Journal Name

ARTICLE TYPE

Cite this: DOI: 00.0000/xxxxxxxxxx

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

Jyler Menard,[a] and RA Mansbach*[a]

Received Date
Accepted Date

DOI: 00.0000/xxxxxxxxxx

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 in 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 a theoretical 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.

[a] *Department of Physics, Concordia University, Montréal, QC, H4B 1R6, Canada. E-mail: re.mansbach@concordia.ca*

† Electronic Supplementary Information (ESI) available: Further detailed analyses, tables and figures available in PDF form (4 sections, 10 tables, and 24 figures). Code for model training and analysis can be found at `https://github.com/Mansbach-Lab/compare-latent-spaces-amps/tree/main`. Trained model checkpoints and datasets are available at `https://doi.org/10.5281/zenodo.17872434`.

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

S1 Support Vector Regression Oracle

To serve as an oracle to rapidly evaluate our "true" objective function, we trained a Support Vector Regression (SVR) model on data compiled by Witten & Witten (2019)[1] by using the scikit-learn SVR implementation[2]. This dataset contains peptide sequences and their associated Minimum Inhibitory Concentration (MIC), which we split into a training (4145 sequences) and a validation (2615 sequences) set. This dataset primarily contains sequences of lengths < 50 amino acids.

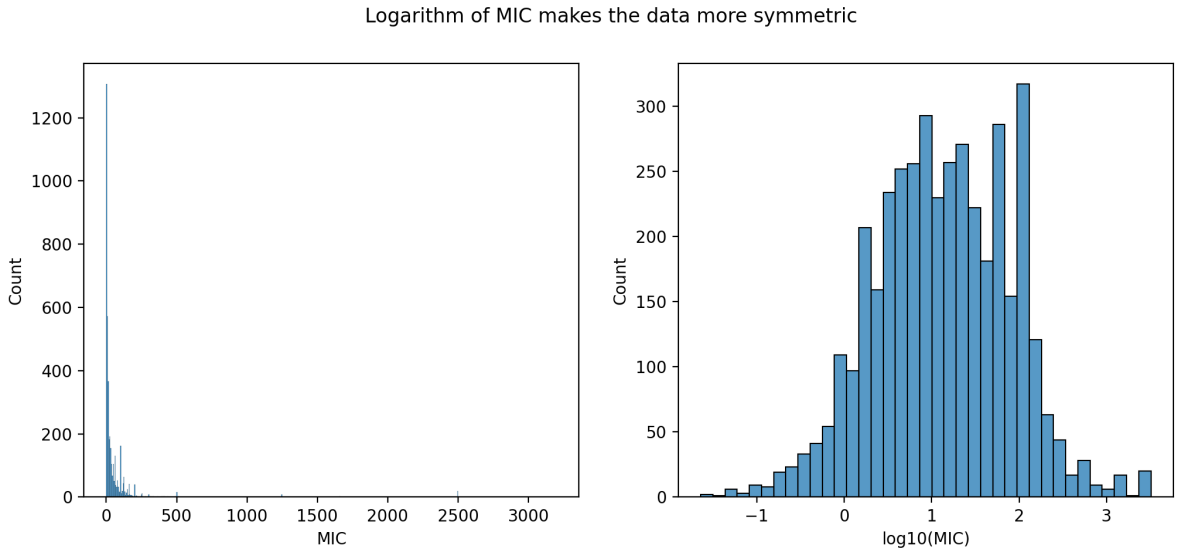

Fig. S1 Distribution of MIC values in the oracle training dataset. Before applying a base-10 logarithm transformation the data is heavily skewed (left). After applying a base-10 logarithm transformation the data becomes more normally distributed (right).

For the outputs of the oracle, we employed the base-10 logarithm of the MIC values $\log_{10}(\text{MIC})$ to ensure the values are distributed more symmetrically (Fig. S1). For the inputs to the oracle, we used easily-computed physico-chemical descriptors. We used a previously-validated feature selection procedure[3] to identify the most relevant subset from a larger set of possible descriptors.

Computing all descriptors available in the ProPy3 Python package[4] resulted in 1529 features per sequence (Table S1). We reduce the number of features necessary for the model via a two-step feature selection procedure. We perform an initial feature screening stage using the minimum Redundancy Maximum Relevance (mRMR) algorithm[5], in which one attempts to identify those features that have the least overlap that are the most relevant to the prediction of the output of interest. We defined *redundancy* as the mutual information between pairs of physico-chemical features and *relevancy* as the mutual information between features and predicted outputs (i.e. $\log_{10}(\text{MIC})$).

After mRMR, we use LASSO[6] regression to select a sparse subset of high-performing features, in a procedure analogous

Abbreviation	Name	Description
AAC	Amino Acid Composition	The fraction of a sequence composed of a given amino acid.
DPC	Dipeptide Composition	The fraction of a sequence composed of a given pair of amino acids.
MBauto	Normalized Moreau-Broto Autocorrelation	Normalized autocorrelation of properties, where property similarities are computed.
Moranauto	Moran autocorrelation	Autocorrelation of properties, where deviations from the average are computed.
Gearyauto	Geary Autocorrelation	The squared-difference of a property at different sequence locations is computed, and normalized by the variance.
CTD	Composition, Transition, Distribution	Amino acids are grouped based on 7 different properties. Then for each property the composition of the sequence, the number of transition between groups, and the distribution of groups is computed.
SOCN	Sequence Order Coupling Numbers	Quantifies the extent to which amino acids are 'coupled' in a sequence based on amino acid physicochemical distances.
QSO	Quasi Sequence Order	Composition and amino acid correlation, computed using amino acid physicochemical "distance" matrices.
PAAC	Pseudo Amino Acid Composition	Computes the amino acid composition of a sequence while maintaining a notion of how correlated amino acids' hydrophilicity, hydrophobicity, and mass are throughout the sequence.

Table S1 Physicochemical descriptors computed using ProPy3 python package[4].

to that outlined in Bi *et al.* (2003)[7¶], and applied in Lee *et al.* (2016)[3]. We first augment our relevant, non-redundant training data features with 10 'dummy features', each sampled from a Normal distribution of mean zero and variance one. Then, we perform $T = 10$ random 80-20 splits of the entire training data into training and validation subsets for an initial hyperparameter search. For each split, we fit a LASSO regression model, performing a grid-search over the regularization hyperparameter α over values $[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.5, 1.0, 1.5, 2.5, 5.0, 7.5, 10, 50, 10^{2}]$ for the α value with lowest mean squared error (MSE) on the validation set. We keep the coefficient vector of the best model of each split. We average the magnitude of the coefficient of every dummy feature to determine a threshold coefficient magnitude. Any coefficient with magnitude less than or equal to the dummy feature threshold contributes to the prediction less than a random variable does, so its coefficient is set to zero (*i.e.* that feature is removed). SI Table S3 provides a summary of the number of features after each stage.

Next, we determined hyperparameters for a nonlinear SVR with Radial Basis Function kernel. Using 5-fold cross-validation, we performed a grid search for hyperparameters $C = 10^{\beta}$ and $\varepsilon = 10^{\beta}$ over β values running from -2 to 2 in 45 equidistant steps. Once hyperparameters were selected, we fit a final nonlinear SVR on the full training set, yielding a mean-squared error of 0.328 on the test set while a simpler linear regressor on the same features yielded a less-optimal, higher error of 0.407 and a predictor using only the mean of the training data set yielded a mean-squared error of 0.602.

¶ Bi *et al.* (2003) use L1-regularized Linear SVCs to perform feature selection.

Rank	Name	LASSO Weight	Brief Description
1	tausw2	-0.19	Quasi Sequence Order coupling number
2	PolarizabilityC3	-0.14	Polarizability Composition
3	ChargeT12	-0.13	Number of charge transition between charged and neutral AAs
4	NormalizedVDWVC2	-0.09	van der waals volume
5	PAAC20	0.08	Type I Pseudo amino acid composition descriptors; it uses hydrophobicity, hydrophilicity, residue mass.
6	QSOSW39	-0.07	Quasi Sequence Order using Schneider-Wrede distance matrix
7	PAAC21	0.06	Type I Pseudo amino acid composition descriptors
8	MoreauBrotoAuto_Steric2	0.06	Moreau-Broto Autocorrelation
9	ChargeD2100	0.05	Charge distribution group 2 100%
10	MoranAuto_Hydrophobicity5	0.05	Moran Autocorrelation

Table S2 The top-10 features selected after mRMR and LASSO. Note that this is a subset of the 149 features we kept after our feature selection procedure.

Stage	No. of Features
Initial	1529
Post-mRMR	248
Post-LASSO	149

Table S3 Number of features at each stage of feature selection when building the SVR oracle.

This final SVR was used to predict $\log_{10}(\text{MIC})$ to obtain our 'true' MIC value; throughout we use this as our 'oracle'.

Hyperparameter	Value	Description
d_{latent}	64	Dimensionality of latent space
d_{model}	512	Dimensionality of model
N_{block}	3	Number of Encoder/Decoder blocks
d_{ff}	256	Dimensionality of feedforward layers
h	4	Number of attention heads per multi-head attention layer
d_{pp}	64	Dimensionality of property predictor layers
depth_{pp}	2	Number of property predictor layers
$d_{\text{pp,out}}$	1,2,3	Output dimensionality of property predictor
$\beta_{\text{pp},0}$	0	Initial value for property predictor annealing coefficient
β_{pp}	1	Final value for property predictor annealing coefficient
$\beta_{\text{KL},0}$	0	Initial value for KL annealing coefficient
β_{KL}	0.05	Final value for KL annealing coefficient
N_{batch}	256	Batch size during training
β_1	0.9	Adam optimizer's β_1
β_2	0.98	Adam optimizer's β_2
lr	$(d_{model})^{-0.5} \cdot \min\left(\text{step}^{-0.5}, \text{step} \cdot (n_{\text{warmup}})^{-1.5}\right)$	Learning rate set adaptively using the Noam optimizer
n_{warmup}	10000	Number of warmup steps for the Noam optimizer
p_{dropout}	0.1	Dropout rate

Table S4 Hyper-parameters for the model architecture and training. The TransVAE consists of an encoder and decoder module, each consisting of N_{block} blocks of multi-head self-attention feeding into feedforward layers.

S2 Additional TransVAE information and analysis

S2.1 Additional model details

We trained variational autoencoders (VAEs) with transformer architectures based on Renaud and A. Mansbach[8]. Model architecture was chosen based on work done in Renaud and A. Mansbach[8], with specific hyper-parameter values and descriptions listed in Table S4. To optimize the weights of the neural networks, we used the Noam optimizer[9], a variant of the Adam optimizer[10] with a varying learning rate. The output dimensions of the property predictor module varied over none, 1, 2, 3, depending on the number of organizing properties. All models were trained to 100 epochs, with the total training loss < 0.9 and validation losses < 0.95, with most models have a total validation loss < 0.8 (Fig. S3).

S2.2 Additional PCA projection analysis

Principal Component Analysis (PCA) determines a set of ordered basis vectors for a set of data points, where each basis vector in the ordering captures a decreasing amount of variance in the original cloud of data points. In the main text, we frequently use only the top five principal components as a lower dimensional representation of a full latent space. Here we provide the full distribution of principal components (Fig. S4). We observe that the top five principal components capture between 17% and 22% of the variance in the VAE training set.

Along with explained variance, we calculate the correlation between each of the top five principal components, and the organizing property (or each individual property in the case of multi-property organization), allowing us to best visualize the property-induced organization of the latent space. In Fig. 2, we visualized the extent to which the latent space is organized by Boman index, charge, and hydrophobicity, when the underlying TransVAE model was being organized by all three simultaneously. Table S5 provides the PC pair used for visualization. Charge is frequently most organized along (most correlated with) PCs separate from Boman and Hydrophobicity. Boman and Hydrophobicity frequently have

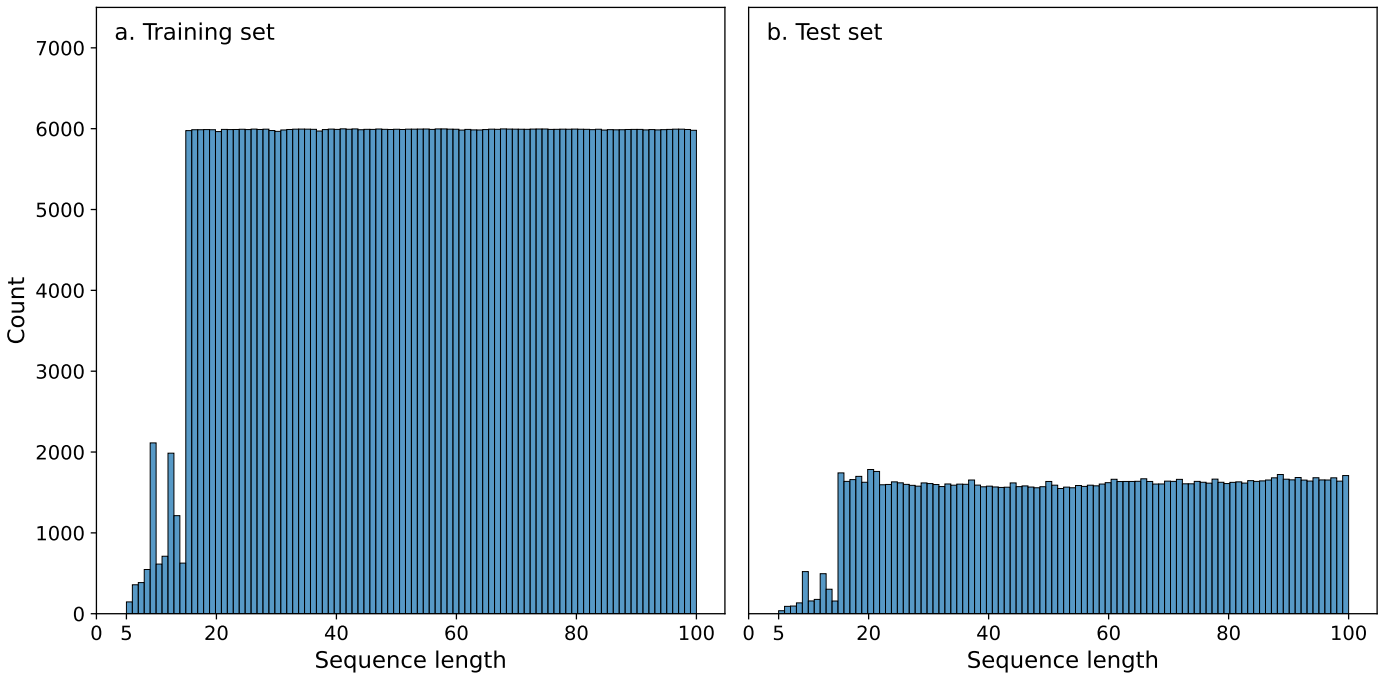

Fig. S2 Distribution of peptide sequence lengths in our dataset.

