Mixture of Gaussian Processes for Bayesian Active Learning

DOI

10.36227/techrxiv.23702517.v1

Mixture of Gaussian Processes for Bayesian Active Learning

Christoffer Riis, Filipe Rodrigues, and Francisco Câmara Pereira, *Member, IEEE*

Abstract—This paper introduces a Mixture of Gaussian processes (MGP) model and explores its application in the context of Bayesian active learning. The MGP offers an alternative approach to fully Bayesian Gaussian processes by leveraging the benefits of 'fully' Bayesian active learning while circumventing the computationally expensive Monte Carlo sampling of the Gaussian process's hyperparameters. Through a detailed empirical analysis, we demonstrate that the MGP equipped with Bayesian Active Learning by Disagreement (BALD) improves querying efficiency and delivers competitive performance compared to both standard and fully Bayesian Gaussian processes. Across six classic simulators, our experiments reveal that the MGP with BALD achieves, on average, the lowest negative log probability with the fewest iterations. Moreover, these models are more than seven times faster than fully Bayesian Gaussian processes with BALD. Furthermore, we extend our evaluation to a real-world simulator from the air traffic management domain, where MGP outperforms both Gaussian processes and fully Bayesian Gaussian processes. Additionally, we demonstrate the applicability of the MGP within the Bayesian optimization framework, where it yields the best minimum on five out of the six simulators considered.

Index Terms—Gaussian process, Bayesian active learning, Bayesian optimization

1 Introduction

In many machine learning applications, gathering and labeling the data needed for good performance is often expensive. Active learning aims to reduce this cost while attaining high predictive performance with as few collected data samples as possible. This is achieved by an iterative process, in which new data is gathered based on an information criterion, such as the predictive uncertainty of the model, after which the model is updated, taking the new data point into account [1], [2]. One such application is surrogate modeling (also known as metamodeling) of computer experiments and simulators, and here active learning has proven itself an effective tool to minimize the labeling cost in multiple areas, e.g., in physics [3], for medical services [4], and in transportation [5].

For active learning surrogates for simulators, it is common practice to use a Gaussian process (GP) as the surrogate model [6]. The GP is known for its ability to provide the necessary flexibility without overfitting to the data while also providing predictive uncertainty estimates [7], which are powerful indicators to be used in the active learning acquisition function since they can be used to directly minimize the predictive uncertainty of the surrogates [1].

Often acquisition functions are optimal only when the optimal hyperparameters are known [8]. However, in general, the hyperparameters of the GP are unknown. The cyclic nature of active learning gives rise to a chicken-and-the-egg problem, where the optimal data acquisition is dependent on the model, but the optimal model is dependent on the current data [8]. This issue is particularly relevant when there are only a few data points, giving rise to multimodal posteriors of the hyperparameters [9], [10]. As a remedy, one can extend the GP to a fully Bayesian GP [11], where the hyperparameters are inferred using, for example, Hamiltonian Monte Carlo sampling (HMC). The hyperparameters inferred from the HMC have been found to give better fits than hyperparameters estimated based on maximum likelihood estimates [11]. Secondly, from the HMC sampling, one can also estimate the joint posterior of the hyperparameters, which can be applied to extend the fully Bayesian GP to a Bayesian mixture of GPs (MGP). The uncertainty from the MGP captures the predictive uncertainty arising from the limited amount of data alongside the entropy coming from the posterior of the hyperparameters. As a result, data points queried based on this uncertainty measure will now simultaneously reduce the predictive uncertainty and help estimate the hyperparameters faster [9].

However, the fully Bayesian GP is a particularly expensive model to train due to the HMC sampling. Within the setting of active learning, the computational burden of the HMC sampling is further amplified since the model must be refitted in each iteration of the active learning loop. Considering that some computer experiments require only a few minutes per run [12], this makes the HMC sampling a significant computational overhead in the active learning setup. Within the setting of active learning, we reduce this cost by approximating the Bayesian mixture of GPs with a mixture of GPs (MGP), coming from initializing multiple GPs with different sets of hyperparameters, which are optimized to recreate an estimation of the uncertainty measure of the Bayesian MGP.

Our contribution We introduce a novel approach to construct a Mixture of Gaussian processes (MGP) efficiently, showing that it is at least seven times faster than the Bayesian MGP without losing performance competitiveness. We extend current state-of-the-art active learning acquisition functions to apply to the proposed model and show that the MPG can utilize the 'fully' Bayesian active learning acquisition functions similarly to the fully Bayesian GP but

All authors were with Machine Learning for Smart Mobility, DTU Management, Technical University of Denmark, e-mails: {chrrii,rodr,camara}@dtu.dk.

without the computationally expensive Monte Carlo sampling. Additionally, we apply it to a real-world application within air traffic management, showing faster and better convergence. Finally, we demonstrate its applicability within Bayesian optimization.

2 Related work

2.1 Gaussian processes in surrogate modeling

Gaussian processes (GPs) have proven themselves effective models for surrogate modeling [6]. In surrogate modeling, we want to spend the fewest resources on data gathering while achieving good predictive performance, and thus, there is much research on the design of experiments, trying to construct an optimal data set in advance to querying any data [13]. For a GP with known hyperparameters, the a priori design based on the frequently used measure ENTROPY can be optimized in advance [8]. However, such designs are often non-optimal for a GP with unknown hyperparameters, making the more costly iterative designs as active learning (or sequential designs) more efficient [14]. Moreover, the advantage of using active learning compared to the computationally efficient a priori designs decreases as the optimal hyperparameters of the GPs have been estimated [8]. This has led to the development of advanced methods for more accurate estimation of hyperparameters, such as initial designs tailored for space-filling and hyperparameter learning [15], or the use of sophisticated inference techniques like HMC sampling [11]. However, the effectiveness of the former initial design relies on a sufficiently large number of data points, typically around $4n$ points for an n-dimensional problem [15]. Regardless of the initial design, the latter inference method enables the use of fully Bayesian Gaussian processes, which yield improved estimates of optimal hyperparameters compared to maximum likelihood estimation, thus leading to better predictive performance [11].

2.2 Mixture of Gaussian processes

Originally, the mixture of GPs (MGP) was proposed to model input-dependent hyperparameters [16], but it has later been extended to model multi-modal data, alongside reducing the training complexity [17]–[19]. The training complexity has recently been reduced further, thus making it manageable to handle, for example, 500k data points [20]. Parallel to this, the MGP has also been adjusted to work for small data sets, e.g., in the setting of active learning in classification tasks [21]. Overall, the research on MGPs, is focusing on using the MGPs to construct different GP components with each component learning a single-mode hyperplane. Similar to these works, is the ensemble of GPs with different kernels [22]–[24]. However, for these ensembles of GPs the goal is not to model multi-modal data, but rather to find an optimal kernel function. In contrast to these models, the fully Bayesian GP [11] and the Bayesian mixture of GPs [9] are using the extra Bayesian information to effectively estimating the hyperparameters of a given kernel, thus increasing the predictive performance compared to a standard GP. These models are tailored to small to medium-sized data sets, since they are computationally expensive to use due to the HMC sampling. Our work expands the research on MGPs in the direction of the effective estimation of the hyperparameters. We introduce an approach to efficently construct the MGP, achieving the same predictive performance while avoiding the computationally expensive HMC sampling.

3 Background

This section lays out the background on Gaussian processes and active learning, including Bayesian active learning.

3.1 Gaussian processes

A Gaussian process is a non-parametric model over the space of functions with a distribution specified by a mean function $m_{\boldsymbol{f}}(\cdot)$ and a kernel (or covariance function) $k_{\boldsymbol{f}}(\cdot,\cdot)$ [7]. Given the data $\mathcal{D} = (X, \boldsymbol{y}) = \{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$, where $\boldsymbol{x}$ is an input with d features and y_i is the corrupted observations of some latent function values $\boldsymbol{f}$ with Gaussian noise ε, i.e., $y_i = f_i + \varepsilon_i,\ \varepsilon_i \in \mathcal{N}(0, \sigma_\varepsilon^2)$, a GP is typically denoted as $\boldsymbol{f} \sim \mathcal{GP}(m_{\boldsymbol{f}}(\boldsymbol{x}), k_{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{x}'))$. The kernel function is parameterized with some hyperparameters $\boldsymbol{\theta}$, and it is common practice to standardize the data and set the mean function equal to zero, thereby making the GP fully determined by the kernel. A popular choice for the kernel function is the radial basis function (RBF) with automatic relevance determination (ARD). Let $\boldsymbol{\ell}$ be a vector of length scales $\ell_1, ..., \ell_d$, then the RBF-ARD kernel is given by

$$k_{\text{RBF-ARD}}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-||\boldsymbol{x} - \boldsymbol{x}'||^2 / 2\boldsymbol{\ell}^2\right) . \tag{1}$$

The variance of the Gaussian noise is added to the kernel by an indicator variable, i.e., $\sigma_\varepsilon^2 \mathbb{I}_{\{\boldsymbol{x}=\boldsymbol{x}'\}}$, and then the hyperparameters are defined as $\boldsymbol{\theta} = (\boldsymbol{\ell}, \sigma_\varepsilon^2)$. The hyperparameters are generally optimized by maximum likelihood estimation or maximum a posteriori. Given the hyperparameters $\boldsymbol{\theta}$, the predictive posterior for the unknown test inputs $X^\star$ is given by $p(\boldsymbol{f}|\boldsymbol{\theta}, \mathcal{D}, X^\star) = \mathcal{N}(\boldsymbol{\mu}^\star, \Sigma^\star)$ with

$$\boldsymbol{\mu}^\star = K_{\boldsymbol{\theta}}^\star (K_{\boldsymbol{\theta}} + \sigma_\varepsilon^2 \mathbb{I})^{-1} \boldsymbol{y} , \tag{2}$$

$$\Sigma^\star = K_{\boldsymbol{\theta}}^{\star\star} - K_{\boldsymbol{\theta}}^\star (K_{\boldsymbol{\theta}} + \sigma_\varepsilon^2 \mathbb{I})^{-1} K_{\boldsymbol{\theta}}^{\star\top} , \tag{3}$$

where $K_{\boldsymbol{\theta}}^{\star\star}$ denotes the covariance matrix between the test inputs, and $K_{\boldsymbol{\theta}}^\star$ denotes the covariance matrix between the test inputs and training inputs.

