Dimensionality Reduction Techniques for Global Bayesian Optimisation

Luo Long LUO.LONG@MATHS.OX.AC.UK
Coralia Cartis CARTIS@MATHS.OX.AC.UK
Paz Fink Shustin PAZ.FINKSHUSTIN@MATHS.OX.AC.UK
Mathematical Institute, University of Oxford, Oxford, UK

Abstract

Bayesian Optimisation (BO) is a state-of-the-art global optimisation technique for black-box problems where derivative information is unavailable and sample efficiency is crucial. However, improving the general scalability of BO has proved challenging. Here, we explore Latent Space Bayesian Optimisation (LSBO), that applies dimensionality reduction to perform BO in a reduced-dimensional subspace. While early LSBO methods used (linear) random projections (Wang et al., 2013 [27]), we employ Variational Autoencoders (VAEs) to manage more complex data structures and general DR tasks. Building on Grosnit *et al.* (2021) [12], we analyse the VAE-based LSBO framework, focusing on VAE retraining and deep metric loss. We suggest a few key corrections in their implementation, originally designed for tasks such as molecule generation, and reformulate the algorithm for broader optimisation purposes. Our numerical results show that structured latent manifolds improve BO performance. Additionally, we examine the use of the Matérn-$\frac{5}{2}$ kernel for Gaussian Processes in this LSBO context. We also integrate Sequential Domain Reduction (SDR), a standard global optimization efficiency strategy, into BO. SDR is included in a GPU-based environment using *BoTorch*, both in the original and VAE-generated latent spaces, marking the first application of SDR within LSBO.

1. Introduction

Global Optimisation (GO) aims to find the (approximate) global optimum of a smooth function f within a region of interest, possibly without the use of derivative problem information and with careful handling of often-costly objective evaluations. In particular, we focus on the GO problem,

$$f^* = \min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}), \tag{P}$$

where $\mathcal{X} \subseteq \mathbb{R}^D$ represents a feasible region, and f is a black-box, continuous function in (high) dimensions D. Bayesian Optimisation (BO) is a state-of-the-art GO framework that constructs a probabilistic model, typically a Gaussian Process (GP), of f, and uses an acquisition function to guide sampling and efficiently search for the global optimum [10]. BO balances exploration and exploitation but suffers from scalability issues in high-dimensions [15]. To mitigate this, Dimensionality Reduction (DR) techniques can be used, allowing BO to operate in a lower-dimensional subspace where it is more effective.

Contributions. We investigate scaling up BO using DR techniques, focusing on VAEs and the Random Embedding Global Optimisation (REGO) framework [5]. Our work extends the algorithm [12], incorporating the Matérn-$\frac{5}{2}$ kernel, which enhances the flexibility and robustness of the method. Moreover, we conduct a comparative analysis of these approaches with standard BO techniques enhanced by Sequential Domain Reduction (SDR) [23]. The first contribution of this work is that we **propose and implement SDR with BO within the *BoTorch* framework [1]**, utilising GPU-based computation for efficiency. Furthermore, we **propose three BO-VAE algorithms, two of which are innovatively combined with SDR in the VAE-generated latent space** to boost optimisation performance. This marks the first instance of SDR being integrated with BO in the context of VAEs. We also investigate the effects of VAE retraining [25] and deep metric loss [12] on the optimisation process, emphasising the advantages of having a well-structured latent space for improved performance. Finally, we **compare our BO-VAE algorithms with the REMBO method** [27] on low effective dimensionality problems, evaluating VAEs versus random embeddings as two different DR techniques in terms of optimisation performance.

2. Preliminaries

Bayesian Optimisation. BO relies on two fundamental components: a GP prior and an acquisition function. Given a dataset of size n, $\mathcal{D}_n = \{\mathbf{x}_i, f(\mathbf{x}_i)\}_{i=1}^{n}$, the function values $\mathbf{f}_{1:n}$ are modelled as realisations of a Gaussian Random Vector (GRV) $\mathbf{F}_{1:n}$ under the GP prior. The distribution is characterised by a mean $\mathbb{E}_{\mathbf{F}_{1:n}}$ and covariance $K_{\mathbf{F}_{1:n}\mathbf{F}_{1:n}}$, where $\mathbf{F}_{1:n} \sim \mathcal{N}(\mathbb{E}_{\mathbf{F}_{1:n}}, K_{\mathbf{F}_{1:n}\mathbf{F}_{1:n}})$, using the Matérn-5/2 kernel. For an arbitrary unsampled point $\mathbf{x}$, the predicted function value $f(\mathbf{x})$ is inferred from the posterior distribution: $F(\mathbf{x}) \sim \mathcal{N}(\mu(\mathbf{x}|\mathcal{D}_n), \sigma^2(\mathbf{x}|\mathcal{D}_n))$, with $\mu(\mathbf{x}|\mathcal{D}_n) = \mathbb{E}_F + K^{T}_{\mathbf{F}_{1:n}F} K^{-1}_{\mathbf{F}_{1:n}\mathbf{F}_{1:n}} (\mathbf{f}_{1:n} - \mathbb{E}_{\mathbf{F}_{1:n}})$, $\sigma^2(\mathbf{x}|\mathcal{D}_n) = K_{FF} - K^{T}_{\mathbf{F}_{1:n}F} K^{-1}_{\mathbf{F}_{1:n}\mathbf{F}_{1:n}} K_{\mathbf{F}_{1:n}F}$. BO uses the posterior mean $\mu(\cdot)$ and variance $\sigma^2(\cdot)$ in an acquisition function to guide sampling. In this work, we focus on Expected Improvement (EI): $u(\mathbf{x}|\mathcal{D}_n) = \mathbb{E}[\max\{F(\mathbf{x}) - f_n^*, 0\}|\mathcal{D}_n]$, where $f_n^* = \max_{m \leq n} f(\mathbf{x}_m)$ is the highest observed value. To enhance BO, we incorporate SDR to refine the search region based on the algorithm's best-found values, updating the region every few iterations to avoid missing the global optimum. To accelerate BO process, we propose to implement SDR [23] within the traditional BO framework such that the search region can be refined to locate the global minimiser more efficiently according to the minimum function values found so far by the algorithm. Compared to the traditional SDR implementation that updates the search region at each iteration, we propose updating the region after a set number of iterations to avoid premature exclusion of the global optimum. Algorithm 2 in Appendix B.1 outlines the BO-SDR approach.

