

Supporting Information:

Data-driven discovery of cardiolipin-selective probe molecules by computational active learning

Bernadette Mohr^{†a}, Kirill Shmilovich^{†b}, Isabel Kleinwächter^c, Dirk Schneider^c, Andrew L. Ferguson^{¶b} and Tristan Bereau^{¶a,d}

^aVan 't Hoff Institute for Molecular Sciences and Informatics Institute, University of Amsterdam, Amsterdam 1098 XH, The Netherlands.

^bPritzker School of Molecular Engineering, University of Chicago, Chicago, Illinois 60637, USA.

^cDepartment of Chemistry - Biochemistry, Johannes Gutenberg University Mainz, 55128 Mainz, Germany

^dMax Planck Institute for Polymer Research, 55128 Mainz, Germany

1 Supporting Methods

1.1 Molecular dynamics simulations

1.1.1 Simulation parameters

Coarse-grained molecular dynamics simulations were performed with the GROMACS 2018.6¹ simulation suite using the Martini 2² force field to represent the lipids and the refined polarizable models for the Martini water and ions.^{3,4} The small molecules were modeled with a *5+1* bead type coarse-grained (CG) model we created based on the *5+0* five-bead-type reduced Martini model published by Kanekal *et al.*,⁵ which is compatible with Martini (Fig. S1). This reduced set of bead types facilitates a more efficient coverage of chemical space while still accounting for all relevant chemical and physical properties to a level of accuracy comparable with Martini. We applied the standard force-field parameters introduced for GPU acceleration⁶ and commonly found in recent Martini papers, with an integration time step of $\delta t = 0.02\tau$, where τ is the natural unit of time for the model. The simulations were kept at constant temperature ($T = 300$ K) and pressure ($P = 1$ bar) using the Langevin thermostat⁷ and the Parrinello-Rahman barostat.^{8,9} both as implemented in GROMACS. Semi-isotropic pressure coupling was used for membrane simulations and isotropic pressure coupling for simulations in octane and water. The corresponding coupling constants were $\tau_P = 12\tau$ and $\tau_T = \tau$. Electrostatic interactions were calculated with particle-mesh Ewald (PME) summation¹⁰ in systems containing polarizable water and ions. A soft-core potential with the parameters set to *sc-alpha* = 0.5, *sc-power* = 1, and *sc-sigma* = 0.3 was applied in the free-energy calculations to ensure convergence around the fully uncoupled state.¹¹⁻¹⁴

The membranes were generated with the CHARMM-GUI Martini maker.¹⁵ Phosphatidylglycerol (PG) membrane simulations contained 118 lipid molecules (59 per layer) solvated in 1754 water beads and 118 sodium ions; Cardiolipin (CL) membrane simulations contained 98 lipids (49 per layer) in 3287 water beads with 198 sodium ions. The bulk simulations were done in systems containing 336 octane molecules and 974 water beads, respectively. Additionally, 28 sodium and 28 chloride ions were added to the water system to adjust it to the same ion concentration as the membrane systems.

[†]Contributed equally to this work.

[¶]E-mail: andrewferguson@uchicago.edu, t.bereau@uva.nl

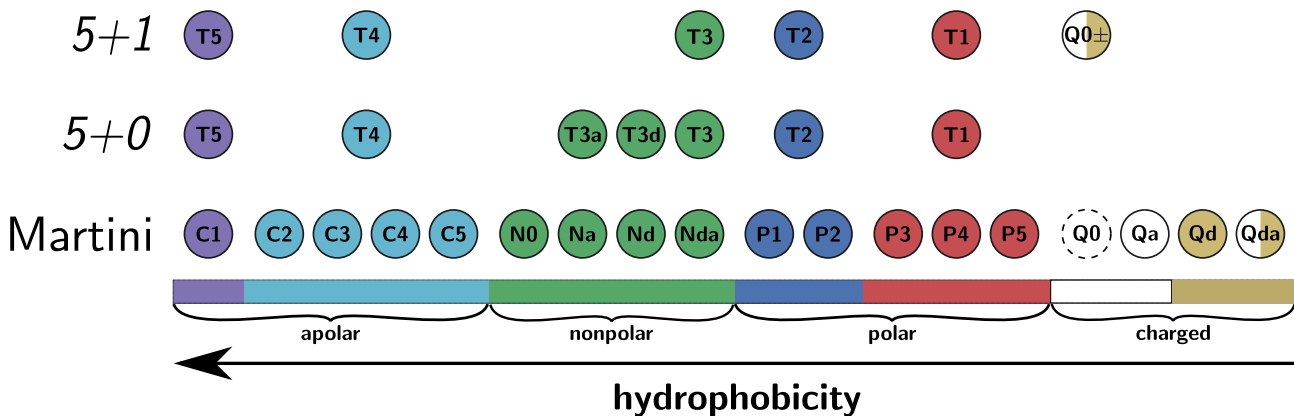


Figure S1: Coarse grained $5+1$ model created by extending the $5+0$ five-bead-type reduced Martini forcefield⁵ to include a charged bead type $Q0\pm$ that can carry one positive or negative charge. In the bottom row, the Martini 2 bead types are shown for reference. The T3 bead adopted from the $5+0$ forcefield encodes both hydrogen-bond donor and acceptor properties, thereby relating closest to the Martini 2 ($14+4$) Nda bead type.

1.1.2 Free-energy calculations

Solvation free energies¹⁶ ΔG were calculated by successively coupling the van der Waals (VdW) and electrostatic interactions between the candidate structures and the environments. The free-energy difference between a quasi-vacuum state, where no interactions between the solute and the environment are considered (state 0), and a fully solvated state (state 1) is estimated by transforming the Hamiltonian with a coupling parameter λ . This corresponds to the horizontal processes in Fig. S2: state 0 is represented by the white colored solute, state 1 by the yellow solute. In the case of charged compounds the systems were not kept at electrostatic equilibrium over all states. The net charges were non-zero ($q \neq 0$) at state 0, and successively equilibrated during coupling of the electrostatic interactions until they reach $q = 0$ at state 1. Although this increases the emergence of finite-size effects^{17,18} introducing errors into our calculations, they are ameliorated by the fact that the errors are comparable between the positive and negative net charges, whereas the selectivity signal clearly favors positive net charges. We quantify these errors by performing Gaussian error propagation and verify that our uncertainties are smaller than 10% of the corresponding free energy difference $\Delta\Delta G$.

The efficiency of free-energy calculations depends on an optimal choice of intermediate states that aims at increasing phase-space overlap. The simplest approach is to follow a linear path between the two end-point Hamiltonians: $\mathcal{H}(\lambda) = \lambda\mathcal{H}_1 + (1 - \lambda)\mathcal{H}_0$. Here, the spacing of the intermediate states was linear for the VdW interactions. For the electrostatic interactions a spacing of $\Delta\lambda \propto \sqrt{\lambda}$ was applied, in addition to two additional steps early on to ensure smooth and small free-energy contributions (Fig. S3). The absence of electrostatic interactions for neutral compounds and for all compounds in the octane environment allows us to reduce the computational costs by only considering vdW interactions. The solutes were constrained to an area of 1 nm around the interface region of the membranes using two flat-bottom potentials as implemented in Gromacs (Fig. S4),

$$V_{\text{fb}}(r_i) = \frac{1}{2}k_{\text{fb}} [d_g(r_i; R_i) - r_{\text{fb}}]^2 H [d_g(r_i; R_i) - r_{\text{fb}}], \quad (1)$$

with the reference position R_i being the bilayer midplane ($z=0$ nm), the distance r_{fb} from R_i the flat-bottom potential is applied at, the force constant $k_{\text{fb}} = 1000 \text{ kJmol}^{-1}\text{nm}^{-2}$, the Heaviside step function¹⁹ and the distance $d_g(r_i; R_i)$ of the constrained solute from the reference position. The solute was constrained in a layer with the layer normal parallel to the membrane normal z , see Fig. S7. The detailed parameters can be found in the Gromacs simulation parameter files included in the ESI†²⁰. This was done to prevent the solutes from moving too far towards the bilayer midplane or into the water phase, while still allowing them to adjust to different positions relative to z along the membrane

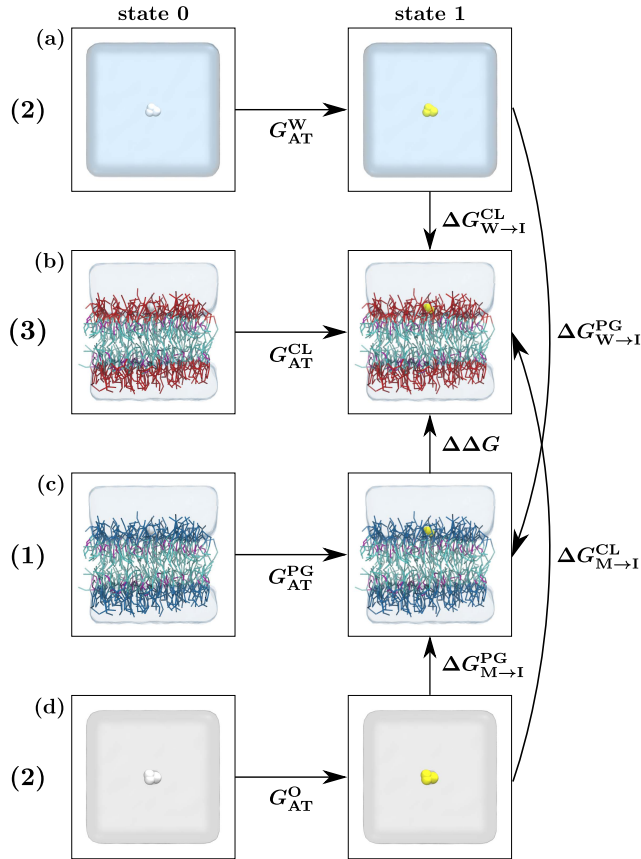


Figure S2: Free energy calculation workflow, with numbers showing the three steps as in Figure S7. Interactions are turned off in state 0 (white solute) and fully on in state 1 (yellow solute). Horizontal arrows represent absolute solvation free energies, vertical arrows represent partitioning free energies between different environments. Environments are (a) bulk water and ions, (b) CL bilayer, a water phase and ions, (c) PG bilayer, water phase and ions, and (d) bulk octane. In (b) and (c), a section of the water phase was removed from the images for aesthetic purposes.

interfaces. The position of the interface region for each lipid type was determined by the minima of potentials of mean force (PMFs) calculated for all bead types making up the $5+1$ model (Fig. S5).