Another approach to learning the hyperparameters is to infer the posterior over the hyperparameters in a hierarchical specification of GPs known as fully Bayesian Gaussian processes (FBGP) [11]. The FBGP extends a GP with priors over the hyperparameters $p(\boldsymbol{\theta})$, such that the joint posterior is

$$p(\boldsymbol{f}, \boldsymbol{\theta}|\boldsymbol{y}, X) \propto p(\boldsymbol{y}|\boldsymbol{f})\, p(\boldsymbol{f}|\boldsymbol{\theta}, X)\, p(\boldsymbol{\theta}) . \tag{4}$$

The hierarchical predictive posterior is a multivariate mixture of M Gaussians and can be estimated by Hamiltonian Monte Carlo (HMC) sampling as proposed by Lalchand

and Rasmussen [11]. For the test inputs $X^\star$, the predictive posterior is

$$p(\boldsymbol{y}^\star|\boldsymbol{y}) = \int p(\boldsymbol{y}^\star|\boldsymbol{y},\boldsymbol{\theta})\, p(\boldsymbol{\theta}|\boldsymbol{y})\, d\boldsymbol{\theta} \tag{5}$$

$$\approx \frac{1}{M}\sum_{j=1}^{M} p(\boldsymbol{y}^\star|\boldsymbol{y},\boldsymbol{\theta}_j), \quad \boldsymbol{\theta}_j \sim p(\boldsymbol{\theta}|\boldsymbol{y}) \tag{6}$$

$$= \frac{1}{M}\sum_{j=1}^{M} \mathcal{N}(\mu_{\boldsymbol{\theta}_j}, \Sigma_{\boldsymbol{\theta}_j}), \tag{7}$$

where $\mu_{\boldsymbol{\theta}_j}$ and $\Sigma_{\boldsymbol{\theta}_j}$ denote the predictive mean and covariance computed with the hyperparameters $\boldsymbol{\theta}_j$. The mean and variance are given by

$$\mu_{\mathrm{MGP}}(\boldsymbol{x}) = \frac{1}{M}\sum_{j=1}^{M} \mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}), \tag{8}$$

$$\sigma^2_{\mathrm{MGP}}(\boldsymbol{x}) = \frac{1}{M}\sum_{j=1}^{M} \left(\sigma^2_{\boldsymbol{\theta}_j}(\boldsymbol{x}) + (\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}) - \mu_{\mathrm{MGP}}(\boldsymbol{x}))^2\right). \tag{9}$$

When the hyperparameters are inferred with HMC, the fully Bayesian GP can be used as a Bayesian mixture of Gaussian processes (MGP). Using the HMC samples to compute the predictive posterior, the MGP has a natural weighting of GPs representing the modes in the hyperparameters' posterior [9]. This model has proven itself in active learning, showing state-of-the-art performance across six standard simulators [9]. However, the HMC sampling is computationally expensive and time-consuming, thus making it impractical to apply in active learning cases where the time available to query a new data point is low.

3.2 Active learning

The main idea of active learning is to *actively* choose a new data point to label and add to the current training data set, to iteratively improve the performance of the model [2]. For active learning surrogates, the underlying problem is very often a regression problem [6]. Additionally, the simulator provides a flexible way of generating and labeling new data points, giving rise to *population-based* active learning (in contrast to *pool-based* active learning with a predefined data set [25]).

A common distinction between active learning acquisition functions is model-based and model-free functions [26]. The former utilize information from the model and are often based on uncertainty measures [1], [27], [28], whereas the latter use no information from the model and are typically based on distance metrics in the input space [29], [30]. However, for active learning surrogates the model-based functions are superior to the model-free functions [6], where a particular powerful subgroup of the model-based functions are those within Bayesian active learning [14], [31], [32].

Bayesian active learning The most common acquisition function is the Bayesian acquisition function based on the predictive entropy [1]:

$$\textsc{Entropy}(\boldsymbol{x}) = \mathrm{H}[y|\boldsymbol{x},\mathcal{D}]\,. \tag{10}$$

For a GP, the predictive entropy is proportional to the predictive uncertainty $\sigma^2(\boldsymbol{x})$ (cf. appendix F), meaning that ENTROPY queries the data point, where the model is most uncertain. The predictive uncertainty has become the standard acquisition function to use in active learning surrogate modeling [6] and is denoted as:

$$\textsc{Variance}(\boldsymbol{x}) = \sigma^2(\boldsymbol{x})\,. \tag{11}$$

Another common acquisition function is Bayesian Active Learning by Disagreement (BALD), which maximizes the expected decrease in the posterior entropy [31]:

$$\textsc{Bald}(\boldsymbol{x}) = \mathrm{I}[\hat{\boldsymbol{\theta}}, y|\boldsymbol{x},\mathcal{D}] \tag{12}$$

$$= \mathrm{H}[y|\boldsymbol{x},\mathcal{D}] - \mathbb{E}_{p(\hat{\boldsymbol{\theta}}|\mathcal{D})}\left[\mathrm{H}[y|\boldsymbol{x},\hat{\boldsymbol{\theta}}]\right], \tag{13}$$

where $\hat{\boldsymbol{\theta}}$ is the model's parameters. For a standard GP, $\hat{\boldsymbol{\theta}}$ corresponds to the latent functions $\boldsymbol{f}$, thus making BALD proportional to ENTROPY, cf. appendix G. The Bayesian MGP also gives rise to an acquisition function directly based on the predictive uncertainty in eq. (9):

$$\textsc{Qb-mgp}(\boldsymbol{x}) = \sigma^2_{\mathrm{MGP}}(\boldsymbol{x})\,, \tag{14}$$

and the embedded acquisition function based on only the variance of the mean functions:

$$\textsc{B-qbc}(\boldsymbol{x}) = \frac{1}{M}\sum_{j=1}^{M} (\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}) - \mu_{\mathrm{MGP}}(\boldsymbol{x}))^2\,. \tag{15}$$

4 METHOD

In this section, we describe the new approach to constructing the Mixture of Gaussian processes (MGP) and extend the current Bayesian active learning acquisition functions to the MGP.

4.1 Mixture of Gaussian processes

The fully Bayesian Gaussian process [11] is a hierarchical specification of a GP, where the hyperparameters are learned by inferring from the hyperparameters' posterior, often approximated with Hamiltonian Monte Carlo (HMC) sampling. The hierarchical predictive posterior is a multivariate mixture of M Gaussians given by eq. (5). When the hyperparameters are inferred with HMC, the fully Bayesian GP is a Bayesian mixture of Gaussian Processes (MGP) [9]. Using the HMC samples to compute the predictive posterior, the MGP has a natural weighting of GPs representing the modes in the hyperparameters' posterior. However, the HMC is computationally expensive.

To address this issue, we propose to approximate the Bayesian MGP by learning the hyperparameters of the individual GPs with a few standard GPs, thereby significantly decreasing the computational time for the active learning scheme. The construction of the MGP consists of three steps:

1) Draw M sets of hyperparameters $\boldsymbol{\theta}_j$ from a prior. Each set of hyperparameters represents a GP.
2) For each set of hyperparameters, update the set using a gradient descent method with n_s learning steps.
3) Computing a weighting of each model such that the models that perform the best will have a larger impact in the predictive posterior.

In the following, we detail the three steps.

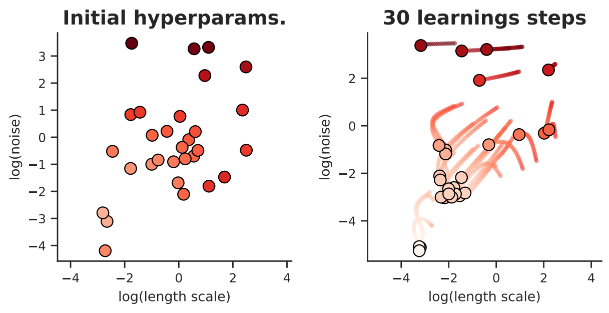

Fig. 1. Illustration of step 1 and 2. **Left**: 30 initial sets of hyperparameters sampled from $\mathcal{N}(0,3)$ in log space. **Right**: The same sets after 30 learning steps. The colors denote the log-likelihood of the models: the brighter, the better.

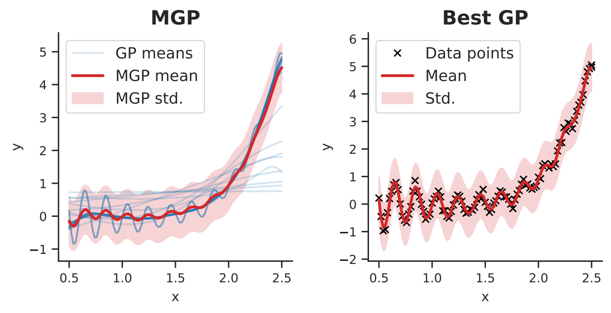

Fig. 2. The MGP and the best GP for **Gramacy1d** with 100 random data points. **Left**: The MGP with the 30 models from Figure 1 with 30 learning steps. **Rigtht**: The single GP in the MGP with the highest likelihood.

Step 1: We need a prior over the hyperparameters $p(\boldsymbol{\theta})$, and the better the prior is, the better each individual model will be. However, in general, it is difficult to choose a strong prior, so we adopt the uninformative prior $\mathcal{N}(0,3)$ in log space. The hyperparameter M specifies how many models there are in the MGP and represents a trade-off between predictive performance and resources: with too few models there is no effect of using the MGP compared to a standard GP, while with too many models there is a risk of spending unnecessary resources.

Step 2: We fine-tune the hyperparameters of the GPs for n_s gradient steps to get better individual models for the MGP. The hyperparameter n_s represents an implicit measure of the diversity in the individual models and a trade-off between predictive performance and resources. If n_s is low, the individual models will be as diverse as the samples from the prior but likely also have low performance. Conversely, if n_s is high, there is a high chance that the models will converge to the same mode(s), thus leading to reduced information about the posterior of the hyperparameters. To properly leverage the power of the MGP, the n_s should be chosen such that the MGP consists of diverse and good-performing models. Overall, the second step can be seen as a "recovery step" from the bad initialization of the hyperparameters due to the weak prior used in step 1. In Figure 1, we illustrate steps 1 and 2 with the initialization of $M = 30$ sets of hyperparameters and the corresponding updated sets after $n_s = 30$ iterations.

Step 3: We compute a weight w_j for each GP based on its marginal likelihood on the training data. Given that the prior and the evidence are the same for all the models, the posterior is proportional to the likelihood. We therefore let each weight w_j be the corresponding likelihood and then normalize all weights to sum to 1, i.e., $\sum_j^M w_j = 1$.

Finally, the predictive posterior of the approximate MGP is given as

$$p\left(\boldsymbol{y}^{\star}|\boldsymbol{y}\right) \approx \frac{1}{M}\sum_{j=1}^{M} w_j\,\mathcal{N}(\mu_{\boldsymbol{\theta}_j^{(n_s)}}, \Sigma_{\boldsymbol{\theta}_j^{(n_s)}})\,, \tag{16}$$

where $\boldsymbol{\theta}_j^{(n_s)}$ is the hyperparameters after n_s learning steps.

For the active learning acquisition function, we use the predictive posterior above, but since the posterior is of limited use when it is multimodal, we use the best individual GP for the actual predictions. In Figure 2, we illustrate the MGP: to the left, we show the mean functions of 30 models together with the mean and the standard deviation of the MGP given by eq. (8) and (9) respectively, to the right, we show the best individual GP.