Variational Autoencoders. DR methods reduce the number of features in a dataset while preserving essential information [26]. They can often be framed as an *Encoder-Decoder* process, where the *encoder* maps high-dimensional (HD) data to a lower-dimensional latent space, and the *decoder* reconstructs the original data. We focus on VAEs [8, 19], a DR technique using Bayesian Variational Inference (VI) [13, 17]. VAEs utilise neural networks as encoders and decoders to generate latent manifolds. The probabilistic framework of a VAE consists of the encoder $q_{\boldsymbol{\phi}}(\cdot|\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{Z}$ parameterised by $\boldsymbol{\phi}$ which turns an input data $\mathbf{x} \in \mathbb{R}^D$ from some distribution into a distribution on the latent variable $\mathbf{z} \in \mathbb{R}^d$ ($d \ll D$), and the decoder $p_{\boldsymbol{\theta}}(\cdot|\mathbf{z}) : \mathcal{Z} \rightarrow \mathcal{X}$ parameterised by $\boldsymbol{\theta}$ which reconstructs $\mathbf{x}$ as $\hat{\mathbf{x}}$ given samples from the latent distribution. The VAE's objective is to maximise the Evidence Lower BOund (ELBO):$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = \ln p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})] = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\ln p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - D_{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})]$, where $\ln p_{\boldsymbol{\theta}}(\mathbf{x})$ is the marginal log-likelihood, and $D_{KL}(\cdot\|\cdot)$ is the non-negative Kullback-Leibler Divergence between the true and the approximate posteriors. The prior is usually set to $\mathcal{N}(\mathbf{0}, \mathbf{I})$, and the posterior is parametrised as Gaussians with diagonal covariance matrices, making ELBO optimisation tractable via the ”reparameterisation trick”

[19]. Given $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\Sigma}(\mathbf{x}))$, the latent variable $\mathbf{z}$ is sampled as $\mathbf{z} = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\Sigma}(\mathbf{x})\boldsymbol{\xi}$, $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, enabling gradient-based optimisation with Adam [18].

3. Algorithms

As mentioned above, DR techniques help reduce the optimisation problem's dimensionality. Using a VAE within BO allows standard BO approach to be applied to larger scale problems, as then, we solve a GP regression sub-problem in the generated (smaller dimensional) latent space $\mathcal{Z}$. The BO-VAE approach[1], instead of solving (P) directly, attempts to solve

$$f^* = \min_{\mathbf{z} \in \mathcal{Z}} \mathbb{E}_{p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z})} \left[f(\mathbf{x}) \right], \tag{1}$$

where $\boldsymbol{\theta}^*$ is the optimal decoder network parameter. Therefore, it is implicitly assumed that the optimal point $\mathbf{x}^*$ can be obtained from the optimal decoder with some probability given some latent data $\mathbf{z}$ by the associated optimal encoder $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x})$ [12], $\exists\, \mathbf{z} \in \mathcal{Z}, \mathbb{P}\left[\mathbf{x}^* \sim p_{\boldsymbol{\theta}^*}(\cdot|\mathbf{z})\right] > 0$.

When fitting the GP surrogate, we follow [12] and use Deep Metric Loss (DML) to generate well-structured VAE-generated latent spaces. Specifically, we apply the soft triplet loss and retrain the VAEs following [25] to adapt to new points from the GP and optimise the black-box objective efficiently. Additionally, we implement SDR in the latent space to accelerate the BO process. Algorithm 1 outlines our BO-VAE approach with SDR, consisting of the pre-training of a standard VAE on the unlabelled dataset $\mathcal{D}_{\mathbb{U}}$ (line 1) and optional retraining with soft triplet loss to structure the latent space by gradually adjusting the network parameters of the encoder and decoder. When the soft triplet loss is used in retraining the VAE, the modified VAE ELBO $\mathcal{L}_{DML}(\cdot)$ is used in line 4 instead; see Appendix B.3 for details. The BO-VAE algorithm with DML is included in Appendix B.3 as Algorithm 4. For comparison, we provide a baseline BO-VAE algorithm without retraining or DML (Appendix B.1, Algorithm 3). Theorem 1 in [12] offers a regret analysis with a sub-linear convergence rate, providing a valuable theoretical foundation. However, the proof relies on the assumption of a Gaussian kernel, limiting its direct applicability when using the Matérn kernel, as we do here. Despite this limitation, the theorem provides key insights supporting the BO-VAE approach. Our ongoing work addresses this gap, and a similar result specifically tailored to the Matérn kernel is delegated to future work.

4. Numerical Study

We conduct numerical experiments with the three BO-VAE algorithms (Algorithms 1, 3, 4) within the *BoTorch* framework [1]. We explore cases where $d = 2, 5$ for $D = 10$, and $d = 2, 10, 50$ for $D = 100$. The results reveal that, for a fixed ambient dimension D, larger latent dimensions, particularly $d = 50$, tend to degrade performance due to the reduced VAE generalisation capacity. In contrast, smaller latent dimensions ($d = 2, 5$) yield better results, as VAEs then can give more efficient latent data representations, and the BO can solve such reduced problems more efficiently. In this paper, we present experimental results for the case $D = 100$ and $d = 2$, which strongly highlight the advantages of SDR in latent space optimisation and illustrate the three BO-VAE algorithms. The encoder structure of the VAE used is $[100, 30, 2]$[2], and the decoder structure is $[2, 30, 100]$. For

[1] For brevity, we use BO-VAE to refer the approach of combing VAEs with BO.