The individual simulations for each intermediate state were done in succession to guarantee proper equilibration of the systems, with the resulting particle configuration of each intermediate state serving as the initial configuration for the next intermediate state. The free energies ΔG for the transformations from state 0 to state 1 and their corresponding uncertainties were calculated using MBAR²¹ with tools provided by the pymbar²² package.

We focus on the three environments water (W), interface (I), and bilayer midplane (M) and for each membrane compute the transfer free energies $\Delta G_{M \rightarrow I}$ and $\Delta G_{W \rightarrow I}$. These transfer free energies correspond to the vertical processes in Fig. S2. To speed up the calculations we use bulk octane (O) as proxy for the midplane environment²³. In Fig. S6 we present calculated free energies for the transfer of each CG bead from water into the midplane of PG and CL membranes versus those for transfer into octane to demonstrate that this is a very good approximation. The λ multiplication factors are given in the GROMACS parameter files provided with the ESI†²⁰.

1.2 Free-energy workflow

The free-energy calculations were done in three steps, where the end of each step attempts to terminate unsuitable compounds to minimize the overall computational cost. We first determine whether a solute is likely to spontaneously partition at the membrane–water interface. We calculate the water to interface transfer free energy in PG, $\Delta G_{W \rightarrow I}$ (see Fig. S2 and Fig. S7 (1)). The transfer free

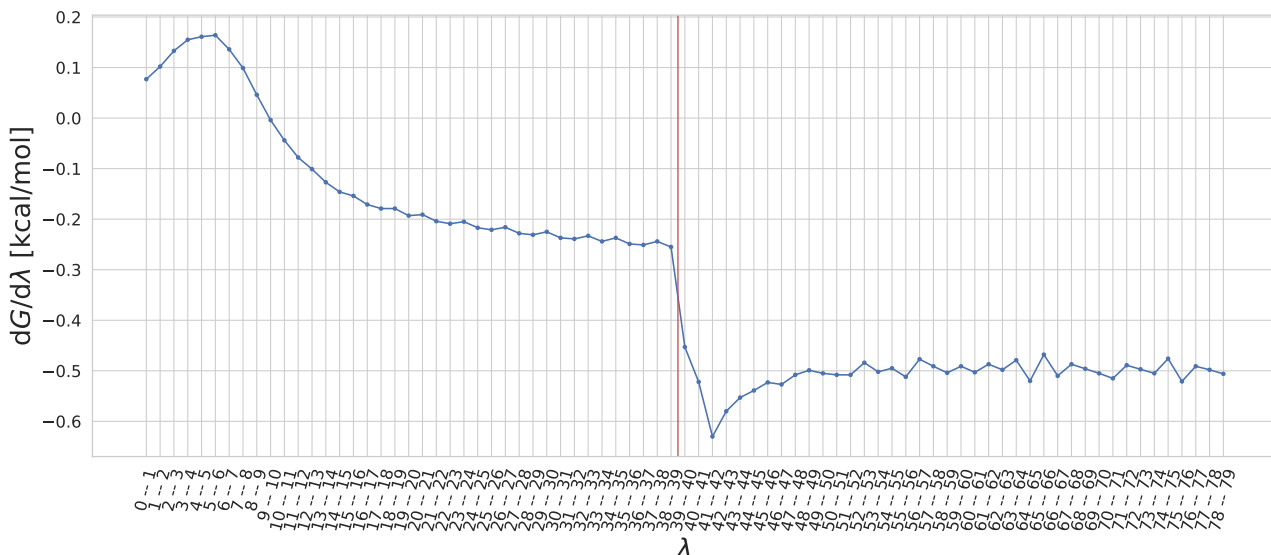


Figure S3: Free energy changes for every pair of coupling parameters λ for charged molecules. In the first 40 steps, to the left of the vertical red line, the VdW interactions are coupled incrementally. In the second 40 steps, to the right of the red line, the Coulomb interactions were coupled. The spacing for the VdW interactions was linear, for the spacing for the Coulomb interactions a spacing of $\Delta\lambda \propto \sqrt{\lambda}$ was applied. Additionally, the first two steps in the Coulomb part were inserted manually to keep the size of the free energy change reasonably consistent over the whole calculation.

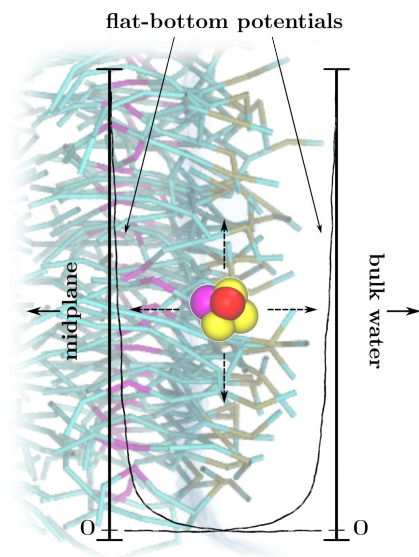


Figure S4: Approximate position of the two flat-bottom potentials used to prevent solutes from leaving the interface region in the direction of the bilayer midplane or the bulk water phase. For the potential function and parameters, see Eqn. 1.

energy requires an alchemical transformation at the membrane interface, where we loosely restrain the compound to stay within an interval of z by means of flat-bottom potentials. Significant sampling outside a narrow region around the interface is interpreted as compounds that are *not* interfacial. We monitor the cumulative probability density for the positions of the solute along z over the simulation trajectory of state 1, as shown by the histogram at step 1 in Fig. S7. The position at which the inflection point occurs is highlighted in Fig. S8, informing us on the likely position of the compound relative to the headgroup interface. We require the inflection point to fall within the third and seventh

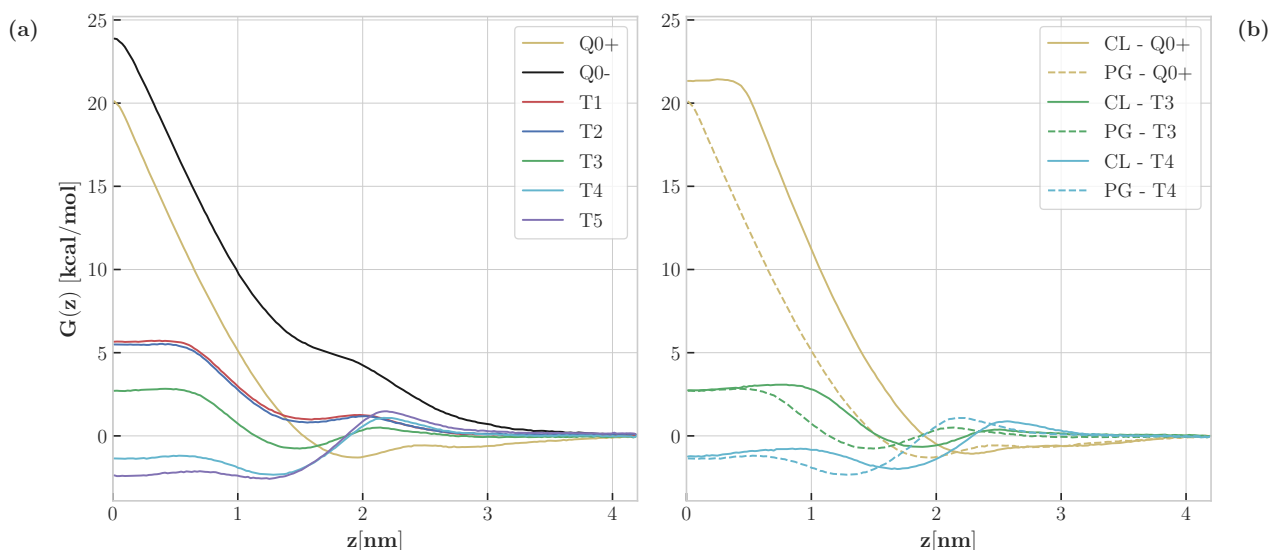


Figure S5: Potentials of mean force of individual CG beads. Minima indicate the position of the interface region of (a) PG and (b) both PG and CL membranes along the membrane normal z . The PMFs also give a clear indication which bead types will facilitate alignment of a small molecule with the interface (T3, T4, Q0+), will drive insertion into the bilayer midplane (T5), or will increase the tendency of the compound to stay in the bulk water phase.

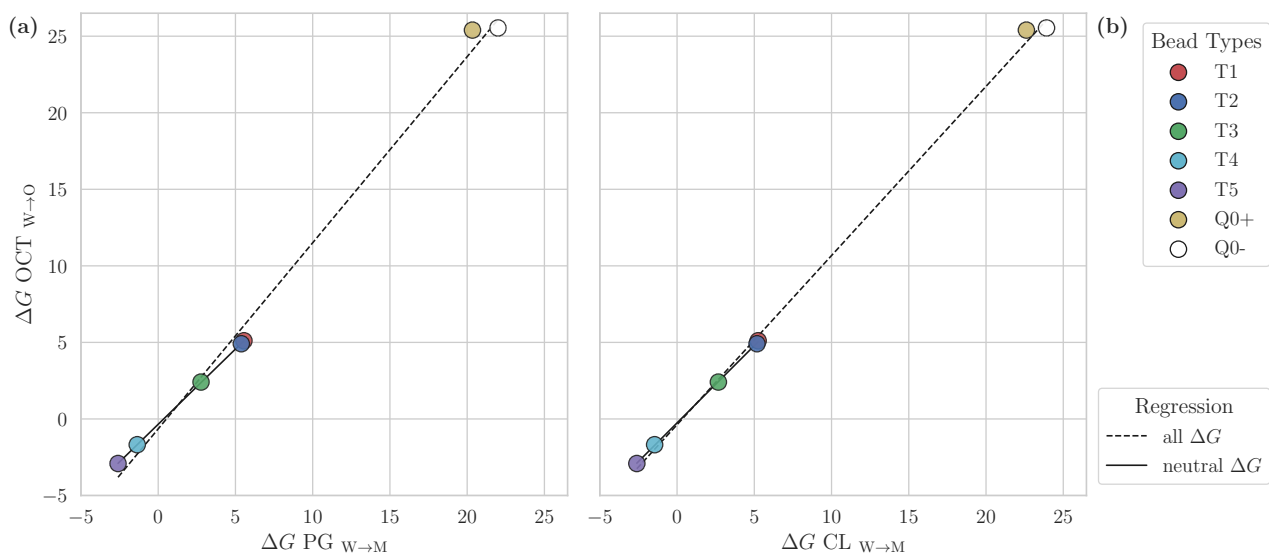


Figure S6: Transfer free energies of single beads from water to bulk octane and bilayer midplane for (a) PG and (b) CL bilayers. A simple linear transformation is suitable to relate bulk octane to bilayer midplane in both membrane environments, in good agreement with Menichetti *et al.*²³ The slight deviation from the linear relationship when involving charged beads can be attributed to long-range electrostatic interactions between the charged beads and the lipid headgroups.

histogram bins, a range optimized from tests on the individual beads of the $5+1$ force field to contain the bead types whose PMFs (Fig. S5) showed a clear minimum around the interface region of the membranes. This area is represented by the gray area in Fig. S8. If the inflection point falls outside this cutoff range, the candidate solute is considered non-interfacial and the free-energy workflow is stopped early.