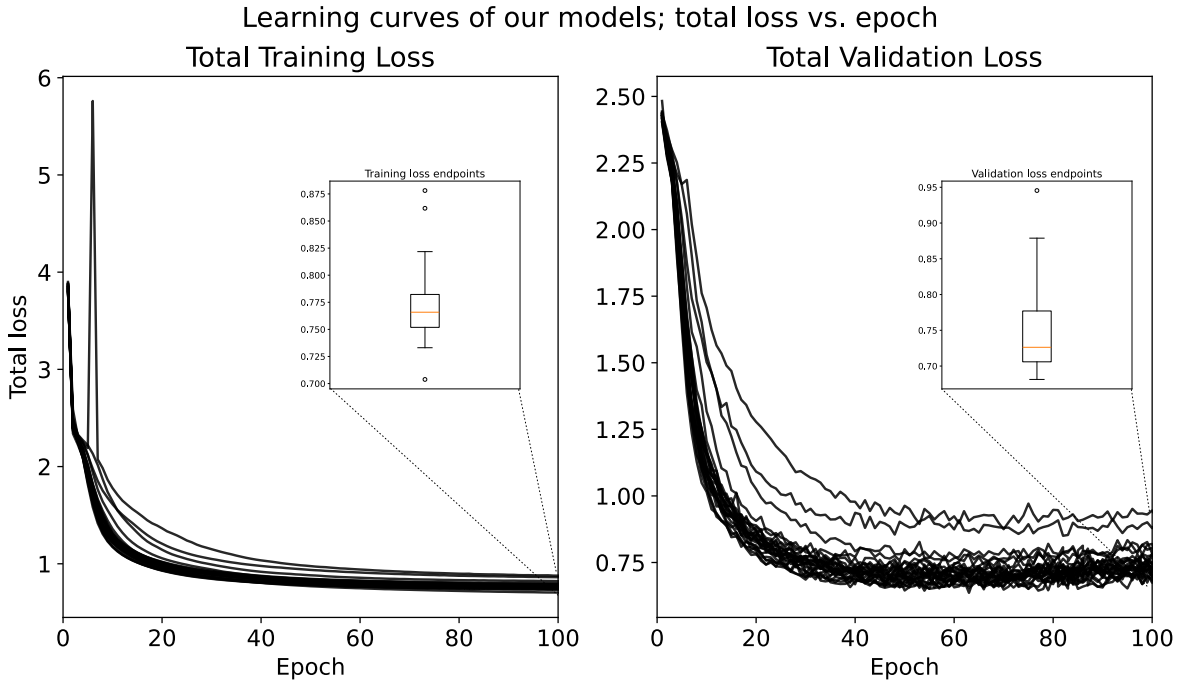

Fig. S3 Learning curves for all models trained. Inset boxplots of total loss values at epoch 100. Total loss is the sum of reconstruction loss, KL Divergence, and property prediction loss. At epoch 100, all models have total training loss < 0.9 and total validation loss < 0.95. The majority of models have even lower loss values that are all relatively near each other. The validation loss is frequently lower than the training loss due to dropout layers being turned off during inference on the validation set.

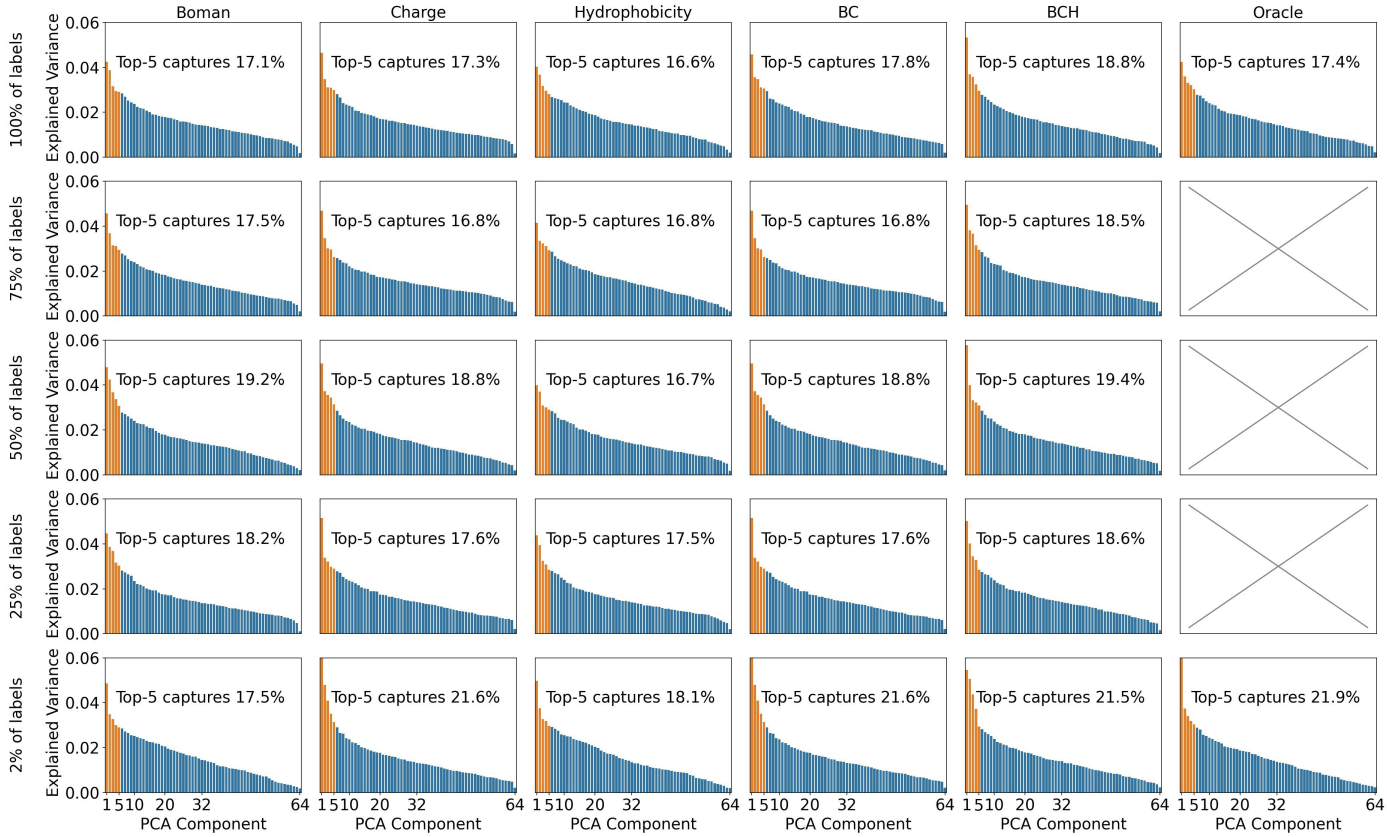

Fig. S4 Principal components explained variance ratios for each TransVAE model. The cumulative explained variance, as a percentage, of the top 5 PCs is reported in its corresponding panel. Top 5 PCs are coloured orange.

Label Percentage	Boman	Charge	Hydrophobicity
100%	(2,3)	(5,2)	(2,3)
75%	(2,3)	(5,3)	(2,4)
50%	(2,3)	(4,5)	(2,1)
25%	(2,1)	(3,4)	(2,1)
2%	(1,3)	(3,1)	(1,3)

Table S5 Principal component pairs used for visualization in Fig. 2. The location of each pair corresponds to their subpanel in Fig. 2

overlapping, but not always the same, PC pair; while Boman and hydrophobicity have similar trends, they are not quite the same, this can suggest that the 2D projection, and the linear correlation used to select PCs, is hiding differences in organization.

S2.3 Relation between organizing properties and predictive oracle

We investigated in some detail the correlations between the principal components and the "bridge variables," that is different physicochemical properties, including organizing variables. We observed that sequence length was most correlated with the first principal component across 21/27 of the models (SI Fig S5g), with the second or third principal component being correlated with the property used during training to organize the latent space (SI Fig. S5h for Boman index; SI Fig. S5i for charge; SI Fig. S5j for hydrophobicity; SI Fig. S5k for oracle). Although we may desire the first principal component to be most correlated with the organizing property in most cases, we note that the top five principal components have similar amounts of explained variance, and that they capture on the order of 20% of the total variance (SI Fig. S4). Additionally, previous work found sequence length can be an easy-to-learn pattern[8]. Patterns that are straightforward to recognize may become unexpectedly strongly correlated with a PC direction.

To check whether searches through spaces organized by different properties performed differently due to certain properties providing stronger signal about the objective value (the output of the SVR), we computed two different metrics. First, we computed the correlation between organizing properties (Boman Index, Charge, and Hydrophobicity) and the top-10 features of the SVR. Because correlation is biased towards monotonic trends, we also estimate the mutual information between the organizing properties and the features of the SVR.

Input Feature	Boman Index	Charge	Hydrophobicity
tausw2	0.27 (0.0075)	0.022 (0.011)	-0.27 (0.0058)
PolarizabilityC3	0.34 (0.0099)	0.50 (0.0038)	-0.34 (0.0082)
ChargeT12	0.51 (0.0062)	0.60 (0.0038)	-0.45 (0.0091)
NormalizedVDWVC2	-0.28 (0.0020)	-0.29 (0.0012)	0.33 (0.014)
PAAC20	-0.34 (0.0069)	-0.091 (0.0041)	0.43 (0.013)
QSOSW39	0.037 (0.0084)	-0.0041 (0.0063)	-0.027 (0.019)
PAAC21	0.44 (0.009)	0.21 (0.0036)	-0.36 (0.0098)
MoreauBrotoAuto_Steric2	-0.0021 (0.0043)	0.014 (0.0069)	-0.025 (0.013)
ChargeD2100	-0.20 (0.017)	-0.038 (0.011)	0.17 (0.012)
MoranAuto_Hydrophobicity5	-0.047 (0.015)	0.015 (0.012)	0.045 (0.0074)
Num. with magnitude < 0.2	3	6	3

Table S6 Average (standard deviation) Pearson correlation coefficient between the organizing property (column) and each of the top-10 input features (row) to the SVR. Properties and features are computed on the training set. For each property, the Pearson correlation was computed across five independent subsamples of 10,000 points; the reported values are the mean and standard deviation across these subsamples. Threshold of 0.2 corresponds to an correlation magnitude of very weak.

Compared to Boman index and hydrophobicity, charge had very weak correlations with the most features (6/10, Table S6), but strong mutual information scores. Compared to hydrophobicity, Boman index had similar correlation values (frequently with switched signs), but weaker mutual information scores. Hydrophobicity tended to be most informative about the top-10 features inputted into the SVR. While with the outputted predicted log10(MIC) values, charge is both more strongly correlated and has a higher mutual information score than Boman Index — the caveat being that charge has a weak correlation, and a somewhat low mutual information score (Table S8).