[2] It indicates a three-layer feedforward neural network: the input layer has 100 neurons, followed by a hidden layer with 30 neurons, and finally an output layer with 2 neurons. Similarly for the others.

```
Algorithm 1: Retraining BO-VAE Algorithm with SDR
Data: Labelled dataset $\mathcal{D}^{l=1}_{\mathbb{L}} = \{\mathbf{x}_i, f(\mathbf{x}_i)\}_{i=1}^{N}$, unlabelled dataset $\mathcal{D}_{\mathbb{U}} = \{\mathbf{x}_i\}_{i=1}^{M}$, budget $B$, periodic
      frequency $q$, initial bound $R^0$ in latent space $\mathcal{Z}$, EI acquisition function $u(\cdot)$, the encoder and
      decoder models from a VAE, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) : \mathcal{X} \to \mathcal{Z}$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) : \mathcal{Z} \to \mathcal{X}$.
Result: Minimum function value $f_{min}$ found by the algorithm.
Pre-train the VAE model $V^{l=0}_{\mathcal{D}_{\mathbb{L}}}$ with $\mathcal{D}_{\mathbb{U}}$: $\boldsymbol{\theta}^*_0, \boldsymbol{\phi}^*_0 = \arg\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{D}_{\mathbb{U}})$;
Set $\boldsymbol{\theta}^*_1 \leftarrow \boldsymbol{\theta}^*_0$, $\boldsymbol{\phi}^*_1 \leftarrow \boldsymbol{\phi}^*_0$, $V^{l=1}_{\mathcal{D}_{\mathbb{L}}} \leftarrow V^{l=0}_{\mathcal{D}_{\mathbb{L}}}$;
for $l = 1$ to $L \equiv \lceil B/q \rceil$ do
    Train the VAE model $V^{l}_{\mathcal{D}^l_{\mathbb{L}}}$ on $\mathcal{D}_{\mathbb{L}}$: $\boldsymbol{\theta}^*_l, \boldsymbol{\phi}^*_l = \arg\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{D}^l_{\mathbb{L}})$;
    Compute the latent dataset $\mathcal{D}^l_{\mathbb{Z}} = \{\mathbf{z}_i, f(\mathbf{x}_i)\}_{i=1}^{N+l\cdot q} = \{\mathbb{E}_{q_{\boldsymbol{\phi}^*_l}(\mathbf{z}|\mathbf{x_i})}[\mathbf{z}], f(\mathbf{x}_i)\}_{i=1}^{N+l\cdot q}$;
    Initialise $\mathcal{D}^{l;k=0}_{\mathbb{L}} \leftarrow \mathcal{D}^l_{\mathbb{L}}$ and $\mathcal{D}^{l;k=0}_{\mathbb{Z}} \leftarrow \mathcal{D}^l_{\mathbb{Z}}$;
    Initialise SDR with $R^0$;
    for $k = 0$ to $q - 1$ do
        Fit a Gaussian Process (GP) model $h_{l;k} : \mathcal{Z} \to \mathbb{R}$ on $\mathcal{D}^{l;k}_{\mathbb{Z}} = \{\mathbf{z}_i, f(\mathbf{x}_i)\}_{i=0}^{N+l\cdot q+k}$;
        Solve for the next latent point: $\hat{\mathbf{z}}_{l;k+1} = \arg\max_{\mathbf{z}} u(\mathbf{z}|\mathcal{D}^{l;k}_{\mathbb{Z}})$;
        Obtain the new sample $\hat{\mathbf{x}}_{l;k+1}$: $\hat{\mathbf{x}}_{l;k+1} \sim p_{\boldsymbol{\theta}^*_l}(\cdot|\hat{\mathbf{z}}_{l;k+1})$;
        Evaluate the objective function at the new sample: $f(\hat{\mathbf{x}}_{l;k+1})$;
        Augment the datasets:
         $\mathcal{D}^{l;k+1}_{\mathbb{L}} \leftarrow \mathcal{D}^{l;k}_{\mathbb{L}} \cup \{\hat{\mathbf{x}}_{l;k+1}, f(\hat{\mathbf{x}}_{l;k+1})\}$, $\mathcal{D}^{l;k+1}_{\mathbb{Z}} \leftarrow \mathcal{D}^{l;k}_{\mathbb{Z}} \cup \{\hat{\mathbf{z}}_{l;k+1}, f(\hat{\mathbf{x}}_{l;k+1})\}$;
        Update the search domain: $R^{k+1} \leftarrow R^k$ using SDR given $\mathcal{D}^{l;k+1}_{\mathbb{Z}}$ ;
    end
    Augment the outer-loop datasets: $\mathcal{D}^{l+1}_{\mathbb{L}} \leftarrow \mathcal{D}^{l;q}_{\mathbb{L}}$, $\mathcal{D}^{l+1}_{\mathbb{Z}} \leftarrow \mathcal{D}^{l;q}_{\mathbb{Z}}$;
end
```

the REMBO comparison, the VAE used has $[100, 25, 5]$ for the encoder and $[5, 25, 100]$ for the decoder. The activation function is Soft-plus. We set the budget $B = 350$ and $q = 50$ to retrain 7 times for Algorithms 1 and 4. In practice, to improve computational costs, the input spaces of VAEs can be normalised to a fixed range, such as a hypercube, simplifying both pre-training and retraining steps and avoiding the need to train multiple VAEs. By applying an additional scaling between the specific problem domain and the fixed VAE input space, we leverage the VAE's generalisation ability to construct latent manifolds for datasets from diverse problem domains. In our experiments, we set the fixed VAE input space to $[-3, 3]^D$. Further details of our experiments are provided in Appendix C.

Numerical Illustration of SDR within BO-VAE. To demonstrate the effectiveness of SDR within VAE-generated latent subspaces, we use Algorithm 3 to generate results in Figure 1. The figure shows that SDR leads to faster convergence and lower, potentially optimal, function values. Additionally, Figure 2 illustrates the results of the three BO-VAE algorithms, where Algorithm 4 generally outperforms the others, achieving better global optima. This improvement is due to the soft triplet loss, which better structures the latent subspaces and enhances the GP surrogate's efficiency.

Algorithm Comparisons. We first compare the three BO-VAE algorithms with BO-SDR on Test Set 1 (Table 3) and then against REMBO [27] on Test Set 2 (see Appendix A.2). REMBO addresses (P) by solving a reduced problem in a low-dimensional subspace: $\min_{\mathbf{y}\in\mathbb{R}^d} f(\mathbf{A}\mathbf{y}) =$

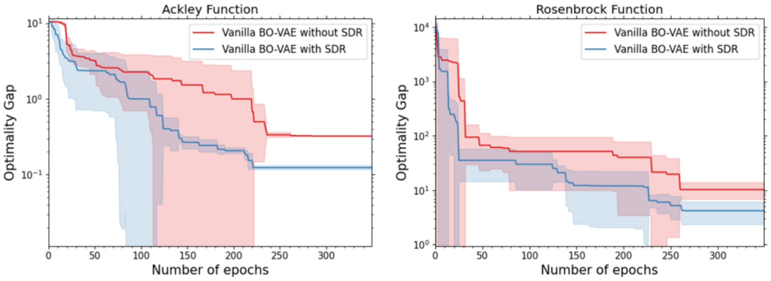

Figure 1: Comparisons of Vanilla BO-VAE algorithm (Algorithm 3) with and without SDR on 100D Ackley and Rosenbrock problems. The means and the standard deviations (shaded areas) of the minimum function values found are plotted across 5 repeated runs.

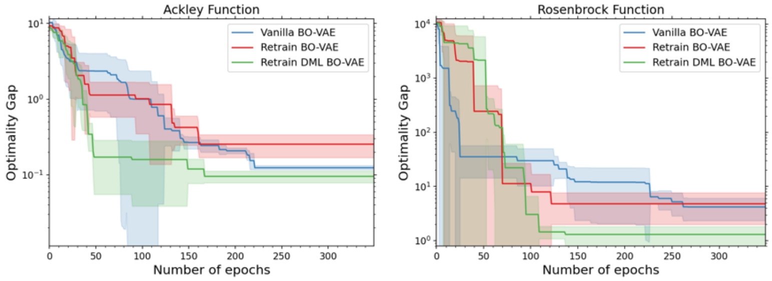

Figure 2: Comparisons of Algorithm 3 (Vanilla BO-VAE), Algorithm 1 (Retrain BO-VAE), and Algorithm 4 (Retrain DML BO-VAE) in solving 100D Ackley and Rosenbrock problems. The means and the standard deviations (shaded areas) of the minimum function values found are plotted across 5 repeated runs.