4.2 Bayesian active learning for MGPs

We extend the current Bayesian active learning acquisition functions to apply to the MGPs, starting with predictive entropy, cf. (10). Exploiting the fact that the predictive entropy is proportional to the predictive variance for a single GP, we can apply VARIANCE to the best sub-GP in the MGP. In this way, we see the MGP as an optimizer with multiple initializations. However, we find that this strategy leads to non-optimal performance compared to using the information from all the GPs (see appendix D).

It is more complicated to use ENTROPY together with an MGP because there is no closed-form solution. The entropy is defined as

$$\mathrm{H}[\boldsymbol{x}] = \mathbb{E}[-\log(p(\boldsymbol{x}))] = -\int p(\boldsymbol{x})\log p(\boldsymbol{x})\,, \tag{17}$$

where, for the MGP, $p(\boldsymbol{x})$ is

$$p(\boldsymbol{x}) = \frac{1}{M}\sum_{j=1}^{M} w_j\,\mathcal{N}(\mu_{\boldsymbol{\theta}_j^{(n_s)}}(\boldsymbol{x}), \Sigma_{\boldsymbol{\theta}_j^{(n_s)}}(\boldsymbol{x}))\ . \tag{18}$$

There is no exact solution for evaluating the log of a sum of exponential functions, and thus the integral must be estimated by numerical methods [33]. BALD also involves the entropy and is equally mathematically complicated to evaluate. Since we are interested in both minimizing the predictive uncertainty and learning the hyperparameters, we let both $\boldsymbol{f}$ and $\boldsymbol{\theta}$ be the main parameters in BALD. Then BALD is

$$\mathrm{I}[\boldsymbol{f},\boldsymbol{\theta},y\,|\,\boldsymbol{x},\mathcal{D}] = \mathrm{H}\left[\mathbb{E}_{p(\boldsymbol{f},\boldsymbol{\theta}|\mathcal{D})}[y\,|\,\boldsymbol{x},\mathcal{D},\boldsymbol{f},\boldsymbol{\theta}]\right] \tag{19}$$

$$-\,\mathbb{E}_{p(\boldsymbol{f},\boldsymbol{\theta}|\mathcal{D})}\left[\mathrm{H}[y\,|\,\boldsymbol{x},\boldsymbol{f},\boldsymbol{\theta}]\right]\ . \tag{20}$$

In the first term, $\boldsymbol{f}$ is integrated out such that the predictive posterior $p(y|\boldsymbol{x})$ now corresponds to the standard GP predictive posterior, i.e., $\mathbb{E}_{p(\boldsymbol{f}|\mathcal{D})}[y\,|\,\boldsymbol{x},\mathcal{D},\boldsymbol{f}] = p(y|\boldsymbol{x})$. Regarding the entropy in the second term, we know $\boldsymbol{f}$, and thus the

TABLE 1
Simulators used in the experiments.

Simulator	d	σ_ε^2	Input space	Iterations
Gramacy1d	1	0.100	$[0.5, 2.5]$	90
Hidgon	1	0.100	$[0, 20]$	60
Gramacy2d	2	0.050	$[-2, 6]^2$	40
Branin	2	11.320	$[-5, 10] \times [0, 15]$	40
Ishigami	3	0.187	$[-\pi, \pi]^3$	200
Hartmann	6	0.019	$[0, 1]^6$	60

only entropy comes from σ_ε^2 (see appendix G), thus leading to BALD being equal to

$$\mathrm{I}[\boldsymbol{f}, \boldsymbol{\theta}, y \mid \boldsymbol{x}, \mathcal{D}] = \mathrm{H}\left[\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[y \mid \boldsymbol{x}, \mathcal{D}, \boldsymbol{\theta}]\right] - \sum_{j}^{M} w_j \sigma_{\varepsilon,j}^2 \ . \tag{21}$$

For the expectation, we use the approximation given by the MGP eq. (16), such that the first term is equal to ENTROPY. Furthermore, since the noise is independent of the input $\boldsymbol{x}$, BALD is, in fact, proportional to ENTROPY.

The two acquisition functions QB-MGP and B-QBC are adapted to the MGP by using the weights w_j of the individual GPs:

$$\text{B-QBC}(\boldsymbol{x}) = \sum_{j=1}^{M} w_j (\mu_{\boldsymbol{\theta}_j}(\boldsymbol{x}) - \mu_{\text{MGP}}(\boldsymbol{x}))^2 \ , \tag{22}$$

$$\text{QB-MGP}(\boldsymbol{x}) = \text{B-QBC}(\boldsymbol{x}) + \sum_{j=1}^{M} w_j \sigma_{\boldsymbol{\theta}_j}^2(\boldsymbol{x}) \ . \tag{23}$$

5 EXPERIMENTS

In this section, we describe the simulators and baselines that we use to evaluate the MGP. Then we detail the experimental setup and how we evaluate the models. Afterward, we present the results, covering the synthetic simulators and a real-world scenario, finishing with an experiment on Bayesian optimization.

Simulators We evaluate the model performance on six standard simulators [9]. The simulator **Gramacy1d** [34] is a 1d simulator with a periodic signal that is hard to reveal with relatively few data points, unless the data points are queried cleverly to separate signal from noise. The simulators **Higdon** (1d) and **Gramacy2d** (2d) [35] have linear and non-linear regions, which can be hard to model for a GP since the linear regions are best modeled with a long length scale, whereas the non-linear regions are better modeled with a short length scale. Then, we compare the performance on the smooth 2d simulator **Branin** [36], which is in contrast to the highly non-linear 3d simulator **Ishigami** [37]. Lastly, we test it on the 6d simulator **Hartmann** [38], a simulator with six local minima and one global optimum, to show the model's performance in higher dimensions. In Table 1, we give an overview of the simulators.

Acquisition functions We evaluate the performance of our model and its applicability to active learning using the three acquisition functions B-QBC, QB-MGP, and BALD. We benchmark the setups against the following baselines: a Gaussian process with VARIANCE, which is the most common pair of a surrogate model and acquisition function in the literature [6], and the fully Bayesian Gaussian process together with both B-QBC, QB-MGP, and BALD. We also use the three models with a RANDOM acquisition function, which randomly selects the next data point. The entropy in BALD is estimated with the numerical integrator based on quadrature `scipy.integrate.quad` using a relative error tolerance at 1 [39].

Models All models are based on a GP with a zero-mean function and an RBF-ARD kernel, where all the hyperparameters have an uninformative $\mathcal{N}(0, 3)$ prior in log space. The hyperparameters of the standard GP and the mixture of GPs are optimized based on the maximum a posterior (MAP), using the stochastic descent variant Adam [40] with a learning rate of 0.1. The GP is optimized for 300 iterations with early stopping if the performance does not improve for 15 iterations. The inference in the FBGP uses five chains, 300 samples, where the first 200 are disregarded as a warm-up period, and done by using NUTS [41] with slice sampling [42] implemented in Pyro [43]. The inference gives 1500 samples, however, to speed up the acquisition function, we afterward apply thinning and only use every 10th sample. For the two hyperparameters of the MGP, the number of models M and learning steps n_s, we performed a grid search with values from 10 to 80 with an interval of 10. To stabilize the performance of the MGP across the active learning iterations, one of M models is the mean of the prior, i.e., both hyperparameters are set to zero in the log space, and another of the M models is the best set of hyperparameters from the previous iteration. We evaluated the performance of each model by comparing the acquisition to a GP with RANDOM. In appendix B, we show the results together with a table of the hyperparameters used for each simulator. For a specific simulator, the hyperparameters were chosen using a cross-validation setup, i.e., based on the performance across the other simulators.

Experimental setup We follow the experimental setup of [9]. The active learning experiments are initialized with three data points from maxi-min Latin Hypercube Sampling. In each iteration, the inputs are rescaled to the unit cube $[0, 1]^d$, and the outputs are standardized to have zero mean and unit variance. A single data point is queried from the unlabeled pool U, consisting of a random subset of 10,000 data points (100 data points for 1-dimensional problems). We run 30 repetitions of each combination of a model and acquisition function. All the models are implemented in GPyTorch [44].

Evaluation The predictive performance is evaluated based on the negative log probability (NLP) of the targets under the model:

$$-\log p(y^\star|\mathcal{D}, \boldsymbol{x}^\star) = \frac{1}{2}\log(2\pi\hat{\sigma}^2) + \frac{(y^\star - \hat{y})^2}{2\hat{\sigma}^2} \ , \tag{24}$$

where the test data $(X^\star, \boldsymbol{y}^\star)$ consists of 1000 random data points. We evaluate the active learning performance by inspecting the performance curves as well as using the

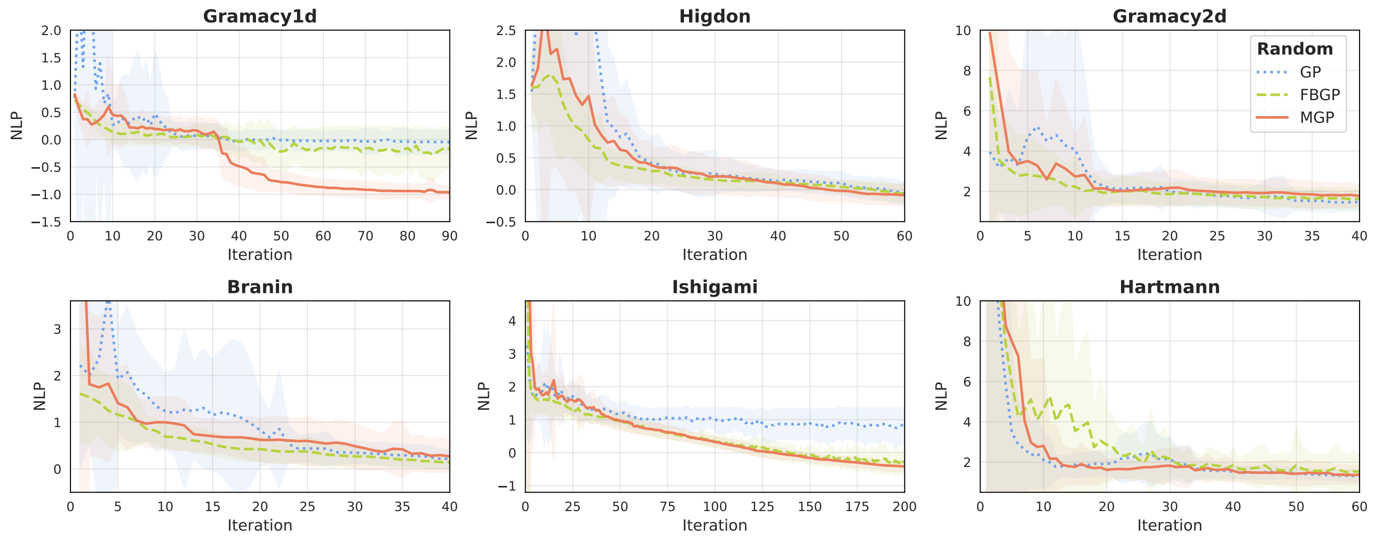

Fig. 3. Active learning curves for the three models using the acquisition function RANDOM. The curves shows the mean negative log probability (NLP) of 30 repetitions as a function of number of active learning iterations. The shaded areas indicate the range of $\mu \pm \sigma$.

