MSDE

PAPER

View Article Online
View Journal

Cite this: DOI: 10.1039/d5me00225g

Towards best practices in low-dimensional semi-supervised latent Bayesian optimization for the design of antimicrobial peptides

Jyler Menard and R. A. Mansbach *

Generative deep learning techniques have demonstrated an impressive capacity for tackling biomolecular design problems in recent years. Despite their high performance, however, they still suffer from a lack of interpretability and rigorous quantification of associated search spaces, which are necessary to unlock their full potential for scientific inquiry beyond efficient design. An area in which they are of particular interest is the design of antimicrobial peptides, which are a promising class of therapeutics to treat bacterial infections. Discovering and designing such peptides is difficult because of the vast number of possible sequences and comparatively small amount of experimental information. In this work, we perform an empirical investigation of latent Bayesian optimization for searching through peptide sequence spaces, with a focus on antimicrobial peptides. We investigate (1) whether searching through a dimensionally-reduced variant of the latent design space may facilitate optimization, (2) how organizing latent spaces by differing amounts of more and less relevant information may improve the efficiency of arriving at an optimal peptide design, and (3) the interpretability of the spaces. We find that employing a dimensionally-reduced version of the latent space is more interpretable and can be advantageous, while the use of less-relevant but more easily-computable physicochemical properties is advantageous to latent space organization in certain contexts and the use of more-relevant but sparser properties associated with the latent Bayesian objective function is advantageous in others. This work lays crucial groundwork for biophysically-motivated peptide design procedures, with an especial focus on antimicrobial peptides.

Received 9th December 2025,
Accepted 6th May 2026

DOI: 10.1039/d5me00225g

rsc.li/molecular-engineering

Design, System, Application

The goal of our work is to develop optimization methods for designing antimicrobial peptides, short proteins that kill bacteria. Most evaluations of candidates are computationally or experimentally intensive tasks where a direct mapping of the peptide composition to activity is not available. Bayesian optimization is an iterative design algorithm that excels in situations where we wish to optimize a black-box function with limited evaluations, exactly the case at hand. We probe different search spaces with a focus on improving BayesOpt efficiency and interpretability. We compare BayesOpt performance in the latent space of a variational autoencoder to its performance in projected, lower-dimensional versions of that space. We ask whether organizing the search space with easily-computable, but less informative, physicochemical properties can improve optimization efficiency and search interpretability over using a limited number of very informative labels. In our toy setting of using a trained predictive model as our objective, we found that using sparse informative labels in a lower-dimensional space can provide an optimization and interpretability advantage. This work lays necessary groundwork for closed-loop antimicrobial peptide design with more realistic objective functions.

1 Introduction

Antimicrobial resistance is a growing global health concern. Through increasing usage in animal agriculture,[1] aquaculture,[2] and humans,[3] it is expected that by 2050, >8.2 million deaths per year will be associated with AMR.[4] With traditional pharmaceutical companies becoming less productive,[5] fewer antibiotics being developed,[6,7] and of the thirteen that have been approved since 2017, ten belonging to antibiotic groups with known resistance mechanisms,[8] there is an urgent need for novel classes of therapeutics to treat bacterial infections, and methods to quickly find them.

One potentially fruitful area of research is the design and development of new and more effective antimicrobial peptides (AMPs). Antimicrobial peptides are short proteins of around 5 to 100 residues with broad-spectrum antibacterial activity.[9] Many AMPs are thought to act by disrupting the bacterial membrane, through the formation of large pores, through inducing micellization of membrane patches (the so-

Department of Physics, Concordia University, Montreal, QC, H4B 1R6, Canada.
E-mail: re.mansbach@concordia.ca

called "carpet method"), or through a combination of such mechanisms.[10] Because of the comparatively indiscriminate nature of the entire membrane as a target rather than a small protein binding pocket like most traditional small-molecule drugs, it was hypothesized that resistance to membrane-active AMPs will develop more slowly than to many traditional small-molecule antibiotics. Evidence supporting this hypothesis includes *in vitro* experiments across four different Gram-negative bacterial species showing less resistance developing to membrane-active antibiotics than to traditional antibiotics within the same time frame.[11] In the same study, it was demonstrated that strong resistance to traditional antibiotics appeared within 120 generations of repeated exposure,[11] compared to the 600 generations required for resistance to develop to an analogue of the peptide magainin-2 that was observed by Perron *et al.*[12] Despite differences in study methodology, the evidence overall supports slower resistance development to peptides. In addition to the possibility of slowing resistance, certain AMPs can be used as part of combination drug cocktails to improve the efficacy of other antibiotics.[13] Such combination therapies have also been found to have slower resistance development.[14] Due to their broad-spectrum activity, slow resistance, synergism with other antibiotics, and the need for new antibiotics, AMPs are a promising therapeutic class.

While peptides are short compared to other proteins, the space of all possible peptide sequences is still vast. Considering only naturally-occurring amino acids, there are on the order of $\sum_{L=5}^{100} 20^L$ possible sequences. Traditional mutagenesis methods are inadequate to explore more than a tiny fraction of this sequence space to find peptides with properties of interest. Traditional computational methods, including machine learning, can improve the speed of candidate identification by rapidly screening databases full of peptide and protein sequences. For example, some of the earliest predictive models for AMP identification were hidden Markov models and shallow neural networks,[15,16] with logistic regression[17] and SVMs[18] following after. More recently, motivated by the success of deep neural networks for (natural language) sequence representations, deep neural network architectures began to be employed.[19] All of these methods, however, function to identify candidates in a database and cannot suggest completely novel AMPs. Because of the number of possible peptides compared to the speed at which they can be tested, finding peptides with desired properties is difficult.

Deep learning generative models can partially obviate the need to pre-generate libraries of sequences by providing a method to sample arbitrarily many new sequences. Recently, such models have been adopted by the biomolecular design community and have demonstrated a powerful capacity to produce peptides with desirable properties. Capecchi *et al.*[20] trained a strain-specific RNN to generate peptide sequences and filtered them using classifiers. Li *et al.*[21] trained a generative model based on a transformer architecture intended to generate high-activity mutants from an input sequence. Zakharova *et al.*[22] used an RNN to generate anticancer peptides. Szymczak *et al.*[23] designed a conditional VAE – conditioned on categorical classifications of AMPness and high/low activity levels – in an attempt to directly sample from the high-activity AMP conditional probability distribution. Das *et al.*[24] used a Wasserstein autoencoder together with a number of classifiers to generate peptides.

While impressive work is being done in improving both predictive and generative models, tying them together in a closed-loop optimization scheme is (currently) less common. Furthermore, the focus of these models has tended to be on design of peptides for specific applications rather than an understanding of the ways in which the specific model and procedure affect the underlying design space. Recently, we developed quantitative benchmarks and investigated the properties of the design space itself in the context of our own generative model for antimicrobial peptide sequence design based on a variational autoencoder (VAE).[25] VAE-based models possess associated latent spaces that can be thought of as explicit continuous design space representations of the discrete peptide sequences. This formulation of the problem facilitates a focus on the quality of the design space itself. In this article, we expand on this work to investigate how the organization of the design space can affect optimization procedures in that space.

While generative deep learning models on their own do not resolve the fundamental needle-in-haystack issue of finding sequences with significant antimicrobial potency and/or other properties of interest, they can produce candidates or design spaces to be used in conjunction with optimization techniques. Broadly speaking, finding a sequence with high antimicrobial activity can be modelled as an optimization problem, with a corresponding objective function that maps from peptide sequences to antimicrobial activity. In a scenario where there is a straightforward function mapping sequences to antimicrobial activity, a litany of optimization algorithms would allow us to find the sequence (or sequences) corresponding to maximal antimicrobial activity. However, how antimicrobial activity relates to the peptide sequence is not sufficiently understood to write down a straightforward function, instead leaving us in the situation of having to treat the relationship as a black box we can probe through expensive experimental assays, machine learning models limited by their data sets, and/or long, resource-intensive MD simulations depending on our available resources.

Bayesian optimization (BayesOpt) is an iterative method of finding the optimum of a black-box function. BayesOpt fits a data-driven, simple, and flexible surrogate model to approximate the relationship between input points (such as peptide sequences) and the objective function (such as antimicrobial potency). To perform BayesOpt one must also define an acquisition function that scores to what extent a point in the design space is predicted to improve on the

current best observed value. At each iteration, the acquisition function combined with the surrogate model identifies a point of interest for the next round predicted to most improve upon the optimal point found so far.

One can employ BayesOpt to search through the latent space of an associated generative model; this is referred to as latent Bayesian optimization (latent BayesOpt).[26–29] Latent BayesOpt has been previously successfully applied to the design of small molecules[26] and synthetic optoelectronic peptides.[30] Although screening peptides sampled from a generative model to design AMPs without using Bayesian optimization also has precedent,[23,24] we focus on latent BayesOpt because of its data efficiency and potential for application with higher-fidelity, more costly oracles.

Other works have explored how the organization of the latent space can impact the efficiency of latent BayesOpt. A number of ways to organize the latent space have been investigated, with particular focus on small molecule discovery where large amounts of data are readily available. In Gómez-Bombarelli *et al.*,[26] the authors investigated whether jointly training a property predictor and a decoder can improve the efficiency of latent BayesOpt for the task of finding optimal small molecules, finding compelling evidence that it can. Lee *et al.*[28] examined how latent BayesOpt performs in a latent space induced to be correlated with the objective they are trying to optimize, finding promising results. In Grosnit *et al.*[27] the authors compared a number of contrastive learning methods, finding that a triplet loss can be effective. Given these works, it seems the efficiency of latent BayesOpt can be improved by organizing the underlying latent space being searched through, but this has not been previously investigated in the context of antimicrobial peptide design. One further consideration for latent BayesOpt is that often the associated neural network has a relatively high-dimensional latent space. BayesOpt can have difficulty in high-dimensional spaces. On the other hand, reducing the dimensionality of the latent space may corrupt the ability of the associated generative model to generate diverse, novel, and realistic sequences. This leaves us with a dilemma where we may expect either worsened BayesOpt performance due to a high-dimensional latent space, or worsened generative model performance due to too small a latent space.

Unlike in the small molecule design space, where large amounts of data relevant to the performance of such molecules as antibiotics are readily available, the corresponding data on antimicrobial peptides is relatively sparse. There are hundreds of thousands or even millions of examples of small-molecule data as experimental drug-target binding affinities and well-established computed properties related to drug-likeness and solubility readily available. Conversely, in the context of peptides, antimicrobial activity in the form of experimentally-measured minimum inhibitory concentration (MIC) values is scarce; specifying particular bacterial species as targets restricts the available data even further. For example, for *E. coli* there are just 11k entries with corresponding MIC measurements, and if we restrict ourselves to those sequences with more than one reported MIC measurement, then we are left with 4k entries.[31] Experimentally-measured peptide hemolysis, which is important for the assessment of host toxicity, is available for even fewer examples, with DBAASP containing 8k sequences with reported hemolysis measurements that fall to 1.6k when restricting ourselves to more than one measurement per sequence.[31] While this may be a sufficient amount for traditional machine learning methods, it is unclear how this amount may impact the effectiveness of generative deep learning models and the organization of their latent space. Moreover, it is unclear whether using easily-computed physicochemical properties as proxies may offer a benefit over the more direct but limited amount of activity data.

In this work, we determine the best practices for performing latent Bayesian optimization in the context of peptide sequence spaces, with a focus on the antimicrobial peptide context in particular. We introduce and assess a modification of latent Bayesian optimization as it has been previously applied in drug discovery applications by performing Bayesian optimization over a reduced space representation of a peptide sequence-based latent space (see Fig. 1 for a schematic overview of the versions of latent Bayesian optimization we investigate). Rather than performing Bayesian optimization over the high-dimensional latent space representation, we explore the possibility of performing Bayesian optimization over a much lower-dimensional projection of that space. The purpose of this approach is twofold: (i) we expect an increase in efficiency by using BayesOpt in a lower-dimensional space without modifying the size of the generative model's latent space, and (ii) we expect an increase in interpretability by performing optimization in a more similar space to that which we employ for visualization compared to using reduced projections solely to visualize (a proxy of) the design space. We also note that we employ a proxy objective function (an "oracle") related to predicted AMP activity for better control of algorithmic design prior to deployment on more realistic, expensive objectives, such as experimental measurements of peptide activity.