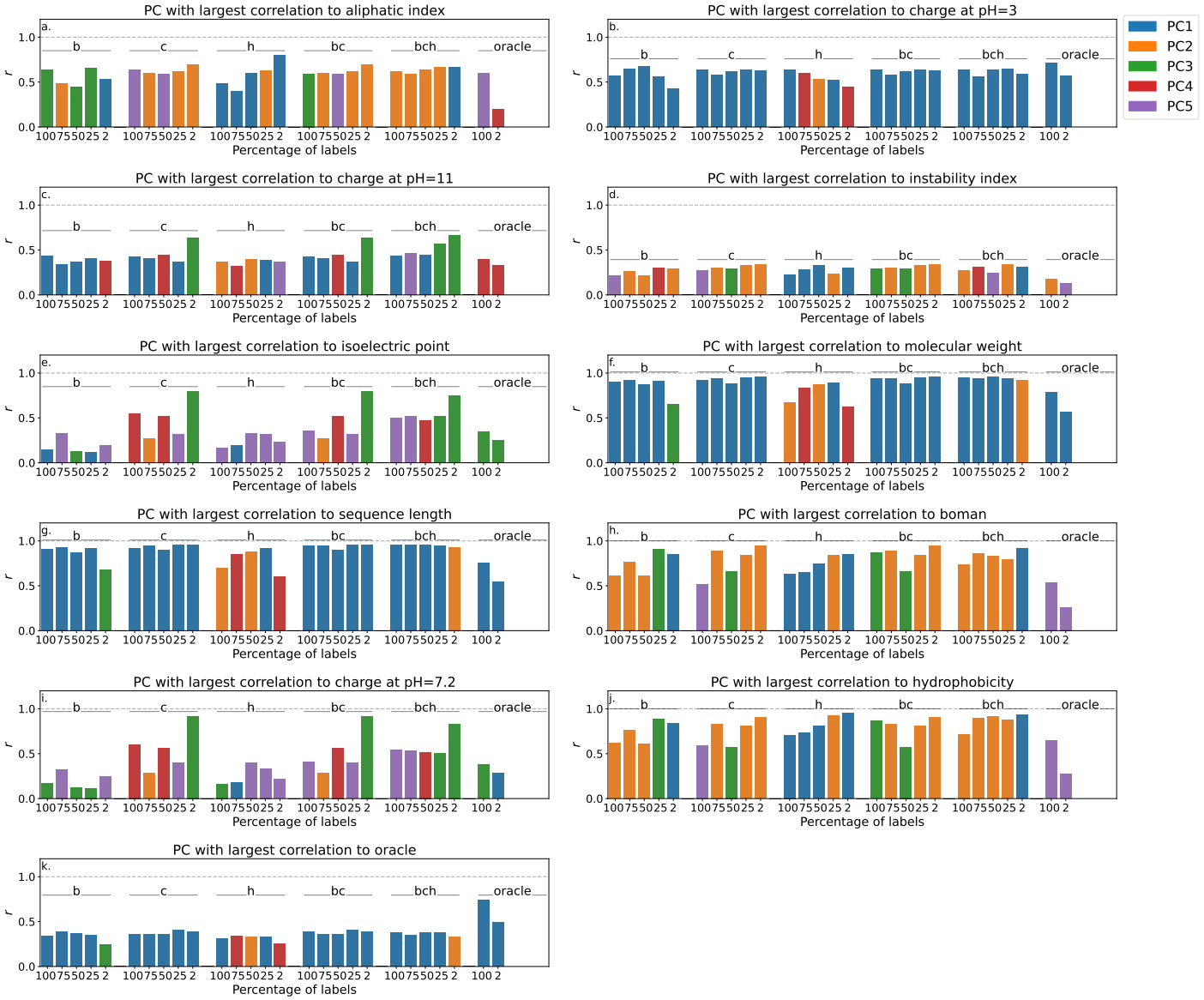

Fig. S5 Top PCs and their correlation scores with physico-chemical properties. (a-k) Pearson correlation coefficients (r) between PCs and (a) aliphatic index, (b) charge at pH$=3$, (c) charge at pH$=11$, (d) instability index, (e) isoelectric point, (f) molecular weight, (g) sequence length, (h) Boman index, (i) charge at pH$=7.2$, (j) hydrophobicity, (k) predicted $\log_{10}$(MIC) values from the SVR oracle. Bars grouped by model, and shown in order of decreasing percentage of property labels available during training. Note: Boman index and hydrophobicity are length-normalized quantities, while charge is not.

Input Feature	Boman Index	Charge	Hydrophobicity
tausw2	0.13	0.22	0.12
PolarizabilityC3	0.11	0.37	0.44
ChargeT12	0.22	0.59	0.35
NormalizedVDWVC2	0.083	0.19	0.37
PAAC20	0.088	0.065	0.15
QSOSW39	0.043	0.15	0.11
PAAC21	0.15	0.076	0.11
MoreauBrotoAuto_Steric2	0.029	0.025	0.025
ChargeD2100	0.033	0.087	0.13
MoranAuto_Hydrophobicity5	0.018	0.028	0.014
Num. with highest value	2	4	4
Average of MI scores	0.26	0.53	0.53

Table S7 Mutual information between organizing properties and the top-10 features of the SVR.

	Boman Index	Charge	Hydrophobicity
PCC	-0.15	-0.30	-0.14
Mutual Information	0.027	0.096	0.021

Table S8 Pearson correlation coefficient and Mutual Information between organizing properties and predicted log10(MIC) over the training set. The Pearson correlation was computed across five independent subsamples of 10,000 points; the reported values are the mean and standard deviation across these subsamples. Charge is calculated at pH$=7.2$.

S2.4 Energy consumption

We estimate the total energy consumption of training the deep learning models to 100 epochs. The system we performed training on was a NVIDIA V100 Volta (32G HBM2 memory) through the Digital Research Alliance of Canada's Cedar cluster. From comet-ml, we observed an average power usage of $\approx 275W$ throughout a given training job[11]. To train a given model to 100 epochs, approximately 24hrs were required, yielding $275 \cdot 24/1000 = 6.6$ kWh per 100 epochs. For the main text, we trained 27 models to 100 epochs, giving an approximate total energy usage of 178.2kWh.

The Hyundai Ioniq 6 is a 2022 battery electric sedan. Its long-range battery capacity is 77.4kWh, corresponding to an estimated range of 614km[12], giving ≈ 0.126kWh/km. Comparing to our total energy usage, we obtain an approximate car range using energy from 100 epochs as ≈ 52.36km. Therefore, the total approximate range of a Hyundai Ioniq 6 using the energy from training our main text models is 1413.63km, or we could fully recharge 2.3 Hyundai Ioniq 6 sedans.

System	NVIDIA V100 Volta
Average GPU energy utilization (Watts)	275
Approx. hours for 100 epochs	24
kWh per 100 epochs	6.6
Hyundai Ioniq 6 battery capacity	77.4 kWh
Hyundai Ioniq 6 estimated range	614 km
Energy per km	0.1261 kWh/km
Approx. car range from 100 epochs	52.36 km
Number of 100 epoch runs	27
Approximate total energy used	178.2 kWh
Approximate equivalent car range	1413.63 km

Table S9 Energy usage of deep learning training on Cedar and its equivalent in electric car distance (Hyundai Ioniq 6).

S3 Additional BayesOpt Results

S3.1 At low numbers, the number of initial data points has little effect.

In the case of antimicrobial peptides, and peptides with hemolytic effect, the number associated with high-quality verification of the mechanism of action — through experimental or simulation methods — is rather low. To establish whether this will limit optimization in our models, we test the effect of varying the number of points with which we initialize the BayesOpt Gaussian Process Regressor. We ensure that each smaller initialization set is contained in the previous, larger initialization set, and that the peptide with strongest predicted activity is maintained throughout all of the sets to ensure that different initializations begin at the same objective value. That is, the best peptide from the 100 data point initialization set is held throughout each subset of it, and we randomly select the remaining points; *e.g.* to construct the 10 data point subset, we begin with the best peptide from the 100, then without replacement randomly select the other 9 peptides from the remaining 99.

When varying the number of initial data points in this way, we observe very little change in the optimization results. In the predicted-$\log_{10}$(MIC)–organized VAE with access to 100% of the property labels during training, varying the number of initial data points does not significantly impact the optimization performance, regardless of whether optimization is done in the VAE's latent space or in a PCA'ed projection of it (Fig S6ac). While it appears that one initial data point results in a slight slowdown (Figure S6a) within the first 100 iterations, by the end of 500 iterations it reaches the same objective value as the other optimization runs. In the case of having access to only 2% of the property labels during VAE training, having 100 initial data points can provide a boost if we optimize in the 64-dimensional latent space (Figure S6d). Ultimately, we observe little change to optimization performance when varying the number of initial data points below 100, implying that in the case when limited data is extant and generating initial data is expensive, few data points may be used without significant loss of optimization performance.

We further perform BayesOpt in the BCH-organized latent spaces, both in the full 64-dimensional space and in the PCA reduced space, with the same sweep in number of initial data points. When optimizing in this differently-organized VAE, we observe little effect on the optimization performance due to the different number of initial points (SI Fig. S9).

S3.2 Vary percentage of labels for given organizing properties

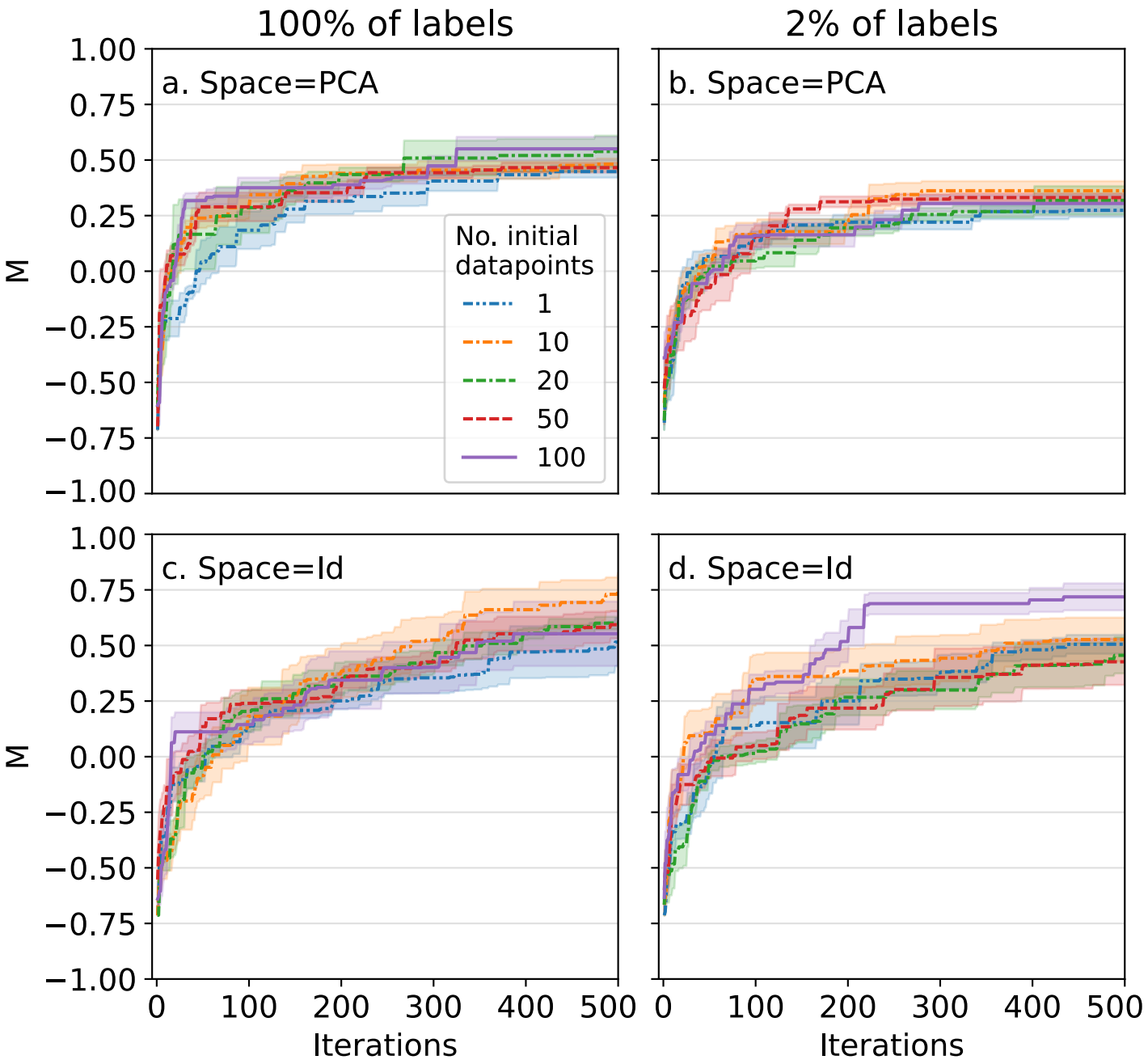

Fig. S6 Bayesian optimization performance as the number of initialization points is varied. Trajectories correspond to the best value encountered up to a given iteration. Error bars correspond to the standard error over five different random initializations. We tested different numbers of initial datapoints in the initial dataset: 1 (blue, dash-dot-dotted), 10 (orange, dash-dotted), 20 (green, dash-dash-dotted), 50 (red, dashed), and 100 (purple, solid). Each initialization is run in a space trained with 100% access to property labels (a,c) and 2% (b,d), and searching through the full 64-dimensional latent space (c,d) and 5-dimensional PCA reduced space (a,b). The organizing property is predicted-log10mic, which is also the (negative) of the objective.