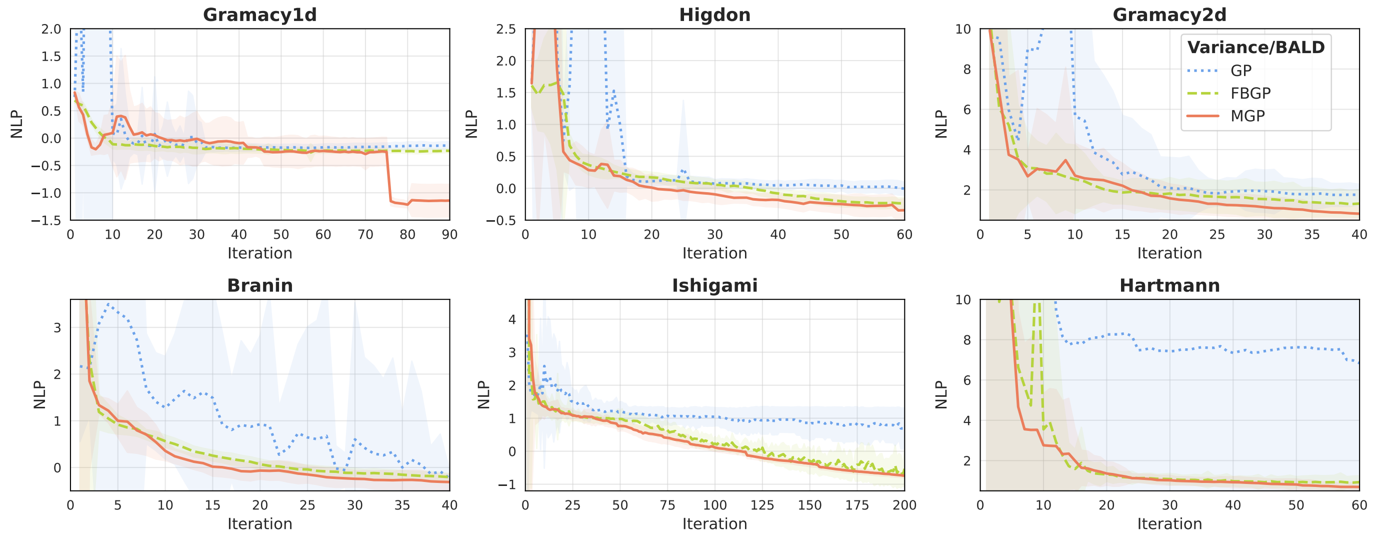

Fig. 4. Active learning curves for the three models using the acquisition function VARIANCE and BALD. The curves shows the mean negative log probability (NLP) of 30 repetitions as a function of number of active learning iterations. The shaded areas indicate the range of $\mu \pm \sigma$.

relative decrease in area under the curve (RD-AUC) [9]. We compute the RD-AUC based on the NLP and relative to the baseline acquisition function RANDOM using the MGP:

$$\text{RD-AUC}(x) = \frac{\text{AUC}_{\text{RANDOM}} - \text{AUC}_{x}}{\text{AUC}_{\text{RANDOM}}}, \quad (25)$$

where x is the acquisition function to be evaluated. The AUC is dependent on the lower bound of the metric, but since the NLP has no lower bound, we create an artificial lower bound by using the lowest NLP across all the experiments obtained for each simulator.

The computational time of the active learning setup is also a central metric. In the experiments, we measure the overall runtime of the active learning iterations, where the time difference between the models and acquisition functions will come from the time it takes to train the model and apply the acquisition function.

5.1 Results on synthetic simulators

We benchmark the performance of the model and acquisition functions on six standard simulators by a visual and quantitative analysis of the active learning loss curves. We compare the predictive performance alongside the runtime.

Predictive performance We investigate the performance of the MGP in two steps. First, we examine how the MGP works as an optimizer, compared to maximum a posteriori estimation for a single GP and HMC sampling for an FBGP, by using the three models together with RANDOM. In Figure 3, we see that all the models converge to a similar NLP, except for Gramacy1d, where both the GP and the FBGP do not converge, and Ishigami, where the GP does not converge. Additionally, the GP is more unstable in the first iterations, where only little data is available. In summary, we can observe that the MGP is the best optimizer across

TABLE 2
The relative decrease in the area under the active learning curves (RD-AUC) based on the negative log probability (NLP) compared to the active learning curve of the baseline acquisition function RANDOM with MGP.

MODEL	ACQUISITION	GRAMACY1D	HIGDON	GRAMACY2D	BRANIN	ISHIGAMI	HARTMANN
NLP: RELATIVE DECREASE IN AUC (%)							
GP	RANDOM	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	**0.0 ±0.0**
	VARIANCE	-9.7 ±5.1	-24.5 ±28.3	-80.6 ±47.0	-5.2 ±12.6	-0.6 ±3.1	-400.6 ±108.0
FBGP	RANDOM	14.0 ±2.7	47.3 ±6.3	19.0 ±9.5	27.5 ±4.5	28.8 ±1.9	-57.0 ±30.9
	BALD	21.4 ±2.0	58.6 ±5.5	15.6 ±8.8	45.7 ±3.9	35.0 ±1.6	-17.4 ±26.1
	B-QBC	36.4 ±1.9	**63.4 ±4.7**	0.9 ±14.4	53.3 ±2.5	33.9 ±5.0	-53.3 ±40.0
	QB-MGP	35.0 ±1.9	58.7 ±5.6	21.8 ±10.2	50.6 ±2.9	27.1 ±2.5	-110.5 ±58.0
MGP	RANDOM	35.7 ±2.3	40.4 ±8.9	1.2 ±12.3	13.8 ±6.7	27.2 ±2.0	-18.4 ±25.9
	BALD	28.0 ±3.2	57.2 ±6.2	24.1 ±12.6	**54.4 ±2.1**	**40.2 ±1.6**	-2.4 ±20.1
	B-QBC	**50.7 ±2.3**	54.0 ±6.7	18.8 ±11.0	29.2 ±12.8	26.7 ±2.9	-26.7 ±35.6
	QB-MGP	45.6 ±3.1	58.3 ±5.6	**25.7 ±8.0**	47.0 ±10.0	24.4 ±7.4	-16.9 ±18.7

TABLE 3
The average runtime (minutes) for each pair of model and acquisition function. The time is computed for the full active learning setup.

MODEL	ACQUISITION	GRAMACY1D	HIGDON	GRAMACY2D	BRANIN	ISHIGAMI	HARTMANN
RUNTIME (MIN)							
GP	RANDOM	**0.7 ±0.0**	**0.5 ±0.0**	**0.5 ±0.0**	**0.4 ±0.0**	2.7 ±0.1	0.8 ±0.0
	VARIANCE	**0.7 ±0.0**	**0.5 ±0.0**	0.6 ±0.0	**0.4 ±0.0**	**2.6 ±0.1**	**0.7 ±0.0**
FBGP	RANDOM	50.1 ±0.2	34.6 ±0.1	35.2 ±0.2	25.5 ±0.1	210.2 ±0.4	77.0 ±0.3
	BALD	57.2 ±0.4	41.4 ±0.4	37.0 ±0.4	32.1 ±0.3	214.3 ±0.4	74.5 ±0.5
	B-QBC	56.3 ±0.2	34.5 ±0.1	28.8 ±0.1	25.2 ±0.1	244.4 ±0.6	59.2 ±0.3
	QB-MGP	53.7 ±0.2	34.6 ±0.1	33.5 ±0.2	26.1 ±0.1	230.4 ±0.5	60.5 ±0.2
MGP	RANDOM	2.0 ±0.0	0.7 ±0.0	0.9 ±0.0	1.4 ±0.1	27.4 ±0.1	1.7 ±0.0
	BALD	1.9 ±0.0	1.5 ±0.0	2.2 ±0.0	2.4 ±0.0	31.0 ±0.1	2.9 ±0.1
	B-QBC	3.4 ±0.0	1.1 ±0.0	1.6 ±0.0	1.9 ±0.0	34.9 ±0.1	0.8 ±0.0
	QB-MGP	1.2 ±0.0	0.8 ±0.0	0.3 ±0.0	1.8 ±0.0	27.3 ±0.1	4.2 ±0.1

these six simulators. It is not surprising that the MGP is better than the GP since it has long been known that multiple initializations of the hyperparameters are beneficial [45]. On the other hand, it is surprising that the computationally cheaper and non-Bayesian model MGP is slightly better than the FBGP. Overall, these results suggest that finding the best hyperparameters through multiple initializations is a good approach. However, in terms of active learning, useful information is thrown away if only the best hyperparameters are used. In appendix D, we show that active learning with a GP is better when the GP is optimized with the MGP, although not as good as using the "full" MGP.

Second, we examine the model's performance using the active learning acquisition function BALD (proportional to VARIANCE for the GP). The acquisition function BALD is the best-performing criteria for both the FBGP and MGP, and thus we only show the active learning curves for this criteria. The active learning curves for all the combinations of models and acquisition functions can be found in section A of the Appendix. In Figure 4, we see that the GP is the worst-performing model and that FBGP is significantly better but is still outperformed by the MGP, which is better across all simulators.

In Table 2, we summarize the performance using the relative decrease in the area under the curve (RD-AUC). For five out of the six simulators, the MGP achieves the highest RD-AUC, and in two out of six the MGP with BALD is the best. If we compare the performance of BALD with the MGP to the same acquisition function with FBGP, we see that the MGP is best for all, except on Higdon where the two setups achieve on-par performance. Note how the RD-AUC on Hartmann across the acquisition functions is negative due to the instability in the models in the first iterations. For completeness, we also show the NLP at the last iteration in Table 6 in appendix C.

Runtime In Table 3, we compare the runtime of the models for different acquisition functions. Unsurprisingly, the standard GP is the fastest, with an average of 50 seconds, whereas the FBGP uses between 25 to 244 minutes, due to the computationally expensive HMC. However, the MGP is significantly faster than the FBGP, and the three acquisition functions RANDOM, B-QBC, and QB-MGP are between 7 and 100 times faster with the MGP compared to the respective combination with the FBGP. Specifically, the MGP with BALD is significantly faster than the fully Bayesian GP as it is 6.9 - 30 times faster than the FBGP with BALD.