We further investigate three important questions related to incorporation of prior knowledge to latent Bayesian optimization in peptide sequence-based design spaces. We first ask whether organizing the latent space with easily-computable physicochemical properties that are not directly related to predicted AMP activity is more informative or higher-performing than organizing the latent space with a sparse amount of predicted AMP activity data. We secondly ask whether organizing by more properties leads to a more structured, easier to optimize in, latent space. We thirdly ask whether increasing the percentage of labelled data available has a direct relationship to latent BayesOpt efficiency and latent space organization, or if there are diminishing returns. By providing an analysis of latent BayesOpt in both a semi-supervised and latent space organization context, we hope to

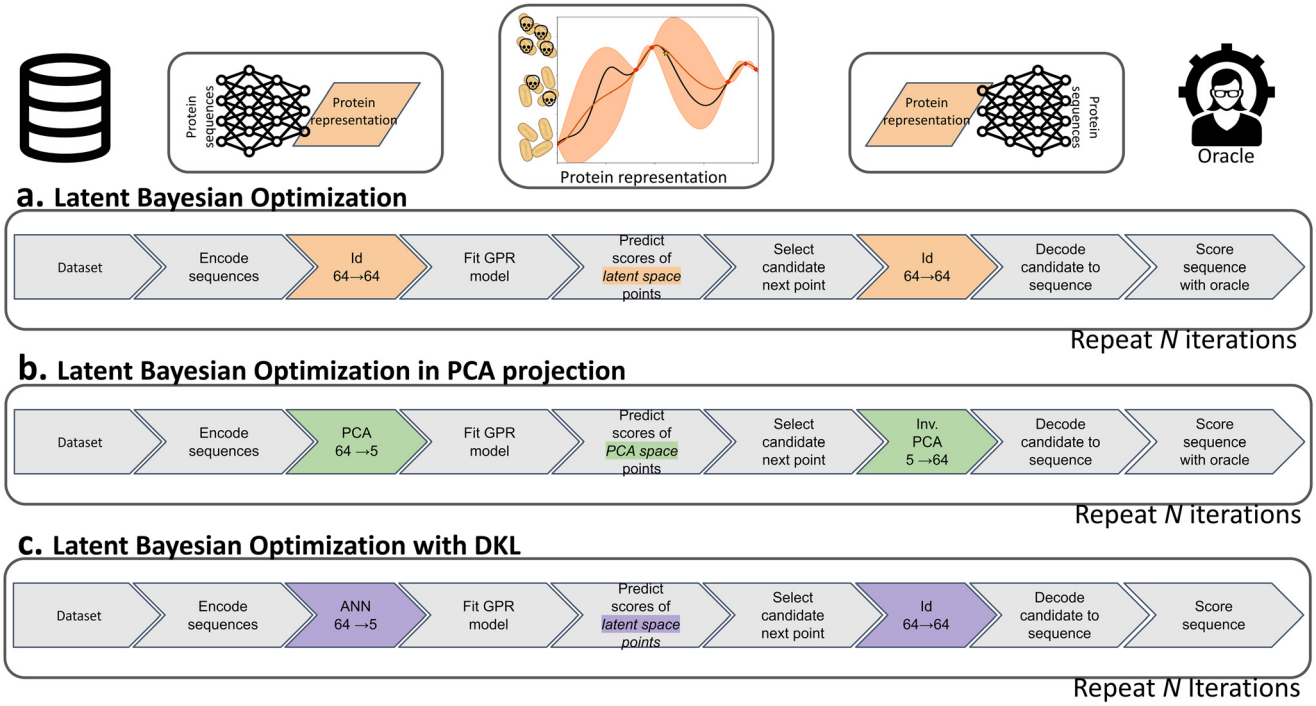

Fig. 1 Schematic overview of three approaches to latent Bayesian optimization investigated herein (differences highlighted). Latent BayesOpt begins assuming a pre-trained generative model with an encoder and a decoder module. Each method follows a similar procedure: encode the optimization dataset into the latent space of a TransVAE model; fit a Gaussian process regressor; maximize the expected improvement to select the next candidate to score using the oracle; decode the candidate latent point to an amino acid sequence; score the sequence using the oracle; add the now-labelled sequence to the optimization dataset, and repeat N times. The differences lie in whether (a) we use the full latent space representation directly in fitting the GPR, (b) we use a PCA projection of the latent space when fitting the GPR, or (c) we use a small neural network that projects the latent points into a smaller-dimensional space that is fit simultaneously with the GPR. Following (b) for one BayesOpt iteration: encode optimization dataset into the 64-dimensional latent space of a TransVAE; project those points into a five-dimensional space constructed using PCA; fit a GPR and select a candidate five-dimensional point; inverse project back to the 64-dimensional latent space; decode the candidate latent point to a candidate sequence; score the sequence using the oracle.

further the discussions of how best to perform Bayesian optimization in latent spaces in a manner that informs us about the design space that is being traversed by the optimization algorithm, and introduce the technique to the therapeutic peptide design problem in the context of AMPs.

2 Methods

In this section, we describe the design of our AMP optimization process (*cf.* Fig. 1). We begin with an overview of our Bayesian optimization procedure and parameters, then define the associated objectives and the manner in which they are assessed. Then we describe the generative models for which the associated latent spaces become the design spaces of possible peptide sequences and present the methods we chose to organize the latent spaces. Lastly, we describe how we implemented Bayesian optimization in the latent space of this generative model.

2.1 Bayesian optimization

To find highly active antimicrobial peptides, we use Bayesian optimization (BayesOpt) with a single objective function, as implemented in the BoTorch Python package using the SingleTaskGP class.[32] BayesOpt is an iterative method of efficiently identifying design optima by using a surrogate (usually a quicker, less accurate assessment) of a desired objective function in combination with an acquisition function that selects the next potential design to be evaluated according to the true objective function (usually slower and more accurate assessment).[33] A good acquisition function balances exploration of the design space with exploitation of observed data points;[33] in other words, it balances assessing new regions in design space with searching near known high-performing designs.

We use a Gaussian process regression (GPR) model for the surrogate objective function. A GPR is a probabilistic model where, for any point $\vec{x}$ in our design space, we assume we have a Gaussian distribution (our prior distribution), fully determined by a mean m and covariance matrix $\sigma = (\sigma_{ij})$. The mean and covariance are parameterized by a mean function $m(\vec{x})$ and kernel function $k(\vec{x}_i, \vec{x}_j)$, each of which depend on data points assessed according to the true objective function. For a data set $\mathcal{D}$ of n points in an F-dimensional space with associated objective values $\mathcal{D} = \{(\vec{x}_j, y_j)\}_{j=1}^{n}$, where $\vec{x} \in \mathbb{R}^F$, at iteration l, we fit the mean and covariance,

$$m_l(\vec{x}) = m_l = \frac{1}{|\mathscr{D}|}\sum_{j=1}^{|\mathscr{D}|} y_j \tag{1}$$

$$k_l(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{1}{2}(\vec{x}_i - \vec{x}_j)^T \Theta_l^{-2}(\vec{x}_i - \vec{x}_j)\right) \tag{2}$$

where $\Theta_l \in \mathbb{R}^{F\times F}$ is a diagonal matrix of fitted length-scales, which are fitted using the exact marginal log-likelihood. Here we use the BoTorch default choice of a constant mean function – specifically the sample mean of the observed objective function values – so it does not depend on the input vector. Together these give a *prior* distribution

$$\mathscr{G}(m_l, \sigma_l). \tag{3}$$

with which we compute a point-dependent *posterior* distribution using Bayes' rule. Using this set of posterior distributions, we make predictions on unobserved points that have not been assessed according to the true objective function, and to compute an uncertainty in those predictions. The posterior distribution's mean M_l and variance $\sum_l$ are determined to be:

$$M_l(\vec{x}) = m_l(\vec{x}) + \sigma_l(\vec{x}, x_D)^T \sigma_l(x_D, x_D)^{-1}\cdot(f(x_D) - m_l(x_D)) \tag{4}$$

$$\sum_l(\vec{x}) = \sigma_l(\vec{x}, \vec{x}) - \sigma_l(\vec{x}, x_D)^T \sigma_l(x_D, x_D)^{-1}\sigma_l(\vec{x}, x_D), \tag{5}$$

where f is the true objective function, x_D is short-hand for the dataset of n previously-assessed input points $\{\vec{x}_i\}$, $\sigma_l(x_D, x_D)$ is an $n \times n$ matrix with entries $[\sigma_l(x_D, x_D)]_{i,j} = k(\vec{x}_i, \vec{x}_j)$, and $\sigma_l(\vec{x}, x_D)$ is a vector of length n with entries $[\sigma_l(\vec{x}, x_D)]_i = k(\vec{x}, \vec{x}_i)$. We use the radial basis function kernel (default when using BoTorch's SingleTaskGP[32]), but other choices exist. Because the radial basis function kernel is distance-dependent, there is an implicit assumption that points near each other have similar objective values. Through the distance-dependence, points near each other in the design space will be predicted to have similar surrogate objective values. With the posterior distribution in hand, predictions are made either by sampling from the posterior distribution

$$\hat{f}(\vec{x}) \sim \mathscr{G}(M_l(\vec{x}), \sum_l(\vec{x})) \tag{6}$$

or, more simply, by using the posterior mean (eqn (4)) as a point estimate, and the posterior covariance to compute the uncertainty range (eqn (5)).

As mentioned, in addition to the Gaussian process surrogate model, we must select an acquisition function, which identifies the next point to be evaluated with the true objective. A common choice is the Expected improvement, defined as,

$$\mathrm{EI}(\vec{x}) \equiv \mathbb{E}_{\hat{f}(\vec{x})}\left[\left(\hat{f}(\vec{x}) - f_l^*\right)^+\right],$$

where f_l^* is the best value we have observed up to iteration l, $\hat{f}(\vec{x})$ is the predicted value of the surrogate function at point $\vec{x}$, and $(\hat{f}(\vec{x}) - f_l^*)^+ = \max(0, \hat{f}(\vec{x}) - f_l^*)$ is the improvement. $\mathrm{EI}(\vec{x})$ encourages exploitation by searching for a point that is likely to improve on the current best point, and it encourages exploration through the standard deviation and cumulative distribution of the GPR function.

For our case, rather than using the expected improvement, we use the log expected improvement, which has been found to be nearly identical but is easier to numerically optimize.[34] It is calculated as the logarithm of the expectation value of the improvement over the previous best value according to the surrogate model,

$$\mathrm{LogEI}(x) = \log \mathbb{E}_{\hat{f}(\vec{x})}\left[\left(\hat{f}(\vec{x}) - f_l^*\right)^+\right]$$

The next point to be evaluated with the objective function is therefore

$$\vec{x}_{l+1} = \mathrm{argmax}_{\vec{x}}(\mathrm{LogEI}(\vec{x}))$$

After assessing the $\vec{x}_{l+1}$-th point using the true objective function, we add the pair $(\vec{x}_{l+1}, f(\vec{x}_{l+1}))$ to the dataset $\mathscr{D} = \mathscr{D} \cup \{(\vec{x}_{l+1}, f(\vec{x}_{l+1}))\}$ and start the next iteration.

2.2 Latent Bayesian optimization

Latent Bayesian optimization attempts to resolve the issue of optimizing over large, discrete design spaces, such as, in our case, amino acid sequences describing peptide candidates. Rather than attempting to search for optimal peptide sequences in the discrete combinatorial space, we instead perform the optimization over a continuous-valued latent space, where we have access to both an encoder $E: X \to Z$ encoding sequences into a latent space, and a decoder $D: Z \to X$ that reconstructs the sequence corresponding to a latent point. Here Z is typically the latent space of a generative model. The conventional choice is to use the latent space of a VAE, where the encoded mean $\vec{\mu}_\theta(\vec{x})$ of a trained VAE with fitted model weights θ is used in the optimization.[26] More formally, latent Bayesian optimization is the process of solving

$$\underset{\vec{x}\in X}{\mathrm{argmax}}\, f(\vec{x})$$

by instead approximating the problem as

$$\underset{\vec{z}\in Z}{\mathrm{argmax}}\, f(D(\vec{z}))$$

i.e. Bayesian optimization over the latent space of a generative model or other invertible mapping. The surrogate model $\hat{f}: Z \to \mathbb{R}$ approximates the relationship between latent points and their corresponding objective value; for a peptide sequence $\vec{x}$ and its encoded representation $\vec{\mu}_\theta(\vec{x})$, the surrogate model approximates $\vec{\mu}_\theta(\vec{x}) \to f(\vec{x})$. At iteration l, the acquisition function determines the subsequent point in the latent space $\vec{\mu}_{l+1}$ to decode into a peptide sequence $\vec{x}_{l+1}$

we can evaluate with the objective function $f(\vec{x}_{l+1})$. In other words, rather than performing the Bayesian optimization on what we want to design (peptide sequences) we perform it over a (learned) representation of those sequences.

2.3 "True" objective function for assessment of AMP design spaces

In this work we are interested in finding peptide sequences with strong antimicrobial activity. We choose to use minimum inhibitory concentration (MIC) as a measure for antimicrobial activity. The MIC refers to the smallest concentration of peptide that inhibits the growth of bacteria in a standardized assay.[35] We make this choice because it is a theoretically continuous, standard, and meaningful measure of antimicrobial activity, and standardized data is available for measurements on it.[36] Because a lower MIC corresponds to stronger antimicrobial activity, we maximize the negative of MIC: $M = -\log_{10}(\text{MIC})$, thereby minimizing MIC.

In this pilot study, we wished to assess the impact of algorithmic choices and ensure our understanding of the design space prior to engaging in time-consuming experimental work. Therefore, we use a proxy model (also called an "oracle") based on traditional machine learning techniques (see next section) as our "ground truth" to estimate the MIC values of sequences for which no experimentally-measured MIC is available. This is analogous to proof-of-concept small-molecule discovery projects that optimize for computed estimates of drug-likeness and solubility; a discipline-relevant proxy model is used to motivate an algorithmic exploration. In future work, we plan to leverage the procedure here to inform peptide design in more realistic contexts.

2.4 Variational autoencoder

For the generative model whose associated latent space becomes our continuous design space of peptide sequences, we employ a variational autoencoder (VAE) based neural network. A VAE is a probabilistic deep learning architecture comprising two neural networks: the encoder $E_\phi: X \rightarrow Z$, which takes an input sequence and projects it to a continuous-valued latent space $z = E_\phi(x)$; and the decoder $D_\theta: Z \rightarrow X$ modules, which maps a point in the latent space back to the original input sequence space $\hat{x} = D_\theta(z)$ with ϕ and θ representing learned model weights.[37] Once the model has been adequately trained, we can use the latent space and the decoder to generate new peptides, sampling from a distribution that is similar to the distribution found in the training set.

Training a VAE entails minimizing the evidence lower bound (ELBO), a combination of a reconstruction loss, which measures the difference between the input sequence and the corresponding generated output sequence, and the Kullback–Leibler divergence which functions to push the probability distribution of the latent points to be close to a Gaussian prior modeled by the latent space. We define the reconstruction loss as is typically done for sequence-based tasks[38] as the cross-entropy of the probability of predicting the next token x_{i+1} conditioned on a context $\vec{c}$, usually the previous tokens in the sequence:

$$\mathscr{L}_{\text{CE}} = -\sum_{i=1}^{N_{\text{batch}}} \sum_{c=1}^{A} p_c \log\left(q\left(x_{i+1,c}|\vec{c}\right)\right)$$

where p_c is the probability distribution over the target token values, and $q(x_{i+1,c}|\vec{c})$ is the predicted probability distribution, and A is the number of different tokens in the alphabet. Because the target token values come from the training dataset, and because we are using the maximum likelihood principle, we say that $p_c = 1$ when c is the correct class and $= 0$ when c is not, leading to the simpler $= -\sum_{i=1}^{N_{\text{batch}}} \log\left(q\left(x_{i+1,c}|\vec{c}\right)\right)$.

