Received: 22 February 2021 | Revised: 31 March 2021 | Accepted: 26 April 2021

DOI: 10.1002/ail2.24

LETTER

WILEY

Good practices for Bayesian optimization of high dimensional structured spaces

Eero Siivola[1] | **Andrei Paleyes**[2] | **Javier González**[3] | **Aki Vehtari**[1]

[1]Department of Computer Science, Aalto University, Espoo, Finland

[2]Machine Learning at Computer Lab, Univeristy of Cambridge, Cambridge, UK

[3]Microsoft Research Cambridge, Microsoft, Cambridge, UK

Correspondence
Eero Siivola, Department of Computer Science, Aalto University, Espoo, Finland.
Email: eero.siivola@gmail.com

Abstract

The increasing availability of structured but high dimensional data has opened new opportunities for optimization. One emerging and promising avenue is the exploration of unsupervised methods for projecting structured high dimensional data into low dimensional continuous representations, simplifying the optimization problem and enabling the application of traditional optimization methods. However, this line of research has been purely methodological with little connection to the needs of practitioners so far. In this article, we study the effect of different search space design choices for performing Bayesian optimization in high dimensional structured datasets. In particular, we analyses the influence of the dimensionality of the latent space, the role of the acquisition function and evaluate new methods to automatically define the optimization bounds in the latent space. Finally, based on experimental results using synthetic and real datasets, we provide recommendations for the practitioners.

KEYWORDS

Bayesian optimization, Gaussian processes, variational autoencoders

1 | INTRODUCTION

Science and engineering often involve optimization of expensive black-box functions. The meaning of "expensive" is context-dependent, but usually refers to monetary cost or time it takes to evaluate the function. Examples of such optimization applications include experimental design,[1] hyperparameter optimization of algorithms,[2] gait optimization in robotics,[3] and process optimization in manufacturing industry.[4,5] The challenge of high evaluation cost of the objective function can be exacerbated by non-trivial structure and high dimensionality of the optimization space. These additional complications are often seen in many problems in bioinformatics, chemical engineering[6] and computer science[7] which involve optimizing structured objects, such as graphs or images. One important example is designing new molecules, which can be laborious since testing the chemical properties requires wet room experiments, that require manual work by an expert and expensive special equipment. Black-box nature, evaluation cost, structure and high dimensionality are all properties that make the traditional optimization techniques inefficient. Mathematically this optimization problem can be formulated as follows. Let $\mathcal{X}$ be an input space and $f : \mathcal{X} \rightarrow \mathbb{R}$ be a continuous *black-box* function. We are interested in solving the global optimization problem of finding the unknown minimum of f:

$$\mathbf{x}_{\min} = \arg\min_{x \in \chi} f(\mathbf{x}) \tag{1}$$

We make three assumptions:

1. We can query f using noisy queries $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathrm{N}(0, \sigma)$.
2. $\mathcal{X}$ is structured and high dimensional.
3. We can access a large unlabeled dataset $\mathbf{X}$ in the input space s.t. $X_i \in \mathcal{X}\ \forall i = 1...N$.

The goal is to find $\mathbf{x}_{\min}$ by limiting the number of queries, which areac assumed to be expensive.

In traditional applications with a low number of continuous parameters, Bayesian optimization (BO) is the de facto solution for these gradient-free black-box optimization problems. The core idea of BO is to build a surrogate probabilistic model that efficiently guides the sequential acquisition of new data. However, building and optimizing these probabilistic surrogates in structure high dimensional spaces is challenging and often leads to poor performance.

The recent increase in the availability of high volume datasets has made it possible to use semi-supervised deep generative models to efficiently embed structured high dimensional objects into a lower-dimensional Euclidean space. These models have enabled the use of gradient-free optimization methods for optimizing the structure in the low dimensional manifold[8,9] However, this research has so far been merely methodological and with no emphasis on how to design the optimization task itself. There is no research on (a) how to decide on the dimensionality of the low dimensional embedding and how this decision affects the optimization task; (b) how to optimize the acquisition function in the low dimensional manifold; and (c) how to balance between exploration and exploitation when selecting new points. We systematically study the effects of these design choices applied to a variety of high dimensional structured optimization tasks. We hope that our findings will help practitioners to better design their high dimensional structured optimization problems.

The rest of the article has the following structure. Section 2 presents the related work and Section 3 introduces the theoretical background. Section 4 describes the framework to perform BO by exploiting deep generative models and the main design choices analyzed in the article. Section 5, describes the experimental set-up and presents the main results. Finally, in Section 6, the article is concluded with discussion.

2 | RELATED WORK

BO is considered as the method of choice for sample-efficient gradient-free optimization of low dimensional Euclidean spaces.[10,11] The biggest limitation of BO has been its scalability with respect to the dimensionality of the search space.

The earliest solutions to scale BO to higher-dimensional spaces are based on projecting the input space to low dimensional spaces using linear transformations. Wang et al[12] use random linear projections where BO is performed. Garnett et al[13] optimize the linear projection during the optimization to further improve the performance. Tripathy et al[14] find a lower-dimensional manifold using orthogonal projection which parameters are treated as hyperparameters. Groves and Pyzer-Knapp[15] use linear encoding to reduce the dimensionality of the optimization space and decide the optimal dimensionality with two-step iterative shrinkage/thresholding-method.[16] Unlike the previous linear dimensionality reduction methods, their approach is able to automatically decide the optimal dimensionality. The main disadvantage of these methods is the fact they are only able to find linear manifolds.

Another strategy for high dimensional optimization is to better understand the structure of the search space and impose additional assumptions that simplify the problem. Kandasamy et al[17] assume that the search space is composed of disjoint low dimensional subspaces that can be optimized separately. Mutny and Krause[18] extend the approach by allowing overlapping subspaces. Oh et al[19] propose cylindrical kernel to better scale with the dimensionality of the latent space and avoid the over-exploration of the boundaries of the search space. Espinasse et al[20] use a Gaussian process (GP) kernel designed for high dimensional graphs. In some of the most recent works, Jaquier et al[21] exploit the non-Euclidean geometry of the parameter space in a parameter tuning problem in robotics and perform the optimization with a GP that is efficient on high dimensional Riemannian manifolds. However, these methods rely on handcrafted or problem specific assumptions that may be violated in most real-world applications.

The most recent works use deep generative models to both reduce dimensionality and take advantage of the structure of the latent space. If lots of data are available, it is possible to find nonlinear low dimensional manifolds. Hebbal et al[22] use Deep GPs in BO, that allow the optimization to be performed in non-linear subspaces. Some of these approaches also assume an access to a reservoir of unlabeled data that can be used in learning a nonlinear, low dimensional manifold from the data in an unsupervised way. Huang et al[23] use a two-step solution by first training an auto encoder with all unlabeled data to find a low dimensional representation of the problem and then performing regression in the low-dimensional latent space. Griffiths and Hérnandez-Lobato[8] use this approach in BO. Kusner et al[9] extend this work to better handle the uncertainty by resorting to deep generative models. In particular, rather than using an auto encoder in the first stage, they resort to a variational auto encoder (VAE) to model the uncertainty to the latent space. In the second stage, they use a Gaussian process latent variable model (GPLVM) to propagate the uncertainty of the latent space improving the overall performance.