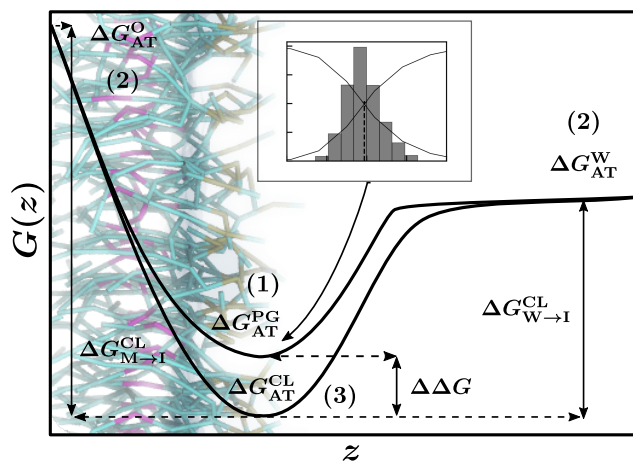


Figure S7: PMFs of a solute in a PG and a CL membrane system with the individual free energy workflow steps indicated at the corresponding sections of the curves. The numbers represent the steps in the workflow at which the respective free energy calculations are done. The inset shows the position probability histogram of the solute at the membrane interface region and the curves of the cumulative probabilities once summed up from the midplane side and once from bulk water. The subscript "AT" here clarifies that the free-energy changes are calculated using an alchemical transformations approach, the PMF curves in the illustration are shown to clarify the positions in the system at which the individual free energies are calculated.

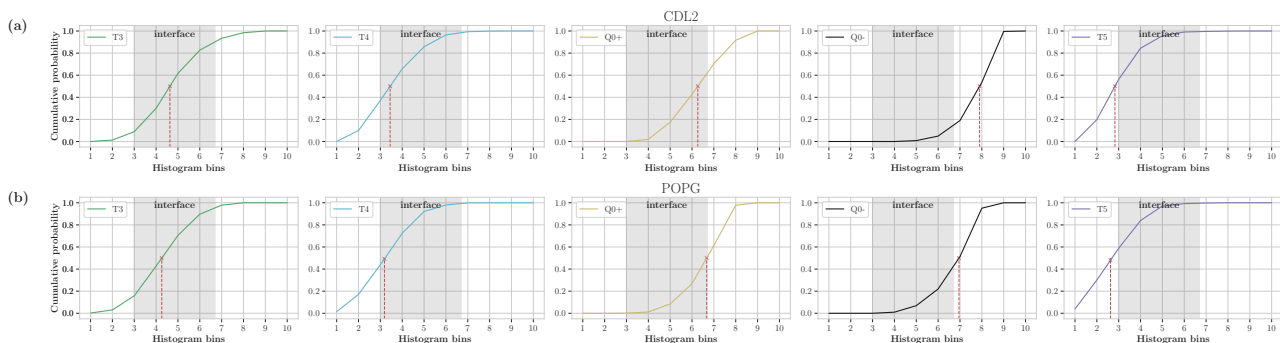


Figure S8: Cumulative probabilities of the positions of individual beads during the alchemical transformation at the fully coupled state $\lambda = 1$. The beads were constrained between two flat-bottom potentials as illustrated in Fig. S4, to give them sufficient translational freedom to adjust their position towards the membrane midplane or the bulk water phase. The grey shaded area denotes the histogram bins containing the positions around the interface region. If the positions fall within the gray area, we predict that the candidate molecule will preferentially align within the interface region of the membranes and proceed to the next step of the alchemical free energy calculations to quantitatively verify this prediction based on transfer free energies. This approach was used as an early exit scheme to quickly eliminate candidate molecules unlikely to preferentially partition near to the membrane interface and therefore unlikely to exhibit selectivity between the PG and CL membrane environments.

If the solute passed step 1, we ran simulations in bulk water and bulk octane to compute the transfer free energies $\Delta G_{O \rightarrow I}$ and $\Delta G_{W \rightarrow I}$ for the PG membrane (Fig. S2 and Fig. S7 (2)). We now quantitatively verify that the solute is indeed interfacial by monitoring the signs of both transfer free energies. If either $\Delta G_{O \rightarrow I} > 0$ or $\Delta G_{W \rightarrow I} > 0$, insertion at the membrane interface is not spontaneous and we terminate the free-energy workflow. Otherwise, we confirm that the candidate molecule does indeed partition to the membrane interface and we declare the molecule "interfacial".

If steps 1 and 2 both complete without early exit, we then move on to step 3 comprising free-energy calculations within the CL membrane environment (Fig. S2 and Fig. S7 (3)). The alchemical transformation in the CL membrane environment allows us to finally compute the desired selectivity measure $\Delta\Delta G = \Delta G_{W \rightarrow I}^{CL} - \Delta G_{W \rightarrow I}^{PG}$. A candidate with $\Delta\Delta G < 0$ has a higher binding affinity to CL than to PG, and it is this quantity that we seek to minimize in our molecular discovery active learning pipeline.

1.3 Chemical-space embedding

The chemical space considered here is composed of a large number of small organic compounds, which are inherently high-dimensional and discrete objects. Performing optimization within this design space to identify candidates that display selectivity to CL-based membranes requires establishing a notion of proximity between different compounds. Gómez-Bombarelli *et al.*²⁴ proposed a framework for performing optimization in discrete molecular spaces by representing molecules as continuous, real-valued vectors within a learned low-dimensional embedding. This embedding provides a compact and continuous representation that can be directly exposed to off-the-shelf optimization algorithms.

Here we generate a low-dimensional representation of our design space of small coarse-grained compounds using regularized autoencoders (RAEs)²⁵, a deterministic adaptation of the variational autoencoder (VAE)²⁶ encoder–decoder architecture capable of generative modeling and interpolation. Active learning using Bayesian optimization is subsequently performed to navigate this embedded chemical space with the objective of discovering structures that preferentially permeate into CL membranes. The following sections outline the data processing, neural network architecture, and training procedures that are used to generate the continuous representation of our discrete chemical space and deployed within our active learning workflow.

1.3.1 Permutation-invariant graph RAEs

Coarse-grained compounds may be represented as graph-structured data, similar to atomic graphs, where node features correspond to the identity of CG beads and the collection of edges connecting the nodes reflect information on the coarse-grained topology. While graph neural networks²⁷ have proven to be powerful tools in processing graph-structured data while respecting permutational symmetries, graph reconstruction from latent codes in encoder–decoder architectures stymies training due to no efficient way to account for permutational invariance of the output in the reconstruction loss. Similar in motivation to Simonovsky *et al.*²⁸, where they evaluate reconstructions using approximate graph matching with soft discretization, we perform exact graph-matching by explicitly considering all permutations of the reconstructed graph to minimize the loss with respect to the input. While full graph matching is certainly not scalable to large atomistic molecular graphs, this approach is tractable when dealing with coarse-grained representations where relatively small five bead structures cover a large fraction of the space of small drug-like molecules. Henceforth, we proceed to exhaustively consider and embed within a unified latent space all possible coarse-grained topologies containing five or fewer beads.

1.3.2 Representing coarse-grained compounds as graph-structured objects

Each coarse-grained structure k in our design space is represented as a graph $G^{(k)} = G(N^{(k)}, A^{(k)}, E^{(k)})$, where $N^{(k)}$ is a matrix of bead type identities, $A^{(k)}$ is a binary adjacency matrix, and $E^{(k)}$ is a tensor of edge features. In our application we use the $5+1$ CG model with five non-charged unique bead types and a charged bead representing two charge states adapted from Kanekal *et al.*⁵, where each node feature $N_i^{(k)} \in \{\text{T1}, \text{T2}, \text{T3}, \text{T4}, \text{T5}, \text{Q0}\pm\}$ with $\{\text{T1}, \text{T2}, \text{T3}, \text{T4}, \text{T5}\}$ carrying a charge of $Q_i^{(k)} = 0$ and $\text{Q0}\pm$ being the same bead type of Q0 but carrying a charge of $Q_i^{(k)} \in \{+1, -1\}$, respectively. These node features $N_i^{(k)}$ can then be represented as the concatenation of a six-dimensional one-hot vector $\{0, 1\}^6$ for the six unique bead types (one of $\{\text{T1}, \text{T2}, \text{T3}, \text{T4}, \text{T5}, \text{Q0}\}$) along with a scalar value $Q_i^{(k)} \in \{-1, 0, +1\}$ for the associated bead charge such that $N_i^{(k)} = [\{0, 1\}^6 || Q_i^{(k)}]$, where the $||$

operator represents concatenation. As we consider coarse-grained compounds with up to $n = 5$ beads the adjacency matrix $A^{(k)} \in \mathbb{R}^{5 \times 5}$ defines the connectivity of the beads within a graph. Self-loops are also included corresponding to each node also being connected to itself. In the case where a molecule has fewer than $n = 5$ nodes the additional rows and columns are padded with zeroes in $A^{(k)}$. Lastly, the edges connecting the nodes $E_{i,j}^{(k)}$ are featurized according to the scalar parameter $\epsilon_{i,j}$ defining the depth of the potential well describing the Lennard-Jones 6-12 interaction between each pair of associated bead types $N_i^{(k)}, N_j^{(k)}$. Fig. S9 illustrates the data processing procedure of how coarse-grained structures are converted into these graph-structured representations that comprise our training dataset.