We use a VAE with a transformer architecture as described previously,[25,39] known therein as a TransVAE. The transformer architecture is largely similar to the architecture originally proposed in Vaswani *et al.*[40] Input sequences were tokenized into integers then fed through a learned embedding layer. During training, sequences were predicted autoregressively, with the $(n + 1)$-th token predicted from a context of the previous n tokens, where the previous n tokens are given directly from the training set and the reparameterized latent point representation $\vec{z}$. The embedding layer and a positional encoding layer then together feed into the encoder module consisting of self-attention blocks followed by convolution blocks. Each self-attention block is a sequence of layers: layer normalization, multi-head self-attention, dropout, a residual connection, then layer normalization, feed-forward, dropout, and another residual connection. After the self-attention blocks, there is a convolution block consisting of three layers of convolution, pooling, and leaky ReLU. The convolution block outputs the mean and variance for the VAE latent space. The decoder module consists of largely the same architecture in reverse: convolution blocks followed by masked self-attention blocks. In our previous results[25] we found evidence suggesting that the TransVAE architecture may limit latent space interpretability compared to other architectures, however we did observe high performance at the assigned tasks. Here, we chose to use a transformer architecture because of their effectiveness at sequence-to-sequence tasks, and because traversing their latent space can correlate well with a predicted property,[25] an important aspect of this work. Additionally, to limit the scope of this work, we focused on only the TransVAE architecture, but in future work will expand to other architectures and representation learning choices.

As previous work has shown, jointly training a property predictor can induce latent space organization.[25,26] In this work, we investigate the impact of the choice of predictive

task and the availability of data associated with that task on the performance of the latent BayesOpt algorithm. To do so, we train multiple TransVAE models jointly with a property predictor consisting of two feedforward layers (see the next section). The property predictor loss term $\mathscr{L}_{\text{PropPred}}$ is a simple mean-squared error between the property predictor's outputs y_{pred}, and the computed property values y_{true},

$$\mathscr{L}_{\text{PropPred}} = \sum_{i}^{N_{\text{properties}}} \left(\frac{1}{N_{\text{batch}}} \sum_{j}^{N_{\text{batch}}} \left(y_{\text{pred},i,j} - y_{\text{true},i,j} \right)^2 \right) \quad (7)$$

for a batch of size N_{batch} and for $N_{\text{properties}}$ number of different properties we are predicting.

2.5 TransVAE models and training

To investigate the importance of additional information on the LBO algorithm and on the latent space that it traverses, we trained TransVAE models in conjunction with property predictors of different combinations of the physicochemical characteristics Boman index, hydrophobicity, and charge. Jointly training property predictors requires target values of these characteristics for corresponding (latent representations of) sequences. We computed the Boman index, net charge at a pH of 7.2, and hydrophobicity of every sequence in both the training set and test set. The Boman index is defined by summing the solubility values of individual amino acids in the peptide, normalized by length.[41] It functions as an estimate of whether a peptide will bind to membranes or other proteins. For hydrophobicity, we use the Kyte–Doolittle scale[42] and average the hydrophobicity values of amino acids in a given peptide for an overall estimate of peptide affinity with water. Charge at a pH of 7.2 is simply computed by summing the predicted charge of each amino acid in the peptide at a pH of 7.2. These physicochemical properties are of interest in AMP design because they relate to whether an AMP will interact with or penetrate a cell membrane.

To compute these physicochemical properties, we used the python package peptides.[43] The properties were then normalized to have zero mean and unit standard deviation using their respective training set means and standard deviations; the test set property values were normalized using the training set means and standard deviations.

Overall, we studied the following models: (i) TransVAE trained jointly with a regressor directly predicting oracle values, *i.e.* $\log_{10}$MIC values estimated by a predictor model; (ii) TransVAE trained jointly with a regressor predicting Boman index values; (iii) TransVAE trained jointly with a regressor predicting charge values; (iv) TransVAE trained jointly with a regressor predicting hydrophobicity values; (v) TransVAE trained jointly with a regressor predicting both the Boman index and charge; and (vi), TransVAE trained jointly with a regressor predicting the Boman index, charge, and hydrophobicity values.

To assess the effect of data sparsity, we trained each model with different numbers of available labels for the associated regressor. For (i), we trained models with 100% and 2% (corresponding to 10 538 points in the training set) of the labels in the training set. For (ii)–(vi), we trained models with 100%, 75% (corresponding to 392 586 points in the training set), 50% (corresponding to 261 647 points in the training set), 25% (corresponding to 130 746 points in the training set), and 2% (corresponding to 10 538 points in the training set) of the labels. The loss function for jointly training the TransVAE with a property predictor was the sum of the ELBO loss and the mean-squared error (predictor's loss), the latter of which was simply set to 0 for any datapoint without an associated value in the training dataset. Thus, training proceeded in a semi-supervised or fully-supervised manner for the predictors, depending on the extent to which labels were made available for the training dataset.

All $2 + 5 \times 5 = 27$ models were trained on the associated loss function for 100 epochs using PyTorch.

2.6 Dataset

To train our TransVAE models we used a peptide sequence dataset consisting of 665 772 sequences. We sourced sequences from AMP-specific databases DRAMP V3.0,[44] APD3,[45] GRAMPA,[36] dbAMP 2.0,[46] StarPep,[47] and general protein databases SwissProt and TREMBL.[48] Using peptide sequences from databases of known antimicrobial peptides ensures that the data distribution—and therefore the distribution being learned by the TransVAE—contains examples of sequences that are AMPs. Additionally, to ensure the training distribution also contained non-AMPs, we excluded those peptides from SwissProt annotated with the keywords antifungal, antimicrobial, antibiotic, antiviral defense, antiviral protein, secreted; although the impact of keyword combinations on predictive model performance has been discussed elsewhere.[49] Lastly, we clustered TREMBL sequences at 90% identity using CD-HIT,[50,51] then randomly sampled 7500 peptides at each length, yielding a subset of 668 343 peptides (the smallest length in the TREMBL set after clustering was 15). Once we removed duplicates, the final dataset of unique peptides from every database was 665 772 sequences. This final amount was split into a training set of 523 848 sequences and a test set of 141 924 sequences where we ensured that the training set and test set had the same proportion of verified AMP and verified non-AMP peptides (Fig. S2).

2.7 Principal component analysis for dimensionality reduction

We leverage dimensionality reduction for two reasons. The first is to obtain low-dimensional visualizations of the 64-dimensional latent spaces associated with the different trained TransVAE models, enabling us to see whether the space has been organized by the property-of-interest. The second reason is to examine whether performing Bayesian

optimization over a reduced representation of the latent space impacts its performance.

We employ principal component analysis (PCA) for dimensionality reduction, a commonly employed linear dimensionality reduction method.[52] In PCA, the centered covariance matrix of the data is decomposed into its eigenvectors, which are then ordered by their corresponding eigenvalues. The top eigenvectors—the top principal components—capture the most variance in the data points. PCA has the advantage of being a mathematically-invertible function and of being highly interpretable due to its linear nature; however, it suffers from an inability to accurately project nonlinear surfaces.

2.8 Latent Bayesian optimization procedure

To perform latent BayesOpt, we use the BoTorch Python package,[32] leveraging its SingleTaskGP Gaussian process object with input unit hypercube and output standard normalization, as suggested by the codebase. As noted above, we use the log expected improvement (LogEI) acquisition function, which offers improved numerical stability compared to conventional expected improvement.[34] For each BayesOpt experiment, we perform five different optimization runs of 500 iterations ('oracle' calls). Each run is initialized with 100 points randomly sampled from predicted $\log_{10}$(MIC) values (see SI for additional details). Every BayesOpt experiment uses the same 5 × 100 initialization points, unless otherwise specified.

For each of the models we trained, we compare searching through their 64-dimensional latent spaces with searching through different dimensional PCA projections of their latent space. To obtain the projected space, for a given VAE, we encoded the peptide training data set into the VAE's latent space, then performed a PCA decomposition of those points to obtain the components capturing most variation in the data. We used the top-n components to make the BayesOpt search space, with n varying across 2, 5, 10, 20, 32. During a given BayesOpt iteration l, optimization of the acquisition function yields a candidate point $\vec{\mu}_{l+1}$ in the 64-dimensional latent space; alternatively if we are searching through a PCA-reduced space, then the candidate point is mapped back to its corresponding 64-dimensional latent space through a PCA inverse projection, giving $\vec{\mu}_{l+1}$. We then decode $\vec{\mu}_{l+1}$ to get a peptide sequence $\vec{x}_{l+1}$ that is given to the oracle to get a predicted $\log_{10}$(MIC) value. We take the negative of that prediction to obtain the objective value for that point in the latent space (or in the PCA projection of the latent space).

3 Results

3.1 Property prediction leads to latent space organization, even in data-sparse conditions

To assess the latent spaces associated with the different models, we quantified the extent to which the top five principal components of a PCA projection correlated with different physicochemical properties used in training. We then employed these PCs to visualize relevant two-dimensional slices of the latent space. To do this we first computed each property over the training set of sequences. Next, we embedded those sequences in the full latent space of each model. Then we decomposed the space into its top five principal components. For each principal component, we computed the Pearson correlation coefficient between it and the property values.

In Fig. 2, we visualize the PC components most highly correlated with the Boman index, charge, and hydrophobicity for the models trained with all three. In line with previous results,[25,26] we observe latent space organization when jointly-training a decoder and a property predictor. Additionally, training a property predictor predicting multiple different properties results in an organized latent space where different properties correlate most with different pairs of dimensions (Fig. 2 rows; Table S5, particularly the Boman index and charge). Furthermore, varying percentage of property labels—thereby limiting the amount of labels the model has access to in training—still results in latent space organization, as indicated by the persistence of an orderly color gradient between the first and last rows of Fig. 2. Overall, increasing the number of properties and masking even a large proportion of the information for the predictor did not prevent the latent space from being organized (for a more detailed discussion of correlations between PC components and organizing variables, we refer the interested reader to section S2.3 in the SI).

Given that for 64-dimensional latent spaces we are visualizing just two-dimensional slices through PCA projections, it is possible that the organization is being induced by the projection. To check the extent to which PCA is distorting the original, high-dimensional manifold, and thus the extent to which it may be providing an illusion of organization when the high-dimensional cloud of points is not actually organized, we compute four manifold distortion metrics as we have done previously:[25] trustworthiness and continuity,[53] steadiness and cohesiveness.[54]

These distortion metrics quantify different ways in which arrangements of points can be disrupted when transforming between the high-dimensional manifold and the low-dimensional manifold ("reduced space"). All run from zero, indicating high distortion (low faithfulness), to one, indicating low distortion (high faithfulness). Trustworthiness measures how frequently points that are not clustered together in the high-dimensional space become clustered together in the reduced space. Conversely, continuity measures the extent to which clustered points in the high-dimensional space become no longer clustered together in the reduced space. Unlike trustworthiness and continuity, which measure the number of nearest neighbours being lost or gained, steadiness and cohesiveness estimate the amount of stretching and compressing occurring when transforming between spaces. Steadiness quantifies the extent to which groups of points that are separated in the high-dimensional space remain separated in the reduced space. Cohesiveness

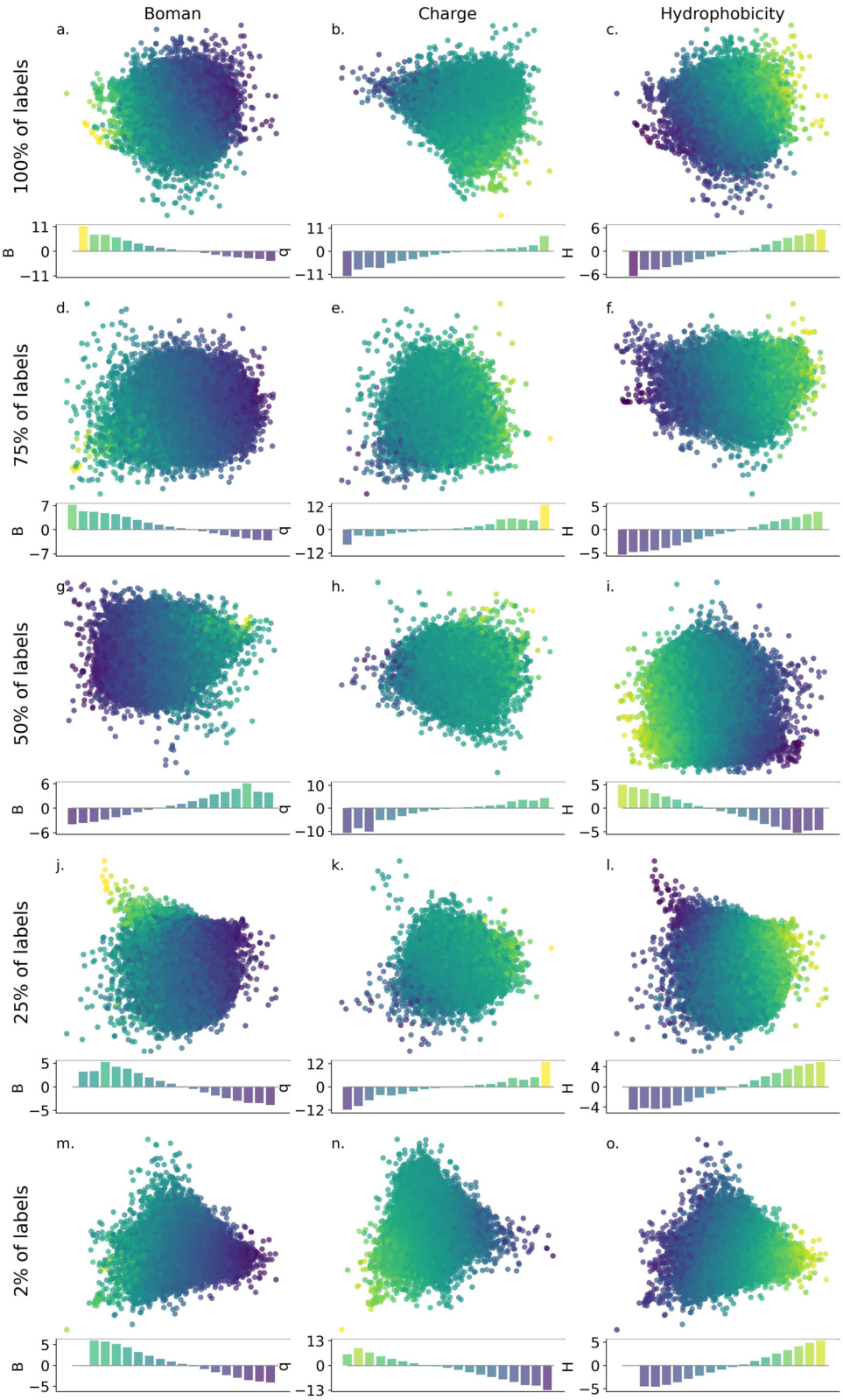

Fig. 2 PCA-projected TransVAE latent spaces jointly-trained by the Boman index, hydrophobicity, and charge, at varying percentages of property values. The left column (subpanels a, d, g, j and m) is coloured by the Boman index. The middle column (subpanels b, e, h, k and n) is coloured by charge. The right column (subpanels c, f, i, l and o) is coloured by hydrophobicity. The visualized principal components are the two PCs most correlated with the property value used for coloring the plot, as measured by a Pearson *r* value (see Table S5 for each PC pair). Bar plots share *x*-axes with their corresponding scatterplots; bar heights are the average value of the column's property averaged over the points lying in a bar's *x*-axis interval width; monotonic trends suggest the (average in) property changes monotonically along the shared *x*-axis. Both the scatterplot and its shared bar plot have the same colourmap. We observe property organization at each percentage of masking, and for each property, although with changes in relevant PC and direction.

quantifies the extent to which groups of points near each other in the high-dimensional space remain near each other in the reduced space. A more detailed description, including algorithmic details, is available in Jeon *et al.*[54]

We compute the extent of manifold distortion between the full latent space and the 5-dimensional PCA projections using a held out test set of sequences. We observe similar levels of distortion between high-dimensional latent spaces and their PCA projections for models trained with different organizing properties and different percentages of labels (Fig. 3). Trustworthiness, continuity, and steadiness are all relatively high, with each organizing method exhibiting values >0.75 (Fig. 3a–c), suggesting high-quality projections with few artificial clusters (trustworthiness), minimal loss of clusters (continuity), and little stretching of distances between points (steadiness). Cohesiveness, lying near 0.6 for all, is moderate, suggesting that compression is occurring (Fig. 3d), as it necessarily must when moving from high to low dimensions. The similarity of the results and the comparatively low-to-moderate distortions observed provide confidence that the visualizations reflect extant trends between, and patterns in, the high-dimensional latent spaces.