The biggest problem with these two-step strategies is that the latent space learned while only using the unlabeled data might not be optimal for the optimization task. Eissman et al[24] address this issue by jointly learning the latent space with the labeled data and iteratively modifying the latent representation as new data is being collected. Tripp et al[25] further exploit the retraining of the latent space by weighing the samples used in training the VAE based on their observed values so that good observations have more importance in training the latent space.

We also explore different strategies for bounding the latent space when optimizing the acquisition function. Prior to this work, this has only been studied for linear embeddings. Wang et al[12] project the input producing the maximum of the acquisition function to the closest point on the boundary of the search space if it is outside it. Binois et al[26] and Binois et al[27] use projections to non-closest point to reduce the over-exploration of the boundaries of the search space.

The solutions combining deep generative models with Gaussian Processes seem the most prominent approach to the problem at the moment. The strength of this is in its ability to use the unlabeled data together with the labeled data. The use of the unlabeled data allows the use of non-linear manifolds for optimization and is thus suitable for complex real world problems. This is the reason why in this article we concentrate on these methods. Specifically, we use VAEs as deep generative models due to them having become the de facto method for the problem. The popularity of VAEs has led to them being applied on different types of structured data, including but not limited to images,[28] text,[29] sound,[30] and structured physical objects like molecules.[31] This allows practitioners from many fields of work to benefit from our work. In the remaining of the article, we analyze the effect of different design choices in the optimization strategies based on VAEs. There is a need for this kind of work as these approaches appear very promising to the practitioners, but the existing research does not yet study the effect of different design choices when applying these methods in practice.

3 | THEORY AND DERIVATIONS

In this section we introduce all necessary parts to define the general framework we will use for the analysis of BO using deep generative models. The section first introduces VAEs, then GPLVMs and finally how to combine VAEs and GPLVMs.

3.1 | Variational Auto Encoders

In VAEs the aim is to find a latent representation $\mathbf{z} \in \mathbb{R}^d$ for data $\mathbf{x} \in \mathcal{X}$. Let θ parameterize a probabilistic decoder $p_\theta(\mathbf{x} \mid \mathbf{z})$ with prior distribution $p(\mathbf{z})$. The posterior distribution $p_\theta(\mathbf{z} \mid \mathbf{x}) \propto p_\theta(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z})$ can be interpreted as a probabilistic encoder, which in most cases is intractable. In order to address this, the posterior needs to be approximated with a tractable distribution $q_\phi(\mathbf{z} \mid \mathbf{x})$, where ϕ parameterizes the encoder. The parameters θ and ϕ can jointly be learnt by maximizing the evidence lower bound (ELBO)

$$\mathscr{L}(\phi,\theta;\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathrm{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \| p(z)\right), \tag{2}$$

which can be done with gradient descent as long as decoder and encoder approximation are differentiable with respect to θ and ϕ and are computable pointwise. A normal choice for the encoder distribution is multivariate normal, $q_\theta(\mathbf{z} \mid \mathbf{x}) = \mathrm{N}(\mathbf{z} \mid \mu_\theta(\mathbf{x}), \Sigma_\theta(\mathbf{x}))$, where mean and (usually) diagonal covariance are outputs of a neural network. Decoder distributions vary more, depending on the type of the data. Bernoulli distributions can be used to decode binary data, continuous Bernoulli for bounded data[32] and (log) normal distribution for (half-bounded) continuous data.

3.2 | Gaussian process latent variable models

Traditional BO approaches assume that the true latent black-box function f is a realization of random variables sampled from a Gaussian process, $p(f) = \mathcal{GP}$, fully specified by a prior mean and some covariance function K.[33] The prior mean, which is often zero, defines the prior mean of the latent function and the covariance function specifies the covariance of the latent function between any two points. The problem with full GP is scalability. Assuming N observations from the latent function, computing the posterior of a full GP requires inverting a $N \times N$ matrix, which has $\mathcal{O}(n^3)$ computational cost.

GPs can be approximated, and made more scalable, by using inducing inputs. In this method we use inducing latent values $\mathbf{u}$ (at locations $\mathbf{z}_u$) in the latent space instead of latent values $\mathbf{f}$ with observations $\mathbf{y}$ (at input locations $\mathbf{z}$). Using this notation, the posterior of the data becomes

$$p(\mathbf{y}|\mathbf{u},\mathbf{z}_u,\mathbf{z}) = \mathbb{E}_{p(\mathbf{f}|\mathbf{u})}[p(\mathbf{y}|\mathbf{f})]. \tag{3}$$

The posterior of the inducing latent values, $p(\mathbf{u}\,|\,y, \mathbf{z}_u, \mathbf{z}) \propto \int p(\mathbf{u}\,|\,\mathbf{f}, \mathbf{z}_u, \mathbf{z})p(\mathbf{f}\,|\,\mathbf{y}, \mathbf{z})\mathrm{d}\mathbf{f} \propto \int p(\mathbf{u}\,|\,\mathbf{f}, \mathbf{z}_u, \mathbf{z})p(\mathbf{y}\,|\,\mathbf{f})p(\mathbf{f}\,|\,\mathbf{z})\mathrm{d}\mathbf{f}$, is intractable for general likelihood $p(\mathbf{y}\,|\,\mathbf{f})$ and for efficient prediction needs to be approximated. Approximating it with normal $q(\mathbf{u}) = \mathrm{N}(\mathbf{u}\,|\,\mathbf{m}, \mathbf{L}^T\mathbf{L})$, with general mean and general lower triangular matrix provides computationally tractable properties.

The GP log marginal likelihood can be approximated with a lower bound

$$\log p(\mathbf{y}) \geq \mathbb{E}_{q(\mathbf{u})}[\log p(\mathbf{y}|\mathbf{u})] - \mathrm{KL}[q(\mathbf{u}) \parallel p(\mathbf{u})], \tag{4}$$

where, $p(\mathbf{u}) = \mathrm{N}(\mathbf{u}\,|\,\mu_{\text{prior}}, \Sigma_{\text{prior}})$, with μ_{prior} and Σ_{prior}, which are the GP prior mean and prior covariance. Furthermore, with $p(y\,|\,f) = \mathrm{N}(y\,|\,f, \sigma^2)$, the likelihood inside the expectation in Equation (4) reduces as

$$p(\mathbf{y}|\mathbf{u}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})\mathrm{d}\mathbf{f} = \mathrm{N}\left(\mathbf{f}|\mathbf{A}\mathbf{m}, \mathbf{K}_{ff} + \mathbf{A}\left(\mathbf{L}^T\mathbf{L} - \mathbf{K}_{uu}\right)\mathbf{A}^T\right),$$

where, $\mathbf{A} = \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}$. This model is referred to as Stochastic Variational Gaussian process (SVGP).[34,35]

Since the latent mapping of VAEs is uncertain, we can also add uncertainty to the locations $\mathbf{z}$ by assuming that the projections in the latent space follow an unknown distribution $q_\phi(\mathbf{z}\,|\,\mathbf{x}) = \mathrm{N}(\mathbf{z}\,|\,\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$, parameterized by the encoder. This uncertainty can be added to the lower bound as

$$\log p(\mathbf{y}|\mathbf{z}) \geq \mathbb{E}_{q(\mathbf{f},\mathbf{z})}[\log p(\mathbf{y}|\mathbf{f})] - \mathrm{KL}[q(\mathbf{f}) \parallel p(\mathbf{u})] = \mathbb{E}_{q(\mathbf{z})}\left[\mathbb{E}_{q(\mathbf{f}|\mathbf{z})}[\log p(\mathbf{y}|\mathbf{f})] - \mathrm{KL}[q(\mathbf{f}) \parallel p(\mathbf{u})]\right]. \tag{5}$$

This model, GP with uncertainty in the inputs, is called Gaussian process latent variable model.[36]

3.3 | Gaussian processes in the latent space of variational auto encoder

To the best of our knowledge, there are two widely used alternatives for combining GPLVM with VAE. For general notation, let $\{\mathbf{X}_o, \mathbf{Y}_o\}$ be the observed data and observations and $\mathbf{X}_u$ be the unobserved data.