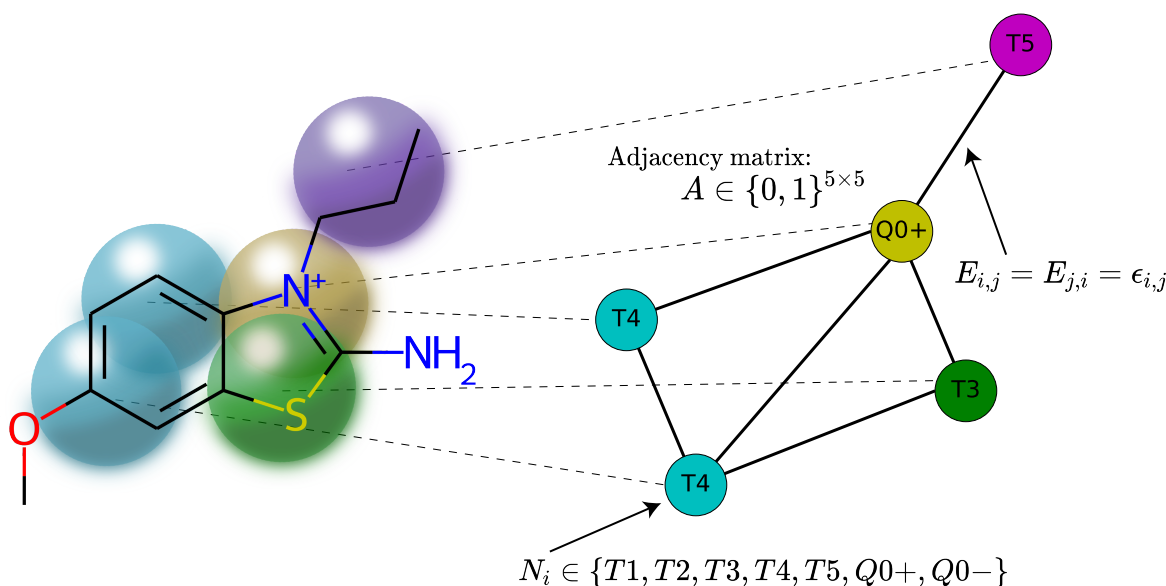


Figure S9: Schematic illustration of how each coarse-grained compound is represented as a graph-structured object. A binary adjacency matrix A captures the connectivity of the beads reflected in the coarse-grained topology. Features for each node N_i encode the identity of each coarse-grained bead one of $\{T1, T2, T3, T4, T5, Q0+, Q0-\}$, where $\{T1, T2, T3, T4, T5\}$ carry a charge of $Q_i = 0$ and $\{Q0+, Q0-\}$ carry a charge of $Q_i = \{+1, -1\}$, respectively. These node features N_i are represented mathematically as the concatenation of a one-hot vector $\{0, 1\}^6$ corresponding to the six unique bead types (one of $\{T1, T2, T3, T4, T5, Q0\}$) and a scalar value corresponding to the associated bead charge $Q_i \in \{-1, 0, +1\}$ such that $N_i \in \{\{0, 1\}^6 || Q_i\}$. Finally, the edge features $E_{i,j}$ encode the well-depth of the Lennard-Jones 6-12 interaction $\epsilon_{i,j}$ between beads N_i and N_j .

1.3.3 Encoder

To encode a coarse-grained molecular graph $G^{(k)}$ into a fixed-size latent code $\text{Encoder}(G^{(k)}) = \mathbf{z}^{(k)} \in \mathbb{R}^d$ we use a message passing neural network (MPNN) architecture similar to that originally employed by Glimer *et al.*²⁹ to predict quantum mechanical properties of small organic molecules. First, a learned embedding layer $\Xi_\theta : \{0, 1\}^6 \mapsto \mathbb{R}^{d_e}$ transforms the one-hot identity of each bead type (one of $\{\text{T1}, \text{T2}, \text{T3}, \text{T4}, \text{T5}, \text{Q0}\}$) into a d_e -dimensional dense vector. Embedding layers are frequently used in the context of natural language processing and word embeddings as they offer the ability to derive more complex relations between categorical data. The embedding layer representation $\Xi_\theta(N_i)$ for each node i is then concatenated with a $Q_i \in \{+1, 0, -1\}$ scalar specifying the bead charge on node i and is then processed with a fully connected layer $\Omega_\theta : \mathbb{R}^{d_e+1} \mapsto \mathbb{R}^{d_h}$ that assigns an initial hidden state $\Omega_\theta([\Xi_\theta(N_i)||Q_i]) = h_i^{t=0} \in \mathbb{R}^{d_h}$ to each node i . We then perform message passing by successively updating the hidden states $h_i^{(t+1)}$ for each node using information accumulated from neighboring nodes in the form of messages $m_i^{(t+1)}$:

$$m_i^{(t+1)} = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \Psi_\Theta(E_{i,j}) \cdot h_j^t \quad (2)$$

$$h_i^{(t+1)} = \text{GRU}(m_i^{(t+1)}, h_i^t), \quad (3)$$

where \cdot represents matrix multiplication, $\mathcal{N}(i)$ is the set of nodes connected to node i , GRU is a Gated Recurrent Unit³⁰ cell as used in the gated graph neural network variant of message passing³¹, and $\Psi_\Theta : \mathbb{R}^1 \mapsto \mathbb{R}^{d_h \times d_h}$ is a multi-layer perceptron (MLP) that maps each edge feature $E_{i,j}$ into a $d_h \times d_h$ dimensional matrix. After total of $T = 5$ rounds of message passing is performed where Eqn. 2 and Eqn. 3 are successively applied. Following this message passing phase, we use the set2set model of Vinyals *et al.*³² as a global pooling operator to aggregate the terminal hidden states $\{h_i^T\}_{i \in G^{(k)}}$ to yield a permutationally invariant graph-level output:

$$h_{G^{(k)}} = \text{set2set}(\{h_i^T\}_{i \in G^{(k)}}). \quad (4)$$

Finally, another MLP Φ_Θ processes this representation $h_{G^{(k)}}$ to finally yield the latent representation $\mathbf{z}^{(k)} \in \mathbb{R}^d$ of the coarse-grained structure within our embedded chemical space.

$$\mathbf{z}^{(k)} = \Phi_\Theta(h_{G^{(k)}}). \quad (5)$$

1.3.4 Decoder

The purpose of the decoder is to reconstruct the composition matrix $\hat{N}^{(k)}$ and adjacency matrix $\hat{A}^{(k)}$ of compound k from the latent code $\mathbf{z}^{(k)} = \text{Encoder}(G^{(k)})$ derived from the encoder such that $(\hat{N}^{(k)}, \hat{A}^{(k)}) = \text{Decoder}(\mathbf{z}^{(k)})$. We use a partially auto-regressive model for the decoder where first the bead compositions $\hat{N}^{(k)}$ are reconstructed directly from the latent code $\mathbf{z}^{(k)}$ and subsequently the adjacency matrix $\hat{A}^{(k)}$, and optionally the edge features $\hat{E}^{(k)}$, are reconstructed conditioned on both the latent code $\mathbf{z}^{(k)}$ and the reconstructed composition $\hat{N}^{(k)}$. For our purposes we elect not to reconstruct the edge features $\hat{E}^{(k)}$ because the identity of each compound is fully specified by the collection of bead identities and their connectivity alone.

Two MLPs ϕ_B and ϕ_C are used to recover the reconstructed composition $\hat{N}^{(k)}$ from the latent code $\mathbf{z}^{(k)}$. The MLP $\phi_B : \mathbb{R}^d \mapsto \mathbb{R}^{5 \times 6}$ maps the latent code $\mathbf{z}^{(k)} \in \mathbb{R}^d$ to a matrix where each row represents softmax probabilities for each of the bead types (one of $\{\text{T1}, \text{T2}, \text{T3}, \text{T4}, \text{T5}, \text{Q0}\}$). The MLP $\phi_C : \mathbb{R}^d \mapsto \mathbb{R}^5$ predicts a vector of bead charges $\hat{Q}^{(k)} = \phi_C(\mathbf{z}^{(k)})$ such that $\hat{Q}_i^{(k)} \in [-1, +1]$ is the predicted charge of node i in molecule k . The concatenation of ϕ_B and ϕ_C defines the reconstructed composition matrix $\hat{N}^{(k)}$ sufficient for predicting the identity of each input bead (bead type and charge):

$$\hat{N}^{(k)} = [\phi_B(\mathbf{z}^{(k)})||\phi_C(\mathbf{z}^{(k)})]. \quad (6)$$

A function ϕ_A uses the i, j rows of the reconstructed composition matrix $\hat{N}_i^{(k)}, \hat{N}_j^{(k)}$, and the latent code $\mathbf{z}^{(k)}$ to assign entries for each component of the reconstructed adjacency matrix $\hat{A}_{i,j}^{(k)}$,

$$\hat{A}_{i,j}^{(k)} = \phi_A(\{\hat{N}_i^{(k)}, \hat{N}_j^{(k)}\}, \mathbf{z}^{(k)}). \quad (7)$$

As we know that the output adjacency matrix must be symmetric, we can construct ϕ_A such that the function is invariant to the order the $\hat{N}_i^{(k)}, \hat{N}_j^{(k)}$ to explicitly enforce that $\hat{A}_{i,j}^{(k)} = \hat{A}_{j,i}^{(k)}$. Three separate MLPs $\phi_A^{(i)}$, $\phi_A^{(ii)}$, and $\phi_A^{(iii)}$ are used to compose the function ϕ_A . First, $\phi_A^{(i)} : \mathbb{R}^7 \mapsto \mathbb{R}^{d_n}$ is used to transform the reconstructed node features $\hat{N}_i^{(k)}$ into a d_n -dimensional intermediate representation while the MLP $\phi_A^{(ii)} : \mathbb{R}^d \mapsto \mathbb{R}^{d_z}$ similarly transforms the latent code $\mathbf{z}^{(k)}$ into a d_z -dimensional intermediate representation. The MLP $\phi_A^{(iii)} : \mathbb{R}^{d_n+d_z} \mapsto \mathbb{R}^1$ then combines the concatenation of the outputs of $\phi_A^{(i)}$ and $\phi_A^{(ii)}$ to predict a single scalar value for each adjacency matrix entry $\hat{A}_{i,j}^{(k)} = \phi_A^{(iii)}([\frac{\phi_A^{(i)}(\hat{N}_i^{(k)}) + \phi_A^{(ii)}(\hat{N}_j^{(k)})}{2} || \phi_A^{(ii)}(\mathbf{z}^{(k)})])$. By additive construction of $\phi_A^{(i)}(\hat{N}_i^{(k)}) + \phi_A^{(ii)}(\hat{N}_j^{(k)})$ as an input we ensure that ϕ_A remains invariant to the order of $\hat{N}_i^{(k)}$ and $\hat{N}_j^{(k)}$, therefore explicitly enforcing the reconstructed adjacency matrix is symmetric such that $\hat{A}_{i,j}^{(k)} = \hat{A}_{j,i}^{(k)}$. Although not done here, the edge features $\hat{E}_{i,j}^{(k)}$ can analogously be predicting by specifying another function $\hat{E}_{i,j}^{(k)} = \phi_E(\hat{N}_i^{(k)}, \hat{N}_j^{(k)}, \mathbf{z}^{(k)})$.