We note in particular similar levels of distortion at various label percentages. Comparing within subpanels of Fig. 3, the bar plots are relatively stable in value. From these results, we infer that the original high-dimensional latent spaces are being organized even when label percentage is low, and that we are not merely seeing an illusion of organization induced from the PCA projection. In conjunction with Fig. 2, this confirms that the number of labels can be relatively sparse and still result in the latent space being organized; perhaps most surprisingly in the case of having access to only 2% of the property labels.

For a final note, each model was trained to the same stopping point (100 epochs). At the end of training, the models all had similar training and validation loss values (Fig. S3). We interpret this to mean that the peptide representations learned by the models are the essential difference between them, not their reconstruction accuracy.

3.2 Bayesian optimization in a linearly-projected space can outperform optimization in the latent space directly

Because BayesOpt can be more efficient in lower-dimensional search spaces, and because PCA provides a search space where each dimension is ordered by the amount of explained variance, we hypothesized that reducing the dimensionality of the search space prior to BayesOpt *via* a simple linear projection may improve performance. To test this hypothesis, we compare the performance of BayesOpt in the full latent space associated with the 64-dimensional VAE with the performance of BayesOpt in lower-dimensional projections of this space and compare the results. To obtain the projected space, for a given VAE, we encode the peptide training data set into the VAE's latent space, then performed a PCA decomposition of those points to obtain the components capturing most variation in the data. We used the top-n components to make the BayesOpt search space, with n varying across 2, 5, 10, 20, 32.

Somewhat to our surprise, we observe that BayesOpt in higher-dimensional linearly-projected spaces tends to result in a higher average final objective score than in lower dimensional ones, $\langle \mathscr{M} \rangle$ (Fig. 4). Indeed, for the BCH-organized space when 100% of the property labels were available at VAE training time (Fig. 4a), we observe a monotonic increase with dimensionality in the final objective value achieved, although scores are comparatively similar for all linearly-projected spaces for about the first 100–200 iterations. For the oracle-organized space when 100% of the property labels were available at VAE training time and the BCH-organized space when the 2% of the property labels were available at training time, we observe similar behavior for most of the searches in linearly-projected spaces but 10, 20, and 32 dimensions slightly outperform 5 dimensions by the final iteration, and all outperform 2 dimensions, which flat-lines rather early (Fig. 4b and c). The trends are less clear

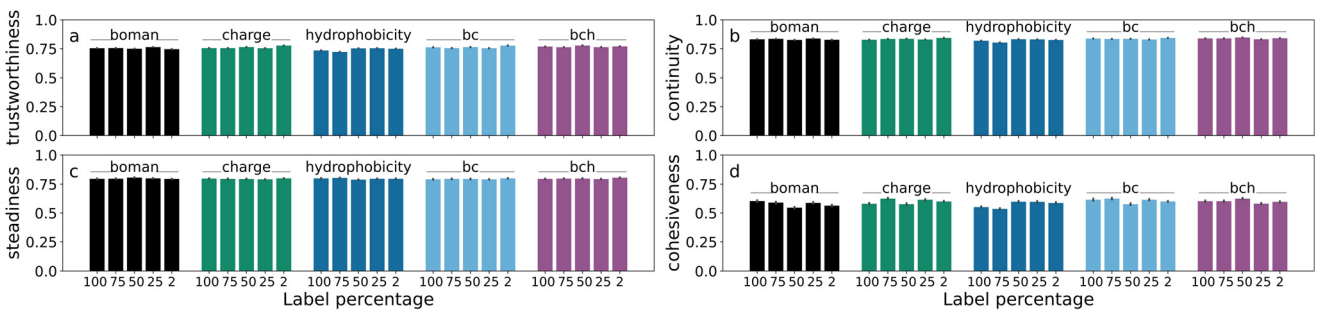

Fig. 3 Quantifying the distortion induced by using PCA on the latent spaces. For each quadrant, from left to right we have the property (or properties) used to organize the latent space; starting on the left, Boman index, charge (pH = 7), hydrophobicity, Boman and charge (pH = 7) (bc), and all three (bch). Variability in each quantity is estimated with 35 equally-sized subsamples of 42 000 different points in the latent space. (a) The estimated trustworthiness of the PCA space compared to the original 64-dimensional latent space. (b) The estimated continuity of the PCA space compared to the latent space. (c) The estimated steadiness of the PCA space compared to the latent space. (d) The estimated cohesiveness of the PCA space compared to the latent space.

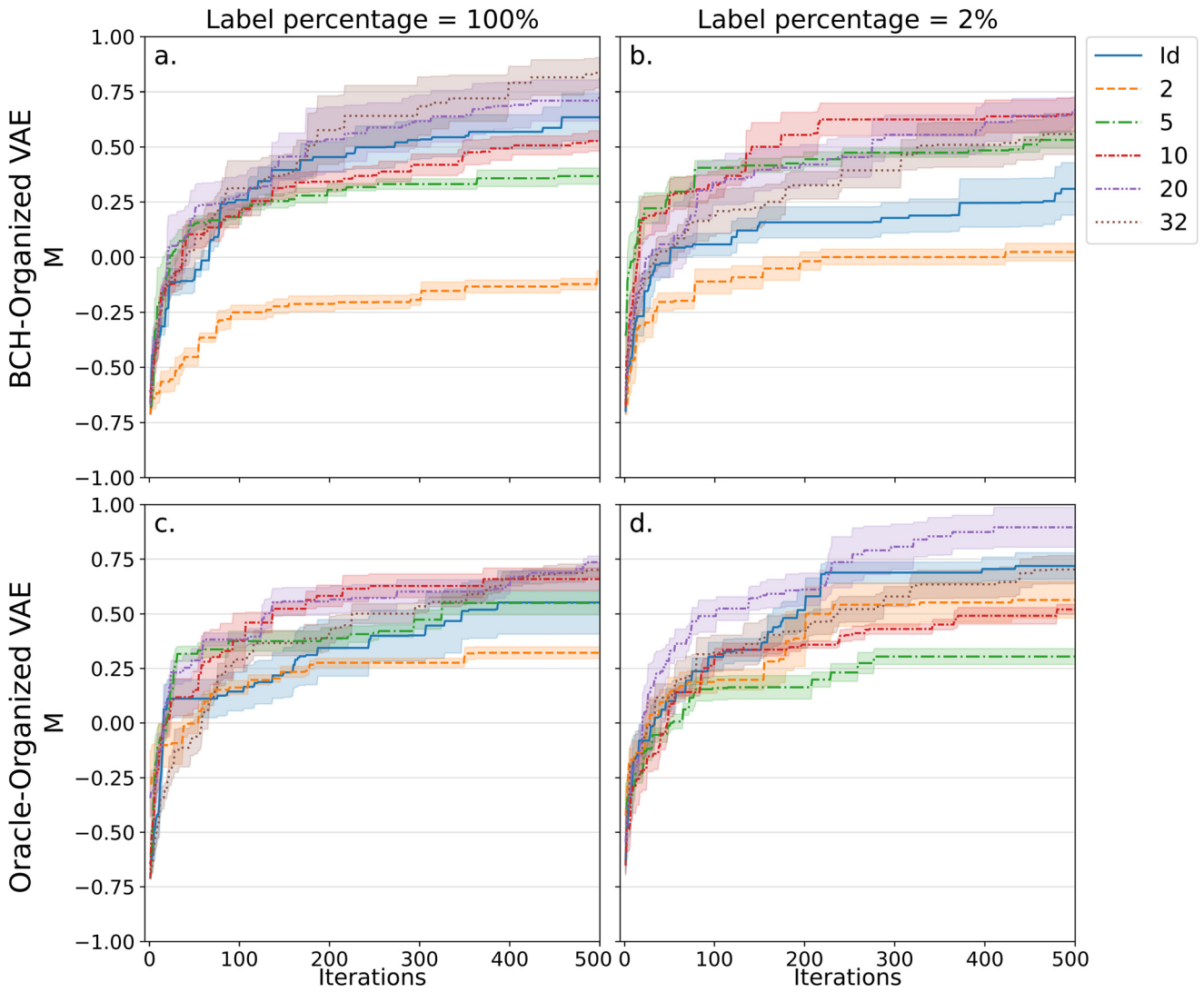

Fig. 4 Best scores identified through Bayesian optimization runs while varying the search space dimensionality. Bayesian optimization results for searching through the 64-dimensional latent space (blue line, "Id" for "identity projection"), and for searching through the projected latent space with varying numbers of PCA components, of BCH organized VAE (a and b) and of oracle-organized VAE (c and d). Label percentage corresponds to the percentage of property labels available during VAE training. Objective scores $\langle \mathscr{M} \rangle$ averaged over five BayesOpt runs with randomly initialized starting points. Error bars represent standard errors across the five runs.

for the oracle-organized space at 2% available label percentage; however, we do note that 20 and 32 dimensions outperform 2, 5, and 10 dimensions (Fig. 4d).

Despite the fact that we observe a correlation between available search space dimensionality and performance in the linearly-projected spaces, better performance is available in all cases in at least one of the lower-dimensional linearly-projected spaces than in the higher-dimensional full 64-dimensional space. When just 2% of the labels were available at training time, BayesOpt in a linearly-projected space of the BCH-organized VAE outperformed BayesOpt in the original latent space for all but the 2-dimensional projection (Fig. 4b). In the case when 100% of property labels were available, we observe better performance when optimizing in a 20 or 32-dimensional PCA reduced space than when optimizing in the latent space directly (Fig. 4a) for the BCH-organized space, while we observe better performance when optimizing in a 10, 20, or 32-dimensional PCA reduced space than when optimizing in the latent space directly for the oracle-organize space (Fig. 4c). Perhaps most interesting is the behavior we observe in the case when few labels are available to organize the VAE latent space, which is closely-related to real scenarios of finding optimally antimicrobial peptides with relevant experimental labels. In this case (Fig. 4d), optimizing in 20 PCA components clearly outperforms optimizing in the corresponding latent space directly, with scores rising more rapidly to a higher final average score after 500 iterations ($\langle \mathscr{M}_{\text{final}} \rangle$ = 0.896 ± 0.092 compared to 0.719 ± 0.061). Optimizing in 32 PCA components performs about the same as optimizing directly in the latent space, reaching a final average score of $\langle \mathscr{M}_{\text{final}} \rangle$ = 0.702 ± 0.063. Surprisingly optimizing in just 2 PCA components performs nearly as

well: although the scores plateau at low values between ~100–175 iterations, they rise rapidly thereafter, attaining a final average score of $\langle \mathscr{M}_{\text{final}} \rangle$ = 0.564 ± 0.084, mostly outperforming optimization in 10 components, and quite substantially outperforming optimization in 5 components.

When we expand to investigate all trained models, we also observe weak evidence of early higher performance in PCA-projected spaces (Fig. S10. Models are labeled using the following convention: S-prop-*l*%, where S is 'id' for searching through the full latent space and 'PCAX' for searching through *X*-dimensional PCA projections, 'prop' is the organizing property, and *l*% is the label percentage). After just 50 iterations, most models are not performing particularly well yet, though we observe more variation in performance (both positive and negative) in BayesOpt runs in PCA-projected spaces than in the full latent space. The highest-performing models at this point are PCA5-bch-25% (average score above 0.4) and PCA5-oracle-100%, PCA20-oracle-100%, PCA5-bch-2%, PCA10-bch-2%, and PCA20-oracle-2% (average score above 0.3). Since all highest-performing models are in PCA projections, this provides weak evidence that PCA projections, particularly if given more properties or more relevant information can provide an early advantage; however, given that, *e.g.*, PCA2-bch-100% has an average score of −0.4, it is clear that this is not a reliable or easily-accessible advantage.

Overall, we observe that performing BayesOpt in a PCA projection of VAE latent spaces may provide certain overall benefits. While the optimal number of PCA components to use is not obvious (*e.g.* 32 *vs.* 10 in Fig. 4a and b), the number is less than the original latent space and in almost all cases provides a performance advantage. Additionally, it is substantially easier to visualize and interpret the search trajectory in lower dimensions, with two dimensions allowing direct visualization of the trajectory.

3.3 Searching through a PCA projection with a more relevant organizing property can provide an advantage

Given the number of different physicochemical properties by which a latent space may be organized, we perform an ablation study testing different numbers and combinations.