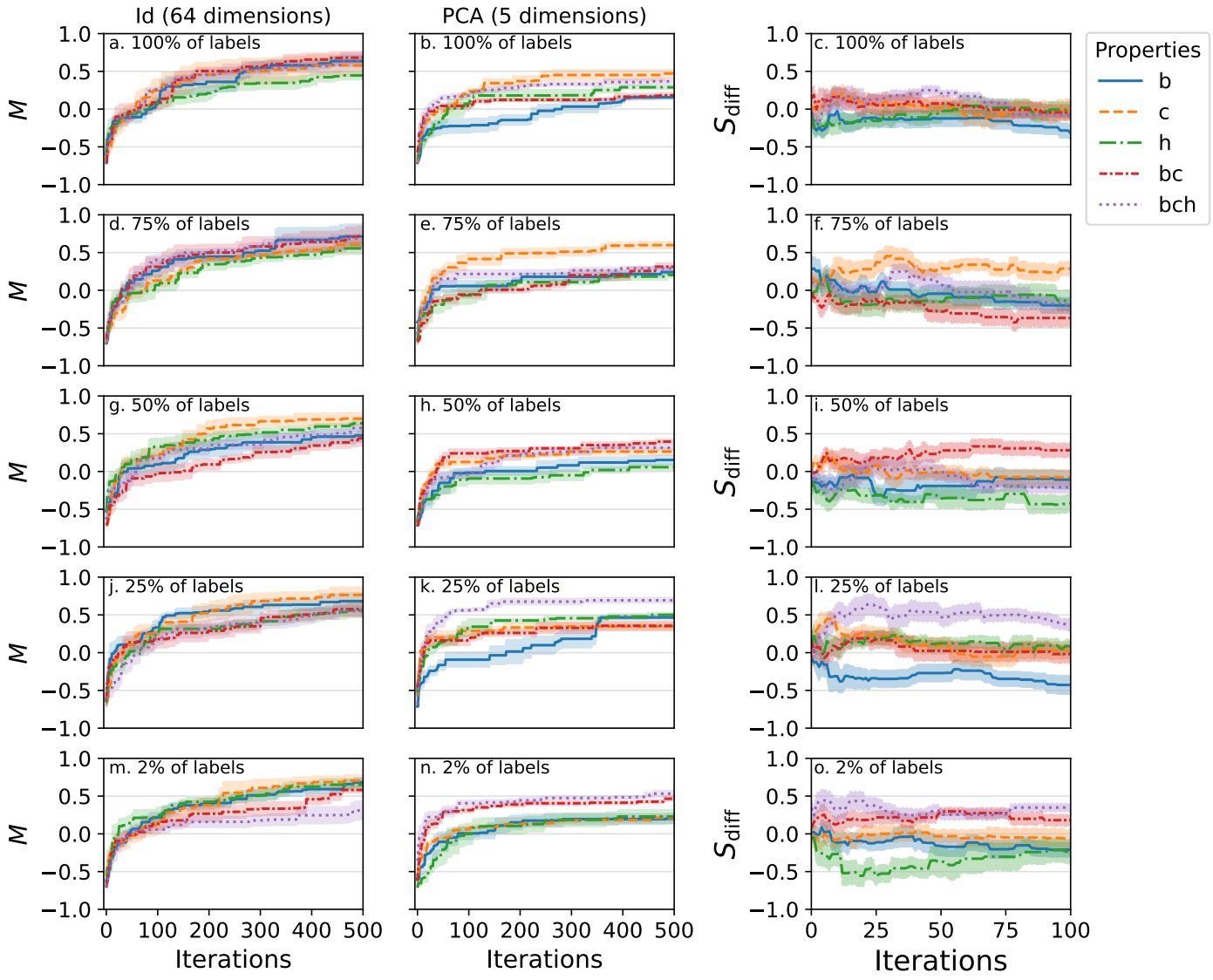

Fig. S7 Bayesian optimization performance in different percentage of labels available while varying the VAE-organizing properties. Optimization was initialized with 100 data points. Error bars correspond to standard error over 5 different initializations. (third column, cfili) Difference in Bayesian optimization results S_{diff}, the difference in best score at each iteration. Error bars correspond to the standard error in the difference of scores, with the variance computed through $\sigma^2(S_{\text{diff}}) = \sigma^2(M_{\text{Id}}) + \sigma^2(M_{\text{PCA}}) - 2\sigma(M_{\text{Id}}, M_{\text{PCA}})$. Note that the the x-axis range is only up to iteration 100 to more clearly see early optimization behaviour. We observe that searching through the PCA'ed latent space offers an advantage in earlier iterations compared to searching through the 64-dimensional latent space.

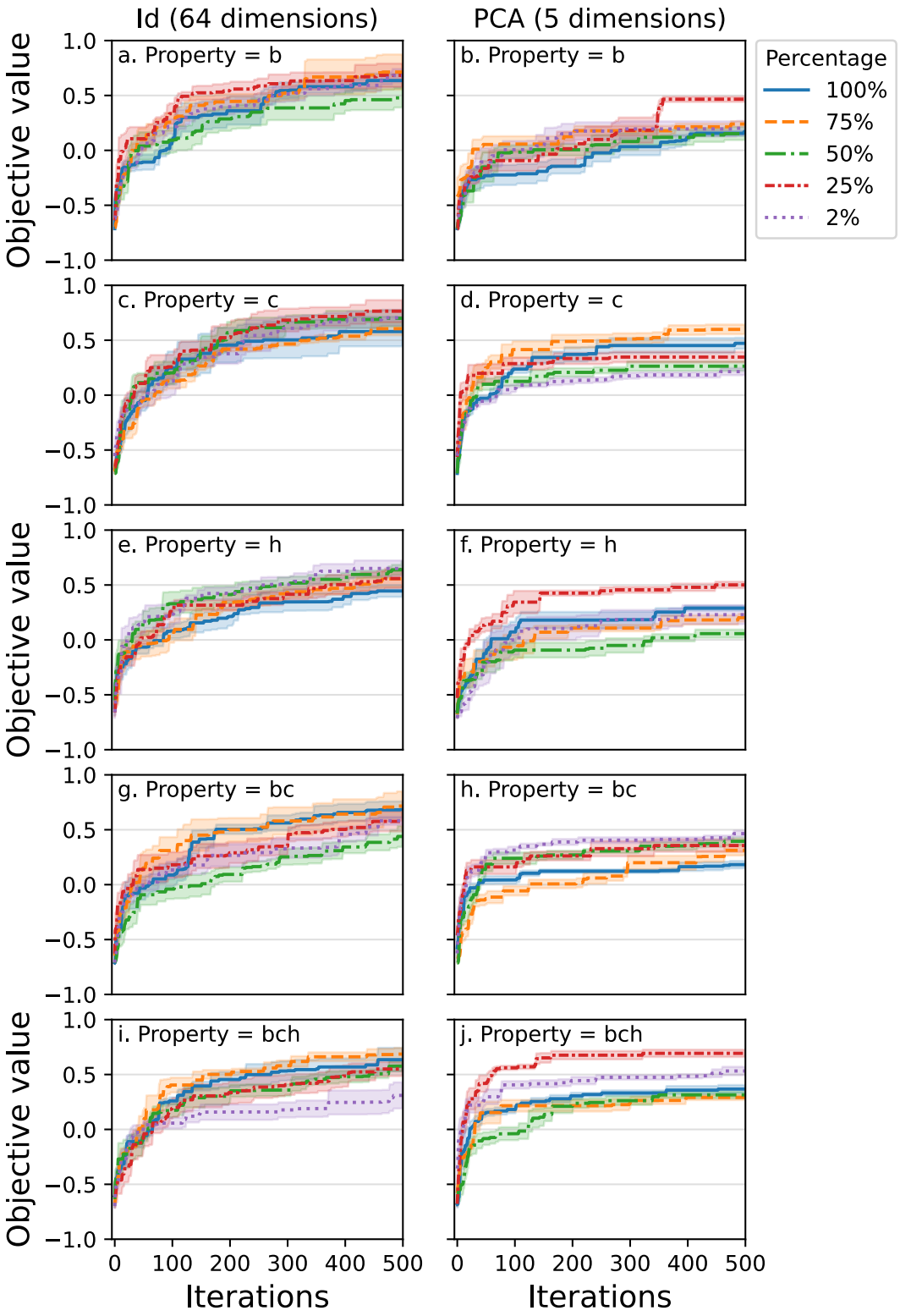

Fig. S8 Bayesian optimization performance in differently-organized spaces with varying percentage of property labels.

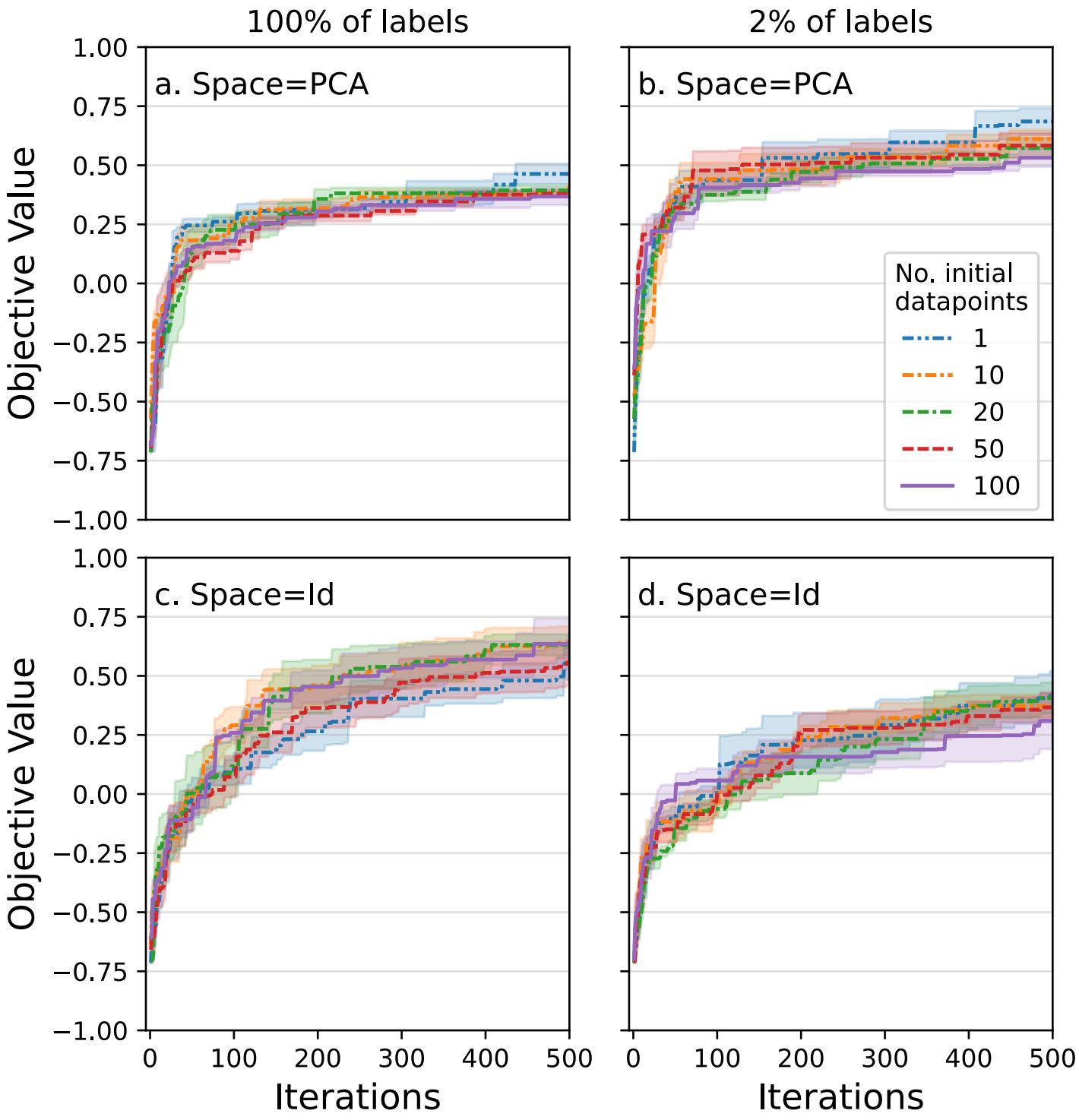

Fig. S9 BayesOpt optimization performance over BCH-organized latent space while varying the number of initial points used to initialize the GPR.