3.3.1 | Training variational auto encoder and Gaussian process latent variable model disjointly

The original and simplest way of combining GPLVM with VAE was introduced by Ref. 9. Their strategy is to train the VAE with $\mathbf{X}_u$ prior to any observations using Equation (2). After training the VAE, its parameters are fixed and the GPLVM is trained by maximizing Equation (5) with $p_\phi(\mathbf{Z}_o\,|\,\mathbf{X}_o)$, where ϕ is fixed.

3.3.2 | Training variational auto encoder and Gaussian process latent variable model jointly

An alternative solution is to train the parameters of the VAE and GPLVM jointly like[24] Their approach uses separate cost functions for the labeled and unlabeled data. For labeled data costs of Equations (2) and (5) are combined as,

$$\mathscr{L}(\phi,\theta;\mathbf{x}_o,\mathbf{y}_o)=\mathscr{L}(\phi,\theta;\mathbf{x}_o)+\mathbb{E}_{q_\phi(\mathbf{z}_o|\mathbf{x}_o)}[\log p(\mathbf{y}_o|\mathbf{z}_o)]. \tag{6}$$

For unlabeled data, the cost defined in Equation (2) is used.

4 | BAYESIAN OPTIMIZATION WITH VARIATIONAL AUTO ENCODERS

BO is a gradient-free black-box optimization method. The iterative steps of any BO algorithm are: (a) Train the probabilistic surrogate model, usually a Gaussian process or in our case GPLVM, using the available data; (b) evaluate the black-box function at the maximum of the acquisition function; (c) Update the existing data with new evaluation; and (d) Repeat steps 1, 2, and 3 until a certain stopping criterion is met. When dealing with high dimensional structured spaces, the first step also includes (re)-training the latent space either jointly or disjointly with the surrogate model. Also, the optimization bounds of the acquisition function need to be re-learned on every iteration as the mapping to the latent space constantly changes (see Algorithm 1). In this section we discuss different design choices for applying BO to tasks in high dimensional structured spaces: (a) the dimensionality of the latent space, (b) the choice of the acquisition function, and (c) the choice of the optimization bounds in the latent space.

The dimensionality of the latent space.

Choosing the right dimensionality of the latent space is complex and task dependent. A too low dimensional latent space affects the quality of the samples in $\mathcal{X}$ produced by the decoder. On the other hand, a too high dimensional latent space makes fitting the GPLVM in the latent space harder and not as sample efficient. In addition, a too high dimensional space leads to overfitting and poor generalization. The aim of iteratively learning the latent space with the collected observations is to make the optimization task easier in the latent space, but it is yet an open research question how the methods perform when the latent space dimensionality is varied.

4.1 | The choice of acquisition function

It has been deeply studied how different acquisition functions perform in low dimensional Euclidean spaces that are prevalent in traditional BO applications. All acquisition functions balance between exploration and exploitation; the tendency of sampling from regions with lots of uncertainty versus tendency of sampling from regions with known good values. Here we explore the role of the acquisition in structured high-dimensional spaces. In particular, the acquisition functions used in this article include Thompson Sampling (TS),[37,38] Expected Improvement (EI),[39,40] Probability of Improvement (PI),[41] and Lower Confidence Bound (LCB).[42]

4.2 | Optimization bounds of the acquisition function

Unlike in the regular low dimensional BO, the selection of the optimization bounds of the acquisition function is a difficult design choice. Normally optimization bounds for the acquisition function are selected based on expert knowledge

```
Algorithm 1
Bayesian optimization with VAEs
Input: Unlabeled data $\mathbf{X}_u$, labeled data $\{\mathbf{X}_o, \mathbf{Y}_o\}$, acquisition function $A(\cdot)$, black-box function $f(\cdot)$.
  while there still is evaluation budget and acquisition function values are larger than a threshold do
  (Re-)Learn the encoder, decoder and GP parameters $\theta$, $\phi$ using $\mathbf{X}_u$, $\mathbf{X}_o$ and $\mathbf{Y}_o$
  (Re-)Learn the space $\mathcal{Z} \in R^d$ where the acquisition function is optimized using $\mathbf{X}_u$, $\theta$, and $\phi$
  Find the next location $\mathbf{z}_*$ in the latent space by maximizing the acquisition function $\mathbf{z}_* = \text{argmax}_{\mathbf{z} \in \mathcal{Z}} A(\mathbf{z})$
  Project $\mathbf{z}_*$ to the original data space as $\mathbf{x}_*$ using the learned decoder parameters $\theta$
  Find label $y_*$ using $f(\mathbf{x}_*)$
  Append $\{\mathbf{X}_o, \mathbf{Y}_o\}$ with $\{\mathbf{x}_*, y_*\}$
end while
```

or physical constraints. As the latent space formed by the VAE is an abstraction and as such is not tied to the business domain of the problem at hand, selecting the bounds for it is much harder. So far, the de facto method has been to bound the optimization of the acquisition function by a hypercube containing the projection of the training data in the latent space. To the best of our knowledge, no alternatives to this method have been explored.

We demonstrate three methods for restricting the optimization space. An easy and scalable approximation is to find a minimum volume *n*-ellipsoid, $(\mathbf{x}-\mathbf{x}_0)^{\mathrm{T}}\mathbf{A}(\mathbf{x}-\mathbf{x}_0)=1$, that contains (the means of) all the training data. An easy and scalable way of doing this is via the Khachiyan algorithm.[43] Another easy, but not as scalable way is to find a set of hyperplanes restricting the data and form a set of linear inequalities of the form $\mathbf{Ax}\preccurlyeq\mathbf{b}$. It is easy to find this set of inequalities by first finding a convex hull for the existing samples and then perform Delaunay triangulation for the edge points and use the edge points of each simplex to find the hyperplanes. The combined time complexity of these operations is $\mathcal{O}(n^2)$, where n is the total number of unlabeled points. The benefit of both these methods is that the optimization of the acquisition function can be performed in a convex set.