$\min_{\mathbf{y}\in\mathbb{R}^d} g(\mathbf{y})$, subject to $\mathbf{y} \in \mathcal{Y} = [-\delta, \delta]^d$. Here $\mathbf{A}$ is a $D \times d$ Gaussian matrix for random embedding, with $d \ll D$. Solving $g(\mathbf{y})$ in the reduced subspace is equivalent to solving $f(\mathbf{Ay})$. In this context, the Gaussian matrix $\mathbf{A}$ serves as a (linear) encoder (for dimensionality reduction), while $\mathbf{A}^T$ acts as a decoder. For the REMBO comparisons, we used $d = d_e + 1$, where d_e is the effective dimensionality, and set $\delta = 2.2\sqrt{d_e}$ based on [4]. Each (randomised) algorithm was run twice on each problem in Test Set 1. Each (randomised) problem in Test Set 2 was run twice, yielding 10 test problems. The algorithms we are comparing are denoted by BO-SDR (Algorithm 2), V-BOVAE (Algorithm 3), R-BOVAE (Algorithm 1), and S-BOVAE (Algorithm 4).

Results with accuracy levels $\tau = 10^{-1}$ and $\tau = 10^{-3}$ are summarised in Tables 1, 2 and Figure 3. From these results, it can be seen that BO-VAE algorithms solve more problems compared to BO-SDR and REMBO algorithms. While BO-SDR may struggle with scalability, REMBO's low problem-solving percentages are likely due to the over-exploration of the boundary projections [21] and the embedding subspaces failing to accurately capture the global minimisers. To address this, [27] recommends restarting REMBO to improve the success rate. Meanwhile, Algorithm 4 (S-BOVAE) consistently performed best due to its structured latent spaces.

	$\tau = 10^{-1}$	$\tau = 10^{-3}$
BO-SDR	10%	0%
V-BOVAE	100%	50%
S-BOVAE	100%	50%
R-BOVAE	100%	50%

Table 1: Average percentage of problems solved in Test Set 1 for $\tau = 10^{-1}$ and 10^{-3}.

	$\tau = 10^{-1}$	$\tau = 10^{-3}$
BO-SDR	20%	0%
V-BOVAE	90%	20%
S-BOVAE	100%	40%
R-BOVAE	100%	30%
REMBO	50%	10%

Table 2: Average percentage of problems solved in Test Set 2 for $\tau = 10^{-1}$ and 10^{-3}.

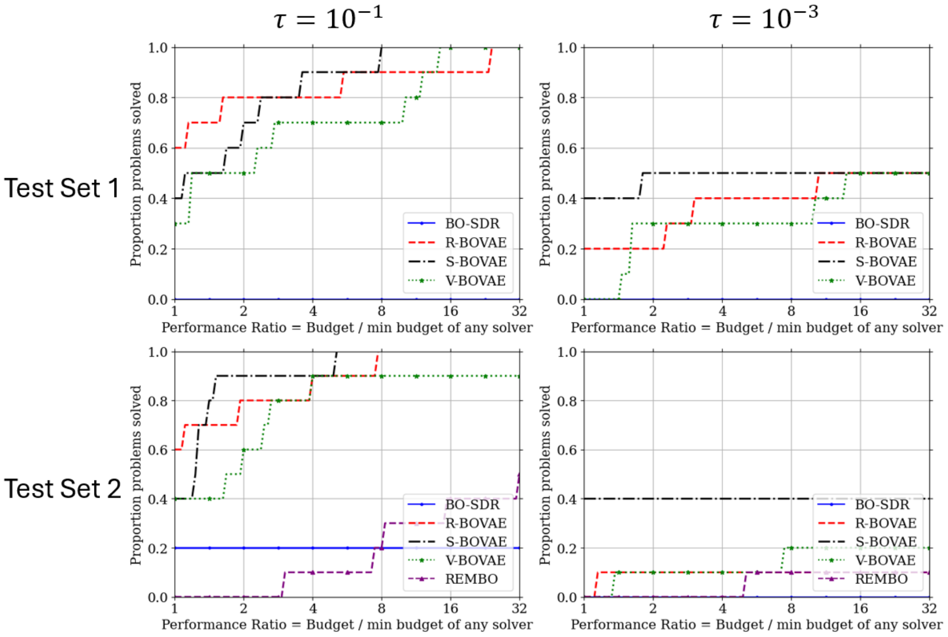

Figure 3: Performance Profiles on Test Sets 1 & 2 when $\tau = 10^{-1}$ and 10^{-3}.

5. Conclusion and Future Work

In this work, we have explored dimensionality reduction techniques to enhance the scalability of BO. The use of VAEs offers an alternative and more general approach for GP fitting in low-dimensional latent subspaces, alleviating the curse of dimensionality. Unlike REMBO, which primarily targets low-rank functions, VAE-based LSBO is effective for both high-dimensional full-rank and low-rank functions. Although BO-VAE reduces function values effectively, the optimality gap remains constrained by noise from the VAE loss, as seen in Figures 1 and 2. To address this, implementations of data weights and different GP initialisation are the potential future directions. Additionally, SDR struggles in high dimensions; adopting methods like domain refinement based on threshold probabilities [22] may improve performance.

Acknowledgments The second author's (CC) research was supported by the Hong Kong Innovation and Technology Commission's Center for Intelligent Multi-dimensional Analysis (InnoHK Project CIMDA).

References

[1] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020. URL http://arxiv.org/abs/1910.06403.

[2] Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space, 2016. URL https://arxiv.org/abs/1511.06349.

[3] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae, 2018. URL https://arxiv.org/abs/1804.03599.

[4] Coralia Cartis and Adilet Otemissov. A dimensionality reduction technique for unconstrained global optimization of functions with low effective dimensionality, 2020. URL https://arxiv.org/abs/2003.09673.

[5] Coralia Cartis, Estelle Massart, and Adilet Otemissov. Global optimization using random embeddings, 2021. URL https://arxiv.org/abs/2107.12102.

[6] Coralia Cartis, Lindon Roberts, and Oliver Sheridan-Methven. Escaping local minima with local derivative-free methods: a numerical investigation. *Optimization*, 71(8):2343–2373, February 2021. ISSN 1029-4945. doi: 10.1080/02331934.2021.1883015. URL http://dx.doi.org/10.1080/02331934.2021.1883015.