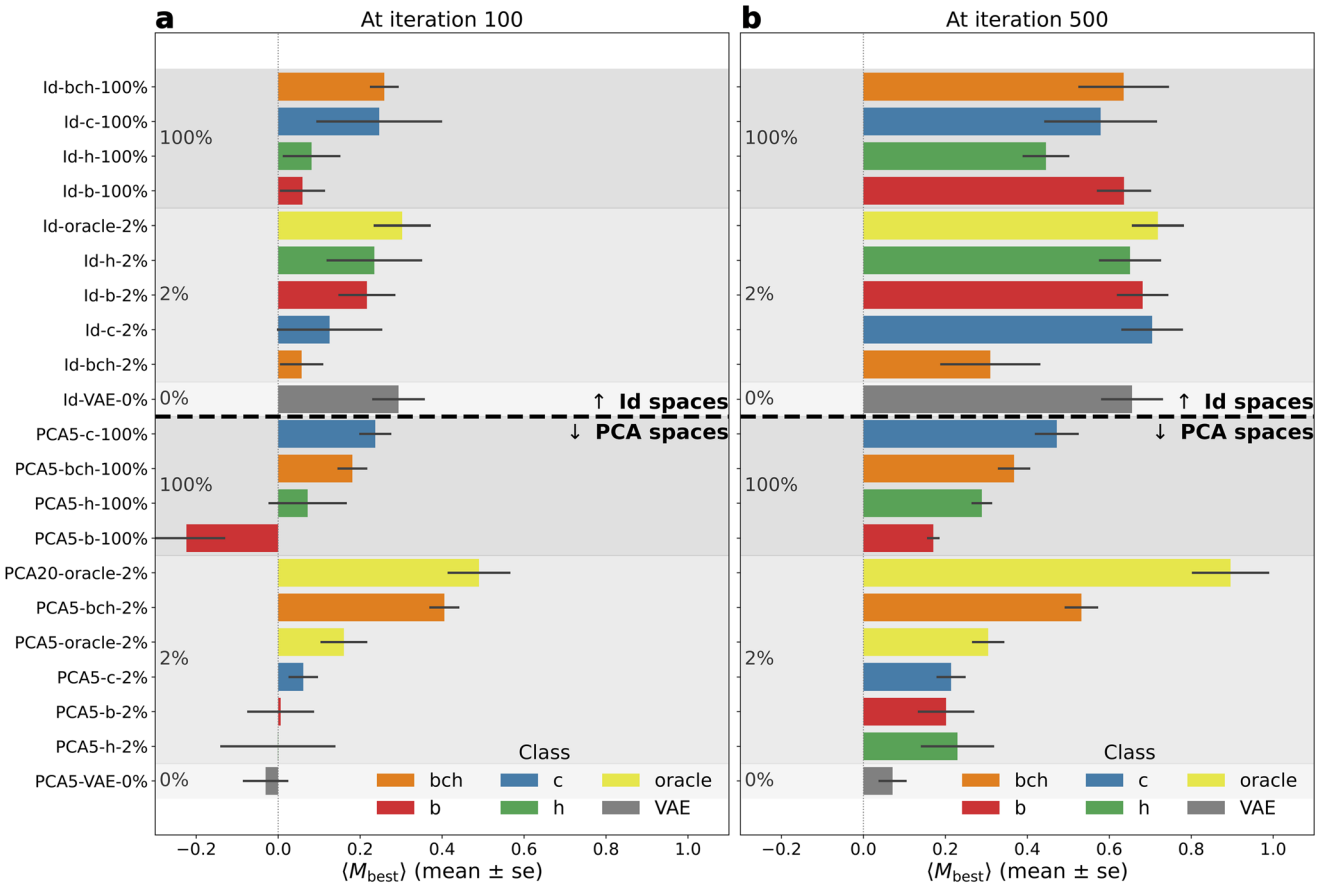

Fig. 5 Average best objective scores found after (a) 100 iterations and after (b) 500 iterations. Error bars correspond to the standard error of M_{best} over five BayesOpt runs. Black dashed line separates those runs performed in the high-dimensional latent space (above the dashed line), and those runs performed in a PCA projection of the latent space (below the dashed line). Shaded regions denote the label percentage available at VAE training time, with darker regions corresponding to a higher label percentage. In (a) values are sorted within groups from largest to smallest, while (b) uses the same ordering as (a). Each class refers to the properties used for organizing the latent space, with "VAE" referring to an unorganized latent space. The full BayesOpt trajectories are depicted in Fig. S7.

We hypothesized that organizing the latent space with more properties would lead to peptide representations more amenable to BayesOpt. We summarize the performance at different iterations in Fig. S10 and 5a and b, where we investigate average best scores found under different conditions. We primarily focus on PCA5 models. We choose the top five components to balance capturing a larger amount of explained variance (near 20%, Fig. S4) with losing interpretability, and because three of the four scenarios where we varied the number of PCA components demonstrated five components performing better than two components (previous section).

We consider single property organization first. There is no clear advantage of any single property when searching through the 64-dimensional latent spaces – the best-performing single-property models compared to other models with only organizing property changed are id-c-100%, id-b-75%, id-h-50%, id-h-2% at 100 iterations (Fig. 5a), and id-b-100% and id-c-100%, id-b-75%, id-d-50%, id-c-25%, and id-b/c-2% at 500 iterations (Fig. 5b). For PCA-projected spaces, we observe that charge-organized spaces tend to demonstrate better performance: at 100 iterations, PCA5-c-100% clearly outperforms b/h-100%, PCA5-c-75% clearly outperforms b/h-75%, PCA5-c-50% clearly outperforms b/h-50%, and PCA5-c-2% clearly outperforms b/h-2%. Only for the PCA5-25% models is charge not the clear winner, and in that case, it performs nearly as well as PCA5-h-25%. At 500 iterations, PCA5-c-100% clearly outperforms b/h-100%, PCA5-c-75% clearly outperforms b/h-75%, and PCA5-c-50% clearly outperforms b/h-50%. However, PCA-5-c-25% performs worse than b/h-25%, and all PCA5-2% models perform roughly the same, displaying rather poor performance at <0.3 average best scores. Thus, charge is a better organizing property than the Boman index or hydrophobicity, particularly when more labels are available. This trend matches with a similar trend in relevancy: charge correlates more strongly and has higher mutual information with the oracle values when looking at the full peptide training set (section S2.3).

Considering multi-property organization, we still do not see clear trends in when searching through the full 64-dimensional latent space. After 100 iterations (Fig. 5a), id-bch-100% is one of the best performers (with performance about equivalent to id-c-100%), and id-bch-75% outperforms other models, but the highest-performing model for lower label percentages is one or more of the single-property models. After 500 iterations (Fig. 5b), we see similar model performance across the board, though id-bch-100% is one of the highest-performing models (tied with id-b-100%) and id-bc-75% is one of the highest-performing models (tied with id-b-75%) but for lower label percentages single-property models are the best, and the performance is not too dissimilar within error bars.

When searching through a PCA projection, there is not a clear advantage to using multi-property organization. After 100 iterations (Fig. 5a), at 100% and 75% label percentage, charge clearly outperforms multi-property organization. At 50% label percentage, PCA5-bc-50% outperforms PCA5-c-50%, at 25% label percentage, PCA-bch-25% by far outperforms any other models, and at 2% the highest performing model is PCA5-bch-2%. After 500 iterations (Fig. 5b), at 100% and 75% label percentage, charge still outperforms multi-property organization. For lower label percentage, multi-property organization does outperform single-property organization (PCA5-bc-50% $>$ PCA5-bch-50% $>$ PCA5-c-50%, PCA5-bch-25% is clearly the best performer, PCA5-bch-2% outperforms PCA5-bc-2% and both clearly outperform the single-property organizers), but neither bc nor bch is consistently better than the other. This suggests that multi-property organization may be better at lower label percentage, an observation which is supported by the fact that the best-performing model we observed in the study was PCA-20-oracle-2%, with a final performance of $\langle \mathscr{M}_{\text{final}} \rangle = 0.896 \pm 0.092$, where the oracle is itself a multi-property organizer, although it was not part of the ablation study.

Overall, we find that performance depends more heavily on latent space organization when performing BayesOpt in PCA-projected latent spaces than when performing it in the full latent space. We find that for PCA-projected latent spaces, more relevant variables result in improved performance, with more types of information in the form of multiple properties or highly-relevant oracle values providing an advantage particularly in the very low percent label regime. Moreover, at iteration 500, using a linear projection of the oracle-organized latent space leads to the highest $\langle M_{\text{best}} \rangle$ over all experiments we tried. This suggests that the case of having very relevant data but very low label percentage (the situation we find ourselves in with AMP datasets associated with experimentally-measured antimicrobial activity) could be a good scenario for using a linear projection of the latent space as the BayesOpt search space.

3.4 Bayesian optimization in a linearly-projected space may outperform that in a nonlinearly projected space

We additionally compared performing BayesOpt in a linearly-projected space to performing BayesOpt in a nonlinearly-projected lower dimensional space on the fly by employing Gaussian process deep kernel learning (GP-DKL).[55] GP-DKL is a variant of a Gaussian process that uses a neural network to augment the kernel function. By learning a projection $g_\theta(\vec{x})$ mapping design points into a feature space that the kernel function operates on, the conventional BayesOpt kernel, $k(\vec{x}_1, \vec{x}_2)$, is replaced by $k(g_\theta(\vec{x}_1), g_\theta(\vec{x}_2))$, where g_θ is a neural network. The weights θ of g_θ are fit simultaneously with the parameters of the kernel $k(,)$ at each BayesOpt iteration. GP-DKL is an a promising alternative approach to deriving a low-dimensional space for BayesOpt to traverse, as it operates directly on the high-dimensional space and leverages a learnable non-linear projection (as opposed to the static, linear one we have used) enabling the GP to determine the projection necessary to find an optimal point.

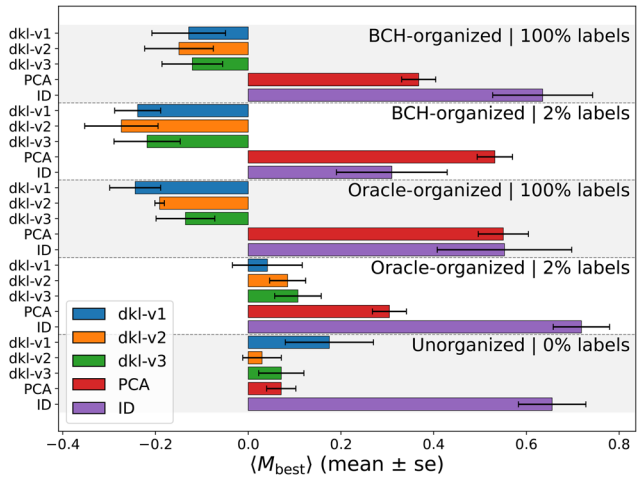

Fig. 6 Average best objective scores found after 500 iterations. Error bars correspond to the standard error of M_{best} over five BayesOpt runs. Abbreviations correspond to neural network architectures described in section S3.3. The full BayesOpt trajectories are depicted in Fig. S7.

Here we implemented GP-DKL, and performed BayesOpt with five-dimensional GP-DKL in five of our TransVAE spaces: unorganized, BCH-organized with either 100% or 2% of the labels, and oracle-organized with either 100% or 2% of the labels. Across three different architectures for the GP-DKL neural network (dklv1, dkl-v2, dkl-v3; see section S3.3 for more details) we find that, in all but one case, BayesOpt in a five-dimensional PCA-projected space is able to consistently perform as well as, or better than, GP-DKL (Fig. 6 and S13).

In an unorganized TransVAE model, two architectures from GP-DKL (dkl-v2, orange; dkl-v3, green) perform as well as BayesOpt in a PCA projection (red), but one GP-DKL variant (dkl-v1, blue) does slightly better than searching in the PCA projection. However, we observe that searching directly in the high-dimensional space (purple) results in substantially higher values attained than either GP-DKL or BayesOpt in the linear projection, thus pointing to fundamental limitations of searching in a lower-dimensional space without additional information to permit an informed choice of directions.

Once we move to organized spaces, there is a clearer difference. In the BCH-organized spaces, all three GP-DKL variants have average best scores below zero while the BayesOpt searching through the high-dimensional space and searching through the PCA projection are able to attain average greater-than-zero scores by at latest the 100th iteration. This trend also holds when we look at the BayesOpt runs done in the oracle-organized spaces. This may be a result of differences in data availability to the different algorithms. The PCA projector is fit to the (embedded) training set of the VAE, which is $O(100k)$ points. This projector is then not updated throughout the BayesOpt loops. In DKL, the neural network projector is fit during the BayesOpt iterations on the fly, meaning it can search in different directions while severely restricting the amount of data available ($O(100)$ points in our scenario) compared to the PCA projection. Further work could be done to optimize a nonlinear projection scenario.

3.5 Searching through a PCA projection is associated with more exploration

To further understand BayesOpt performance in our spaces, we examine how much exploration occurs; that is, how much of the space is probed by the search. We quantify the amount of exploration that occurs throughout a BayesOpt run through four lenses: (1) the amount of the total latent space (input) a BayesOpt run spans, which we estimate using the hypervolume of the sampled points; (2) the amount of the scores (output) a BayesOpt run spans, which we estimate using the variance in objective scores sampled; (3) the distance from the oracle SVR's training set; and (4) the amount of the relevant latent space (relevant input) a BayesOpt run spans, which we estimate using the total path length of the sampled points. We first characterize the exploration and then assess its relationship with exploitation as previously quantified by achieved best scores.

We compute the hypervolume of the input points sampled in each BayesOpt run, which is essentially a measure of the extent to which we explore different objective function inputs. The hypervolume was computed using the pygmo python package;[56] it computes the hypervolume of a set of points compared to a reference point. To choose our reference point, we used the corner of the search box with the maximum value in each dimension (10 in each dimension). When looking at the distribution of resulting hypervolumes, we observe more hypervolume is explored on average when optimizing in a PCA search space compared to the high-dimensional search space; the PCA distribution (Fig. 7a orange) is shifted to the right of the high-dimensional search space distribution (Fig. 7a, blue), with $\mu_{PCA,HV} = 1.1 \times 10^{-17} > 0.55 \times 10^{-17} = \mu_{Id,HV}$. This suggests that a broader slice (about 50% more) of the underlying high-dimensional latent space was explored despite the PCA runs being confined to a subspace of it.