Third approach of limiting the optimization space is setting an upper limit to the allowed distance between a point in the latent space $\mathbf{z}'$ and the expected value of the encoder of the expected value of the decoded $\mathbf{z}'$

$$\left\|\mathbf{z}'-\int \mathbf{z}p_{\theta}\left(\mathbf{z}\middle|\int \mathbf{x}q_{\phi}(\mathbf{x}|\mathbf{z})\mathrm{d}\mathbf{x}\right)\mathrm{d}\mathbf{z}\right\|. \tag{7}$$

This approach guarantees that the uncertainty is reduced in the regions where the acquisition function is originally meant to be evaluated. A good strategy for selecting the upper bound of the Euclidean distance is to see how much all the points in the training data move and select, for example, the 90% percentile of these distances. The drawback of this solution is that the allowed region is not necessarily convex, making the optimization harder. All these strategies are visualized in Figure 1 for a VAE trained with the Shape dataset (see details of the Shape dataset in Section 5).

5 | EXPERIMENTAL RESULTS

5.1 | Experimental setup

All case studies are performed with the two variations of the high dimensional structured BO as described in Section 3.3. We used MXNet[44] for modeling and Emukit[45] to run the customized BO routine. All the code written to run the experiments is available at https://github.com/esiivola/hdssbo.

The VAE used in the experiments has three layers in both encoder and decoder with [{num inputs}, {num inputs}, {dim latent space} × 2] units in the encoder and [{dim latent space}, {num inputs}, {num inputs}] units in the decoder. The parameters are trained with a learning rate 10^{-3}. The GPLVM model has 150 inducing points and a squared

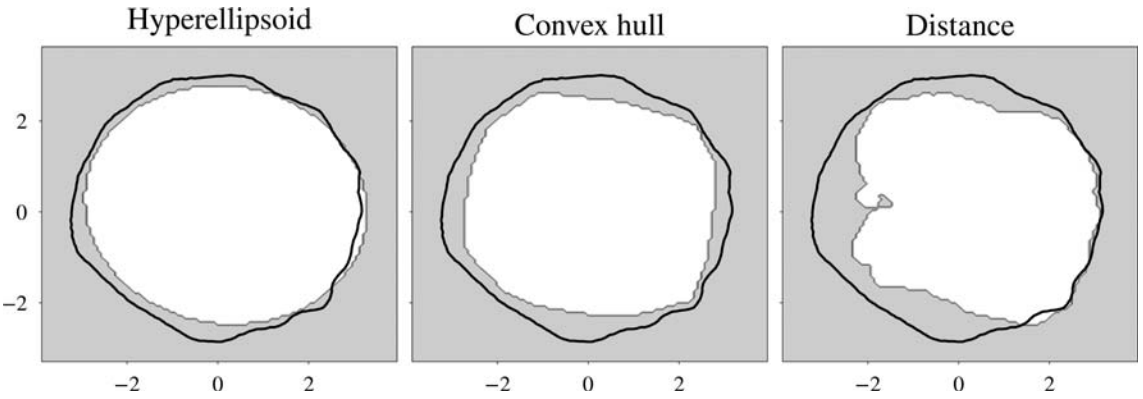

FIGURE 1 Different space restriction strategies for acquisition function optimization visualized from left to right ellipse, convex hull, and distance method. In the distance method the maximum distance the point can move is computed as a 90% of the maximum of the move distances of latent space means of the train data. The white regions indicate the area where the acquisition function is optimized. The black continuous line visualizes the region which contains 99% of the data used to train the VAE

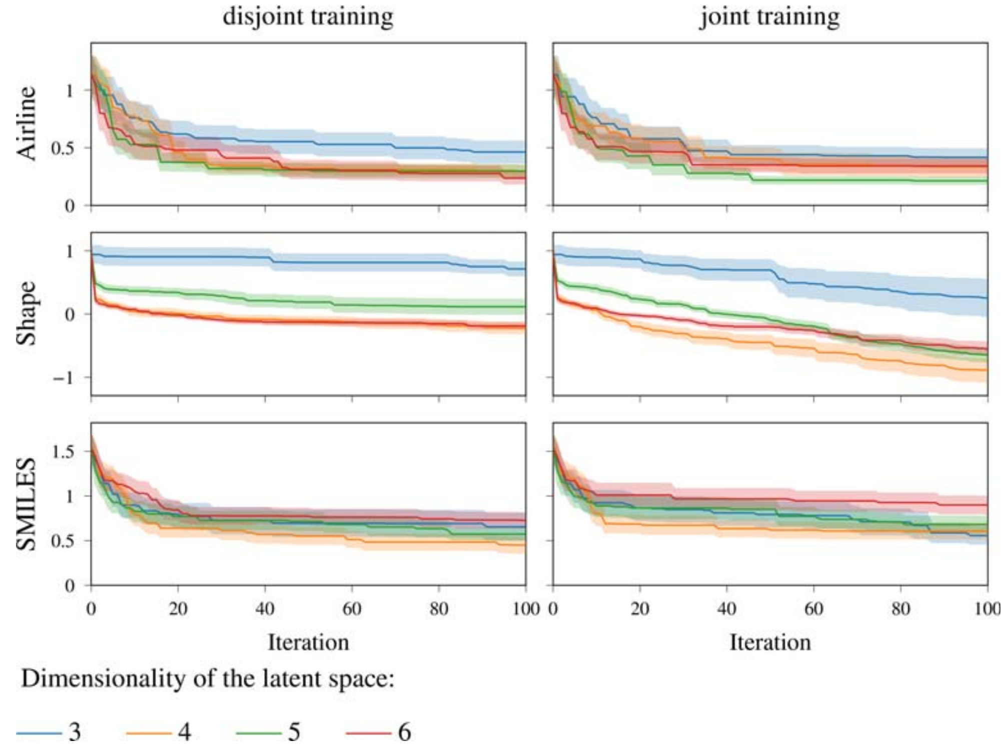

FIGURE 2 The effect of the latent space dimensionality on the performance of the BO algorithm. Each subplot visualizes the best observed value so far as a function of optimization iterations for different latent space dimensionalities. Each line visualizes the mean of 10 separate runs and the colored band around the solid line visualizes the SD of the mean. Different rows show results for different datasets and columns show results for different methods. The values of the black-box function are as follows. For airline data, the mean-square error of the fitted model on test data. For shape data, the area of the shape. For SMILES data, the penalized water-octanol partion. The black-box function values are normalized so that the best value in the training data is 0 and the SD of the values of the training data is 1. The black-box function values are normalized so that comparing methods between datasets is easier. In addition to this, the normalization makes the goodness of the optimization more visible as the best black-box function value in the training data is 0

exponential kernel with each latent dimension having its own length scale parameter. Learning rate 10^{-1} is used for the GPLVM parameters. All parameters are trained using Adam algorithm.[46] Prior to starting the optimizations routine, the GPLVM model is initialized with 10 observations sampled uniformly at random from the training data. Following,[25] the VAE parameters are allowed to change only every 10 iterations. The purpose of this is to save computation time and reduce overfitting.

5.2 | Data sets

We design three structured high-dimensional optimization task based on the following datasets:

Airline passenger dataset[47] is a time-series data consisting of the number of monthly airline passengers from January 1949 to December 1960. The black-box function is the mean square error of a model fitted on 66% of the data on a test set consisting of 33% of the data. Following the experimental setting in Lu et al,[48] the fitted model is a Gaussian Process whose kernel is generated by a grammar with four basis kernels (periodic, squared exponential, linear, and rational quadratic) and two operators (+ and *) so that there are at maximum four kernels combined in the produced kernel.