The overall action of the decoder is therefore to reconstruct the composition matrix and adjacency matrix from the latent code $(\hat{N}^{(k)}, \hat{A}^{(k)}) = \text{Decoder}(\mathbf{z}^{(k)})$ such that the reconstructions fully specify the identity of the coarse-grained molecule used to originally derive the latent code from the input graph structured representation $\mathbf{z}^{(k)} = \text{Encoder}(G^{(k)})$. The next section describes the procedures used to train the RAE neural network from these reconstructions and inputs.

1.3.5 Training

The training objective \mathcal{L} consists of three terms: the reconstruction loss \mathcal{L}_{REC} , the RAE loss \mathcal{L}_{RAE} , and the regularization \mathcal{L}_{REG} . In some cases of graph reconstruction information contained in the edge attributes $E^{(k)}$ may be necessary for complete reconstruction, for example in defining the bond order in atomic graphs. In our case, however, only the composition $\hat{N}^{(k)}$ and adjacency matrix $\hat{A}^{(k)}$ are alone sufficient to reconstruct the identity of the input compound and thus we only consider contributions from those terms in the loss as we find empirically this yields higher fidelity reconstructions. The reconstruction loss \mathcal{L}_{REC} for a single training sample is given by:

$$\begin{aligned} \mathcal{L}_{REC} = \operatorname{argmin}_{\pi} & \left[\sum_{i=1}^5 \left[-\log\left(\frac{\exp((P_{\pi}\hat{N}^{(k)})_{i,j^*})}{\sum_{j=1}^6 \exp((P_{\pi}\hat{N}^{(k)})_{i,j})}\right) + ((P_{\pi}\hat{N}^{(k)})_{i,7} - Q_i^{(k)})^2 - \right. \right. \\ & \left. \left. \sum_{j=1}^5 (A_{i,j}^{(k)} \log((P_{\pi}\hat{A}^{(k)}P_{\pi}^T)_{i,j}) + (1 - A_{i,j}^{(k)}) \log(1 - (P_{\pi}\hat{A}^{(k)}P_{\pi}^T)_{i,j})) \right] \right] \end{aligned} \quad (8)$$

Where P_{π} are all valid permutation matrices of size five indexed by π and j^* indicates the correct class label for the bead identity (one of {T1, T2, T3, T4, T5, Q0}) for the given node i determined from the ground truth input $N_i^{(k)}$. The first term in Eqn. 8 is a categorical cross entropy for predicting the correct bead type from the first six columns of $\hat{N}^{(k)}$ to match the one-hot representation of the ground truth bead type (one of {T1, T2, T3, T4, T5, Q0}), the second term is a mean squared error between the ground truth charge $Q_i^{(k)}$ and the predicted bead charge $\hat{N}_{i,7}^{(k)} = \hat{Q}_i^{(k)} \in [-1, +1]$ given by the final column of each \hat{N}_i row, and the third term is a binary cross entropy between the reconstructed adjacency matrix $\hat{A}^{(k)}$ and the ground truth $A^{(k)}$. By explicitly enumerating all valid node permutations π for each reconstruction and selecting the argmin we can ensure that our decoder, and hence our entire network, remains permutationally invariant to the ordering of the nodes. Also, while in this framework the edge attributes are not explicitly learned, the network can nonetheless use

information encoded in the edge attributes to help shape the embedding and improve accuracy. The RAE loss \mathcal{L}_{RAE} takes the form,

$$\mathcal{L}_{RAE} = \frac{1}{2} \|\mathbf{z}^{(k)}\|_2^2 = \frac{1}{2} \sum_{i=1}^d (\mathbf{z}_i^{(k)})^2, \quad (9)$$

where $\mathbf{z}^{(k)} \in \mathbb{R}^d$ is the latent code produced by the encoder $\mathbf{z}^{(k)} = \text{Encoder}(G^{(k)})$. Intuitively, this term serves to restrict the size of the latent space helping to prevent unbounded optimization. Lastly, the \mathcal{L}_{REG} term regularizes the decoder to promote a ‘‘smooth’’ embedding. Comparatively, this regularization is implicitly done when training variational autoencoders (VAE)²⁶ through the injection of noise into the decoder via the reparameterization trick, but ultimately the purpose of this regularization is to enforce that points near one another in the latent space should be decoded to similar structures in the output data space. A number of regularization techniques such as weight decay, spectral normalization, or gradient penalty have shown to be effective regularizers in RAE training by enforcing the decoder have a bounded Lipschitz constant.²⁵ In this work we opt to use an L2 norm weight decay on the parameters of the decoder as our regularizer of choice,

$$\mathcal{L}_{REG} = \mathcal{L}_{L2} = \|W_{\text{Decoder}}\|^2. \quad (10)$$

The complete loss \mathcal{L} which is minimized during training takes the final form,

$$\mathcal{L} = \mathcal{L}_{REC} + \lambda_{RAE} \mathcal{L}_{RAE} + \lambda_{REG} \mathcal{L}_{REG}. \quad (11)$$

The RAE is trained end-to-end with mini-batch gradient descent using the Adam optimizer.³³ Table 1 provides a full list of hyperparameters and specifications for the different neural network components. The PyTorch framework was used to develop, train, and build the RAE model.³⁴ A high-level illustration of the neural network architecture and data flow for our RAE is shown in Fig. S10.

We qualitatively validate our latent space embedding represents a smooth encoding of our coarse-grained topology space by performing latent space interpolation in Fig. S11 and confirming that chemically similar structures are encoded as proximate points within the latent space. Further investigations into the latent space are shown in Fig. S12 displaying the latent space locations of various coarse-grained typologies and in Fig. S13, Fig. S14, Fig. S15, Fig. S16 and Fig. S17 showing the latent space color-coded by different graph-related properties.

Table 1: Regularized autoencoder neural network hyperparameters and specifications. Specifications for MLP structure is defined by a sequence of operations performed on an input. For example, [Linear(10, 50), Leaky_ReLU, Linear(50, 100)] signifies a linear layer transforms a 10-dimensional input into a 50-dimensional output that is then acted upon by a Leaky_ReLU non-linearity and lastly a linear layer transforms these activations into the final 100-dimensional output.

Parameter	Description	Value/Specification
d	Latent space dimension	16
d_e	Intermediate dimension of the bead-type embedding layer within the encoder	16
d_h	Intermediate hidden dimension in the encoder layers	64
d_n	Intermediate hidden dimension of transformed reconstructed node features in the decoder layers	256
d_z	Intermediate hidden dimension of transformed latent space representation in the decoder layers	256

Ω_{Θ}	Layer within the encoder that transforms embedded node features concatenated with a scalar bead charge $[\Xi_{\Theta}(N_i) Q_i]$ into the initial hidden states $h_i^{t=0}$	[Linear(d_e+1, d_h), Leaky_ReLU]
Ψ_{Θ}	MLP within the encoder that transforms edge features $E_{i,j}$ into a $d_h \times d_h$ representation	[Linear(1,128), Leaky_ReLU, Linear(128, $d_h \times d_h$)]
Φ_{Θ}	MLP within the encoder that transforms output of set2set global pooling operator $h_{G^{(k)}}$ into latent space representation $\mathbf{z}^{(k)}$	[Linear($2d_h, d_h$), Leaky_ReLU, Linear(d_h, d)]
ϕ_B	MLP within the decoder that transforms the latent space representation $\mathbf{z}^{(k)}$ into reconstructed bead-types defining the first six columns in the reconstructed node features $\hat{N}^{(k)}$	[Linear($d, 128$), Leaky_ReLU, Linear(128, 256), Leaky_ReLU, Linear(256, 5×6), Sigmoid]
ϕ_C	MLP within the decoder that transforms the latent space representation $\mathbf{z}^{(k)}$ into reconstructed bead charges Q_i defining the last column in the reconstructed node features $\hat{N}^{(k)}$	[Linear($d, 128$), Leaky_ReLU, Linear(128, 256), Leaky_ReLU, Linear(256, 5), Tanh]
$\phi_A^{(i)}$	MLP within the decoder that transforms reconstructed node features \hat{N}_i into a d_n -dimensional intermediate representation	[Linear(7, 128), Leaky_ReLU, Linear(128, d_n), Leaky_ReLU]
$\phi_A^{(ii)}$	MLP within the decoder that transforms the latent space representation $\mathbf{z}^{(k)}$ into a d_z -dimensional intermediate representation	[Linear($d, 128$), Leaky_ReLU, Linear(128, d_z), Leaky_ReLU]
$\phi_A^{(iii)}$	MLP within the decoder that transforms the concatenation $[\frac{\phi_A^{(i)}(\hat{N}_i^{(k)}) + \phi_A^{(ii)}(\hat{N}_j^{(k)})}{2} \phi_A^{(ii)}(\mathbf{z}^{(k)})]$ into scalar-valued entries of the reconstructed adjacency matrix $\hat{A}_{i,j}$	[Linear($d_n + d_z, 128$), Leaky_ReLU, Linear(128, 256), Leaky_ReLU, Linear(256, 128), Leaky_ReLU, Linear(128, 1), Sigmoid]
Learning rate	Learning rate used by the Adam optimizer	3.5×10^{-4}
β_1, β_2	Coefficients used by the Adam optimizer for computing running averages of gradient and its square	$\beta_1=0.9, \beta_2=0.999$
Batch size	Mini-batch size used when performing mini-batch gradient descent	64
λ_{RAE}	Weight of the RAE loss term \mathcal{L}_{RAE}	2.0
λ_{REG}	Weight of the L2 weight decay weight in the regularization loss term \mathcal{L}_{REG} applied to the parameters of the decoder W_{Decoder}	0.1
Epochs	Number of training epochs performed	5000