[7] Coralia Cartis, Jaroslav Fowkes, and Lindon Roberts. Optimization resources: A collection of software and resources for nonlinear optimization. https://lindonroberts.github.io/opt/resources.html#data-performance-profiles, n.d.

[8] Carl Doersch. Tutorial on variational autoencoders, 2021. URL https://arxiv.org/abs/1606.05908.

[9] P.A. Ernesto and U.P. Diliman. MVF—multivariate test functions library in c for unconstrained global optimization, 2005.

[10] Peter I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. URL https://arxiv.org/abs/1807.02811.

[11] Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. Cyclical annealing schedule: A simple approach to mitigating kl vanishing, 2019. URL https://arxiv.org/abs/1903.10145.

[12] Antoine Grosnit, Rasul Tutunov, Alexandre Max Maraval, Ryan-Rhys Griffiths, Alexander I. Cowen-Rivers, Lin Yang, Lin Zhu, Wenlong Lyu, Zhitang Chen, Jun Wang, Jan Peters, and Haitham Bou-Ammar. High-dimensional bayesian optimisation with variational autoencoders and deep metric learning, 2021. URL https://arxiv.org/abs/2106.03609.

[13] G. E. Hinton and D. Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 5–13, 1993.

[14] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015.

[15] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla bayesian optimization performs great in high dimensions, 2024. URL https://arxiv.org/abs/2402.02229.

[16] Haque Ishfaq, Assaf Hoogi, and Daniel Rubin. Tvae: Triplet-based variational autoencoder using metric learning. *arXiv preprint arXiv:1802.04403*, 2018. URL https://arxiv.org/abs/1802.04403.

[17] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. In *Learning in Graphical Models*, pages 105–161. Springer, 1998.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

[20] J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20:172–191, 2009.

[21] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for bayesian optimization in embedded subspaces. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4752–4761. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/nayebi19a.html.

[22] Sudeep Salgia, Sattar Vakili, and Qing Zhao. A domain-shrinking based bayesian optimization algorithm with order-optimal regret performance, 2021. URL https://arxiv.org/abs/2010.13997.

[23] Nielen Stander and Kenneth Craig. On the robustness of a simple domain reduction scheme for simulation-based optimization. *International Journal for Computer-Aided Engineering and Software (Eng. Comput.)*, 19, 06 2002. doi: 10.1108/02644400210430190.

[24] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. https://www.sfu.ca/~ssurjano/, 2013.

[25] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[26] S. Velliangiri, S. Alagumuthukrishnan, and S. I. Thankumar Joseph. A review of dimensionality reduction techniques for efficient computation. *Procedia Computer Science*, 165: 104–111, 2019. doi: 10.1016/j.procs.2020.01.079. URL https://doi.org/10.1016/j.procs.2020.01.079.

[27] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. De Freitas. Bayesian optimization in high dimensions via random embeddings. In *International Joint Conference on Artificial Intelligence*, pages 1778–1784, 2013.

[28] Peter Wu, SaiKrishna Rallabandi, Alan W. Black, and Eric Nyberg. Ordinal triplet loss: Investigating sleepiness detection from speech. In *Proc. Interspeech 2019*, pages 2403–2407, 2019. doi: 10.21437/Interspeech.2019-2278. URL http://dx.doi.org/10.21437/Interspeech.2019-2278.

Appendix A. Test Sets

A.1. High-dimensional Full-rank Test Set

#	Function	Dimension(s)	Domain	Global Minimum
1	Ackley [9]	D	$\mathbf{x} \in [-30, 30]^D$	0
2	Levy [24]	D	$\mathbf{x} \in [-10, 10]^D$	0
3	Rosenbrock [24]	D	$\mathbf{x} \in [-5, 10]^D$	0
4	Styblinski-Tang [24]	D	$\mathbf{x} \in [-5, 5]^D$	$-39.16599 \times D$
5	Rastrigin Function [24]	D	$\mathbf{x} \in [-5.12, 5.12]^D$	0

Table 3: Benchmark high-dimensional full-rank test problems from [6, 7].

A.2. High-dimensional Low-rank Test Set

The low-rank test set, or Test Set 2, comprises D-dimensional low-rank functions generated from the low-rank test functions listed in Table 4. To construct these D-dimensional functions with low effective dimensionality, we adopt the methodology proposed in [27].
Let $\bar{h}(\bar{\mathbf{x}})$ be any function from Table 4 with dimension d_e and the given domain scaled to $[-1, 1]^{d_e}$. The first step is to append $D - d_e$ fake dimensions with zero coefficients to $\bar{h}(\bar{\mathbf{x}})$:

$$h(\mathbf{x}) = \bar{h}(\bar{\mathbf{x}}) + 0 \cdot x_{d_e+1} + \cdots + 0 \cdot x_D.$$

Then, we rotate the function $h(\mathbf{x})$ for a non-trivial constant subspace by applying a random orthogonal matrix $\mathbf{Q}$ to $\mathbf{x}$. Hence, we obtain our D-dimensional low-rank test function, which is given by

$$f(\mathbf{x}) = h(\mathbf{Q}\mathbf{x}).$$

It is noteworthy that the first d_e rows of $\mathbf{Q}$ form the basis of the effective subspace $\mathcal{T}$ of f, while the last $D - d_e$ rows span the constant subspace $\mathcal{T}^{\perp}$.

#	Function	Effective Dimensions d_e	Domain	Global Minimum
1	low-rank Ackley [9]	4	$\mathbf{x} \in [-5, 5]^4$	0
2	low-rank Rosenbrock [24]	4	$\mathbf{x} \in [-5, 10]^4$	0
3	low-rank Shekel 5 [24]	4	$\mathbf{x} \in [0, 10]^4$	-10.1532
4	low-rank Shekel 7 [24]	4	$\mathbf{x} \in [0, 10]^4$	-10.4029
5	low-rank Styblinski-Tang [24]	4	$\mathbf{x} \in [-5, 5]^4$	-156.664

Table 4: Benchmark high-dimensional low-rank test problems from [4, 6].