We further compute the variance in objective scores sampled throughout each BayesOpt run, which is essentially a measure of the extent to which we explore different objective function outputs. When comparing the distribution of variances in objective scores for runs searching through the PCA space (with center $\mu_{PCA,Var(M)} = 0.24$ (± 0.005)) *vs.* the high-dimensional latent space (with center $\mu_{Id,Var(M)} = 0.17$ (± 0.004)), we observe that the BayesOpt runs done in PCA spaces tend to have more variance in objective function scores (Fig. 7b); $\mu_{PCA,Var(M)} = 0.24 > 0.17 = \mu_{Id,Var(M)}$. We test the hypothesis that these two means are the same using an independent samples t-test, yielding a p-value $= 1.58 \times 10^{-21}$, suggesting a statistically robust difference. This suggests that when optimizing through linear-projections of the high-dimensional space, more of the objective function is sampled (about 30% more).

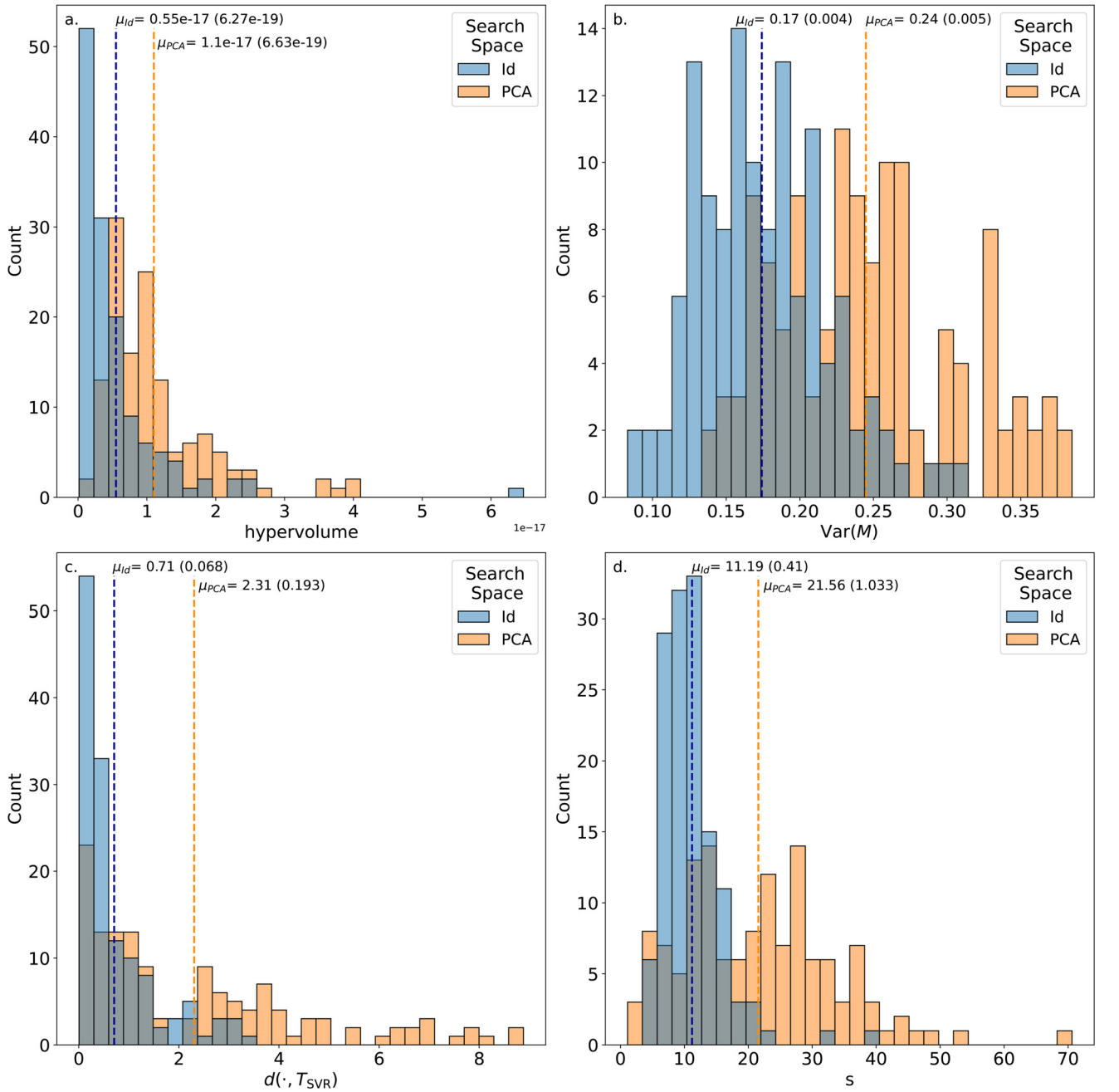

Fig. 7 Distributions of exploration quantities for optimization done in the full latent space ("Id", blue) or the PCA projection of it (PCA, orange). Distribution of each run's: (a) computed hypervolume; (b) variance in scores sampled, Var(M); (c) best sampled point's distance from the oracle's training set (eqn (8)); (d) total path length (eqn (9)). For (c) and (d) the distances are computed in the two PCs most correlated with the oracle values.

Additionally, we compute the distance of the best point in a run from the SVR oracle's training set as an assessment of how exploration of input may be related to exploration of output and an assessment of the algorithm's ability to suggest novel optima. Because computing distances in high-dimensional spaces can be misleading, we computed these distances in a two-dimensional projection of the latent space. We construct this two-dimensional projection using the two principal components most correlated with oracle values. To do this for a run, we first encoded the oracle's training set into the high-dimensional latent space the BayesOpt run was done in. We then compute a PCA projection of the high-dimensional latent space. The oracle's training set is then projected into the PCA-reduced space. Then, we compute the Euclidean distance between the best point found during the run and each point in the oracle's training set, in the PCA-reduced space. The minimum distance found is used as the

distance between the point and the oracle's training set T_{SVR}:

$$d(x, T_{SVR}) = \inf\{d(x, a) : a \in T_{SVR}\} \quad (8)$$

We observe that the BayesRuns searching through linear-projections of latent spaces have a distribution of distances from the oracle's training set with a mean of μ_{PCA} = 2.31 (±0.193). The distribution of runs searching through the high-dimensional latent space has a mean of μ_{Id} = 0.71 (±0.068). From the distribution of distances computed, we observe that the BayesOpt runs searching through linear projections of the latent space tended to result in final points about 225% farther from the oracle's training set when compared to the BayesOpt runs searching through the full high-dimensional latent space (Fig. 7c); μ_{PCA} = 2.31 > 0.71 = μ_{Id}, a statistically robust difference, with a p-value = 1.21×10^{-13}.

Lastly, we compute the total path length of a BayesOpt run as a way of measuring the exploration of the relevant input space. For the k-th point sampled during a BayesOpt run, we compute the Euclidean distance from it to the $(k - 1)$-th point sampled during the run, in the two PCs most correlated with the oracle values of the full VAE training set. We compute the distance between each consecutive pair of points in the PCA-reduced space for the run. Then, we sum the distances to find the total path length in the PCA-reduced space. That is, for a given BayesOpt run with sampled latent space points $\{\vec{\mu}_1, \vec{\mu}_2, \ldots, \vec{\mu}_{500}\}$, projected into a two-dimensional reduced space $\{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_{500}\}$ we compute its total path length as

$$s = \sum_i \|\vec{p}_{i+1} - \vec{p}_i\|_2 \quad (9)$$

From the distribution of path lengths we observe an increase in total path length by about 93% from searching directly in the high-dimensional space compared to the linearly-projected space (Fig. 7d); the centre of the distribution of path lengths for searching in a linearly-projected space is μ_{PCA} = 21.56 (±1.033); compared to the distribution of high-dimensional search space path lengths, whose centre is μ_{Id} = 11.19 (±0.410); this difference is statistically robust, with p-value = 4.18×10^{-18}. We further quantified the exploration in sequence space directly and found similar results (section S3.4).

We further examined the BayesOpt runs in their associated latent spaces qualitatively. To do this, we plotted the peptides sampled during a BayesOpt run in a two-dimensional PCA projection of the associated latent space using the two PCs most correlated to oracle values. In Fig. 8, we depict a full example trajectory showing the latent space (panel a), the structures of 17 selected peptides along the run as predicted by ESMFold (panel b), and their corresponding scores (panel c). This trajectory illustrates the best-scoring

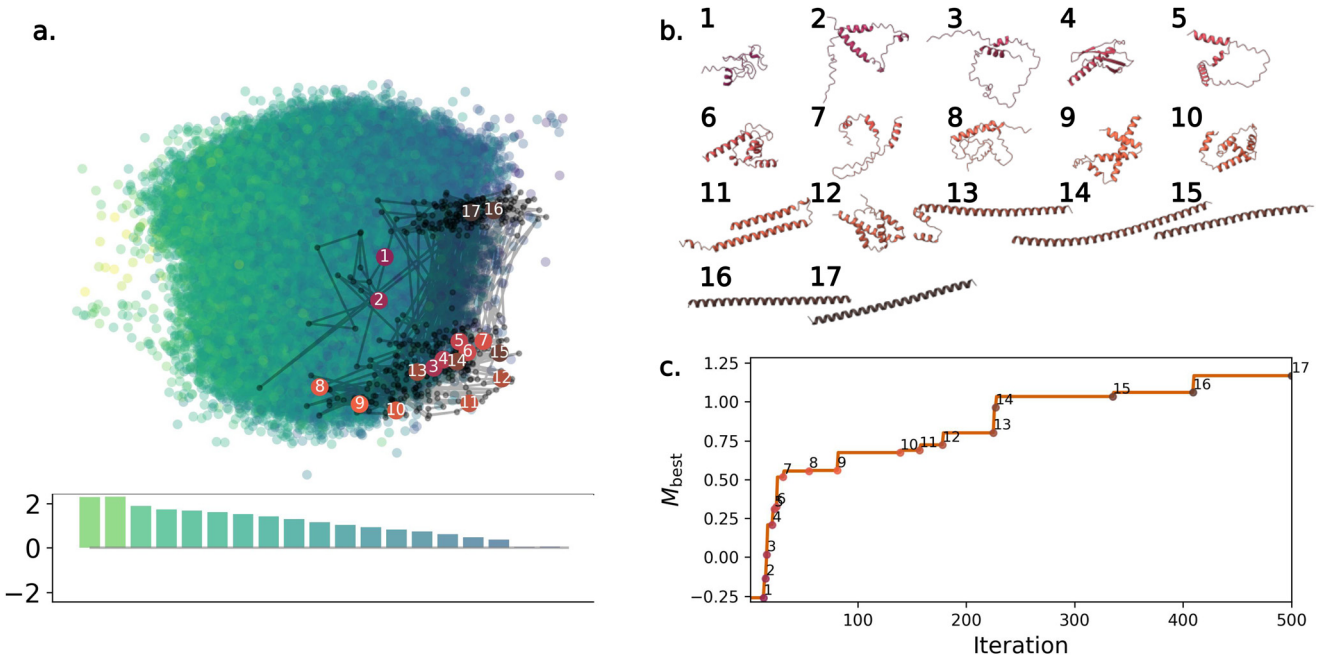

Fig. 8 Best peptides identified during a BayesOpt run. (a) PCA plot of the associated oracle-organized latent space (at 2% of labels available), with bar plot depicting an average decrease in predicted-$\log_{10}$(MIC) from left-to-right across the latent space. The latent space locations of the best peptides are depicted with large circular markers; white integers represent the sequence in which these peptides were identified during a BayesOpt run, but not the iteration at which they were identified. (b) ESMFold-predicted structures[57] of the best peptides found during the BayesOpt run, visualized using ChimeraX.[58] In this case we note a strong increase in helicity with oracle value, a reasonable – but biophysically misleading – proxy that the oracle might learn for anti-microbialness, which may be ameliorated with higher-fidelity evaluations such as simulation. (c) The corresponding best score M_{best} found throughout the BayesOpt run.