The 2D shape area maximization dataset is an image dataset consisting of rotated black rectangles of a varying area on a background of 10 × 10 pixels. The black-box function is the area of the rotated shape. The dataset mimics the dSprites dataset[49] and is used for optimization by Tripp et al.[25] The dataset simulates a situation, where the true optimum is not inside the training data and finding the optimum requires modifying the latent space.

Molecule dataset consists of valid molecules of carbon, oxygen, nitrogen, and iron each containing up to seven atoms in total and was first introduced by Fink and Raymond.[50] The molecules in the dataset are described by the SMILES

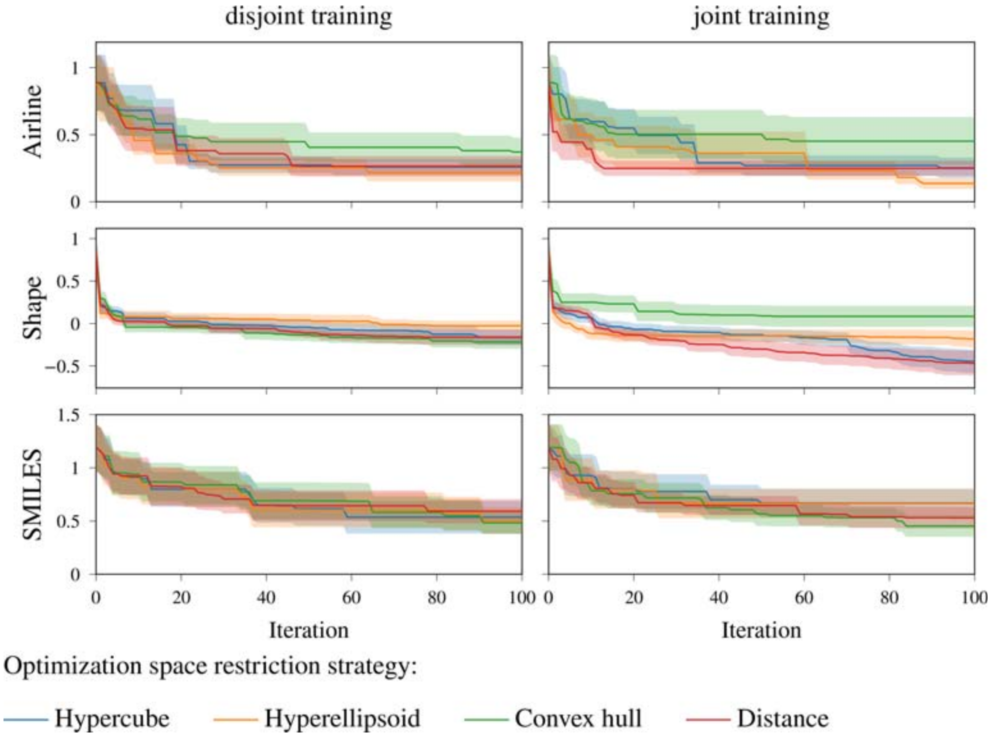

FIGURE 3 The effect of using different strategies of defining the optimization space on the performance of the BO algorithm. Representation of the lines, colored bands, and normalized black-box values is the same as in Figure 2

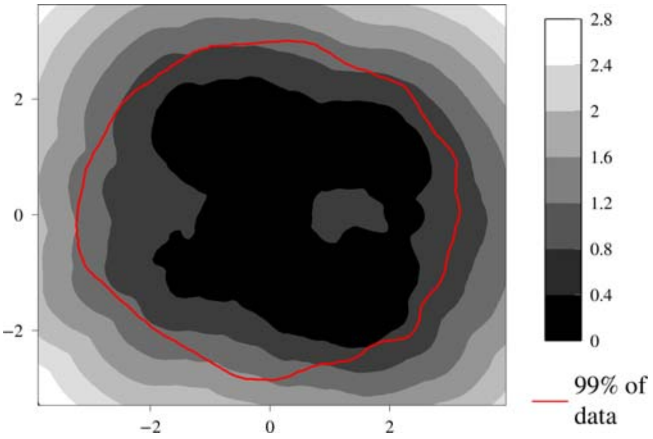

FIGURE 4 The Euclidean distance between a point $\mathbf{z}'$ in a latent space and the location of $\mu_\theta(\mathbf{x}')$ (where $\mathbf{x}'$ is the decoded $\mathbf{z}'$) as a function of $\mathbf{z}'$ in two dimensional latent space. The red continuous line visualizes the region which contains 99% of the data used to train the VAE projected in the latent space of the encoder. Darker colors mean shorter distance and brighter colors mean larger distance. The VAE which latent space is visualized is trained using the Shape-data

grammar.[51] SMILES molecules can be embedded in a low dimensional space using a VAEs as described by Kusner et al.[9] The black-box function to be optimized is the penalized water-octanol partition coefficient that mimics the drug-likeliness of a molecule similarly as Ertl and Schuffenhauer[52] and is also used by Kusner et al.[9]

The outcomes of the black-box functions within the training data are normalized as follows. Let $\mathbf{Y} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\mathrm{T}$ be the *un-normalized* outputs of the black-box function for the whole dataset with N observations. The normalized output for input $\mathbf{x}^* \in \mathcal{X}$ can be computed with

$$f_{\text{normalized}}(\mathbf{x}^*) = \frac{f(\mathbf{x}^*) - \min(\mathbf{Y})}{\mathrm{sd}(\mathbf{Y})}, \tag{8}$$

where, $\mathrm{sd}(\mathbf{Y}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\mathbf{y}_i - \mathrm{mean}(Y))}$ is the SD of the un-normalized values and $\mathrm{mean}(\mathbf{Y}) = \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i$ is the mean of the un-normalized values. The normalization makes comparing different datasets easier as the variation is standardized and the smallest output of the *normalized* black-box function is zero for all datasets.

5.3 | Effect of the dimensionality of the latent space

To study the robustness of changing the dimensionality of the latent space d, we compare the optimization performance on the three datasets and two models when trying different dimensionalities $d \in \{3,4,5,6\}$. The acquisition space is restricted by a hypercube and the acquisition function is set to the Lower Confidence Bound. The results are visualized in Figure 2.

The results show a clear trend for all datasets and both methods. For Airline, Shape, and SMILES the best performance is obtained with four-dimensional latent space. This is true for both tested methods, but for the Shape dataset, at later iterations five-dimensional latent space performs better than four-dimensional latent space. The results also show that if the dimensionality is increased, the performance increases until it plateaus or starts to decrease again. This is caused by one hand the optimization problem becoming harder in high dimensions, but on the other hand the latent space becoming more nuanced with more dimensions. The first property makes BO harder and the second property reduces how noisy the black-box function appears in the latent space.