5.2 Application in air traffic management

We now investigate the performance of the MGP on the simulator *Mercury* [12], which models the air transportation system in Europe. Mercury is event-driven and simulates individual aircraft movements and passenger processes, including passenger connections and delays, alongside modeling different types of agents, such as airlines and airports, and their decision-making based on detailed cost functions. The

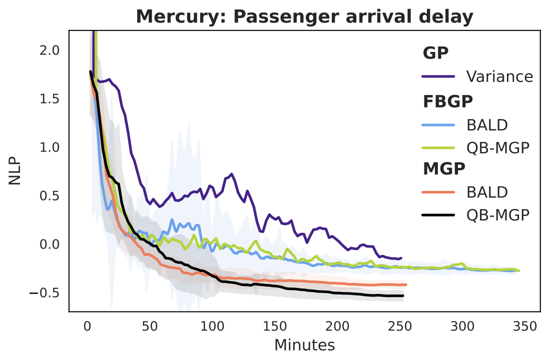

Fig. 5. The mean negative log probability (NLP) of 30 repetitions as a function of the number of minutes (incl. active learning and simulation time). The shaded areas indicate the range of $\mu \pm \sigma$.

simulator has seven inputs and is relatively fast, as a single simulation takes, on average, 2.5 minutes. The simulator is used as a case study for different surrogate modeling techniques, investigating different simulation outputs [46], [47]. We use the simulation output *passenger arrival delay* to test the performance of the MGP. Furthermore, we use the same experimental setup from the other experiments, except starting with ten simulations instead of three, following the setup of previous studies using Mercury [46], [47].

In Figure 5, we see that the MPG with BALD is the best setup for the first 100 minutes. Afterward, the MGP with QB-MGP is catching up and converging to a lower NLP. Overall, the MGP outperforms both the standard GP and the computationally-expensive FBGP.

5.3 Mixture of Gaussian processes for Bayesian optimization

We conclude our empirical analysis with a small experiment on the usability of the MGPs within the setting of Bayesian optimization, where the MGPs are shown to outperform the GPs and FBGPs. We use the acquisition function GP Upper Confidence Bound (GP-UCB) [48] using BALD as the measure for variance. For the MGP, we use the hyperparameters estimated for BALD across the six simulators used in the active learning experiments, cf. Table 5. We benchmark the models on six synthetic, stochastic simulators (specific settings detailed in appendix E) and present how the best *true* function value observed, computed with the corresponding noise-free simulator evaluated at $\arg\min_x \mu(x)$, evolves over the iterations in Figure 6. The MGP is among the best for five out of the six simulators and significantly outperforms the GP and FBGP on the three Hartmann simulators. In Table 4, we evaluate the relative decrease in the areas under the curve (RD-AUC), the minimum in the last iteration as well as the running time. The MGP yields the best RD-AUC on four and the lowest minimum on five of the six simulators. Furthermore, it is between 2.7-8.2 times faster than the FBGP, although 7.3-26.2 slower than the standard GP.

6 Conclusion

We proposed a new model type for active learning called 'Mixture of Gaussian processes' (MGP) and showed their applicability within this framework. The model is able to take advantage of the 'fully' Bayesian active learning in a similar way as the fully Bayesian Gaussian processes, however without the computationally expensive Monte Carlo sampling of the hyperparameters of the GPs.

We empirically showed that MGPs with BALD query new data points more efficiently than the regular and fully Bayesian GPs. Across six classic simulators, we showed that MGPs with BALD, on average, yielded the highest decrease in area under the curve and achieved the lowest negative log probability, while being 6.9 - 30 times faster than the FBGP with BALD. Additionally to the evaluation of the synthetic functions, we showed that the MGP is outperforming the standard GP and the FBGP on a real-world simulator from the air traffic management domain. Lastly, we show that the MGP is applicable within the setting of Bayesian optimization as well, where it finds the lowest minimum on five out of the six simulators.

Acknowledgment

This work was supported by NOSTROMO, framed in the scope of the SESAR 2020 Exploratory Research topic SESAR-ER4-26-2019, funded by SESAR Joint Undertaking through the European Union's Horizon 2020 research and innovation programme under grant agreement No 892517.

Authors Contribution

The authors confirm their contribution to the paper as follows: formulation of mixture of Gaussian processes and code: C. Riis; derivation of acquisition functions: C. Riis and F. Rodrigues; study conception and design: all; draft manuscript preparation: C. Riis. All authors reviewed the results and approved the final version of the manuscript.

References

[1] D. J. C. MacKay, "Information-Based Objective Functions for Active Data Selection," *Neural Computation*, vol. 4, no. 4, pp. 590–604, 1992.
[2] B. Settles, *Active learning literature survey*. University of Wisconsin-Madison Department of Computer Sciences, 2009.
[3] A. Sauer, R. B. Gramacy, and D. Higdon, "Active learning for deep gaussian process surrogates," *Technometrics*, vol. 0, no. 0, pp. 1–15, 2022. [Online]. Available: https://doi.org/10.1080/00401706.2021.2008505
[4] F. Antunes, M. Amorim, F. C. Pereira, and B. Ribeiro, "Active learning metamodeling for policy analysis: Application to an emergency medical service simulator," *Simulation Modelling Practice and Theory*, vol. 97, p. 101947, 2019.
[5] R. Sánchez-Cauce, C. Riis, F. Antunes, D. Mocholí, O. G. C. Ros, F. C. Pereira, R. Herranz, and C. M. L. Azevedo, "Active learning metamodelling for r-nest," in *Proceedings of the 11th SESAR Innovation Days, 2021*, 2022.
[6] R. B. Gramacy, *Surrogates*. Chapman and Hall/CRC, mar 2020. [Online]. Available: https://www.taylorfrancis.com/books/9781000766202
[7] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.
[8] A. Krause and C. Guestrin, "Nonmyopic active learning of gaussian processes: an exploration-exploitation approach," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 449–456.
[9] C. Riis, F. Antunes, F. Hüttel, C. Lima Azevedo, and F. Pereira, "Bayesian active learning with fully bayesian gaussian processes," *Advances in Neural Information Processing Systems*, vol. 35, pp. 12 141–12 153, 2022.

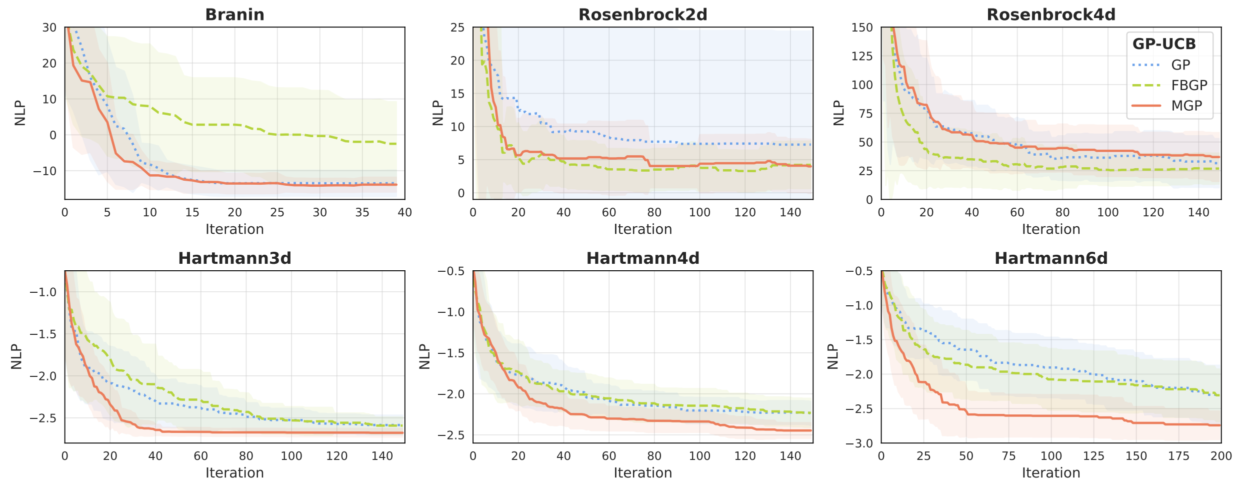

Fig. 6. The mean minimum function value of 30 repetitions as a function of the number of Bayesian optimization iterations. The shaded areas indicate the range of $\mu \pm \sigma$.

TABLE 4
The relative decrease in the area under the active learning curves (RD-AUC) based on the negative log probability (NLP) compared to the active learning curve of the baseline acquisition function RANDOM with MGP.

MODEL	BRANIN	ROSENBROCK2D	ROSENBROCK4D	HARTMANN3D	HARTMANN4D	HARTMANN6D
MIN y: RELATIVE DECREASE IN AUC (%)						
GP	-0.0 ±8.8	0.0 ±17.8	0.0 ±4.6	0.0 ±3.7	0.0 ±2.7	0.0 ±2.6
FBGP	-194.9 ±24.9	**80.7 ±2.6**	**22.8 ±4.5**	-5.9 ±5.2	-14.2 ±3.2	11.9 ±3.2
MGP	**16.9 ±6.2**	58.3 ±5.6	-0.9 ±4.2	**56.6 ±1.4**	**29.0 ±2.7**	**51.5 ±1.8**
MIN y: LAST ITERATION						
GP	-13.49 ±3.7	7.28 ±17.14	**31.58 ±21.4**	-2.59 ±0.12	-2.24 ±0.15	-2.30 ±0.43
FBGP	-2.5 ±11.72	4.21 ±3.59	26.88 ±13.5	-2.59 ±0.10	-2.23 ±0.18	-2.31 ±0.38
MGP	**-13.88 ±2.07**	**3.99 ±4.10**	36.92 ±21.3	**-2.68 ±0.05**	**-2.45 ±0.10**	**-2.74 ±0.21**
RUNTIME (MIN)						
GP	**0.3 ±0.0**	**1.3 ±0.1**	**2.6 ±0.1**	**1.4 ±0.1**	**1.3 ±0.1**	**3.0 ±0.1**
FBGP	23.9 ±0.1	95.0 ±0.2	118.9 ±0.3	107.9 ±0.2	119.7 ±0.3	251.1 ±0.8
MGP	2.9 ±0.0	34.1 ±0.4	19.0 ±0.3	21.7 ±0.3	18.9 ±0.3	43.4 ±0.2