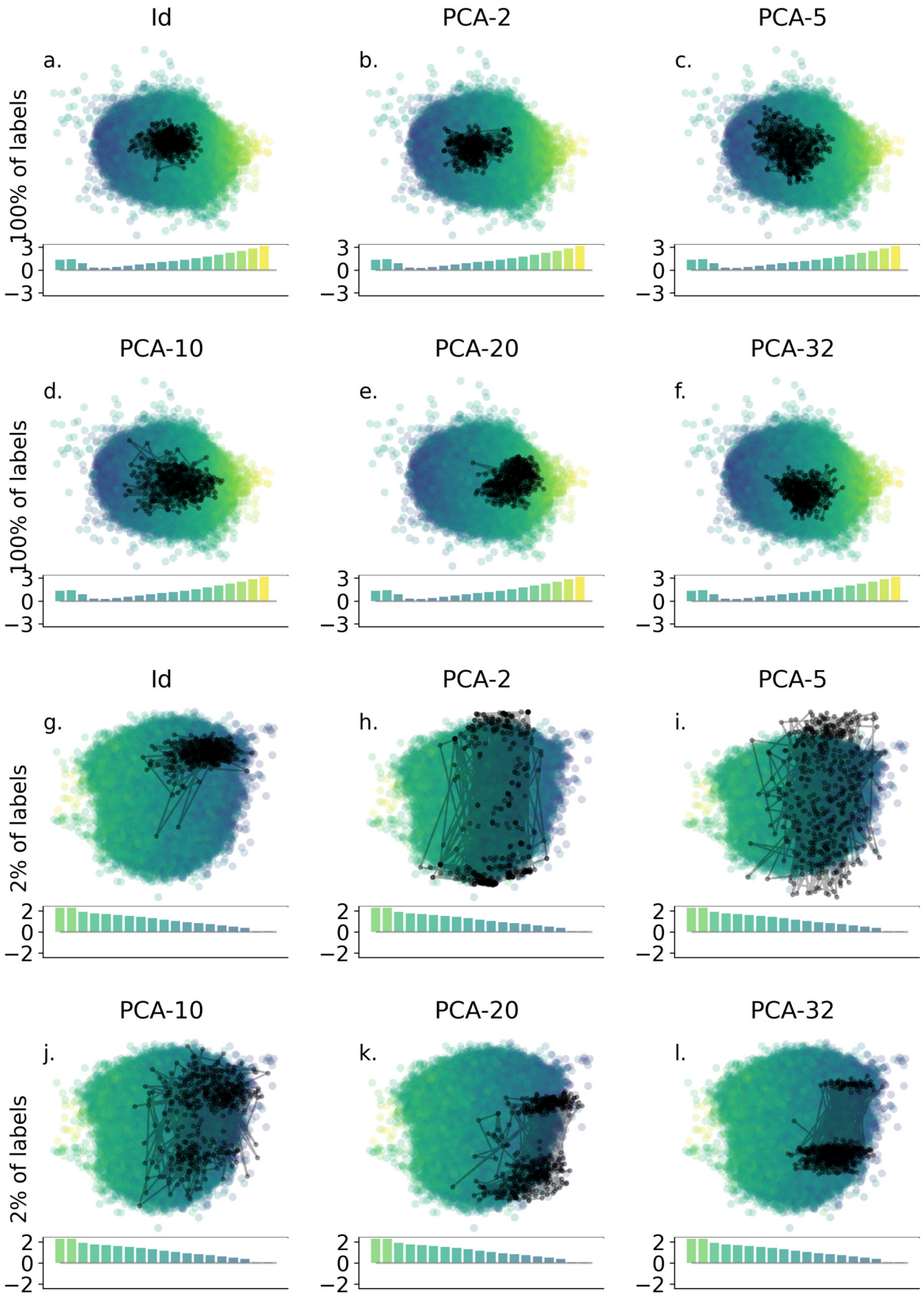

Fig. 9 The BayesOpt run with the best objective score is plotted on a 2D visualization of oracle-organized latent spaces. (a–f) The oracle-organized VAE had access to 100% of the property labels during training. (g–l) Only 2% of the property labels were available during training. Optimization was done in different projections of the underlying latent space: (a) and (g) directly in the 64-dimensional latent space, (b) and (h) in the top 2 PCs of a PCA projection, (c) and (i) in the top 5 PCs of a PCA projection, (d) and (j) in the top 10 PCs of a PCA projection, (e) and (k) in the top 20 PCs of a PCA projection, (f) and (l) in the top 32 PCs of a PCA projection.

run of five that were performed in a 20-dimensional projection of the oracle-organized latent space. This figure has a clear interpretation and allows us to gain a strong physical intuition for the progress of the search. We see the BayesOpt trajectory begin at a low objective value (1) and rapidly increase by moving to the right in the latent space, towards the average direction of increased objective value (lower MIC). After attaining point 7, BayesOpt performs a large explorative jump to the left to point 8, which only marginally increases the best score. It then proceeds by generally drifting to the right and eventually jumping upwards (point 16). Concomitantly, we see that as the search proceeds, the predicted structures increase strongly in helicity. Given the over-representation of α-helical AMPs in most datasets, helicity is a reasonable proxy for anti-microbialness that the BayesOpt is exploiting; a straightforward example of reward hacking. This confirmation of reward hacking provides another reason for using the most relevant data possible (less hackable), even if less of it is available (because *e.g.* it is expensive to compute).

In Fig. 9, we visualize the best-scoring run of five for all oracle-organized latent spaces at 100% and 2% label percentage (for more information *cf.* also Fig. S15–S18). As previously quantified, we visually observe that BayesOpt runs tend to cover more of the 2D visualizations when their associated search space is a PCA projection (*e.g.*, compare Fig. 9g with Fig. 9h–l). Additionally, there is substantially greater spatial exploration when the percentage of labels is low, although as we have seen, BayesOpt is still able to converge to satisfactory optima in this case. In particular, when comparing the low-label regime (2%) with the full label regime (100%), we observe that the BayesOpt runs explore substantially more of the visualization; this is especially the case when BayesOpt runs search through the top two, five, or ten PCA dimensions (more detailed trajectories supporting this discussion may be found in Fig. S15–S24).

The previous observations are certainly due at least partly to the fact that the search procedure is more restricted to the visualized directions for the PCA projections than for the full search space. This leads us to note that interpretation of BayesOpt runs is easiest when the search space is similar to the visualization. We can also observe this in that when the organizing property is a physicochemical property the BayesOpt runs are centered, appearing to only explore a portion in the directions used to make the visualization, since the projection is not well-aligned with the directions of the search. However, when looking at the oracle-organized spaces, we can clearly see the BayesOpt runs identifying the side of the space that contains better objective function values (low value oracle predictions correspond to higher objective scores) after covering more of the visualized region, particularly when fewer labels are available.

3.6 Performance is correlated with sampling a range of objective values but not with exploration of input design space

In this section, we assess whether there is a relationship between exploration as quantified in the previous section and BayesOpt performance as measured by the final best score of the run. We first compare the hypervolume of input points sampled during a run with the best objective score during that run (Fig. 10a). For the BayesOpt runs done in PCA reduced spaces, we observe a weak negative PCC of −0.202 that is statistically robust ($p = 0.019 < 0.05$). For BayesOpt runs done in the 64-dimensional latent spaces, we observe a very weak negative PCC of −0.108 that does not appear statistically robust.

We then compare the variance of scores during a run with the best objective value encountered during that run (Fig. 10b). For BayesOpt runs done in PCA reduced spaces, we observe a strong Pearson correlation coefficient (PCC) of 0.652 that is statistically robust with a p-value < 0.001 (Fig. 10b, crosses). For BayesOpt runs done in the 64-dimensional latent spaces, we observe a moderate PCC of 0.498 that is statistically robust with a p-value < 0.001 (Fig. 10b, dots).

Additionally, we compare the best objective score found in each run to the distance of the corresponding point from the SVR's training set, and to the total path length of a BayesOpt run in two dimensions (Fig. 10c and d). We observe no statistically significant correlation in either of these values.

Overall, sampling a greater range of objective scores results in better final scores, while exploration of the input space as measured here is not strongly correlated with performance.

4 Discussion and conclusions

In this article, we have trained a series of transformer-based VAE models for peptide sequence generation, which we have used as continuous search spaces for latent Bayesian optimization. We presented a version of latent Bayesian optimization with additional dimensionality reduction, and compared it with traditional latent BayesOpt and BayesOpt with deep kernel learning. Inspired by a desire to better align visualization/analysis techniques with algorithmic properties, we took PCA projections of the latent spaces – previously used solely for visualization – and performed BayesOpt directly in them. We assessed the properties of the latent spaces themselves, compared our low-dimensional BayesOpt search with direct Bayesian optimization in the full 64-dimensional latent space of the trained generative model, and connected latent space properties with resultant properties of the BayesOpt search trajectories.

In line with previous work[25,26] we found that jointly-training a property-predictor and a VAE can lead to an organized latent space. Expanding on that work, we investigated whether such organization persists even in a semi-supervised scenario, finding evidence that just 2% of

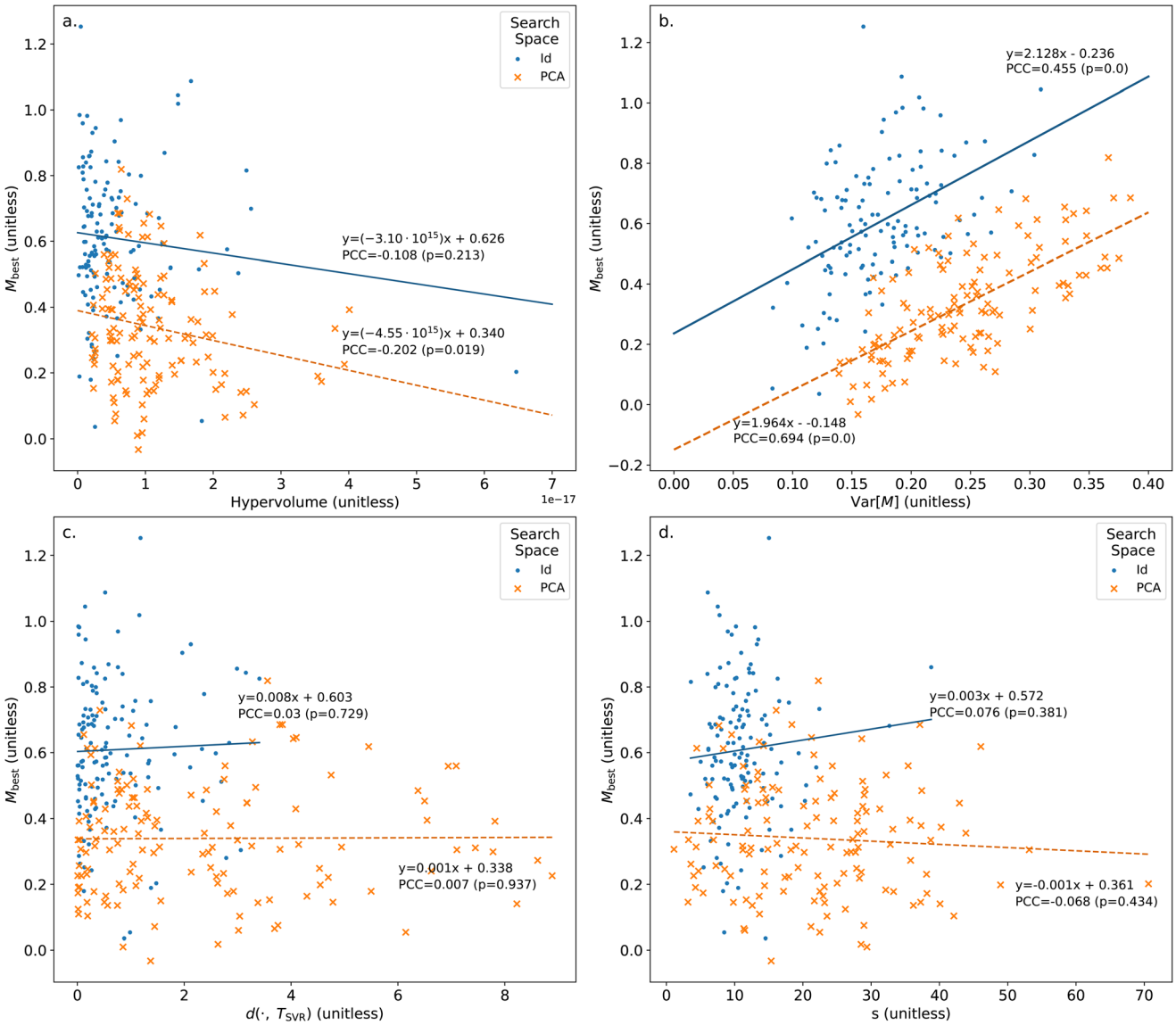

Fig. 10 Relationship between M_{best} and (a) the hypervolume explored in that run; (b) the variance in objective scores Var[M] sampled throughout the run, for each BayesOpt run we performed; (c) the distance between the best sequence's latent representation and the SVR's training set; and, (d) the path lengths of the corresponding BayesOpt run. Each BayesOpt run searched either over a 64-dimensional latent space ("Id", blue dots, solid regression line), or the PCA projection of the latent space ("PCA", orange crosses, dashed regression line), with search box edges ranging from −10 to 10 in each dimension. Solid (dashed) lines correspond to regressing on the dots (crosses). Pearson correlation coefficients (PCCs) are written next to their corresponding point group, with p-values in parentheses. (a) We observe a weak negative correlation between the normalized hypervolume of latent space explored during a BayesOpt run, and the best score found during that run. (b) We observe a moderate (PCC = 0.498 dots) to strong (PCC = 0.652, crosses) correlation between variance in scores sampled throughout a run and the best score found during that run. We observe minimal correlation between the best objective score sampled in a run and (c) that run's final distance from the SVR's training set, or (d) that run's total path length. We compute a run's path length by taking the Euclidean distance between point $\vec{\mu}_k$ and $\vec{\mu}_{k-1}$ and summing over all k = 2, …, 500. For (c) and (d) the distances are computed in the two PCs most correlated with the oracle values.

property labels suffice to induce organization along that property; additionally we showed that jointly-training with multiple properties can lead to organization along each property.

We employed these organized latent spaces for projected latent BayesOpt and traditional latent BayesOpt. When we varied the number of PCA components to optimize in, we found that optimizing in a PCA projection could outperform optimizing directly in the high-dimensional latent space, but it is not obvious *a priori* how to select the optimal number of PCA components to optimize over. In general, using projected latent BayesOpt may provide an early or late advantage and can show impressive performance; however, it is less reliable than performance in the full latent space, which demonstrates less variability in performance with latent space organization and other hyperparameters.

To better understand the use of projected latent BayesOpt, we assessed the effects of different organizing properties of

the latent space. We hypothesized that if the properties provided a weak but ample signal regarding the oracle, it may improve the BayesOpt performance through either increasing the rapidity at which it found potent peptides, or through increasing the max value attained throughout an optimization run. We find that, among the single physicochemical property spaces, charge tends to give better BayesOpt performance in terms of final objective values, and how quickly the objective value increases. This occurs primarily when searching through the linear projection of the original 64-dimensional latent space, but also maintained strong performance when searching through the full high-dimensional space. As charge is the most informative of the physicochemical properties we studied, this demonstrates that BayesOpt in PCA-projected spaces works best when the space is organized by relevant information. We also demonstrated that at low label percentage, we could obtain high performance from employing multiple organizing properties or using the oracle. Overall, in a real-world situation, we expect to see good performance from projected latent BayesOpt if the space is organized by a highly-relevant property; the more types of information available at low label percentage, the better.

Despite the fact that the performance of projected latent BayesOpt can be more variable and more sensitive to latent space organization than latent BayesOpt, projected latent BayesOpt has a significant advantage in terms of interpretability. We can more easily and accurately visualize and understand the trajectories of projected latent BayesOpt. In addition, we showed that projected latent BayesOpt explores more of the search space than latent BayesOpt, in terms of both full hypervolume and exploration along relevant dimensions. Furthermore, the variability in performance may also be framed as heightened exploration of output scores sampled, which we showed to be correlated with better performance. Although projected latent BayesOpt has a weakness, we suggest that it can be ameliorated by running multiple short trajectories in spaces of different dimensionalities, and it comes with the advantage of better understanding of the search procedure. We also note that this work further underscores an important point we have previously raised,[25] which is that naive use of low-dimensional projection algorithms to visualize latent spaces of generative models can be misleading. The community should not over-rely on the assumption that a two-dimensional PCA or t-SNE projection is sufficient to reveal qualities of the latent space, at least not without further analysis.