Appendix B. Additional Details

B.1. Sequential Domain Reduction

To formally introduce SDR [23], let us denote $\mathbf{x}^{(k)} \in \mathbb{R}^D$ be the current optimal position, i.e., the centre point of the current sub-region, at iteration k with each component being bounded, $x_i^{l,k} \leq$

$x_i \leq x_i^{u,k}, i \in \{1, \ldots, D\}$. Initially, when $k = 0$, we construct the first Region of Interest (RoI) centring at $\mathbf{x}^{(0)}$ with lower and upper bounds being

$$x_i^{l,0} = x_i^{(0)} - \frac{r_i^{(0)}}{2},\ x_i^{u,0} = x_i^{(0)} + \frac{r_i^{(0)}}{2},\ i \in \{1, \ldots D\}, \tag{2}$$

where $r_i^{(0)}$ is the initial range value computed from the upper and lower bounds of the initial search domain. Now, suppose we are progressing from iterations $k-1$ to k and that the best observations are $\mathbf{x}^{k-1}$ and $\mathbf{x}^k$ up to the $(k-1)$-th and k-th iterations respectively. To update and contract on the RoI, we first determine an oscillation indicator along dimension i at iteration k as

$$c_i^{(k)} = d_i^{(k)} d_i^{(k-1)} \text{ with } d_i^{(j)} = \frac{2(x_i^{(j)} - x_i^{(j-1)})}{r_i^{(j-1)}},\ j = k, k-1, \tag{3}$$

where $r_i^{(k-1)}$ is the RoI size along dimension i at iteration $k-1$. Then, we normalise it as

$$\hat{c}_i^{(k)} = \sqrt{|c_i^{(k)}|}\ sign(c_i^{(k)}), \tag{4}$$

where $sign(\cdot)$ is the standard sign function. Then, the contraction parameter along dimension i at iteration k is

$$\gamma = \frac{\gamma_p(1+\hat{c}) + \gamma_o(1-\hat{c})}{2}, \tag{5}$$

where the indices i, k have been intentionally omitted for clarity and to avoid complex notations. Here, the parameter γ_o, typically set between 0.5 and 0.7, is a shrinkage factor to dampen oscillation. This parameter controls the reduction of the RoI, facilitating more stable and efficient convergence towards the global optimum. Meanwhiel, γ_p indicates the pure panning behaviour and is typically set as a unity. To shrink the RoI, we utilise a zooming parameter η to update the range along each dimension, i.e,

$$r_i^{(k)} = \lambda_i r_i^{(k-1)},\ \text{where } \lambda_i = \eta + |d_i^{(k)}|(\gamma - \eta). \tag{6}$$

λ_i represents the contraction rate along dimension i and η typically lies in $[0.5, 1)$. Below, we present the Bayesian Optimisation algorithms innovatively with SDR in the ambient and the VAE-generated latent spaces.

B.2. Methodology for Comparing Algorithms and Solvers

To evaluate performances of different algorithms/solvers fairly, we adopt the methodology from [6], using performance and data profiles as introduced in [20].

Performance profiles. A performance profile compares how well solvers perform on a problem set under a budget constraint. For a solver s and problem p, the performance ratio is:

$$r_{p,s} = \frac{M_{p,s}}{\min_{s \in \mathcal{S}} M_{p,s}},$$

where $M_{p,s}$ is a performance metric, typically the number of function evaluations required to meet the stopping criterion:

$$N_p(s;\tau) = \#\ \text{evaluations to achieve } f_k^* \leq f^* + \tau(f_0^* - f^*),$$

Algorithm 2: Bayesian Optimisation with Sequential Domain Reduction

```
Data: Initial dataset $\mathcal{D}_0 = \{\mathbf{X}_0, \mathbf{f}_0\}$, budget $B$, acquisition function $u(\cdot)$, initial search domain
      $\mathcal{X}$, parameters $\gamma_o, \gamma_p, \eta$, minimum region of interest size $t$, and step size $\xi$.
Result: Minimum value $f_{min}$ found by the algorithm.
Initialise SDR by computing the initial Region of Interest (RoI) $R^{(0)}$ according to the bounds;
for $k = 0, \ldots, B-1$ do
    Fit Gaussian process $\mathcal{GP}_k$ to current data $\mathbf{X}_k$ and $\mathbf{f}_k$;
    Find the next iterate $\mathbf{x}_{k+1} \leftarrow \arg\max_{\mathbf{x}\in\mathcal{X}} u(\mathbf{x}|\mathcal{D}_k)$;
    Evaluate function $f$ at $\mathbf{x}_{k+1}$, store result $f_{k+1} \leftarrow f(\mathbf{x}_{k+1})$;
    Augment the data: $\mathbf{X}_{k+1} \leftarrow \mathbf{X}_k \cup \{\mathbf{x}_{k+1}\},\ \mathbf{f}_{k+1} \leftarrow \mathbf{f}_k \cup \{f_{k+1}\}$
    if $k$ mod $\xi = 0$ and $r_i^{(k)} \geq t$ then
        Update RoI $R^{(k)}$ based on bounds using oscillation indicators;
        Trim the updated RoI;
    else
        Continue to next iteration;
    end
end
```

Algorithm 3: BO-VAE Combined with SDR

```
Data: Unlabelled dataset $\mathcal{D}_{\mathbb{U}} = \{\mathbf{x}_i\}_{i=1}^M$, Initial labelled dataset $\mathcal{D}_{\mathbb{L}} = \{\mathbf{x}_i, f(\mathbf{x}_i)\}_{i=1}^N$, budget
      $B$, initial bound $R^0$ in latent space $\mathcal{Z}$, the EI acquisition function $u(\cdot)$, an
      encoder-decoder pair of a VAE, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) : \mathcal{X} \to \mathcal{Z}$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) : \mathcal{Z} \to \mathcal{X}$.
Result: Minimum value $f_{min}$ discovered by the algorithm.
Train the encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ and decoder $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ on $\mathcal{D}_{\mathbb{U}}$: $\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{D}_{\mathbb{U}})$.
Compute the latent dataset $\mathcal{D}_{\mathbb{Z}}^0 = \{\mathbf{z}_i, f(\mathbf{x}_i)\}_{i=1}^N$, where $\mathbf{z}_i = \mathbb{E}_{q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x_i})}[\mathbf{z}]$, on $\mathcal{D}_{\mathbb{L}}$.
Initialise SDR with the initial bound $R^0$;
for $k = 0, \ldots, B-1$ do
    Fit GP model $h_k : \mathcal{Z} \to \mathbb{R}$ on $\mathcal{D}_{\mathbb{Z}}^k$;
    Solve for the next latent point $\hat{\mathbf{z}}_k = \arg\max_{\mathbf{z}} u(\mathbf{z}|\mathcal{D}_{\mathbb{Z}}^k)$ and reconstruct the corresponding
      input, $\hat{\mathbf{x}}_k \sim p_{\boldsymbol{\theta}^*}(\cdot|\hat{\mathbf{z}}_k)$;
    Evaluate the objective function $f_k = f(\hat{\mathbf{x}}_k)$;
    Augment the latent dataset $\mathcal{D}_{\mathbb{Z}}^{k+1} \leftarrow \mathcal{D}_{\mathbb{Z}}^k \cup \{\hat{\mathbf{z}}_k, f_k\}$;
    Update the search domain $R^{k+1} \leftarrow R^k$ using the updated dataset $\mathcal{D}_{\mathbb{Z}}^{k+1}$;
end
```

where $\tau \in (0, 1)$ is an accuracy level. If the criterion is not met, $N_p(s; \tau) = \infty$. The performance profile $\pi_{s,\tau}(\alpha)$ is the fraction of problems where $r_{p,s} \leq \alpha$, representing the cumulative distribution of performance ratios.