[10] Y. Yao, A. Vehtari, and A. Gelman, "Stacking for non-mixing bayesian computations: The curse and blessing of multimodal posteriors," *Journal of Machine Learning Research*, vol. 23, no. 79, pp. 1–45, 2022. [Online]. Available: http://jmlr.org/papers/v23/20-1426
[11] V. Lalchand and C. E. Rasmussen, "Approximate inference for fully bayesian gaussian process regression," in *Symposium on Advances in Approximate Bayesian Inference*. PMLR, 2020, pp. 1–12.
[12] L. Delgado, G. Gurtner, P. Mazzarisi, S. Zaoli, D. Valput, A. Cook, and F. Lillo, "Network-wide assessment of atm mechanisms using an agent-based model," *Journal of Air Transport Management*, vol. 95, p. 102108, 2021.
[13] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*. Springer, 2019.
[14] C. Guestrin, A. Krause, and A. P. Singh, "Near-optimal sensor placements in Gaussian processes," in *Proceedings of the 22nd international conference on Machine learning - ICML '05*, vol. 1. New York, New York, USA: ACM Press, 2005, pp. 265–272. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1102351.1102385
[15] B. Zhang, D. A. Cole, and R. B. Gramacy, "Distance-distributed design for gaussian process surrogates," *Technometrics*, vol. 63, no. 1, pp. 40–52, 2021.
[16] V. Tresp, "Mixtures of gaussian processes," *Advances in neural information processing systems*, vol. 13, 2000.
[17] C. Rasmussen and Z. Ghahramani, "Infinite mixtures of gaussian process experts," in *Advances in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., vol. 14. MIT Press, 2001. [Online]. Available: https://proceedings.neurips.cc/paper/2001/file/9afefc52942cb83c7c1f14b2139b09ba-Paper.pdf
[18] C. Yuan and C. Neubauer, "Variational mixture of gaussian process experts," in *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., vol. 21. Curran Associates, Inc., 2008. [Online]. Available: https://proceedings.neurips.cc/paper/2008/file/f4b9ec30ad9f68f89b29639786cb62ef-Paper.pdf
[19] M. Lázaro-Gredilla, S. Van Vaerenbergh, and N. D. Lawrence, "Overlapping mixtures of gaussian processes for the data association problem," *Pattern recognition*, vol. 45, no. 4, pp. 1386–1395, 2012.
[20] M. Trapp, R. Peharz, F. Pernkopf, and C. E. Rasmussen, "Deep structured mixtures of gaussian processes," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 2251–2261. [Online]. Available: https://proceedings.mlr.press/v108/trapp20a.html
[21] J. Zhao, S. Sun, H. Wang, and Z. Cao, "Promoting active learning with mixtures of Gaussian processes," *Knowledge-Based Systems*, vol. 188, p. 105044, 2020. [Online]. Available: https://doi.org/10.1016/j.knosys.2019.105044
[22] Q. Lu, G. Karanikolas, Y. Shen, and G. B. Giannakis, "Ensemble gaussian processes with spectral features for online interactive learning with scalability," in *Proceedings of the Twenty Third International Conference on Artificial*

Intelligence and Statistics, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 1910–1920. [Online]. Available: https://proceedings.mlr.press/v108/lu20d.html

[23] Q. Lu, G. V. Karanikolas, and G. B. Giannakis, "Incremental ensemble gaussian processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[24] Q. Lu, K. D. Polyzos, B. Li, and G. B. Giannakis, "Surrogate modeling for bayesian optimization beyond a single gaussian process," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[25] M. Sugiyama and S. Nakajima, "Pool-based active learning in approximate linear regression," *Machine Learning*, vol. 75, no. 3, pp. 249–274, 2009.

[26] J. O'Neill, S. Jane Delany, and B. MacNamee, "Model-Free and Model-Based Active Learning for Regression," in *Advances in computational intelligence systems*. Springer, 2017, pp. 375–386. [Online]. Available: http://link.springer.com/10.1007/978-3-319-46562-3{_}24

[27] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine Learning*, vol. 15, no. 2, pp. 201–221, 1994.

[28] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, "Gaussian process regression: Active data selection and test point rejection," in *Mustererkennung 2000: 22. DAGM-Symposium. Kiel, 13.–15. September 2000*. Springer, 2000, pp. 27–34.

[29] D. Wu, C.-T. Lin, and J. Huang, "Active learning for regression using greedy sampling," *Information Sciences*, vol. 474, pp. 90–105, 2019.

[30] Z. Liu and D. Wu, "Integrating informativeness, representativeness and diversity in pool-based sequential active learning for regression," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[31] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel, "Bayesian active learning for classification and preference learning," 2011. [Online]. Available: https://arxiv.org/abs/1112.5745

[32] A. Kirsch, S. Farquhar, P. Atighehchian, A. Jesson, F. Branchaud-Charron, and Y. Gal, "Stochastic batch acquisition for deep active learning," 2021. [Online]. Available: https://arxiv.org/abs/2106.12059

[33] M. F. Huber, T. Bailey, H. Durrant-Whyte, and U. D. Hanebeck, "On entropy approximation for gaussian mixture random vectors," in *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, 2008, pp. 181–188.

[34] R. B. Gramacy and H. K. Lee, "Cases for the nugget in modeling computer experiments," *Statistics and Computing*, vol. 22, 2012.

[35] ——, "Adaptive design and analysis of supercomputer experiments," *Technometrics*, vol. 51, no. 2, pp. 130–145, 2009. [Online]. Available: https://doi.org/10.1198/TECH.2009.0015

[36] A. Keane, A. Forrester, and A. Sobester, *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley & Sons, 2008.

[37] T. Ishigami and T. Homma, "An importance quantification technique in uncertainty analysis for computer models," in *[1990] Proceedings. First International Symposium on Uncertainty Modeling and Analysis*. IEEE, 1990, pp. 398–403.

[38] V. Picheny, T. Wagner, and D. Ginsbourger, "A benchmark of kriging-based infill criteria for noisy optimization," *Structural and Multidisciplinary Optimization*, vol. 48, 2013.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo." *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.

[42] R. M. Neal, "Slice sampling," *The annals of statistics*, vol. 31, no. 3, pp. 705–767, 2003.

[43] E. Bingham, J. P. Chen, M. Jankowiak, F. Obermeyer, N. Pradhan, T. Karaletsos, R. Singh, P. Szerlip, P. Horsfall, and N. D. Goodman, "Pyro: Deep universal probabilistic programming," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 973–978, 2019. [Online]. Available: https://jmlr.csail.mit.edu/papers/v20/18-403

[44] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration," *Advances in Neural Information Processing Systems*, vol. 31, pp. 7576–7586, 2018.

[45] C. Williams and C. Rasmussen, "Gaussian processes for regression," *Advances in neural information processing systems*, vol. 8, 1995.

[46] C. Riis, F. Antunes, G. Gurtner, F. C. Pereira, L. Delgado, and C. M. Lima Azevedo, "Active learning metamodels for atm simulation modeling," in *Proceedings of the 11th SESAR Innovation Days, 2021*, 2021.

[47] C. Riis, F. Antunes, T. Bolic, G. Gurtner, F. C. Pereira, and C. L. Azevedo, "Explainable metamodels for atm performance assessment," in *Proceedings of the 12th SESAR Innovation Days, 2022*, 2022.

[48] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for gaussian process optimization in the bandit setting," *IEEE transactions on information theory*, vol. 58, no. 5, pp. 3250–3265, 2012.

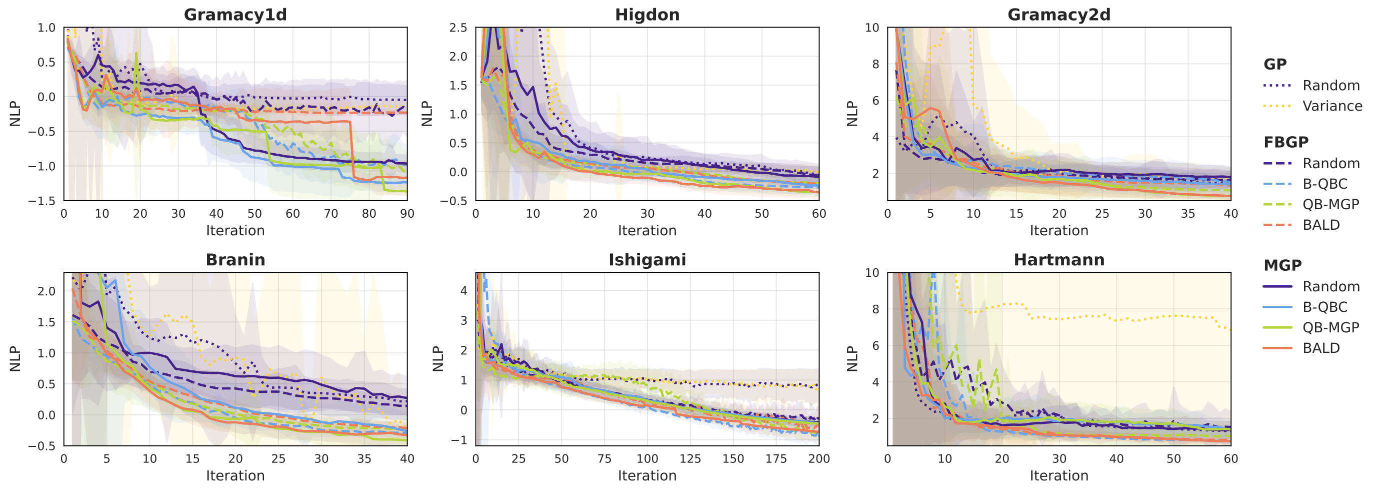

Fig. 7. The mean negative log probability (NLP) of 30 repetitions as a function of the number of active learning iterations. The shaded areas indicate the range of $\mu \pm \sigma$. Nine combinations of models and acquisition functions are evaluated and compared against MGP with RANDOM.

APPENDIX A ACTIVE LEARNING CURVES

Figure 7 shows the active learning curves for all the setups. The Mixture of Gaussian processes with the acquisition function BALD has good performance on all simulators. Only on Gramacy1d, the MGP with B-QBC and RANDOM is significantly better.

APPENDIX B GRIDSEARCH FOR NUMBER OF MODELS AND LEARNING STEPS IN THE MGP

The estimation of the best hyperparameters for each simulator is computed by taking the set of hyperparameters that, on average, achieved the highest decrease in AUC for the other simulators (using BALD), see Table 5. In other words, we do not tune the hyperparameters for the specific simulator, but instead, we mimic what to do if you have a new simulator. For new applications, we recommend using the hyperparameters given by the last row (simulator: ALL).

In Figure 8 and Figure 9, we show contour plots of the grid search together with the best set of hyperparameters based on the simulator itself (yellow cross), and the best set based on the other simulators (red cross). The latter are those we use to benchmark the performance of the model. In general, the crosses are in the same area. However, for example, for Gramacy1d, we see that the optimal hyperparameters estimated based on the other simulators are in a particularly bad location.

APPENDIX C PERFORMANCE AT THE LAST ITERATION

The NLP at the last iteration is shown in Table 6. The MGP with QB-MGP achieves the lowest NLP for four out of the six simulators, however, on average, the MGP with BALD is yielding the lowest NLP.

