innovation engines', *Nature Reviews Genetics* **17**(8), 470.

Shi, J., Khan, M. E. and Zhu, J. (2019), Scalable training of inference networks for Gaussian-process models, *in* 'International Conference on Machine Learning', pp. 5758–5768.

Snelson, E. and Ghahramani, Z. (2006), Sparse gaussian processes using pseudo-inputs, *in* 'Advances in neural information processing systems', pp. 1257–1264.

Snow, R., O'connor, B., Jurafsky, D. and Ng, A. Y. (2008), Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks, *in* 'Proceedings of the 2008 conference on empirical methods in natural language processing', pp. 254–263.

Sun, S., Zhang, G., Shi, J. and Grosse, R. (2019), Functional variational Bayesian neural networks, *in* 'International Conference on Learning Representations'.
URL: *https://openreview.net/forum?id=rkxacs0qY7*

Swain, P. S., Stevenson, K., Leary, A., Montano-Gutierrez, L. F., Clark, I. B., Vogel, J. and Pilizota, T. (2016), 'Inferring time derivatives including cell growth rates using gaussian processes', *Nature communications* **7**(1), 1–8.

Titsias, M. (2009), Variational learning of inducing variables in sparse Gaussian processes, *in* 'Artificial Intelligence and Statistics', pp. 567–574.

Williams, C. K. and Rasmussen, C. E. (2006), *Gaussian processes for machine learning*, Vol. 2, MIT press Cambridge, MA.

Zhang, J., Wu, X. and Sheng, V. S. (2016), 'Learning from crowdsourced labeled data: a survey', *Artificial Intelligence Review* **46**(4), 543–576.