Data profiles. The data profile shows solver performance across different budgets. For a solver s, accuracy level τ, and problem set $\mathcal{P}$, it is defined as:

$$d_{s,\tau}(\alpha) = \frac{|\{p \in \mathcal{P} : N_p(s; \tau) \leq \alpha(n_p + 1)\}|}{|\mathcal{P}|},\ \alpha \in [0, N_g],$$

where n_p is the problem dimension and N_g is the maximum budget. The data profile tracks the percentage of problems solved as a function of the budget.

B.3. Soft and Hard Triplet Losses

In this subsection, we briefly present how [12] integrates the triplet loss with VAEs. We refer the reader to [14, 28] for background knowlegde for triplet deep metric loss. [12] introduces a parameter η to create sets of positive $\mathcal{D}_p(\mathbf{x}^{(b)};\eta) = \{\mathbf{x} \in \mathcal{D} : |f(\mathbf{x}^{(b)}) - f(\mathbf{x})| < \eta\}$ and negative points $\mathcal{D}_n(\mathbf{x}^{(b)};\eta) = \{\mathbf{x} \in \mathcal{D} : |f(\mathbf{x}^{(b)}) - f(\mathbf{x})| \geq \eta\}$ for a base point $\mathbf{x}^{(b)}$ in a dataset $\mathcal{D}$, based on differences in function values. However, the classical triplet loss is discontinuous, which hinders GP models. To resolve this, a smooth version, the soft triplet loss, is proposed. Suppose we have a latent triplet $\mathbf{z}_{ijk} = \langle \mathbf{z}_i, \mathbf{z}_j, \mathbf{z}_k \rangle$ associated with the triplet $\mathbf{x}_{ijk} = \langle \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \rangle$ in the ambient space. Here, $\mathbf{z}_i$ is the latent base point. The complete expression of the soft triplet loss is [12]

$$\mathcal{L}_{s-trip}(\mathbf{z}_{ijk}) = \ln\left(1 + \exp(d_{\mathbf{z}}^{+} - d_{\mathbf{z}}^{-})\right) \omega_{ij}\omega_{ik} \times I_{\{|f(\mathbf{x}_i)-f(\mathbf{x}_j)|<\eta \text{ \& } |f(\mathbf{x}_i)-f(\mathbf{x}_k)|\geq\eta\}},$$

where

$$d_{\mathbf{z}}^{+} = \|\mathbf{z}_i - \mathbf{z}_j\|_p, d_{\mathbf{z}}^{-} = \|\mathbf{z}_i - \mathbf{z}_k\|_p,$$
$$\omega_{ij} = \frac{f_\nu\left(\eta - |f(\mathbf{x}_i) - f(\mathbf{x}_j)|\right)}{f_\nu(\eta)}, \omega_{ik} = \frac{f_\nu\left(|f(\mathbf{x}_i) - f(\mathbf{x}_k)| - \eta\right)}{f_\nu(1-\eta)},$$

for any $\mathbf{z}_j \sim q_{\boldsymbol{\phi}}(\cdot|\mathbf{x}_j), \forall \mathbf{x}_j \in \mathcal{D}_p(\mathbf{x}_i;\eta)$ and $\mathbf{z}_k \sim q_{\boldsymbol{\phi}}(\cdot|\mathbf{x}_k), \forall \mathbf{x}_k \in \mathcal{D}_n(\mathbf{x}_i;\eta)$. Here, $f_\nu(x) = \tanh\left(a/(2\nu)\right)$ is a smoothing function with ν being a hyperparameter such that $\mathcal{L}_{s-trip}(\mathbf{z}_{ijk})$ approaches the hard triplet loss since $\lim_{\nu\to 0} f_\nu(a) = 1$. The function $I_{\{\cdot\}}$ is a indicator function. The penalisation weights ω_{ij} and ω_{ik} are introduced to smooth out the discontinuities. Thus, the modified ELBO of a VAE trained with soft triplet loss is [12, 16]

$$\begin{aligned}\mathcal{L}_{DML}(\boldsymbol{\theta},\boldsymbol{\phi};\{\mathbf{x}_i, f(\mathbf{x}_i)\}_{i=1}^{N}) &= \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\{\mathbf{x}_i\}_{i=1}^{N}) - \mathcal{L}_{metric} \\ &= \sum_{n=1}^{N}\left[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n)}\left[\ln p_{\boldsymbol{\theta}}(\mathbf{x}_n|\mathbf{z}_n)\right] - D_{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}_n|\mathbf{x}_n)\|p(\mathbf{z}_n)\right)\right] \\ &\quad - \sum_{i,j,k=1}^{N,N,N}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}_{ijk}|\mathbf{x}_{ijk})}\left[\mathcal{L}_{s-\text{trip}}(\mathbf{z}_{ijk})\right],\end{aligned}$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}_{ijk}|\mathbf{x}_{ijk}) = q_{\boldsymbol{\phi}}(\mathbf{z}_i|\mathbf{x}_i)q_{\boldsymbol{\phi}}(\mathbf{z}_j|\mathbf{x}_j)q_{\boldsymbol{\phi}}(\mathbf{z}_k|\mathbf{x}_k)$.

The BO-VAE algorithm with the soft triplet loss as the chosen deep metric loss is outlined 4. We note that Algorithm 4 is not implemented with SDR in the latent space, as experiments have shown that SDR and DML methods conflict with each other in excluding the global optimum. Addressing this conflict when implementing SDR in DML-structured latent spaces is left as future work.