In sum, latent Bayesian optimization can provide an approach to identifying highly-potent antimicrobial peptides leveraging the continuous-valued latent spaces of generative models. We found that VAEs based on transformers can have their latent spaces organized by multiple different physicochemical properties simultaneously through the joint-training of multiple property predictors with the VAE. We further compared whether using physicochemical properties to organize the latent space is better than using few oracle values, finding evidence (section 3.3) that using few oracle evaluations can be better when restricting the search space to a linear projection of the latent space; *i.e.* having a more relevant organizing property with few labels may be a good use case for searching through a PCA projection of the latent space. Using the linear projection comes with the additional benefits of being more interpretable and easier to visualize.

Although we have primarily discussed our work in the context of AMP design, given that BayesOpt is an efficient, iterative algorithm for searching a design space to optimize a function under limited data, our results extend straightforwardly to any peptide design question. The only limitation is the definition of a relevant objective function. For example, one could optimize for cell-penetrating peptides through *in silico* measurements of maximum force experienced by a peptide as it is pulled through a lipid bilayer in constant-velocity steered molecular dynamics; alternatively, one could optimize for anticancer peptides through *in vitro* measurements of the minimum peptide concentration required to inhibit cancer cell growth, or through *in silico* approximations of binding against a target protein. More broadly, the application of BayesOpt to a particular peptide sequence design challenge is limited primarily by the availability of an oracle that can be regularly queried for estimates of the underlying objective.

Author contributions

JM: conceptualization; methodology; software; validation; formal analysis; investigation; data curation; writing – original draft; writing – review & editing; visualization. RAM: conceptualization; methodology; resources; writing – original draft; writing – review & editing; supervision; project administration; funding acquisition.

Conflicts of interest

There are no conflicts to declare.

Data availability

The code for model training and analysis can be found at https://github.com/Mansbach-Lab/compare-latent-spaces-amps/tree/main and are released on Zenodo with DOI: https://doi.org/10.5281/zenodo.17872434. We note that for certain model training we employed the publicly available GRAMPA dataset.[36] Trained model checkpoints and datasets are released separately through Zenodo (https://zenodo.org/records/17872449) with DOI: https://doi.org/10.5281/zenodo.17872449.

Supplementary information (SI): further detailed analyses, tables and figures available in PDF form (3 sections, 10 tables, and 24 figures). See DOI: https://doi.org/10.1039/d5me00225g.

Acknowledgements

This research was supported in part by Discovery Grant #RGPIN-2021-03470 from the National Sciences and Engineering Research Council of Canada. This research was enabled in part by support provided by Calcul Quebec (https://www.calculquebec.ca) and the Digital Research Alliance of Canada (https://www.alliancecan.ca). This research was undertaken, in part, thanks to funding from the Canada Research Chairs Program under grant number CRC-2020-00225. JM acknowledges an NSERC CGS-D scholarship, as well as the motivating support of the Mansbach lab, particularly Mohammadreza Niknam Hamidabad and Adam Graves for expressing interest and wanting to see this manuscript finished.

Notes and references

1 R. Mulchandani, Y. Wang, M. Gilbert and T. P. V. Boeckel, *PLOS Glob. Public Health*, 2023, **3**, e0001305.
2 D. Schar, E. Y. Klein, R. Laxminarayan, M. Gilbert and T. P. Van Boeckel, *Sci. Rep.*, 2020, **10**, 21878.
3 E. Y. Klein, T. P. Van Boeckel, E. M. Martinez, S. Pant, S. Gandra, S. A. Levin, H. Goossens and R. Laxminarayan, *Proc. Natl. Acad. Sci. U. S. A.*, 2018, **115**, E3463–E3470.
4 M. Naghavi, S. E. Vollset, K. S. Ikuta, L. R. Swetschinski, A. P. Gray, E. E. Wool, G. R. Aguilar, T. Mestrovic, G. Smith, C. Han and R. L. Hsu, *Lancet*, 2024, **404**(10459), 1199–1226.
5 J. A. DiMasi, H. G. Grabowski and R. W. Hansen, *J. Health Econ.*, 2016, **47**, 20–33.
6 H. W. Boucher, G. H. Talbot, J. S. Bradley, J. E. Edwards, D. Gilbert, L. B. Rice, M. Scheld, B. Spellberg and J. Bartlett, *Clin. Infect. Dis.*, 2009, **48**, 1–12.
7 C. L. Ventola, *Pharm. Ther.*, 2015, **40**, 277–283.
8 WHO, *2023 Antibacterial agents in clinical and preclinical development: an overview and analysis*, World health organization technical report, Geneva, 2024.
9 R. Seyfi, F. A. Kahaki, T. Ebrahimi, S. Montazersaheb, S. Eyvazi, V. Babaeipour and V. Tarhriz, *Int. J. Pept. Res. Ther.*, 2020, **26**, 1451–1463.
10 Y. Huan, Q. Kong, H. Mou and H. Yi, *Front. Microbiol.*, 2020, **11**, 582779.
11 L. Daruka, M. S. Czikkely, P. Szili, Z. Farkas, D. Balogh, G. Grézal, E. Maharramov, T.-H. Vu, L. Sipos, S. Juhász, A. Dunai, A. Daraba, M. Számel, T. Sári, T. Stirling, B. M. Vásárhelyi, E. Ari, C. Christodoulou, M. Manczinger, M. Z. Enyedi, G. Jaksa, K. Kovács, S. van Houte, E. Pursey, L. Pintér, L. Haracska, B. Kintses, B. Papp and C. Pál, *Nat. Microbiol.*, 2025, **10**, 313–331.
12 G. G. Perron, M. Zasloff and G. Bell, *Proc. R. Soc. B*, 2005, **273**, 251–256.
13 A. Lewies, L. H. Du Plessis and J. F. Wentzel, *Probiotics Antimicrob. Proteins*, 2019, **11**, 370–381.
14 L. R. Pizzolato-Cezar, N. M. Okuda-Shinagawa and M. T. Machini, *Front. Microbiol.*, 2019, **10**, article 1703.
15 C. D. Fjell, R. E. Hancock and A. Cherkasov, *Bioinformatics*, 2007, **23**, 1148–1155.
16 C. D. Fjell, H. Jenssen, K. Hilpert, W. A. Cheung, N. Panté, R. E. Hancock and A. Cherkasov, *J. Med. Chem.*, 2009, **52**, 2006–2015.
17 E. G. Randou, D. Veltri and A. Shehu, *2013 IEEE 3rd International Conference on Computational Advances in Bio and medical Sciences (ICCABS)*, 2013, pp. 1–6.
18 E. Y. Lee, M. W. Lee, B. M. Fulan, A. L. Ferguson and G. C. Wong, *Interface Focus*, 2017, 7.
19 S. Spänig and D. Heider, *BioData Min.*, 2019, **12**, 7.
20 A. Capecchi, X. Cai, H. Personne, T. Köhler, C. v. Delden and J.-L. Reymond, *Chem. Sci.*, 2021, **12**, 9221–9232.
21 T. Li, X. Ren, X. Luo, Z. Wang, Z. Li, X. Luo, J. Shen, Y. Li, D. Yuan and R. Nussinov, *et al.*, *Nat. Commun.*, 2024, **15**, 7538.
22 E. Zakharova, M. Orsi, A. Capecchi and J.-L. Reymond, *ChemMedChem*, 2022, **17**, e202200291.
23 P. Szymczak, M. Możejko, T. Grzegorzek, R. Jurczak, M. Bauer, D. Neubauer, K. Sikora, M. Michalski, J. Sroka, P. Setny, W. Kamysz and E. Szczurek, *Nat. Commun.*, 2023, **14**, 1453.
24 P. Das, T. Sercu, K. Wadhawan, I. Padhi, S. Gehrmann, F. Cipcigan, V. Chenthamarakshan, H. Strobelt, C. dos Santos, P.-Y. Chen, Y. Y. Yang, J. P. K. Tan, J. Hedrick, J. Crain and A. Mojsilovic, *Nat. Biomed. Eng.*, 2021, **5**, 613–623.
25 S. Renaud and R. A. Mansbach, *Digital Discovery*, 2023, **2**(2), 441–458.
26 R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams and A. Aspuru-Guzik, *ACS Cent. Sci.*, 2018, **4**, 268–276.
27 A. Grosnit, R. Tutunov, A. M. Maraval, R.-R. Griffiths, A. I. Cowen-Rivers, L. Yang, L. Zhu, W. Lyu, Z. Chen, J. Wang, J. Peters and H. Bou-Ammar, High-Dimensional Bayesian Optimisation with Variational Autoencoders and Deep Metric Learning, 2021, preprint, arXiv:2106.03609 [cs], DOI: 10.48550/arXiv.2106.03609, http://arxiv.org/abs/2106.03609.
28 S. Lee, J. Chu, S. Kim, J. Ko and H. J. Kim, *Adv. Neural Inf. Process. Syst.*, 2023, **36**, 48906–48917.
29 A. Tripp, E. Daxberger and J. M. Hernández-Lobato, *Adv. Neural Inf. Process. Syst.*, 2020, **33**, 11259–11272.
30 K. Shmilovich, R. A. Mansbach, H. Sidky, O. E. Dunne, S. S. Panda, J. D. Tovar and A. L. Ferguson, *J. Phys. Chem. B*, 2020, **124**, 3873–3891.
31 M. Pirtskhalava, A. A. Amstrong, M. Grigolava, M. Chubinidze, E. Alimbarashvili, B. Vishnepolsky, A. Gabrielian, A. Rosenthal, D. E. Hurt and M. Tartakovsky, *Nucleic Acids Res.*, 2021, **49**, D288–D297.
32 M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson and E. Bakshy, *Advances in Neural Information Processing Systems 33*, 2020.
33 B. Shahriari, K. Swersky, Z. Wang, R. P. Adams and N. de Freitas, *Proc. IEEE*, 2016, **104**, 148–175.
34 S. Ament, S. Daulton, D. Eriksson, M. Balandat and E. Bakshy, *Adv. Neural Inf. Process. Syst.*, 2023, **36**, 20577–20612.
35 N. Kadeřábková, A. J. S. Mahmood and D. A. I. Mavridou, *npj Antimicrob. Resist.*, 2024, **2**, 1–9.

36 J. Witten and Z. Witten, Deep learning regression model for antimicrobial peptide design, *BioRxiv*, 2019, preprint, DOI: 10.1101/692681.
37 D. P. Kingma and M. Welling, *Found. Trends Mach. Learn.*, 2019, **12**(4), 307–392.
38 A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, *Dive into Deep Learning*, Cambridge University Press, 2023.
39 O. Dollar, N. Joshi, D. A. C. Beck and J. Pfaendtner, *Chem. Sci.*, 2021, **12**, 8362–8372.
40 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, *Adv. Neural Inf. Process. Syst.*, 2017, **30**, 6000–6010.
41 H. G. Boman, *J. Intern. Med.*, 2003, **254**, 197–215.
42 J. Kyte and R. F. Doolittle, *J. Mol. Biol.*, 1982, **157**, 105–132.
43 M. Larralde, D. Osorio, P. Rondón-Villarreal and R. Torres, peptides, https://pypi.org/project/peptides/.
44 G. Shi, X. Kang, F. Dong, Y. Liu, N. Zhu, Y. Hu, H. Xu, X. Lao and H. Zheng, *Nucleic Acids Res.*, 2022, **50**, D488–D496.
45 G. Wang, X. Li and Z. Wang, *Nucleic Acids Res.*, 2016, **44**, D1087–D1093.
46 J.-H. Jhong, L. Yao, Y. Pang, Z. Li, C.-R. Chung, R. Wang, S. Li, W. Li, M. Luo, R. Ma, Y. Huang, X. Zhu, J. Zhang, H. Feng, Q. Cheng, C. Wang, K. Xi, L.-C. Wu, T.-H. Chang, J.-T. Horng, L. Zhu, Y.-C. Chiang, Z. Wang and T.-Y. Lee, *Nucleic Acids Res.*, 2022, **50**, D460–D470.
47 L. Aguilera-Mendoza, Y. Marrero-Ponce, J. A. Beltran, R. Tellez Ibarra, H. A. Guillen-Ramirez and C. A. Brizuela, *Bioinformatics*, 2019, **35**, 4739–4747.
48 The UniProt Consortium, *Nucleic Acids Res.*, 2023, **51**, D523–D531.
49 K. Sidorczuk, P. Gagat, F. Pietluch, J. Kała, D. Rafacz, L. Bąkała, J. Słowik, R. Kolenda, S. Rödiger, L. C. H. W. Fingerhut, I. R. Cooke, P. Mackiewicz and M. Burdukiewicz, *Briefings Bioinf.*, 2022, **23**, bbac343.
50 W. Li and A. Godzik, *Bioinformatics*, 2006, **22**, 1658–1659.
51 L. Fu, B. Niu, Z. Zhu, S. Wu and W. Li, *Bioinformatics*, 2012, **28**, 3150–3152.
52 I. T. Jolliffe and J. Cadima, *Philos. Trans. R. Soc., A*, 2016, **374**, 20150202.
53 J. Venna and S. Kaski, *Neural Netw.*, 2006, **19**, 889–899.
54 H. Jeon, H.-K. Ko, J. Jo, Y. Kim and J. Seo, *IEEE Trans. Vis. Comput. Graph.*, 2022, **28**, 551–561.
55 A. G. Wilson, Z. Hu, R. Salakhutdinov and E. P. Xing, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 2016, pp. 370–378.
56 F. Biscani and D. Izzo, *J. Open Source Softw.*, 2020, **5**, 2338.
57 Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, N. Smetanin, R. Verkuil, O. Kabeli, Y. Shmueli, A. dos Santos Costa, M. Fazel-Zarandi, T. Sercu, S. Candido and A. Rives, *Science*, 2023, **379**, 1123–1130.
58 E. C. Meng, T. D. Goddard, E. F. Pettersen, G. S. Couch, Z. J. Pearson, J. H. Morris and T. E. Ferrin, *Protein Sci.*, 2023, **32**, e4792.