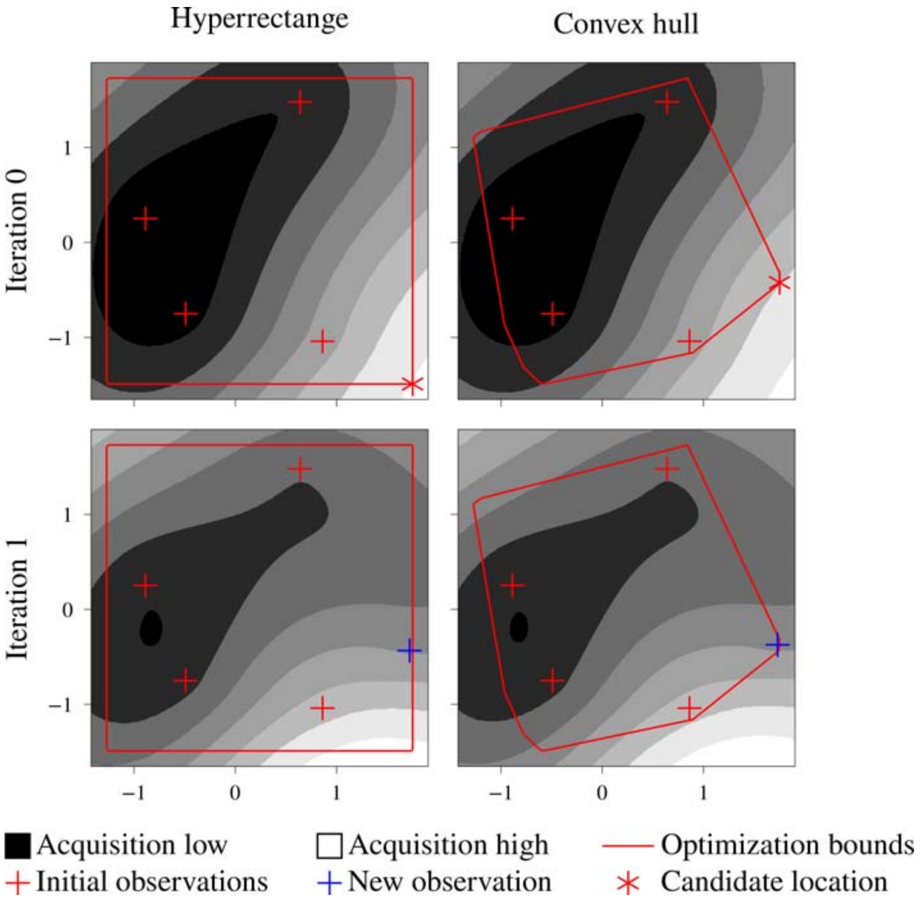

FIGURE 5 The figure shows, on the example of the Airline dataset, how the acquisition function changes during one iteration in a BO for two separate optimization techniques (columns) (A) space bounded with a hyper rectangle bounding the training data; (B) space bounded with a smallest convex hull bounding the training data. On iteration 0, the only difference between the two techniques is the location of the maximum of the acquisition function (marked as *). Even though the acquisition function surfaces are identical the maximums within the optimization area are different. On iteration 1, the maximum of the previous iteration has been projected to the original space to be evaluated and the evaluated point has been projected back to the latent space (marked as +). Since for the hyperrectangle approach, the maximum of the acquisition function is outside the data used to train the VAE, the distance between the maximum of the acquisition function (*) at iteration 0 and the evaluated point (+) at iteration 1 is large. As the maximum of the acquisition function is within the training data for the convex hull approach, the distance is much smaller. Since the maximums of the acquisition function are close to each other at iteration 0 for both restriction strategies, the points are evaluated at the same location and both methods have identical outcome after the first iteration

The only dataset for which the joint training method outperforms the disjoint method is the Shape-dataset. The reason for this is that the joint training often leads to overfitting, which reduces its performance. However, the Shape dataset is optimal for joint training as the minimum of a black-box function is outside the data used to train the VAE. In other words, excellent performance requires big changes in VAE.

5.4 | Effect of different optimization space restriction strategies

To study the effect of different acquisition space optimization strategies, we compare the optimization performance of the three datasets and two models using the three acquisition space restriction strategies described in Section 1, that is, the hyperellipsoid, minimum convex hull and metric based on minimizing the extrapolation error. In addition to these, the simple hyper-rectangle approach traditionally used in the BO literature is used as a baseline. The results are visualized in Figure 3.

The results are surprising as different optimization space restriction strategies seem to only have a very minimal effect on the results. To understand why we need to first understand how the VAE works on the edges of the search space. If a point $\mathbf{z}^{'} \in \mathbb{R}^d$ is selected from the corner of the hypercube in the latent space, it needs to be projected to $\int \mathbf{x} q_\phi(\mathbf{x}|\mathbf{z}')\mathrm{d}\mathbf{x} = \mathbf{x}' \in \mathcal{X}$ to be evaluated. As the corner of the hypercube is outside the data (or on the edge if we are lucky), the decoder needs to extrapolate as it has not been trained using this kind of data. After evaluation, for $\mathbf{x}'$ to be usable for the GPLVM, it needs to be projected back to the latent space R^d (as a distribution $p_\theta(\mathbf{z}^* | \mathbf{x}^{'})$). Since the decoder extrapolates, the point projected back to the latent space is not the same as $\mathbf{z}'$. Figure 4 visualizes the Euclidean distance between the original point in the latent space $\mathcal{X}$ and the point that has first been projected to the original space $\mathbf{x}^{'} = \int \mathbf{x}\, p(\mathbf{x} | \mathbf{z}^{'})\mathrm{d}\mathbf{x}$ and then back to the latent space $\mathbf{z}^* = \int \mathbf{z}^* q_\phi(\mathbf{z} | \mathbf{x}^{'})\mathrm{d}\mathbf{z}$. The figure shows that the further away the point is from the data used to train the VAE, the bigger the distance is. The figure does not show it, but the points sampled from outside the training data travel closer to the training data.

The poor extrapolation of VAE also changes how the BO works. Figure 5 visualizes this on the airline dataset with the acquisition function LCB. The figure visualizes how the acquisition function looks in the latent space of the VAE