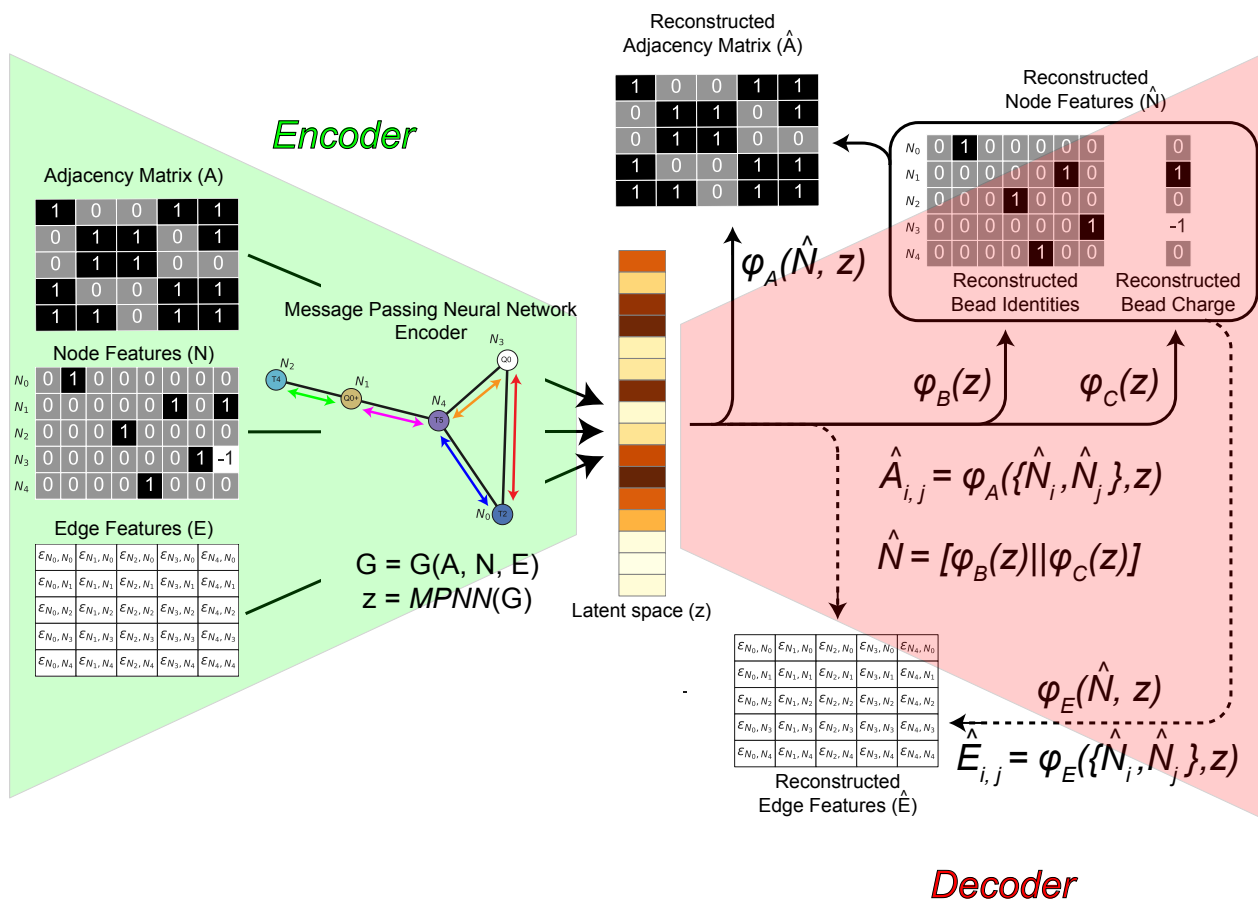


Figure S10: Schematic illustration of data flow in the Regularized AutoEncoder (RAE) neural network architecture used to generate our latent space embedding. Each compound k within our molecular design space is exposed to the RAE as a graph G represented jointly using a node-identity matrix N encoding the identity and charge of each coarse-grained bead, an adjacency matrix A capturing the connectivity of the beads in the graph, and an edge tensor E encoding edge features related to the Lennard-Jones 6-12 parameter in the coarse-grained force field. A message passing neural network (MPNN) encoder is used to process this graph-structured object $G = G(A, N, E)$ yielding the latent code $z = MPNN(G)$. This latent code z is then first used to reconstruct the node features by using two MLPs to separately reconstruct the bead identity and bead charge $\hat{N} = [\phi_B(z) || \phi_C(z)]$. The reconstructed adjacency matrix \hat{A} , and optionally the reconstructed edge features \hat{E} , are then predicted autoregressively using together the reconstructed node features \hat{N} and the latent code z : $\hat{A} = \phi_A(\hat{N}, z)$, $\hat{E} = \phi_E(\hat{N}, z)$. We can further enforce that the reconstructed adjacency matrix be symmetric ($\hat{A}_{i,j} = \hat{A}_{j,i}$) by ensuring the function ϕ_A remain invariant to the order of the rows N_i and N_j : $\hat{A}_{i,j} = \phi_A(\{\hat{N}_i, \hat{N}_j\}, z)$.

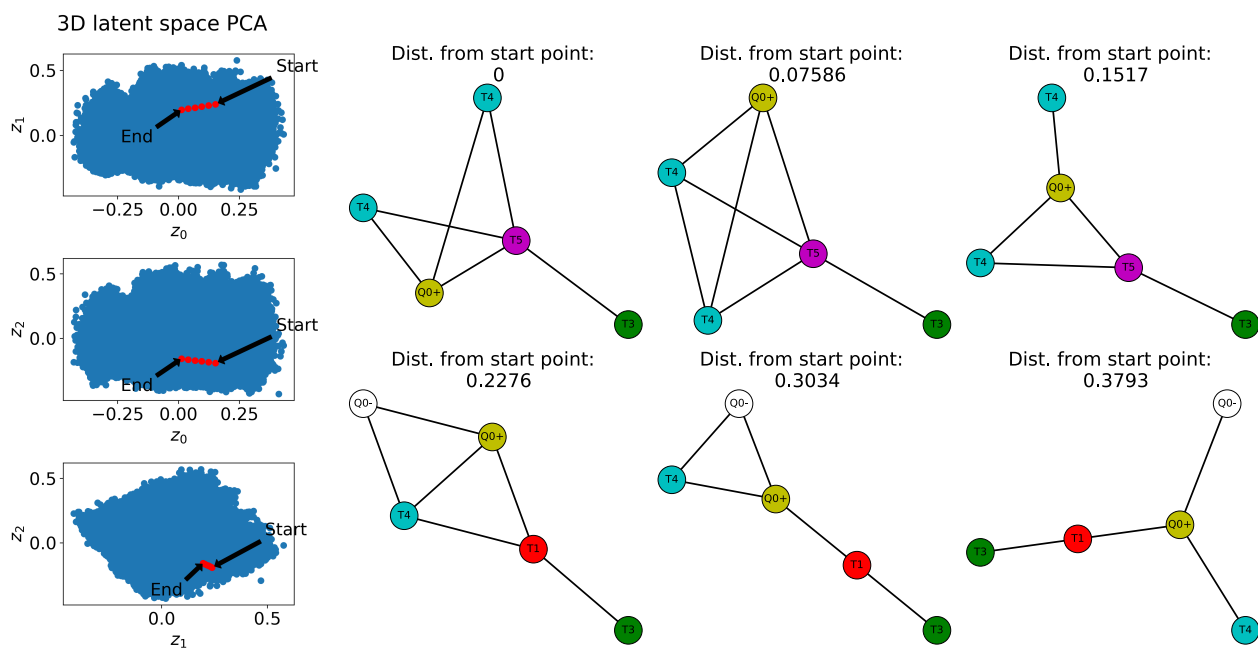


Figure S11: Latent space interpolation between embedded molecules. Two molecular latent space representations are chosen at random and a linear interpolation is performed within the full-dimensional latent space. The embedded molecular representation nearest to each interpolate is displayed alongside the Euclidean latent space distance to the starting molecule on the right. A three-dimensional PCA projection and the corresponding interpolation points is shown on the left. Proximate latent space embeddings are encoded as topologically similar coarse-grained compounds, a property illustrative of a smooth latent space embedding.

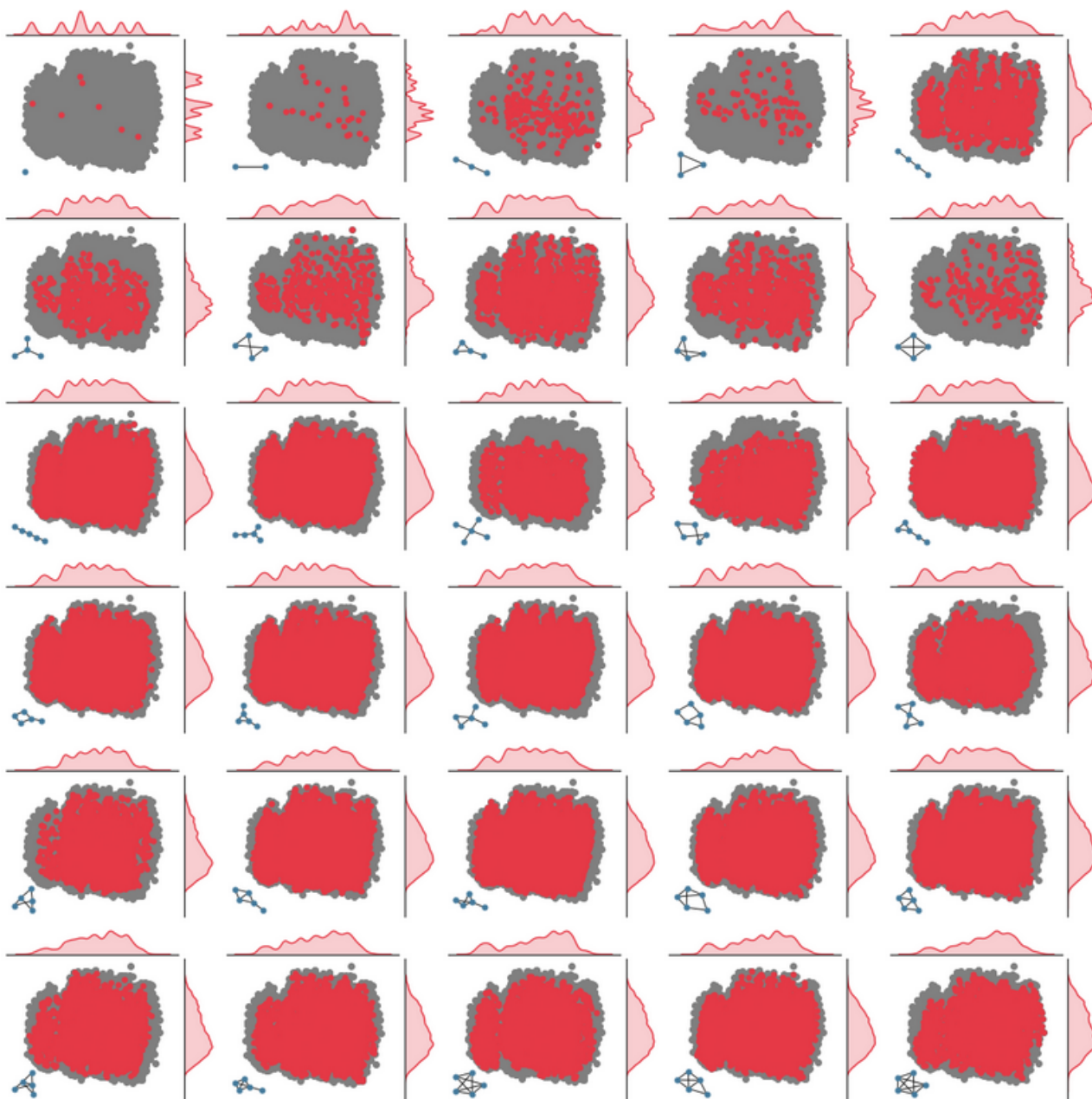


Figure S12: Locations within the latent space of different molecular topologies.