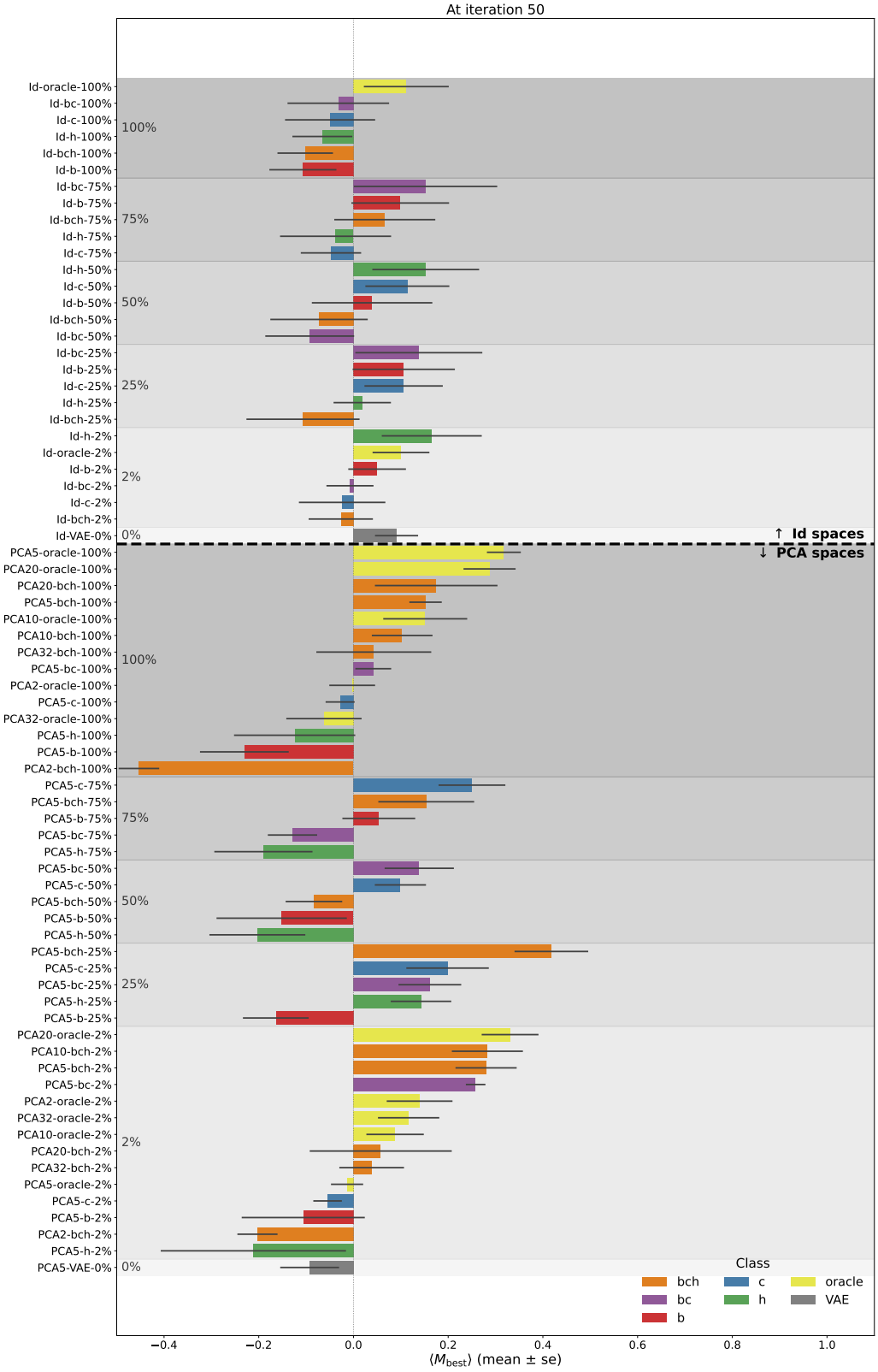

Fig. S10 Average best objective scores found at iteration 50. Error bars correspond to the standard error of M_{best} over five BayesOpt runs. Black dashed line separates those runs performed in the high-dimensional latent space (above the dashed line), and those runs performed in a PCA projection of the latent space (below the dashed line). Shaded regions denote the label percentage available at training time, with darker regions corresponding to a higher label percentage. Values are sorted within groups from largest to smallest. The full BayesOpt trajectories are depicted in SI Fig. S7.

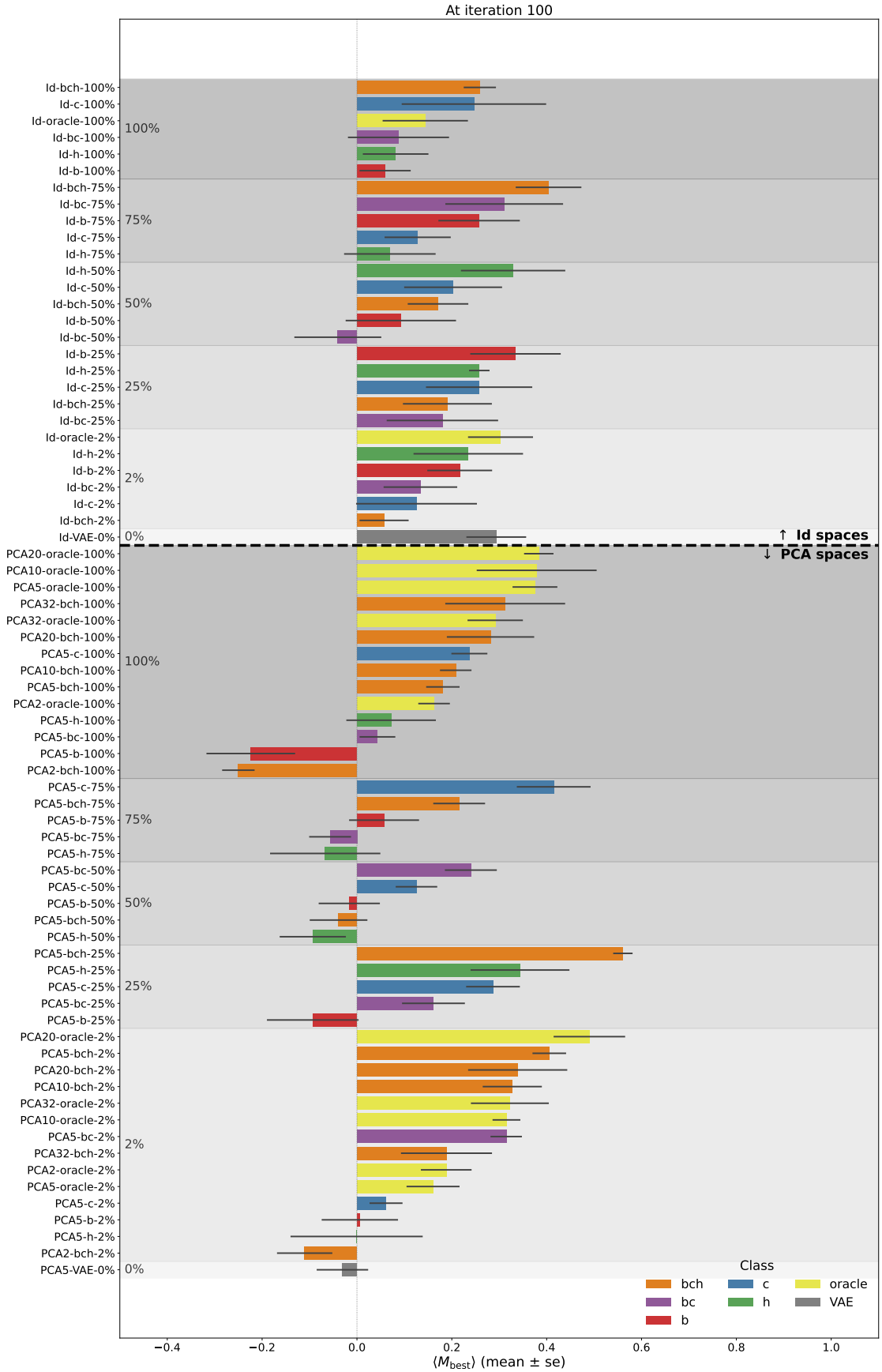

Fig. S11 Average best objective scores found at iteration 100. Error bars correspond to the standard error of M_{best} over five BayesOpt runs. Black dashed line separates those runs performed in the high-dimensional latent space (above the dashed line), and those runs performed in a PCA projection of the latent space (below the dashed line). Shaded regions denote the label percentage available at training time, with darker regions corresponding to a higher label percentage. Values are sorted within groups from largest to smallest. The full BayesOpt trajectories are depicted in SI Fig. S7.

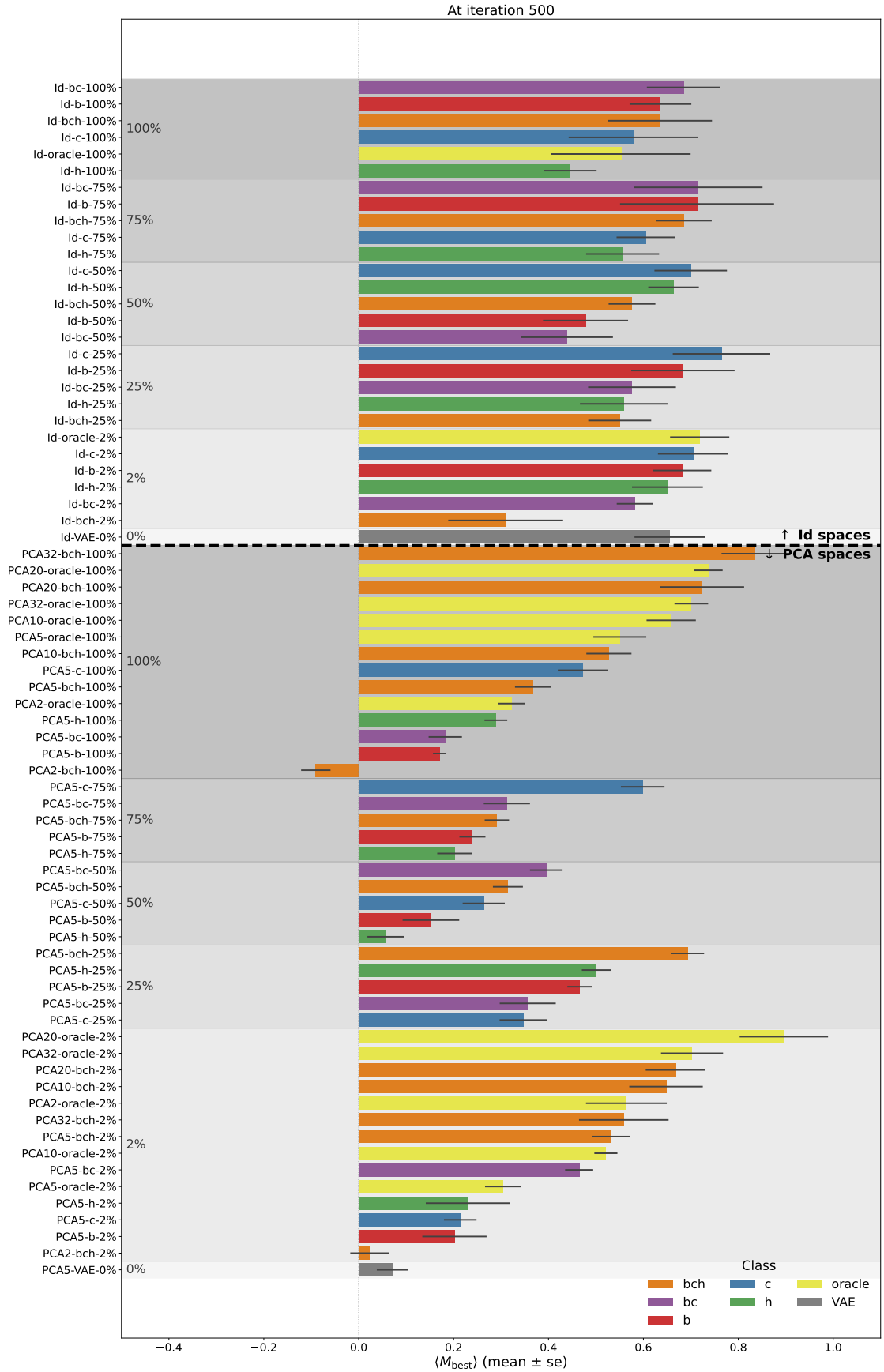

Fig. S12 Average best objective scores found at iteration 500. Error bars correspond to the standard error of M_{best} over five BayesOpt runs. Black dashed line separates those runs performed in the high-dimensional latent space (above the dashed line), and those runs performed in a PCA projection of the latent space (below the dashed line). Shaded regions denote the label percentage available at training time, with darker regions corresponding to a higher label percentage. Values are sorted within groups from largest to smallest. The full BayesOpt trajectories are depicted in SI Fig. S7.

S3.3 Deep Kernel Learning

Deep Kernel Learning[13] augments the kernel k_ϕ with parameters ϕ of a Gaussian Process with a neural network g_θ with weights θ non-linearly transforming design points into a lower-dimensional representation. In the GP-DKL approach the neural network weights θ are fit simultaneously with the kernel parameters ϕ. To do this simultaneous fit, at each iteration of BayesOpt we performed 400 epochs of backpropagation using the Adam optimizer.

We summarize the neural network architectures we tested in Table S10. We took architecture inspiration from the original DKL publication[13], and shrunk each architecture as the amount of data they are exposed to is on the order of a couple of hundred points. In dkl-v1, we used two hidden layers with dimensions 32 and 16, outputting into $\mathbb{R}^5$. In dkl-v2, we used three hidden layers of dimensions 32, 16, and 12, outputting into $\mathbb{R}^5$. In dkl-v3 we used three hidden layers of dimensions 48, 24, and 12, outputting into $\mathbb{R}^5$. Each architecture used simple feed-forward layers and leaky ReLU activate functions. The outputs of the neural networks were then fed into a standard radial basis function kernel. Lastly, 64-dimensional latent space points are inputs to each architecture, rather than raw sequences.

Abbreviation	**Architecture**			**Base kernel**
	Input dimension	Hidden layers	Output dimension	
dkl-v1	64	32->16	5	RBFKernel
dkl-v2	64	32->16->12	5	RBFKernel
dkl-v3	64	48->24->12	5	RBFKernel

Table S10 Neural network architectures used for our Deep Kernel Learning experiments. We used a simple feed-forward layer for each hidden layer, leaky ReLU activation functions, and initialized weights with Xavier normal, and initialized each layer's bias as a vector of zeros. Note that we used relatively small architectures compared to the original DKL publication as the neural network is exposed to much less data.