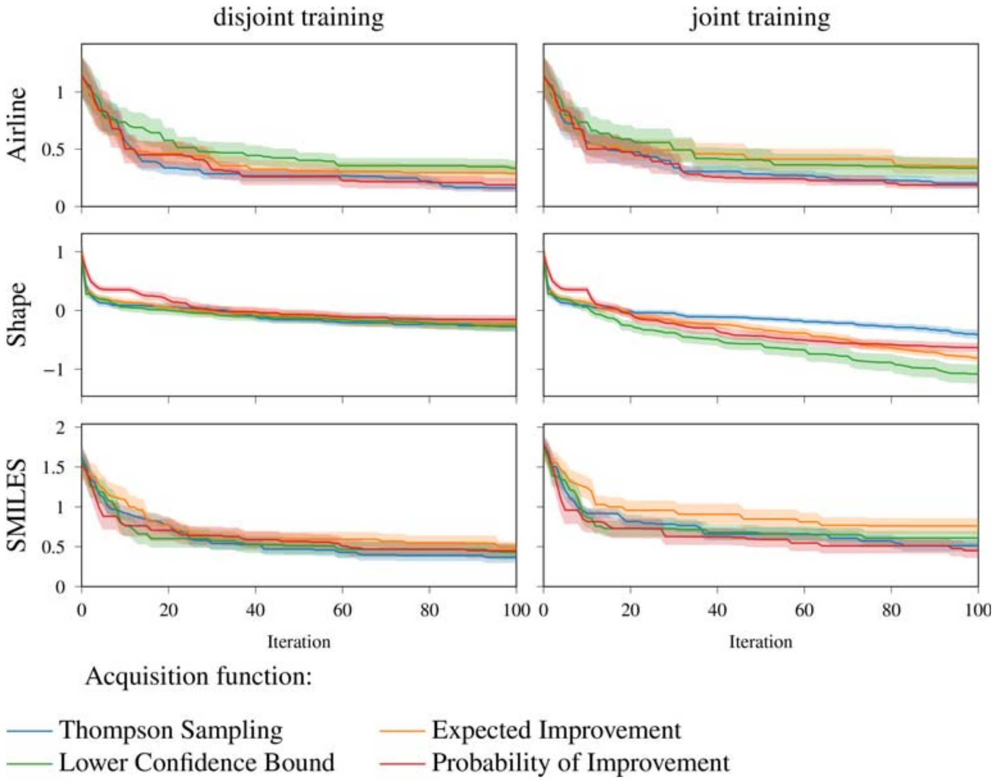

FIGURE 6 The effect of using different acquisition functions on the performance of the BO algorithm. Representation of the lines, colored bands and normalized black-box values is the same as in Figure 2

using two different optimization space restriction strategies in different phases of the BO loop. As the optimization space restriction methods are different, the maximums of the acquisition functions are in different locations. However, for the hyperrectangle approach, as the point is outside the training data, the decoder has to extrapolate and when the extrapolated and evaluated point is projected back to the latent space, it has traveled toward the data set (the distance between the red "*" on the first row and blue "+" on the second row). When comparing to the convex hull approach, the evaluated point has not traveled that much. After all, the effect of both methods is very similar. Counter intuitively, the poor extrapolation of the VAE causes similar behavior as more sophisticated optimization space restriction strategies.

5.5 | Effect of different acquisition strategies

To study the effect of different acquisition functions, we compare the optimization performance of the three datasets and two models using four different acquisition functions using four dimensional latent space. The results are visualized in Figure 6.

The different acquisition strategies show a behavior that is comparable to their expected behavior in regular, low dimensional, BO settings. This means that there is no single best acquisition strategy that would rule them all. For Airline data, exploitative strategies EI and PI perform better than the explorative strategies. For Shape data, explorative strategies perform better than the exploitative strategies, which is natural as the minimum is outside the training data. For SMILES data, explorative strategies perform slightly better. There is no significant difference between different methods for the same acquisition function. The order of the performance is similar for both the tested methods. Also, as demonstrated in the previous experiments, the performance of the joint training for Airline and SMILES datasets is affected by overfitting.

6 | CONCLUSION

The goal of this work was to demonstrate the effect of different design choices in optimization tasks performed in high dimensional structured spaces. This work concentrated on methods that combine deep generative models, such as VAEs, and Gaussian processes. The biggest advantage of these methods is that they allow the optimization to be done in the smooth Euclidean low dimensional space formed by the generative model. This work demonstrated the effect of dimensionality of the latent manifold, optimization space restriction strategy, and acquisition functions on the optimization performance in three different optimization tasks.

The results of this work show that there is an optimal dimension for the deep generative model in which the optimization yields the best performance. In addition to this, the results show that different acquisition functions have similarly different performances in structured problems as they do in the case of regular, low dimensional BO; explorative acquisition functions stay explorative and exploitative ones stay exploitative. The most surprising result of this manuscript is that restricting the optimization space of the acquisition function beyond simple hyper rectangular optimization spaces yields no improvement due to the poor extrapolation properties of deep generative models. Last, the results show that tuning the latent space by jointly learning the latent space and the GP model might lead to overfitting and an overall drop in performance. This can be seen in the case of Airline and Shape datasets, where the disjoint training approach performed better.

The remaining open question and potential topic for further research is how to avoid the apparent overfitting of the VAE parameters when training the VAE and GPLVM models jointly. Another topic for further research is to use sparse VAE[53] in BO to avoid the problem of having to decide the optimal latent space dimensionality (as demonstrated in Section 4.1).

ACKNOWLEDGMENTS

The authors would like to thank Pablo García Moreno for his valuable feedback and discussions that helped us in greatly improving the manuscript. The authors also thank Austin Tripp for his feedback on the manuscript.

DATA AVAILABILITY STATEMENT

Research data are openly available at https://github.com/esiivola/hdssbo.

ORCID

Eero Siivola https://orcid.org/0000-0002-3926-9651
Andrei Paleyes https://orcid.org/0000-0002-3703-8163
Aki Vehtari https://orcid.org/0000-0003-2164-9469

REFERENCES

1. Greenhill S, Rana S, Gupta S, Vellanki P, Venkatesh S. Bayesian optimization for adaptive experimental design: a review. *IEEE Access*. 2020;8:13937-13948.
2. Wu J, Chen XY, Zhang H, Xiong LD, Lei H, Deng SH. Hyperparameter optimization for machine learning models based on Bayesian optimization. *J Electron Sci Technol*. 2019;17(1):26-40.
3. Calandra R, Seyfarth A, Peters J, Deisenroth MP. Bayesian optimization for learning gaits under uncertainty. *Ann Math Artif Intell*. 2016;76(1):5-23.
4. Sano S, Kadowaki T, Tsuda K, Kimura S. Application of Bayesian optimization for pharmaceutical product development. *J Pharm Innov*. 2020;15(3):333-343.
5. Maier M, Zwicker R, Akbari M, Rupenyan A, Wegener K. Bayesian optimization for autonomous process set-up in turning. *CIRP J Manuf Sci Technol*. 2019;26:81-87.
6. Griffiths RR, Hernández-Lobato JM. Constrained Bayesian optimization for automatic chemical design using variational autoencoders. *Chem Sci*. 2020;11(2):577-586.
7. Dhamala J, Ghimire S, Sapp JL, Horáček BM, Wang L. Bayesian optimization on large graphs via a graph convolutional generative model: application in cardiac model personalization. Paper presented at: International Conference on Medical Image Computing and Computer-Assisted Intervention; Springer; 2019:458–467.
8. Griffiths RR, Hernández-Lobato JM. Constrained Bayesian optimization for automatic chemical design. *Chemical science*. 2020;11(2): 577-586.
9. Kusner MJ, Paige B, Hernández-Lobato JM. Grammar variational autoencoder. *Proceedings of the 34th International Conference on Machine Learning, PMLR*. 2017;70:1945-1954. http://proceedings.mlr.press/v70/kusner17a.
10. Brochu E, Cora VM, De Freitas N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv Preprint*. 2010. https://www.cs.ubc.ca/tr/2009/tr-2009-23.
11. Shahriari B, Swersky K, Wang Z, Adams RP, De Freitas N. Taking the human out of the loop: a review of Bayesian optimization. *Proc IEEE*. 2015;104(1):148-175.
12. Wang Z, Zoghi M, Hutter F, Matheson D, De Freitas N, et al. Bayesian optimization in high dimensions via random embeddings. Paper presented at: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence; 2013:1778–1784.
13. Garnett R, Osborne MA, Hennig P. Active learning of linear embeddings for Gaussian processes. *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*. 2013;230-239. https://dl.acm.org/doi/10.5555/3020751.3020776.
14. Tripathy R, Bilionis I, Gonzalez M. Gaussian processes with built-in dimensionality reduction: applications to high-dimensional uncertainty propagation. *J Comput Phys*. 2016;321:191-223.
15. Groves M, Pyzer-Knapp EO. Efficient and scalable batch Bayesian optimization using K-means. Paper presented at: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence; 2019.
16. Bioucas-Dias JM, Figueiredo MA. A new TwIST: two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Trans Image Process*. 2007;16(12):2992-3004.
17. Kandasamy K, Schneider J, Póczos B. High dimensional Bayesian optimisation and bandits via additive models. Paper presented at: International Conference on Machine Learning; 2015:295–304.
18. Mutny M, Krause A. Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features. Paper presented at: Advances in Neural Information Processing Systems; 2018:9005–9016.
19. Oh C, Gavves E, Welling M. BOCK: Bayesian optimization with cylindrical kernels. *Proceedings of the 35th International Conference on Machine Learning*. 2018;80:3868-3877. http://proceedings.mlr.press/v80/oh18a.html.
20. Espinasse T, Gamboa F, Loubes JM. Parametric estimation for Gaussian fields indexed by graphs. *Probab Theory Relat Fields*. 2014;159: 117-155.
21. Jaquier N, Rozo L, Calinon S, Bürger M. Bayesian optimization meets Riemannian manifolds in robot learning. Paper presented at: Proceedings of the Conference on Robot Learning 100 of Proceedings of Machine Learning Research PMLR; 2020: 233–246.
22. Hebbal A, Brevault L, Balesdent M, Talbi EG, Melab N. Bayesian optimization using deep Gaussian processes with applications to aerospace system design. *Optim Eng*. 2020;22:1-41.
23. Huang W, Zhao D, Sun F, Liu H, Chang E. Scalable Gaussian process regression using deep neural networks. Paper presented at: Twenty-Fourth International Joint Conference on Artificial Intelligence; 2015.
24. Eissman S, Levy D, Shu R, Bartzsch S, Ermon S. Bayesian optimization and attribute adjustment. Paper presented at: 34th Conference on Uncertainty in Artificial Intelligence; 2018.
25. Tripp A, Daxberger E, Hernández-Lobato JM. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. Paper presented at: Advances in Neural Information Processing Systems 33 Pre-Proceedings (NeurIPS 2020).