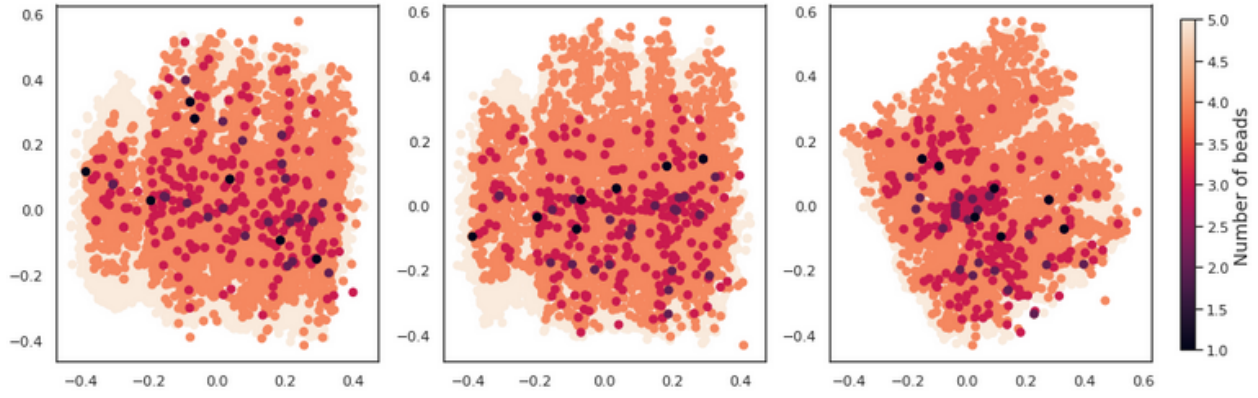


Figure S13: 3D PCA of latent space colored according to the number of beads within each graph.

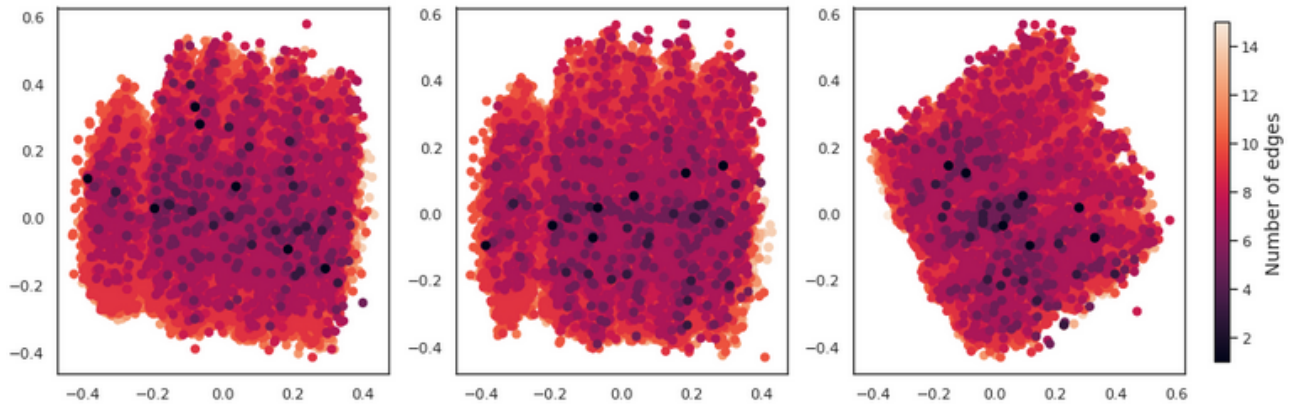


Figure S14: 3D PCA of latent space colored according to the number of edges within each graph.

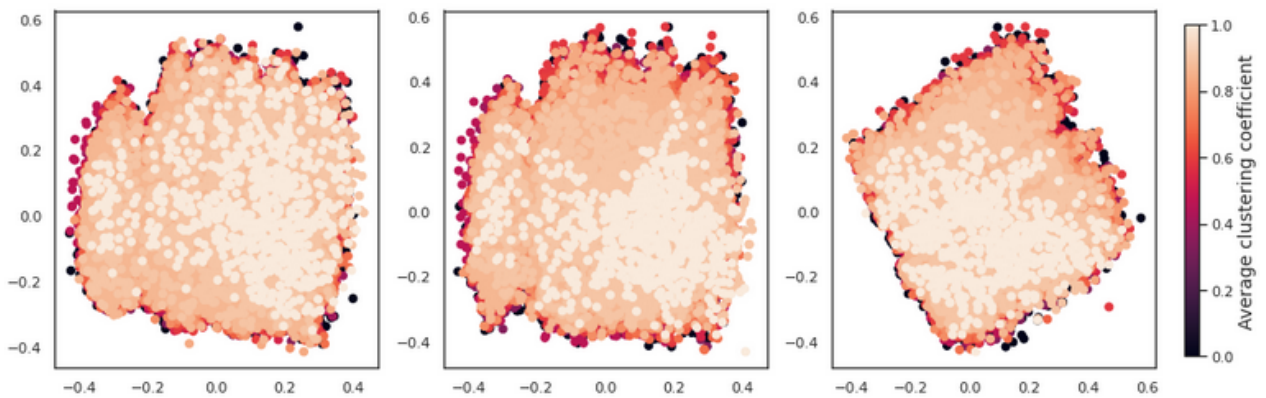


Figure S15: 3D PCA of latent space colored according to the average clustering coefficient of each graph.

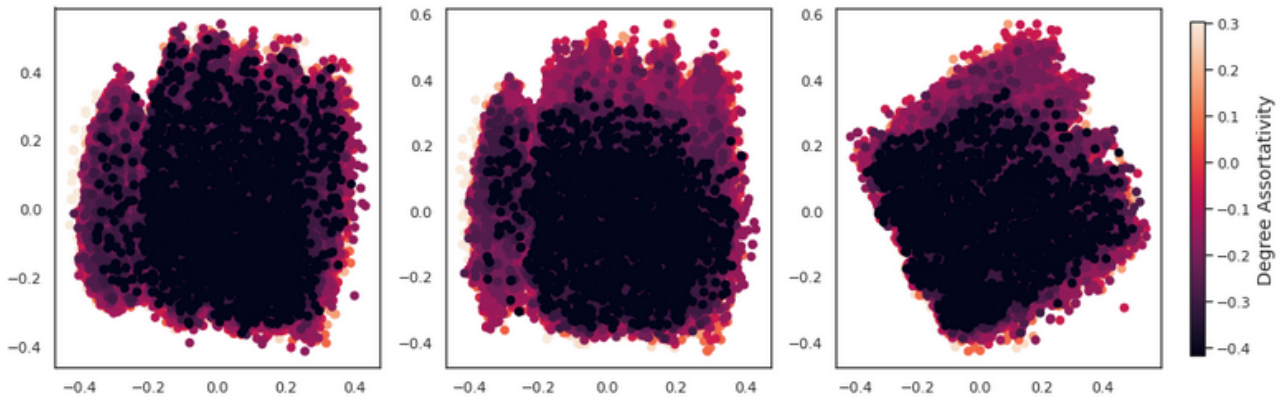


Figure S16: 3D PCA of latent space colored according to the degree assortativity of each graph.

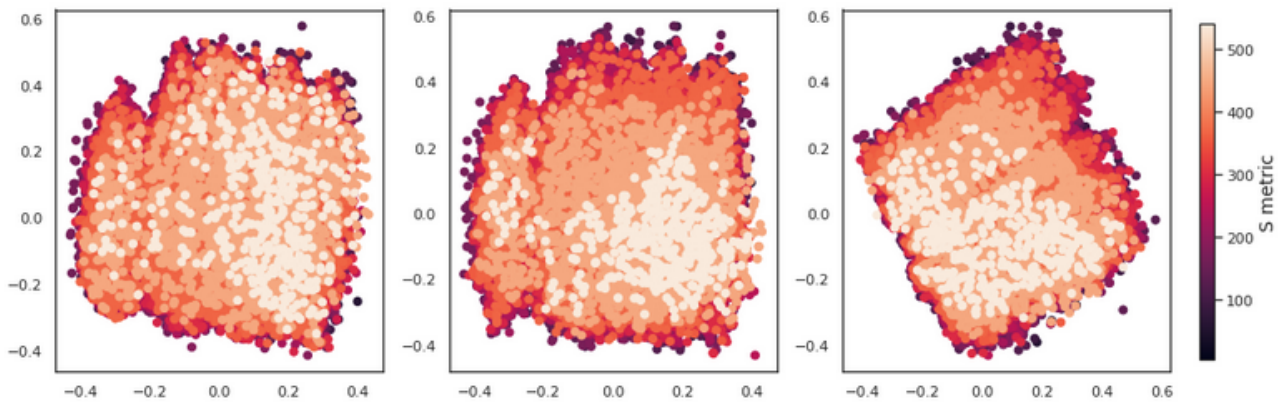


Figure S17: 3D PCA of latent space colored according to the S metric of each graph.