Results for GP-DKL compared to LBO and LBO in a PCA reduce space are presented in SI Fig. S13. We performed GP-DKL BayesOpt over the latent space of five different transVAE models: BCH-organized transVAE with access to 100% of the property labels, BCH-organized transVAE with access to 2% of labels, oracle-organized with access to 100% and 2% of property labels, and an unorganized transVAE. Across all four organized transVAE models, DKL underperforms relative to LBO and LBO in a PCA reduced latent space. Over an unorganized transVAE, each of the approaches perform similarly, with traditional LBO pulling ahead after 500 iterations.

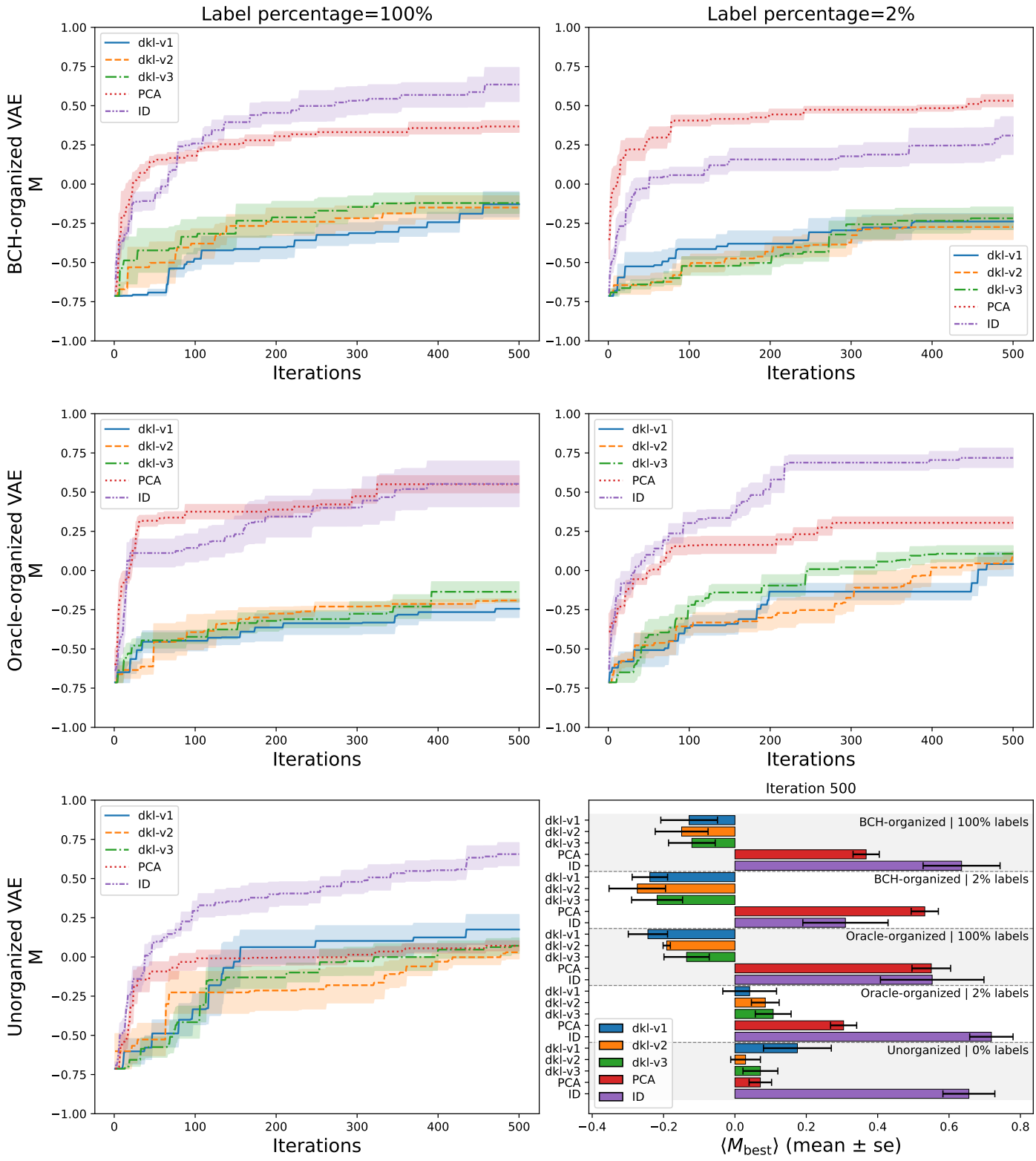

Fig. S13 BayesOpt results comparing DKL to LBO and LBO in a PCA reduced latent space.

S3.4 Sequence space exploration

Similar to Sect. 3.5, we examined the extent to which sequences are different between searching in the full latent space, and searching in a PCA projection of the latent space. Specifically, in response to the question: is there a loss of novelty when searching through the PCA projection compared to the full latent space?

In Fig. S14a we have compared the oracle scores of sampled sequences compared to both the VAE training set, and the oracle's training set, finding that BayesOpt done in the full latent space on average sampled better (i.e. lower) oracle scores compared to searching in the PCA-reduced space. In Fig. S14b we plotted the lengths of the sequences sampled throughout BayesOpt runs, finding that both BayesOpt in PCA reduced spaces and in the full latent space sample similar distributions. However, we note that the PCA reduced spaces sample more short peptides, thus a distribution closer to that of both training sets. On the one hand, these observations indicate a lower novelty traversed by the algorithm in the PCA space; on the other hand, it is possible that this partially protects the algorithm in this space against venturing into territory that is less well-predicted by the oracle. In Fig. S14c we observe that the Levenshtein path lengths of each BayesOpt run were smaller in the full latent space compared to the PCA-reduced space, suggesting that sequences sampled throughout a BayesOpt run in the full latent space were more similar to each other than those sampled when searching through the PCA-reduced space. In Fig. S14d, we check the sequence distance of the best sequence found in each run to the oracle's training set; we observe that sequences sampled from the PCA-reduced space were closer to the oracle's training set compared to the sequences sampled from the full latent space. Figures (a) and (d) together suggest that using a PCA-reduced space can lead to sequences more similar to the oracle's training set compared to using the full latent space. Figure (b) and (c) together suggest that searching through a PCA-reduced space can have a larger breadth of sequences sampled (larger variety of lengths, larger path lengths in sequence space) compared to searching through the full latent space.

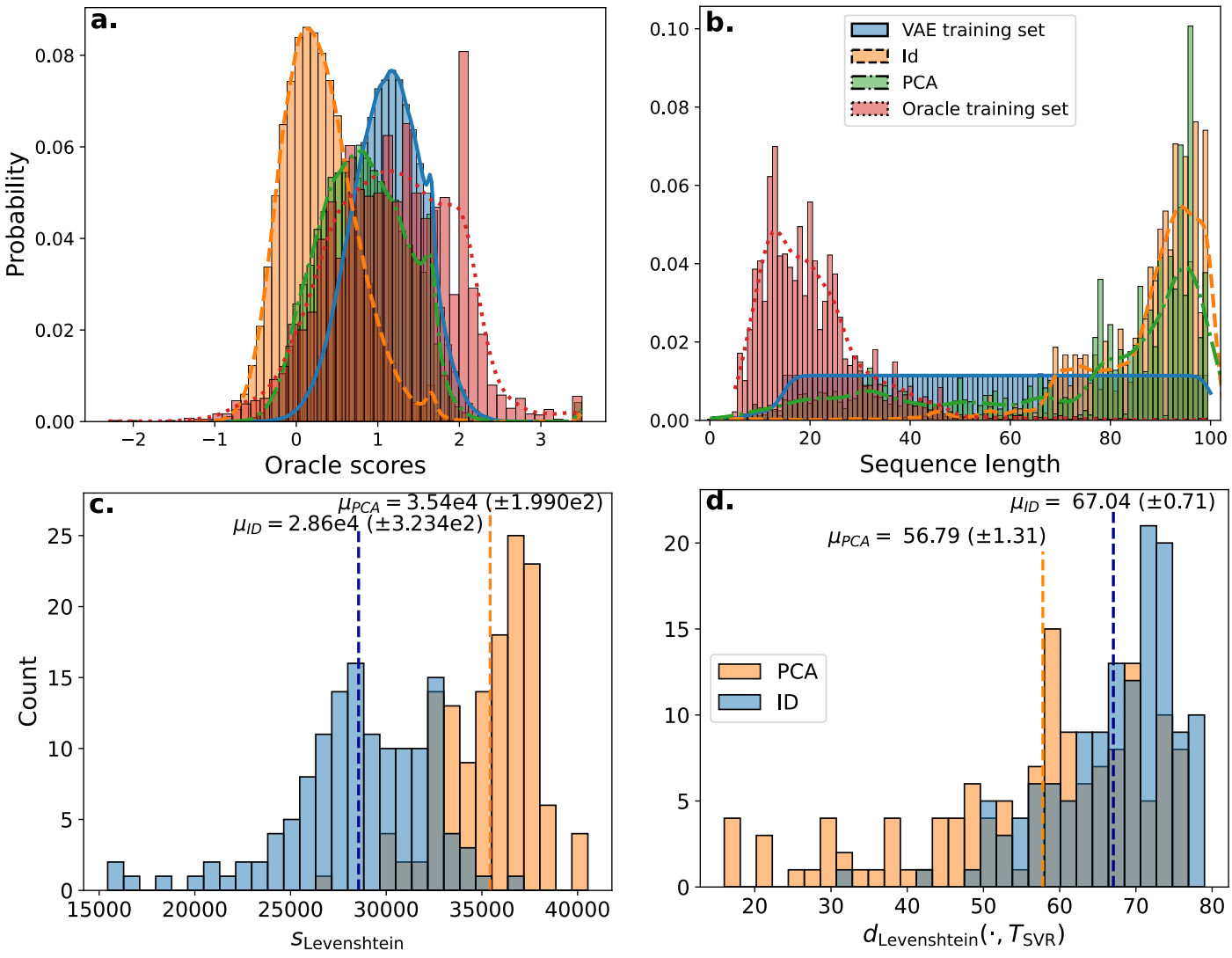

Fig. S14 Exploring sequence space. (a) Distribution of oracle scores for the VAE training set (blue, solid line), the oracle's training set (red, dotted line), the sequences sampled when searching through the full latent space (orange, dashed line) or the PCA projection (green, dash-dotted line). (b) The lengths of sequences from the VAE training set (blue, solid line), oracle training set (red, dotted), or sequences sampled from the full latent space (orange, dashed) or PCA projection (green dash-dotted). (c) The Levenshtein path lengths of sequences sampled in a given BayesOpt run searching through either the full latent space (blue) or PCA projection (orange). (d) The Levenshtein distance of the best sequence identified in each BayesOpt when searching through the full latent space (blue) or PCA projection (orange) to the oracle's training set.

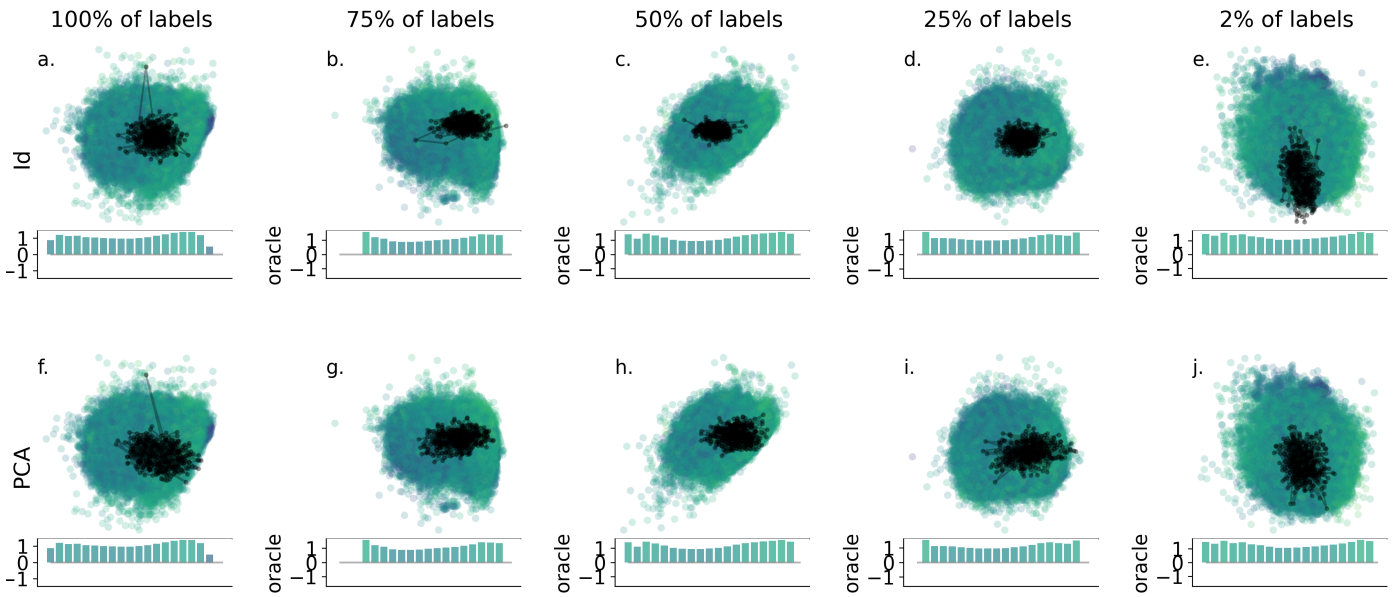

Fig. S15 Boman-organized latent spaces, coloured by peptide training set oracle predictions.