26. Binois M, Ginsbourger D, Roustant O. A warped kernel improving robustness in Bayesian optimization via random embeddings. Paper presented at: International Conference on Learning and Intelligent Optimization, Springer; 2015:281–286.
27. Binois M, Ginsbourger D, Roustant O. On the choice of the low-dimensional domain for global optimization via random embeddings. *J Glob Optim*. 2020;76(1):69-90.
28. Hou X, Shen L, Sun K, Qiu G. Deep feature consistent variational autoencoder. Paper presented at: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV); 2017:1133–1141.
29. Yang Z, Hu Z, Salakhutdinov R, Berg-Kirkpatrick T. Improved Variational autoencoders for text modeling using dilated convolutions. Paper presented at: Proceedings of Machine Learning Research; 2017:3881–3890.
30. Roberts A, Engel J, Eck D., eds., *Hierarchical Variational Autoencoders for Music*; 2017.
31. Gómez-Bombarelli R, Wei JN, Duvenaud D, et al. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent Sci*. 2018;4(2):268-276.
32. Loaiza-Ganem G, Cunningham JP. The continuous Bernoulli: fixing a pervasive error in variational autoencoders. Paper presented at: 33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada; 2019:13287–13297.
33. Rasmussen CE. Gaussian processes in machine learning. Paper presented at: Summer School on Machine Learning, Springer; 2003: 63–71.
34. Snelson E, Ghahramani Z. Sparse Gaussian processes using pseudo-inputs. Paper presented at: Advances in Neural Information Processing Systems 18 (NIPS 2005); 2005:1257–1264.
35. Titsias M. Variational learning of inducing variables in sparse Gaussian processes. Paper presented at: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS) 5 JMLR Workshop and Conference Proceedings; 2009:567–574.
36. Lawrence ND. Gaussian process latent variable models for visualisation of high dimensional data. Paper presented at: Advances in Neural Information Processing Systems; 2004:329–336.
37. Thompson WR. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*. 1933;25(3/4):285-294.
38. Chapelle O, Li L. An empirical evaluation of Thompson sampling. In: Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ, eds. *Advances in Neural Information Processing Systems* 2011;24:2249-2257. https://papers.nips.cc/paper/2011/hash/e53a0a2978c28872a4505bdb51db06dc-Abstract.html.
39. Jones DR, Schonlau M, Welch WJ. Efficient global optimization of expensive black-box functions. *J Glob Optim*. 1998;13(4):455-492.
40. Močkus J. On the Bayes methods for seeking the extremal point. *IFAC Proc*. 1975;8:428-431.
41. Kushner HJ. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *J Basic Eng*. 1964; 86:97-106.
42. Srinivas N, Krause A, Kakade S, Seeger M. Gaussian process optimization in the bandit setting: no regret and experimental design. Paper presented at: Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML), Omnipress; 2010:1015–1022.
43. Moshtagh N. Minimum volume enclosing ellipsoid. *Convex Optim*. 2005;111:1-9.
44. Chen T, Li M, Li Y, et al. MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv Preprint*. 2015.
45. Paleyes A, Pullin M, Mahsereci M, Lawrence N, Gonzalez J. Emulation of physical processes with Emukit. Paper presented at: Second Workshop on Machine Learning and the Physical Sciences, NeurIPS; 2019.
46. Kingma DP, Ba J. Adam: A method for stochastic optimization. *arXiv Preprint*. 2014. https://arxiv.org/abs/1412.6980.
47. Box GE, Jenkins GM, Reinsel GC. *Time Series Analysis: Forecasting and Control*. Vol 734. Philadelphia: John Wiley & Sons; 2011.
48. Lu X, Gonzalez J, Dai Z, Lawrence N. Structured variationally auto-encoded optimization. Paper presented at: International Conference on Machine Learning; 2018:3267–3275.
49. Matthey L, Higgins I, Hassabis D, Lerchner A. dSprites: Disentanglement testing Sprites dataset. https://github.com/deepmind/dsprites-dataset/; 2017.
50. Fink T, Reymond JL. Virtual exploration of the chemical universe up to 11 atoms of C, N, O, F: assembly of 26.4 million structures (110.9 million stereoisomers) and analysis for new ring systems, stereochemistry, physicochemical properties, compound classes, and drug discovery. *J Chem Inf Model*. 2007;47(2):342-353.
51. Weininger D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J Chem Inf Comput Sci*. 1988;28(1):31-36.
52. Ertl P, Schuffenhauer A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J Chem*. 2009;1(1):8.
53. Tonolini F, Jensen BS, Murray-Smith R. Variational sparse coding. Paper presented at: Uncertainty in Artificial Intelligence, PMLR; 2020:690–700.

How to cite this article: Siivola E, Paleyes A, González J, Vehtari A. Good practices for Bayesian optimization of high dimensional structured spaces. *Applied AI Letters*. 2021;2(2):e24. https://doi.org/10.1002/ail2.24