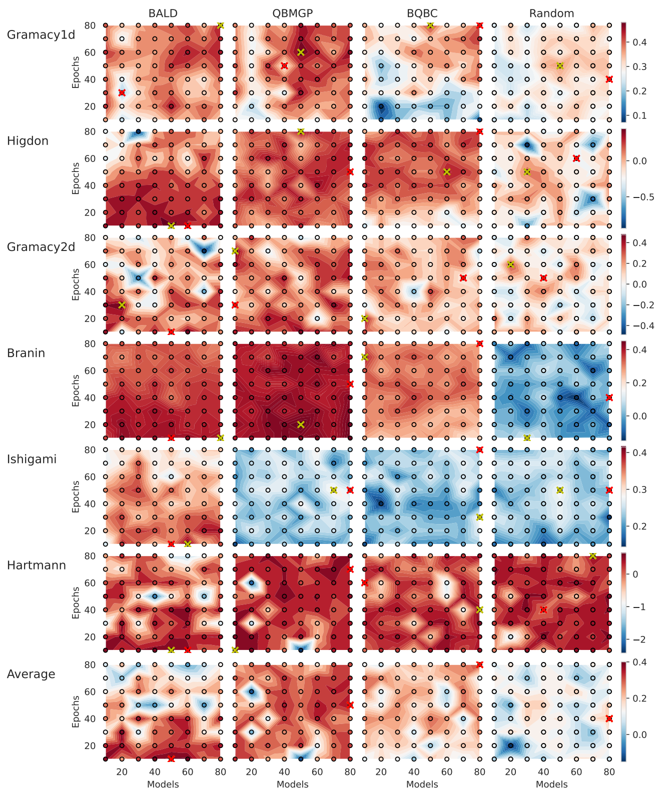

Fig. 8. Contour plots of the grid search based on relative decrease in the area under the curve (RD-AUC), the higher (more red), the better. Yellow crosses mark the best values, whereas red crosses mark the values obtained by using the optimal hyperparameters estimated by using the other simulators.

APPENDIX D THE MGP AS AN OPTIMIZER

By construction, the Mixture of Gaussian processes consists of M sub-GPs. Thus, we can use the MGP as an optimizer with M initializations, using only the best GP to acquire new data. In Table 7, we show the performance of the MGP as an optimizer (MGP-OPT), where we use VARIANCE based on

TABLE 5
The estimation of the best hyperparameters for the MGP for the different simulators.

	BALD		QB-MGP		B-QBC		RANDOM	
SIMULATOR	MODELS	EPOCHS	MODELS	EPOCHS	MODELS	EPOCHS	MODELS	EPOCHS
GRAMACY1D	20	30	40	50	80	80	80	40
HIGDON	60	10	80	50	80	80	60	60
GRAMACY2D	50	10	10	30	70	50	40	50
BRANIN	50	10	80	50	80	80	80	40
ISHIGAMI	50	10	80	50	80	80	80	50
HARTMANN	60	10	80	70	10	60	40	40
ALL	50	10	80	50	80	80	80	40

TABLE 6
The negative log probability (NLP) at the last iteration.

MODEL	ACQUISITION	GRAMACY1D	HIGDON	GRAMACY2D	BRANIN	ISHIGAMI	HARTMANN
NLP: RELATIVE DECREASE IN AUC (%)							
NLP AT LAST ITERATION							
GP	RANDOM	-0.05 ±0.21	-0.06 ±0.18	1.47 ±0.44	0.22 ±0.21	0.75 ±0.59	1.32 ±0.31
	VARIANCE	-0.14 ±0.07	-0.01 ±0.10	1.76 ±0.52	-0.23 ±0.17	0.69 ±0.62	6.84 ±6.77
FBGP	RANDOM	-0.17 ±0.41	-0.06 ±0.15	1.63 ±0.50	0.14 ±0.14	-0.30 ±0.40	1.52 ±0.85
	BALD	-0.23 ±0.04	-0.24 ±0.09	1.32 ±0.71	-0.21 ±0.08	-0.52 ±0.56	0.93 ±0.32
	B-QBC	-0.98 ±0.29	-0.27 ±0.12	1.53 ±0.78	-0.30 ±0.08	**-0.88 ±0.18**	0.75 ±0.25
	QB-MGP	-1.07 ±0.29	-0.20 ±0.09	1.06 ±0.53	-0.22 ±0.16	-0.73 ±0.32	1.03 ±0.44
MGP	RANDOM	-0.97 ±0.12	-0.09 ±0.17	1.78 ±0.54	0.27 ±0.34	-0.42 ±0.17	1.37 ±0.47
	BALD	-1.14 ±0.30	-0.34 ±0.18	0.82 ±0.35	-0.92 ±0.20	-0.74 ±0.07	**0.69 ±0.20**
	B-QBC	-1.24 ±0.14	-0.24 ±0.17	1.39 ±0.76	-0.26 ±0.29	-0.45 ±0.15	1.42 ±0.48
	QB-MGP	**-1.37 ±0.06**	**-0.36 ±0.07**	**0.76 ±0.30**	**-0.41 ±0.12**	-0.46 ±0.12	1.39 ±0.45

the GP with the highest marginal likelihood to query new data points. For easier comparison, we added the GP and the MGP results with BALD to the table. For the MGP-OPT with VARIANCE, we use the hyperparameters estimated based on the MGP with QB-MGP. On average, MGP-OPT is better than a single GP. However, the best-performing setup is the MGP using the extra information from the other GPs, i.e., MGP with BALD.

APPENDIX E
EXPERIMENTAL SETTINGS FOR BAYESIAN OPTIMIZATION

The dimensionality, noise-level, input space, and the number of iterations for the six simulators used in the Bayesian optimization experiments are listed in Table 8. Following [38], the noise-level is computed as 5% of the variance of the outputs from the noise-free simulator evaluated at 10,000 randomly sampled data points from the input space.

APPENDIX F
ENTROPY OF A GAUSSIAN DISTRIBUTION

We show that the entropy of a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ is proportional to the variance σ^2:

$$\begin{aligned}
\mathrm{H}[X] &= \mathbb{E}\left[-\ln p(X)\right] && (26)\\
&= \mathbb{E}\left[\frac{1}{2}\ln 2\pi\sigma^2 + \frac{(X-\mu)^2}{2\sigma^2}\right] && (27)\\
&= \frac{1}{2}\ln 2\pi\sigma^2 + \frac{1}{2\sigma^2}\mathbb{E}\left[(X-\mu)^2\right] && (28)\\
&= \frac{1}{2}\ln 2\pi\sigma^2 + \frac{\sigma^2}{2\sigma^2} && (29)\\
&= \frac{1}{2}\ln 2\pi e\sigma^2, \quad \text{since } \frac{1}{2} = \frac{1}{2}\ln e && (30)\\
&\propto \sigma^2 && (31)
\end{aligned}$$

APPENDIX G
GAUSSIAN PROCESSES

G.1 Predictive posterior for a GP

The predictive posterior for a GP is given as

$$p(y^\star \mid \boldsymbol{x}^\star, \mathcal{D}) = \int_{\boldsymbol{f}^\star} p(y^\star \mid \boldsymbol{f}^\star) p(\boldsymbol{f}^\star \mid \boldsymbol{x}^\star, \mathcal{D}) = \mathcal{N}(\mu^\star_{\boldsymbol{\theta}}, K^\star_{\boldsymbol{\theta}}) \tag{32}$$

where $\mu^\star_{\boldsymbol{\theta}}$ and $K^\star_{\boldsymbol{\theta}}$ is defined in eq. (2).

G.2 BALD for a GP

BALD is motivated by maximizing the decrease in the expected posterior $p(\boldsymbol{\theta}|\mathcal{D})$. But since it is often intractable to compute the entropy of the posterior, BALD is rewritten as

TABLE 7
The performance of the MGP-OPT compared to a GP and MGP.

MODEL	ACQUISITION	GRAMACY1D	HIGDON	GRAMACY2D	BRANIN	ISHIGAMI	HARTMANN
NLP: RELATIVE DECREASE IN AUC (%)							
GP	VARIANCE	-9.7 ±5.1	-24.5 ±28.3	-80.6 ±47.0	-5.2 ±12.6	-0.6 ±3.1	-400.6 ±108.0
MGP-OPT	VARIANCE	**39.1 ±2.5**	44.3 ±7.7	-43.8 ±20.1	48.1 ±3.3	29.4 ±2.4	-478.0 ±262.8
MGP	BALD	28.0 ±3.2	**57.2 ±6.2**	**24.1 ±12.6**	**54.4 ±2.1**	**40.2 ±1.6**	**-2.4 ±20.1**
NLP AT LAST ITERATION							
GP	VARIANCE	-0.14 ±0.07	-0.01 ±0.10	1.76 ±0.52	-0.23 ±0.17	0.69 ±0.62	6.84 ±6.77
MGP-OPT	VARIANCE	**-1.17 ±0.30**	-0.10 ±0.14	2.74 ±2.03	-0.26 ±0.08	-0.43 ±0.11	5.68 ±15.34
MGP	BALD	-1.14 ±0.30	**-0.34 ±0.18**	**0.82 ±0.35**	**-0.92 ±0.20**	**-0.74 ±0.07**	**0.69 ±0.20**
RUNTIME (MIN)							
GP	VARIANCE	**0.7 ±0.0**	**0.5 ±0.0**	**0.6 ±0.0**	**0.4 ±0.0**	**2.6 ±0.1**	**0.7 ±0.0**
MGP-OPT	VARIANCE	5.5 ±0.2	3.1 ±0.1	4.6 ±0.1	4.3 ±0.1	24.0 ±0.3	7.2 ±0.2
MGP	BALD	1.9 ±0.0	1.5 ±0.0	2.2 ±0.0	2.4 ±0.0	31.0 ±0.1	2.9 ±0.1

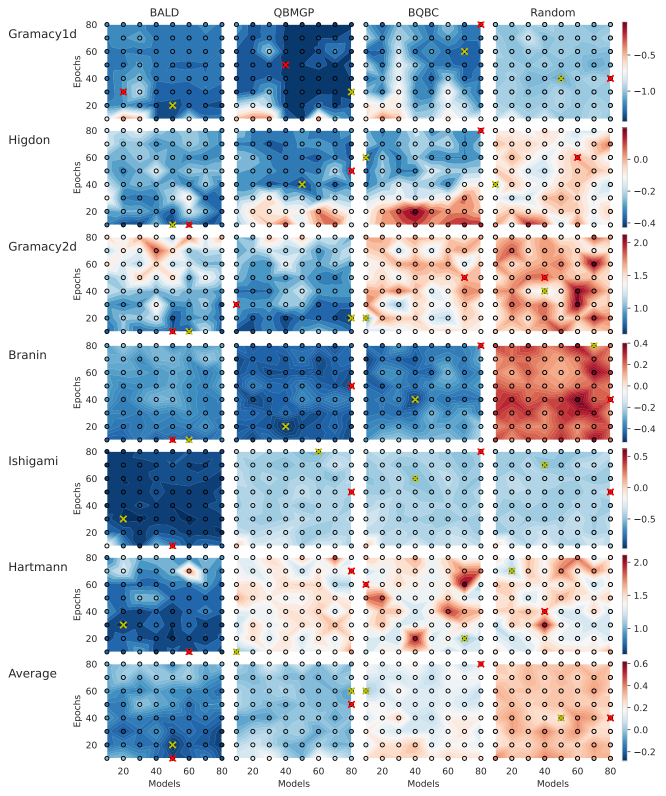

Fig. 9. Contour plots of the grid search based on negative log probability, the lower (the more blue), the better. Yellow crosses mark the best values, whereas red crosses mark the values obtained by using the optimal hyperparameters estimated by using the other simulators.