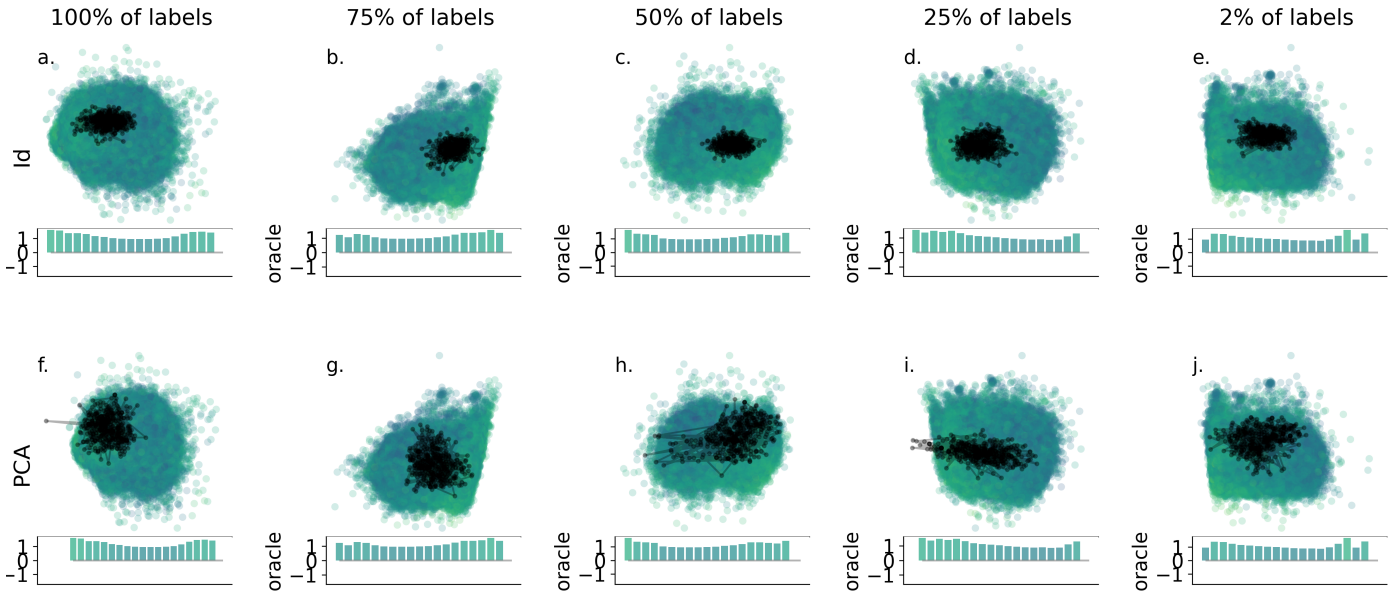

Fig. S16 Charge-organized latent spaces, coloured by peptide training set oracle predictions.

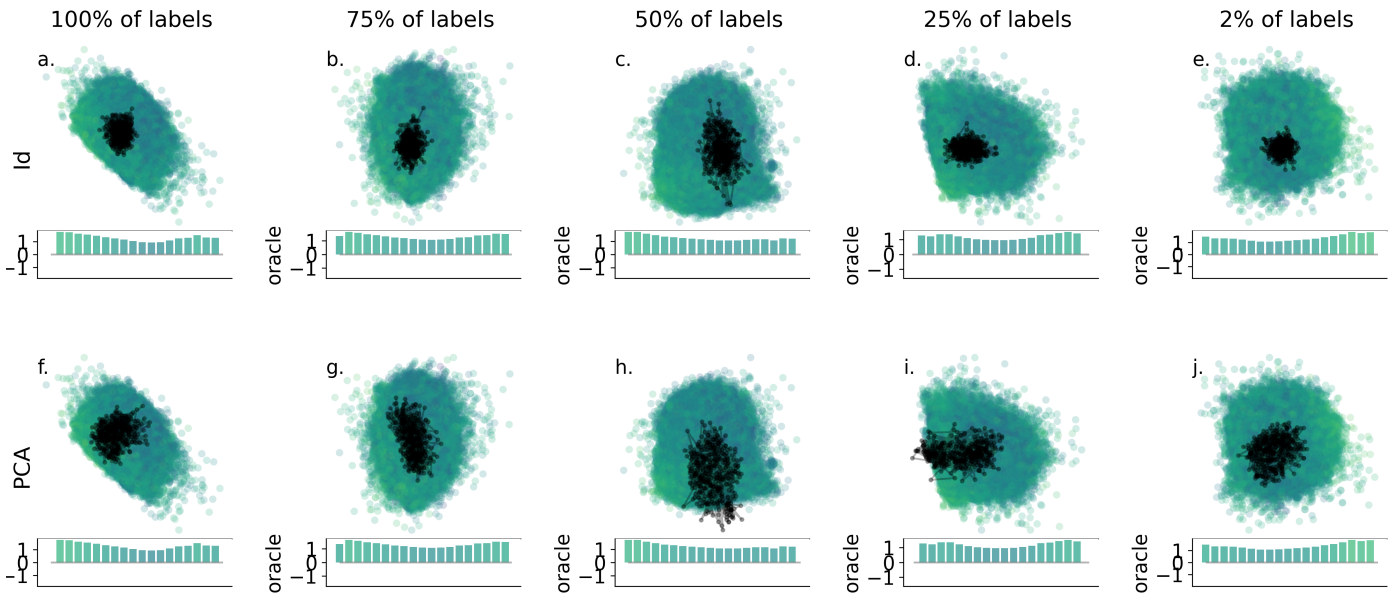

Fig. S17 Hydrophobicity-organized latent spaces, coloured by peptide training set oracle predictions.

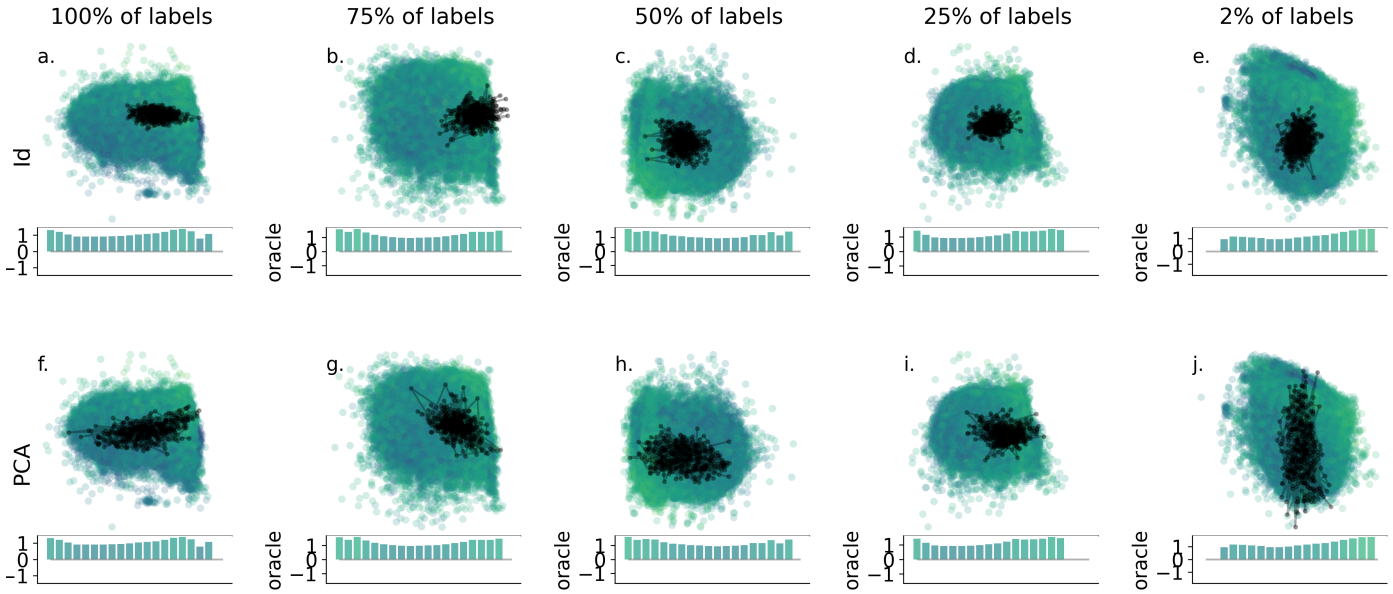

Fig. S18 BCH-organized latent spaces, coloured by peptide training set oracle predictions.

S3.4.1 Coloured by organizing property

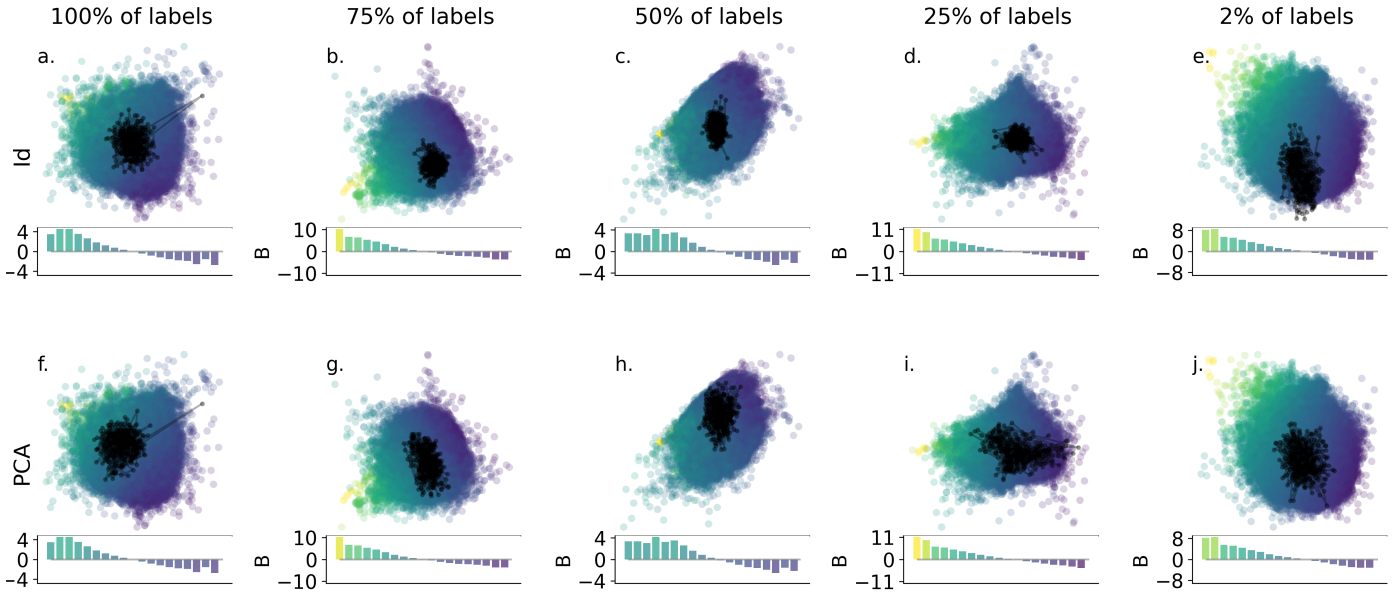

Fig. S19 Boman-organized latent spaces.

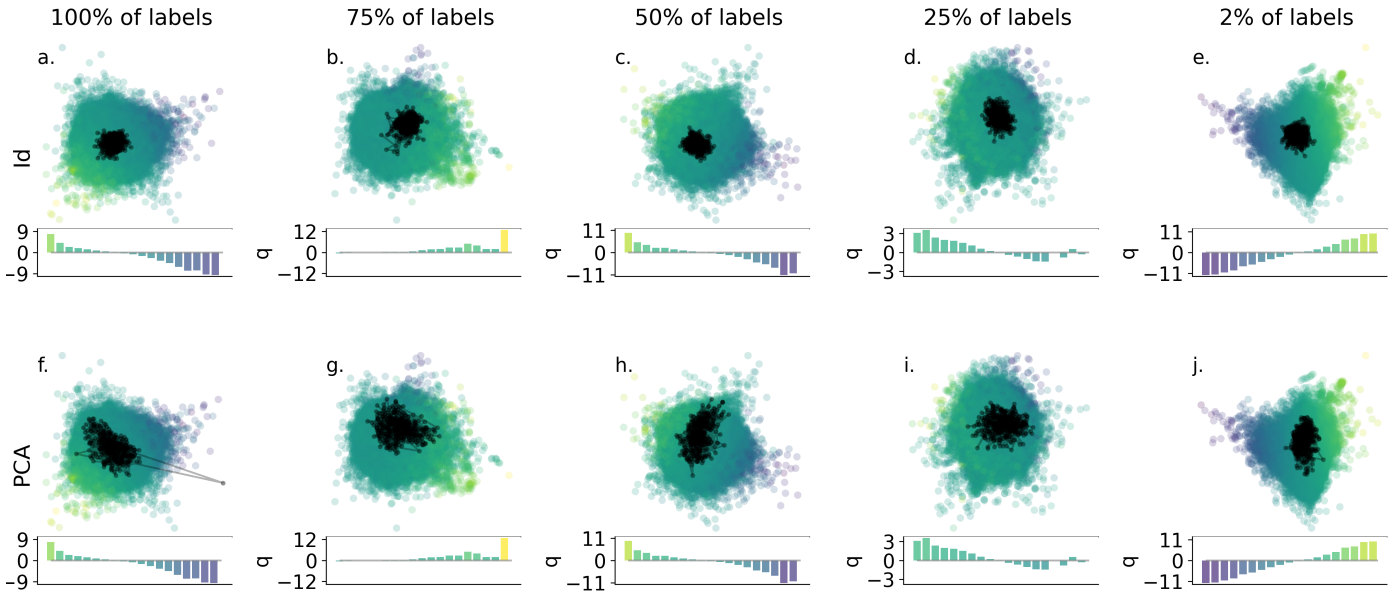

Fig. S20 Charge-organized latent spaces.