Appendix C. Additional Details for Section 4 Numerical Experiments

To train VAEs more robustly, we introduce an additional weight β to the VAE ELBO, known as beta-annealing approach [2, 3]. The modified ELBO becomes

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi};\mathbf{x}) &= \ln p_{\boldsymbol{\theta}}(\mathbf{x}) - \beta D_{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[\ln p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] - \beta D_{KL}[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})],\end{aligned}$$

Algorithm 4: Retraining BO-VAE Algorithm with DML

```
Data: Labelled dataset $\mathcal{D}_{\mathbb{L}}^{l=1} = \{\mathbf{x}_i, f(\mathbf{x}_i)\}_{i=1}^{N}$, unlabelled dataset $\mathcal{D}_{\mathbb{U}} = \{\mathbf{x}_i\}_{i=1}^{M}$, budget $B$,
      periodic frequency $q$, EI acquisition function $u(\cdot)$, the encoder and decoder models from
      a VAE, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) : \mathcal{X} \to \mathcal{Z}$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) : \mathcal{Z} \to \mathcal{X}$.
Result: Minimum function value $f_{min}$ found by the algorithm.
Pre-train the VAE model $V_{\mathcal{D}_{\mathbb{L}}}^{l=0}$ with $\mathcal{D}_{\mathbb{U}}$: $\boldsymbol{\theta}_0^*, \boldsymbol{\phi}_0^* = \arg\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{D}_{\mathbb{U}})$
Set $\boldsymbol{\theta}_1^* \leftarrow \boldsymbol{\theta}_0^*$, $\boldsymbol{\phi}_1^* \leftarrow \boldsymbol{\phi}_0^*$, $V_{\mathcal{D}_{\mathbb{L}}}^{l=1} \leftarrow V_{\mathcal{D}_{\mathbb{L}}}^{l=0}$;
for $l = 1$ to $L \equiv \lceil B/q \rceil$ do
    Train the VAE model $V_{\mathcal{D}_{\mathbb{L}}^l}^l$ on $\mathcal{D}_{\mathbb{L}}$: $\boldsymbol{\theta}_l^*, \boldsymbol{\phi}_l^* = \arg\max_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}_{DML}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathcal{D}_{\mathbb{L}}^l)$
    Compute the latent dataset $\mathcal{D}_{\mathbb{Z}}^l = \{\mathbf{z}_i, f(\mathbf{x}_i)\}_{i=1}^{N+l\cdot q} = \{\mathbb{E}_{q_{\boldsymbol{\phi}_l^*}(\mathbf{z}|\mathbf{x_i})}[\mathbf{z}], f(\mathbf{x}_i)\}_{i=1}^{N+l\cdot q}$
    Initialise $\mathcal{D}_{\mathbb{L}}^{l;k=0} \leftarrow \mathcal{D}_{\mathbb{L}}^l$ and $\mathcal{D}_{\mathbb{Z}}^{l;k=0} \leftarrow \mathcal{D}_{\mathbb{Z}}^l$;
    for $k = 0$ to $q-1$ do
        Fit a Gaussian Process (GP) model $h_{l;k} : \mathcal{Z} \to \mathbb{R}$ on $\mathcal{D}_{\mathbb{Z}}^{l;k} = \{\mathbf{z}_i, f(\mathbf{x}_i)\}_{i=0}^{N+l\cdot q+k}$
        Solve for the next latent point: $\hat{\mathbf{z}}_{l;k+1} = \arg\max_{\mathbf{z}} u(\mathbf{z}|\mathcal{D}_{\mathbb{Z}}^{l;k})$
        Obtain the new sample $\hat{\mathbf{x}}_{l;k+1}$: $\hat{\mathbf{x}}_{l;k+1} \sim p_{\boldsymbol{\theta}_l^*}(\cdot|\hat{\mathbf{z}}_{l;k+1})$
        Evaluate the objective function at the new sample: $f(\hat{\mathbf{x}}_{l;k+1})$;
        Augment the datasets:
            $\mathcal{D}_{\mathbb{L}}^{l;k+1} \leftarrow \mathcal{D}_{\mathbb{L}}^{l;k} \cup \{\hat{\mathbf{x}}_{l;k+1}, f(\hat{\mathbf{x}}_{l;k+1})\}, \mathcal{D}_{\mathbb{Z}}^{l;k+1} \leftarrow \mathcal{D}_{\mathbb{Z}}^{l;k} \cup \{\hat{\mathbf{z}}_{l;k+1}, f(\hat{\mathbf{x}}_{l;k+1})\}$
    end
    Augment the outer-loop datasets: $\mathcal{D}_{\mathbb{L}}^{l+1} \leftarrow \mathcal{D}_{\mathbb{L}}^{l;q}, \mathcal{D}_{\mathbb{Z}}^{l+1} \leftarrow \mathcal{D}_{\mathbb{Z}}^{l;q}$
end
```

where $\beta \geq 0$. The use of β is a trade-off between reconstruction accuracy and the regularity of the latent space and to avoid the case of the vanishing KLD term, where no useful information is learned [2, 11]. A common approach to implementing the beta-annealing technique [2] involves initialising β at 0 and gradually increasing it in uniform increments over equal intervals until β reaches 1.

Experimental Configurations for Test Set 1 The basic training details of the VAE used in the experiments are given in Table 5.

Epochs	Optimiser	Learning Rate	Batch Size	$(\beta_i, \beta_f, \beta_s, \beta_a)$	M
300	Adam	1×10^{-3}	1024	$(0, 1, 10, 0.1)$	50000

Table 5: The (pre-)training details of the VAE used for Test Set 1. β_i and β_f are the initial and final values of β for $\beta-$VAEs respectively. The annealing approach is to increase the weight β by β_a every β_s epochs. M is the size of $\mathcal{D}_{\mathbb{U}}$.

We highlight two ingredients in the implementation of the algorithms.

1. The first thing involves the VAE pre-training. The models are pre-trained according to the details in Table 5. It is crucial that training samples are drawn with high correlations to

construct the VAE training dataset. For instance, samples can be generated from a multivariate normal distribution with a large covariance matrix. This approach facilitates the VAE in learning a meaningful low-dimensional data representation.

2. The second one involves constructing the latent datasets for a sample-efficient BO procedure, as it would be computationally inefficient to use the entire VAE training dataset. Therefore, instead of using the entire $\mathcal{D}_{\mathbb{L}}$, we utilise only 1% of it by uniformly and randomly selecting N points, where N represents 1% of the size of $\mathcal{D}_{\mathbb{U}}$ at the current retraining stage l.

The SDR setting is: $\gamma_o = 0.7$, $\gamma_p = 1.0$, $\eta = 0.9$, $t = 0.5$, $\xi = 1$. The initial search domain R^0 for Algorithms 1 and 3 is $[-5, 5]^d$. For Algorithm 4, the hyperparameters η and ν are set to be 0.01 and 0.2 respectively. For the retraining stage, we use Table 6 as the common setup.

Epochs	Optimiser	Learning Rate	Batch Size	Beta-annealing
2	Adam	1×10^{-3}	256	No

Table 6: The retraining details of the VAE used for Test Set 1.

Experiment Configurations for Test Set 2 The pre-training and retraining details for this VAE are consistent with as before, as shown in Table 5 and Table 6, respectively. In addition to the two key implementation details for BO-VAE algorithms listed in Appendix C, it is important to note that the test problem domains must be scaled to $[-1, 1]^D$ for a fair comparison with REMBO. This adjustment is due to the domain scaling used in constructing Test Set 2. The specific experimental configurations for each BO-VAE algorithm are consistent with as before.