TABLE 8
Simulators used in the Bayesian optimization experiments.

Simulator	d	σ_ε^2	Input space	Iterations
Branin	2	2.850	$[-5, 10] \times [0, 15]$	40
Rosenbrock2d	2	11.430	$[-1.5, 1.5]^2$	150
Rosenbrock4d	4	21.790	$[-1.5, 1.5]^4$	150
Hartmann3d	3	0.048	$[0, 1]^3$	150
Hartmann4d	4	0.050	$[0, 1]^4$	150
Hartmann6d	6	0.019	$[0, 1]^6$	200

the mutual information between the unknown outputs and the parameters $\mathrm{I}[\hat{\theta}, y \mid x, \mathcal{D}]$:

$$\begin{aligned}\textsc{Bald}(\boldsymbol{x}^\star) &= \mathrm{I}[\boldsymbol{f}^\star, y^\star \mid \boldsymbol{x}^\star, \mathcal{D}] && (33)\\ &= \mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \mathcal{D}] - \mathbb{E}_{p(\boldsymbol{f}^\star \mid \mathcal{D})}\left[\mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}^\star]\right]. && (34)\end{aligned}$$

Given that the likelihood of the data $p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}^\star) = \mathcal{N}(\boldsymbol{f}^\star, \sigma_\varepsilon^2)$ and that the entropy of a Gaussian distribution is $\mathrm{H}[X] = \frac{1}{2}\ln 2\pi e\sigma^2$ (cf. eq. (26)), the second term is independent of $\boldsymbol{x}^\star$:

$$\begin{aligned}\mathbb{E}_{p(\boldsymbol{f}^\star \mid \mathcal{D})}\left[\mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}^\star]\right] &= \int_{\boldsymbol{f}^\star} p(\boldsymbol{f}^\star \mid \mathcal{D})\, \mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}^\star] && (35)\\ &= \int_{\boldsymbol{f}^\star} p(\boldsymbol{f}^\star \mid \mathcal{D})\, \frac{1}{2}\ln 2\pi e\sigma_\varepsilon^2 && (36)\\ &= \frac{1}{2}\ln 2\pi e\sigma_\varepsilon^2 \int_{\boldsymbol{f}^\star} p(\boldsymbol{f}^\star \mid \mathcal{D}) && (37)\\ &= \frac{1}{2}\ln 2\pi e\sigma_\varepsilon^2\,. && (38)\end{aligned}$$

Thus BALD is equivalent to the entropy minus a constant. Since the predictive posterior of a GP is a Gaussian distribution (cf. eq. (32)), and the entropy of a Gaussian distribution is proportional to the predictive variance cf. eq. (26), BALD for a GP is then proportional to the predictive variance:

$$\begin{aligned}\textsc{Bald}(\boldsymbol{x}^\star) &= \mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \mathcal{D}] - \frac{1}{2}\ln 2\pi e\sigma_\varepsilon^2 && (39)\\ &= \frac{1}{2}\ln 2\pi e\sigma^2(\boldsymbol{x}^\star) - const. && (40)\\ &\propto \sigma^2(\boldsymbol{x}^\star)\,. && (41)\end{aligned}$$

G.3 Predictive posterior for a fully Bayesian GP

In the following, we derive the predictive posterior for a fully Bayesian GP.

$$p\left(y^\star \mid \boldsymbol{x}^\star, \mathcal{D}\right) \tag{42}$$
$$= \int_{\boldsymbol{f}^\star} p\left(y^\star \mid \boldsymbol{f}^\star\right) p\left(\boldsymbol{f}^\star \mid \boldsymbol{x}^\star, \mathcal{D}\right) \tag{43}$$
$$= \int_{\boldsymbol{f}^\star} p\left(y^\star \mid \boldsymbol{f}^\star\right) \int_{\boldsymbol{\theta}} \int_{\boldsymbol{f}} p\left(\boldsymbol{f}^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}, \boldsymbol{\theta}, \mathcal{D}\right) p(\boldsymbol{f}, \boldsymbol{\theta} \mid \mathcal{D}) \tag{44}$$
$$= \int_{\boldsymbol{f}^\star} p\left(y^\star \mid \boldsymbol{f}^\star\right) \tag{45}$$
$$\int_{\boldsymbol{\theta}} \int_{\boldsymbol{f}} p\left(\boldsymbol{f}^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}, \boldsymbol{\theta}, \mathcal{D}\right) p(\boldsymbol{f} \mid \boldsymbol{\theta}, \mathcal{D}) p(\boldsymbol{\theta} \mid \mathcal{D}) \tag{46}$$
$$= \int_{\boldsymbol{f}^\star} p\left(y^\star \mid \boldsymbol{f}^\star\right) \int_{\boldsymbol{\theta}} p\left(\boldsymbol{f}^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}, \mathcal{D}\right) p(\boldsymbol{\theta} \mid \mathcal{D}) \tag{47}$$
$$= \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) \int_{\boldsymbol{f}^\star} p\left(y^\star \mid \boldsymbol{f}^\star\right) p\left(\boldsymbol{f}^\star \mid \boldsymbol{x}^\star, \boldsymbol{\theta}, \mathcal{D}\right) \tag{48}$$
$$= \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) \, \mathcal{N}\left(y^\star \mid \mu^\star_{\boldsymbol{\theta}}, K^\star_{\boldsymbol{\theta}}\right) . \tag{49}$$

G.4 BALD for a fully Bayesian GP

For a fully Bayesian GP with a homoscedastic Gaussian likelihood ($\mathcal{N}(0, \sigma^2_\varepsilon | \boldsymbol{\theta})$), there are two parameters: $\boldsymbol{f}^\star$ and $\boldsymbol{\theta}$. Since we are interested in both minimizing the predictive uncertainty and learning the hyperparameters, we let both $\boldsymbol{f}$ and $\boldsymbol{\theta}$ be the main parameters in BALD. Then BALD is

$$\mathrm{I}[\boldsymbol{f}, \boldsymbol{\theta}, y \,|\, \boldsymbol{x}, \mathcal{D}] = \mathrm{H}\left[\mathbb{E}_{p(\boldsymbol{f}, \boldsymbol{\theta} | \mathcal{D})}[y \,|\, \boldsymbol{x}, \mathcal{D}, \boldsymbol{f}, \boldsymbol{\theta}]\right] \tag{50}$$
$$- \mathbb{E}_{p(\boldsymbol{f}, \boldsymbol{\theta} | \mathcal{D})}\left[\mathrm{H}[y \,|\, \boldsymbol{x}, \boldsymbol{f}, \boldsymbol{\theta}]\right] . \tag{51}$$

There exists no closed-form solution for computing the entropy of the predictive posterior for a fully Bayesian GP, and thus the first term is approximated with Monte Carlo samples and quadrature techniques. Now, we look at the second term. First, we note that

$$\int_{\boldsymbol{\theta}} \int_{\boldsymbol{f}^\star} p(\boldsymbol{f}^\star, \boldsymbol{\theta} \mid \mathcal{D}) = \int_{\boldsymbol{\theta}} \int_{\boldsymbol{f}^\star} p(\boldsymbol{f}^\star \mid \boldsymbol{\theta}, \mathcal{D}) p(\boldsymbol{\theta} \mid \mathcal{D}) \tag{52}$$
$$= \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) \int_{\boldsymbol{f}^\star} p(\boldsymbol{f}^\star \mid \boldsymbol{\theta}, \mathcal{D}) \tag{53}$$
$$= \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) . \tag{54}$$

Given that the likelihood of the data $p(y^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}^\star) = \mathcal{N}(\boldsymbol{f}^\star, \sigma^2_\varepsilon)$ and that the entropy of a Gaussian distribution is $\mathrm{H}[X] = \frac{1}{2} \ln 2\pi e \sigma^2$ (cf. eq. (26)), the entropy $\mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}^\star, \boldsymbol{\theta}] = \frac{1}{2} \ln 2\pi e \sigma^2_{\varepsilon, \boldsymbol{\theta}}$, where the variance $\sigma^2_{\varepsilon, \boldsymbol{\theta}}$ is dependent on the hyperparameters $\boldsymbol{\theta}$.

The second term is

$$\mathbb{E}_{p(\boldsymbol{f}^\star, \boldsymbol{\theta} | \mathcal{D})}\left[\mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}^\star, \boldsymbol{\theta}]\right] \tag{55}$$
$$= - \int_{\boldsymbol{\theta}} \int_{\boldsymbol{f}^\star} p(\boldsymbol{f}^\star, \boldsymbol{\theta} \mid \mathcal{D}) \mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}^\star, \boldsymbol{\theta}] \tag{56}$$
$$= - \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) \int_{\boldsymbol{f}^\star} p(\boldsymbol{f}^\star \mid \boldsymbol{\theta}, \mathcal{D}) \mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \boldsymbol{f}^\star, \boldsymbol{\theta}] \tag{57}$$
$$= - \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) \int_{\boldsymbol{f}^\star} p(\boldsymbol{f}^\star \mid \boldsymbol{\theta}, \mathcal{D}) \, \frac{1}{2} \ln 2\pi e \sigma^2_{\varepsilon, \boldsymbol{\theta}} \tag{58}$$
$$= - \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) \, \frac{1}{2} \ln 2\pi e \sigma^2_{\varepsilon, \boldsymbol{\theta}} . \tag{59}$$

Thus the expectation of the entropy is independent of the input $\boldsymbol{x}^\star$. Hence, BALD is proportional to the entropy:

$$\mathrm{BALD}_{\text{case 2}}(\boldsymbol{x}^\star) = \mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \mathcal{D}] - \int_{\boldsymbol{\theta}} p(\boldsymbol{\theta} \mid \mathcal{D}) \, \frac{1}{2} \ln 2\pi e \sigma^2_{\varepsilon, \boldsymbol{\theta}} \tag{60}$$
$$\propto \mathrm{H}[y^\star \mid \boldsymbol{x}^\star, \mathcal{D}] . \tag{61}$$