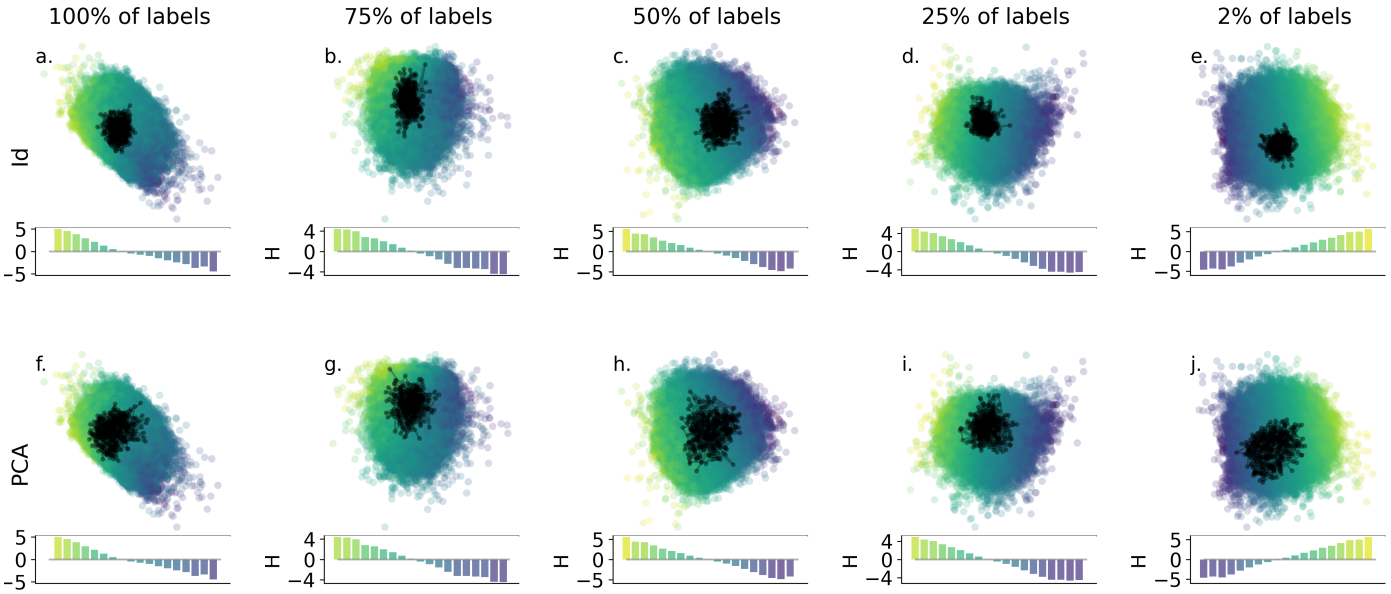

Fig. S21 Hydrophobicity-organized latent spaces.

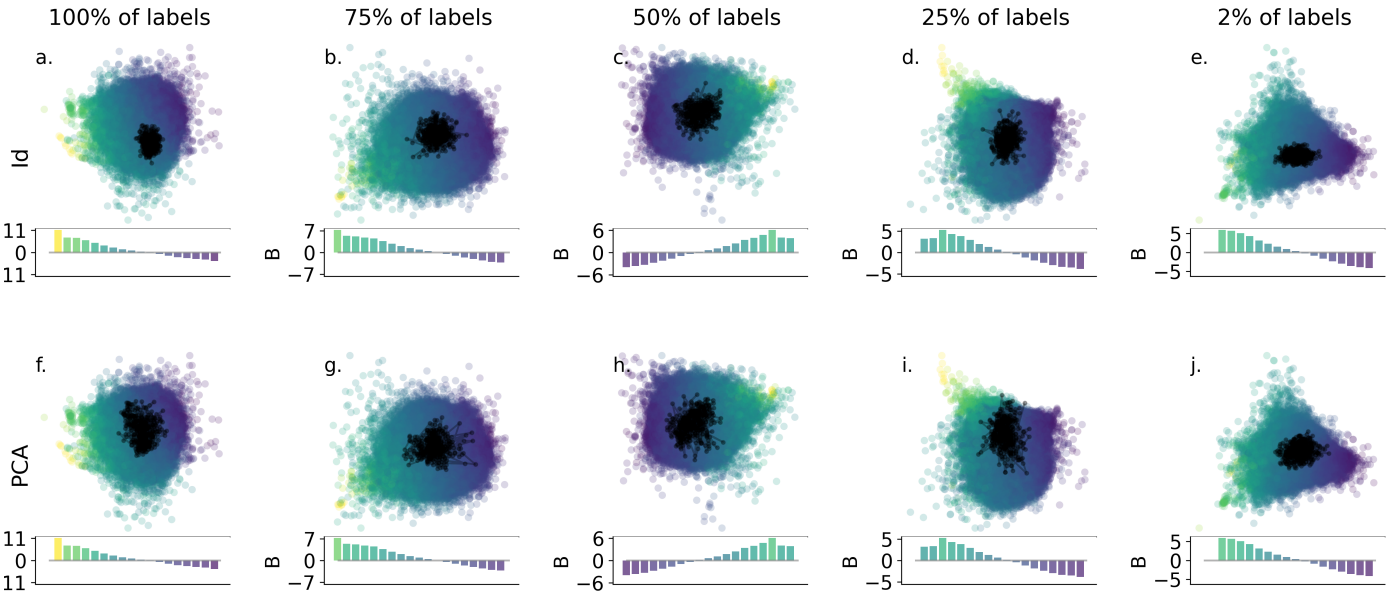

Fig. S22 BCH-organized latent spaces coloured by Boman index.

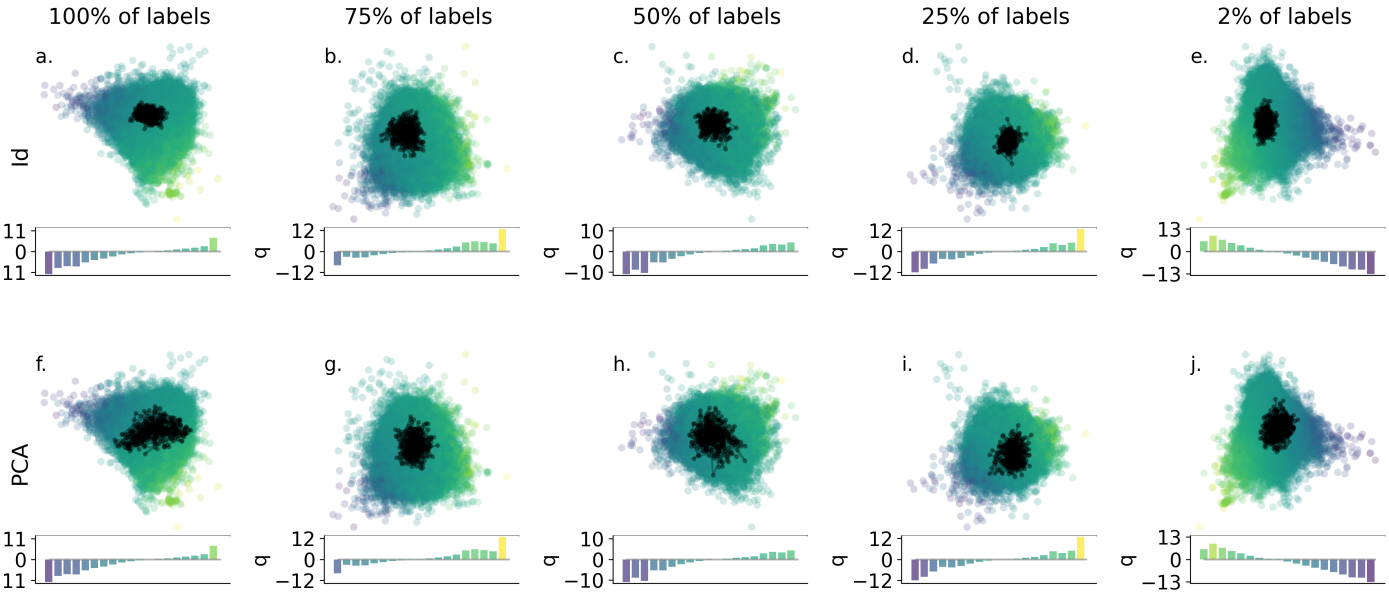

Fig. S23 BCH-organized latent spaces coloured by charge.

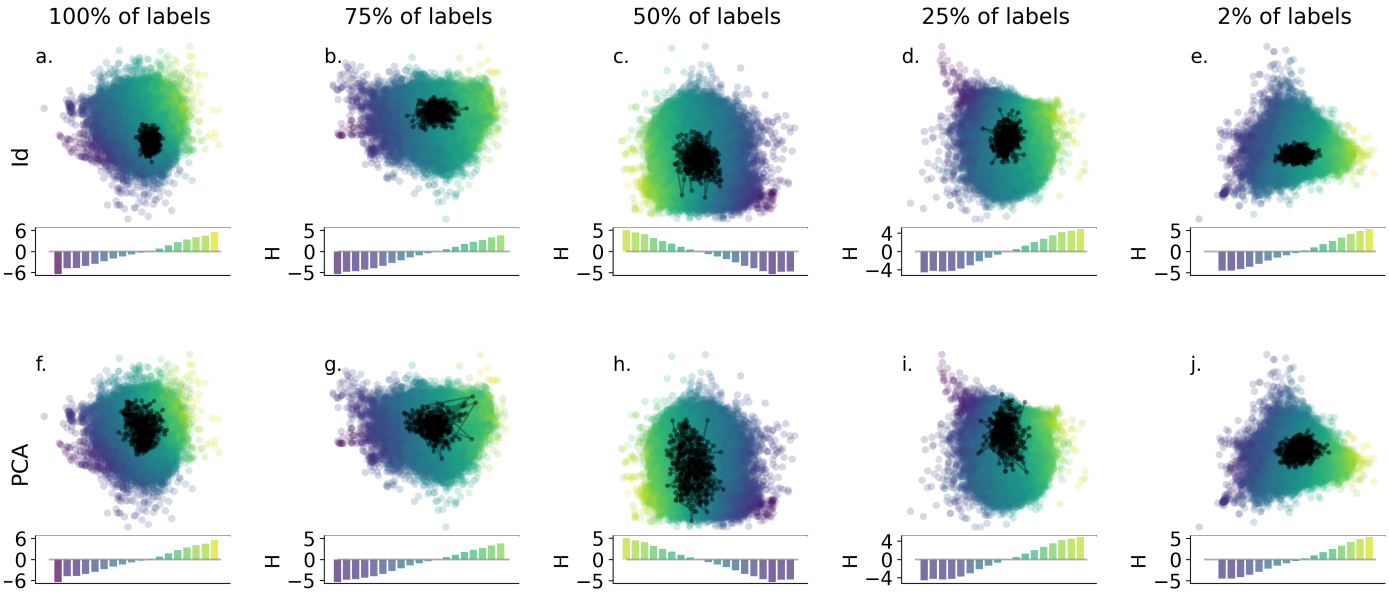

Fig. S24 BCH-organized latent spaces coloured by hydrophobicity.

Notes and references

1 J. Witten and Z. Witten, Deep learning regression model for antimicrobial peptide design, 2019, https://www.biorxiv.org/content/10.1101/692681v1, Pages: 692681 Section: New Results.

2 F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, Journal of Machine Learning Research, 2011, **12**, 2825–2830.

3 E. Y. Lee, B. M. Fulan, G. C. L. Wong and A. L. Ferguson, Proceedings of the National Academy of Sciences, 2016, **113**, 13588–13593.

4 M. Thoma and J. Hahnfeld, ProPy3: Python3 package for computing protein descriptors., https://pypi.org/project/propy3/.

5 C. Ding and H. Peng, Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003, 2003, pp. 523–528.

6 R. Tibshirani, Journal of the Royal Statistical Society. Series B (Methodological), 1996, **58**, 267–288.

7 J. Bi, K. P. Bennett, M. Embrechts, C. M. Breneman and M. Song, 2003.

8 S. Renaud and R. A. Mansbach, Digital Discovery, 2023.

9 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, Attention Is All You Need, 2023, http://arxiv.org/abs/1706.03762, arXiv:1706.03762 [cs].

10 D. P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization, 2017, http://arxiv.org/abs/1412.6980, arXiv:1412.6980 [cs].

11 https://www.comet.com/site/.

12 https://www.hyundai.com/worldwide/en/eco/ioniq6/technology.

13 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.