1.4 Active learning

Having defined a low-dimensional embedding of the available coarse-grained candidates and an objective function that solves the forward problem of evaluating the preferential selectivity of a permeant into CL-based membranes using alchemical free-energy calculations, we set out to navigate our chemical space to identify compounds with exceptionally high CL selectivity. As the evaluation for each molecule involves performing an expensive series of molecular dynamics simulations, it behooves us to minimize the number of such evaluations in the interest of time and computational cost. We choose to tackle this problem with active learning using Bayesian optimization, which has found popular use in molecular and drug design where it has been used, for example, to design self-assembling π -conjugated peptides³⁵ and optimize potency of cyclin-dependent kinase 2 inhibitors.³⁶

Active learning³⁷ is an adaptive framework wherein a Gaussian Process Regression (GPR) surrogate model is fit on a labeled subset of the design space to predict the fitness of the pool of untested candidates. These surrogate model predictions are then purposed to select the next most promising unsampled candidates to query using Bayesian optimization. Newly selected candidates are then evaluated for fitness, using a black-box non-differentiable forward model involving molecular simulations in this application, and incorporated back into the set of labeled data where the surrogate model is refit and new candidates are selected again. This process iteratively proceeds until a time/cost budget is reached, the pool of available candidates is exhausted, or some convergence criterion is met.

1.4.1 Selecting round 0 molecules

Each coarse-grained compound k in our design space is represented as a d -dimensional real-valued vector $\mathbf{z}^{(k)} \in \mathbb{R}^d$ embedded in the learned RAE latent space. Prior to beginning iterations in the active learning cycle we select a subset of molecules that are evaluated using our forward model to comprise our initial labeled data set. We call this initial set of molecules round 0.

The surrogate model used to suggest candidates in future rounds is conditioned on all the molecules for which the forward problem has been evaluated to date. Hence, our initial set of molecules should represent a broad and representative sampling of our design space to allow the GPR surrogate model to fit a better estimate of the property landscape. Randomly selecting these round 0 molecules from the pool of available candidates fails to take into account the data distribution in the embedded latent space, which can potentially lead to high density regions being under-sampled resulting in poor model performance and poor initial latent space coverage.

To ensure our round 0 compounds provide this broad and representative coverage of our embedded latent space we perform k-mean clustering to identify 100 centroids, using the k-means++ initialization scheme³⁸. Clustering is performed on the latent space embedding of all N molecules, each of dimension d , defining the data matrix $X \in \mathbb{R}^{N \times d}$ passed to the k-means algorithm. From the 100 identified centroids $\{\mathbf{c}_i \in \mathbb{R}^d\}_{i=1}^{100}$ the molecule closest to each centroid \mathbf{c}_i is selected as part of the round 0 dataset: $\{\arg\min_k \|\mathbf{z}^{(k)} - \mathbf{c}_i\|_2\}_{i=1}^{100}$. Free-energy calculations are performed on these 100 molecules to furnish our initial round 0 dataset.

1.4.2 Gaussian process regression

Compounds that have undergone alchemical free energy calculations to evaluate their fitness $y^{(k)} = f(\mathbf{z}^{(k)}) \forall k \in \{\text{sampled molecules}\}$ comprise a labeled subset of the full chemical space. In our present application $y^{(k)} = -\Delta\Delta G^{(k)}$ and we seek to maximize $y^{(k)}$ and (i.e., minimize $\Delta\Delta G^{(k)}$) in order to discover the most thermodynamically selective molecules for CL membranes. The remaining molecules $\mathbf{z}^{(k)} \forall k \notin \{\text{sampled molecules}\}$ are all potential candidates for future sampling. By training a surrogate model to predict the fitness of these not simulated candidates, along with an uncertainty to reflect the confidence in our predictions, we can direct sampling to profitable regions of chemical space by balancing selecting points with high predicted fitness (exploitation) and high uncertainty (exploration). We choose Gaussian Process Regression (GPR) as our surrogate model of choice for this supervised regression task, as it represents a non-parametric Bayesian regression technique that comes with built-in uncertainty estimates^{37,39-41}. The quality of these predictions will depend on both the quantity

of training data and “smoothness” of the latent space; the predictions will become more accurate as training data is accumulated round-by-round throughout active learning, while chemically similar molecules embedded near one another in the latent space helps promote a smooth property landscape with respect to the embedding amenable for optimization.

Given a set of n training data points $\mathbf{X} = \{(\mathbf{z}^{(k)}, y^{(k)})\}_{k=1}^n$ representing the collection of learned RAE latent space molecular representations $\{\mathbf{z}^{(k)}\}_{k=1}^n$ and their corresponding evaluated fitness measurements $\{y^{(k)} = f(\mathbf{z}^{(k)})\}_{k=1}^n$, we build a GPR to predict the fitness of all N molecules in our molecular design space $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(N)}\}$. We adopt the squared exponential kernel as our covariance function for the GPR

$$k(\mathbf{z}, \mathbf{z}') = \nu^2 \exp\left(\frac{-\|\mathbf{z} - \mathbf{z}'\|^2}{2\gamma}\right) \quad (12)$$

where the hyperparameter γ is the characteristic length scale of the kernel and ν^2 the marginal function variance. Fitting the GPR involves learning the parameters γ and ν^2 determined via maximum likelihood estimation optimizing the log marginal likelihood of the data with respect to the kernel hyperparameters⁴¹, making use of the scikit-learn⁴² GPR implementation.

Uncertainties inherent to each measurement $\sigma^{(k)}$ are incorporated through Tikhonov regularization⁴³ where $\boldsymbol{\sigma} = [\sigma^{(1)}, \sigma^{(2)}, \dots, \sigma^{(n)}]^T$ is added to the diagonal of the n -by- n covariance matrix K when calculating the predicted mean $\mu(\mathbf{z}_*)$ and uncertainty $\sigma(\mathbf{z}_*)$ of a point \mathbf{z}_* in our design space used to model each prediction as a Gaussian distributed variable $y_* = \mathcal{N}(\mu(\mathbf{z}_*), \sigma(\mathbf{z}_*))$

$$\mu(\mathbf{z}_*) = K(\mathbf{z}_*, \cdot)[K + \boldsymbol{\sigma}^T \mathbf{I}]^{-1} \mathbf{y} \quad (13)$$

$$\sigma^2(\mathbf{z}_*) = k(\mathbf{z}_*, \mathbf{z}_*) - K(\mathbf{z}_*, \cdot)[K + \boldsymbol{\sigma}^T \mathbf{I}]^{-1} K(\mathbf{z}_*, \cdot)^T \quad (14)$$

where $\mathbf{y} = [y^{(0)}, y^{(1)}, \dots, y^{(n)}]^T$, $K_{i,j} = k(\mathbf{z}^{(i)}, \mathbf{z}^{(j)})$, $K_{*,\cdot} = [k(\mathbf{z}_*, \mathbf{z}^{(1)}), k(\mathbf{z}_*, \mathbf{z}^{(2)}), \dots, k(\mathbf{z}_*, \mathbf{z}^{(n)})]$, $K_{*,*} = k(\mathbf{z}_*, \mathbf{z}_*)$, \mathbf{I} is a n -by- n identity matrix, and $*$ denotes a test point (i.e., not part of the training set). Ultimately, this surrogate model serves to bypass expensive direct calculation in favor of cheaply available predictions, eliminating the need to exhaustively simulate all molecules in our design space.

To help provide some intuition into the GPR we show in Fig. S18 and Fig. 1.4.2 the terminal GPR posterior mean μ and standard deviation σ projected onto the leading PCA coordinates of our latent space. A correlation plot between the terminal GPR predicted and ground truth calculated $\Delta\Delta G$ values for all inter-facial molecules is also presented in Fig. S20.

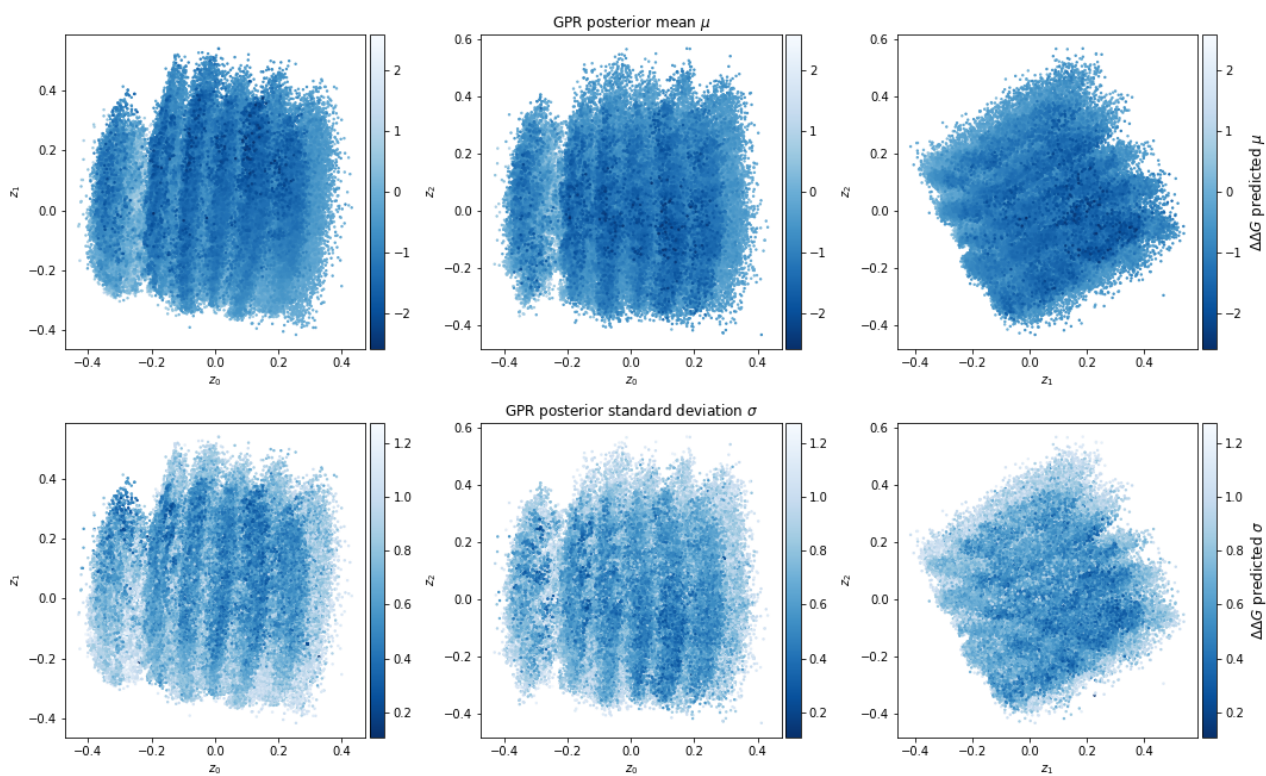


Figure S18: Predicted posterior mean μ (top) and uncertainty σ (bottom) from the terminal GPR projected onto the top 3 PCA coordinates.

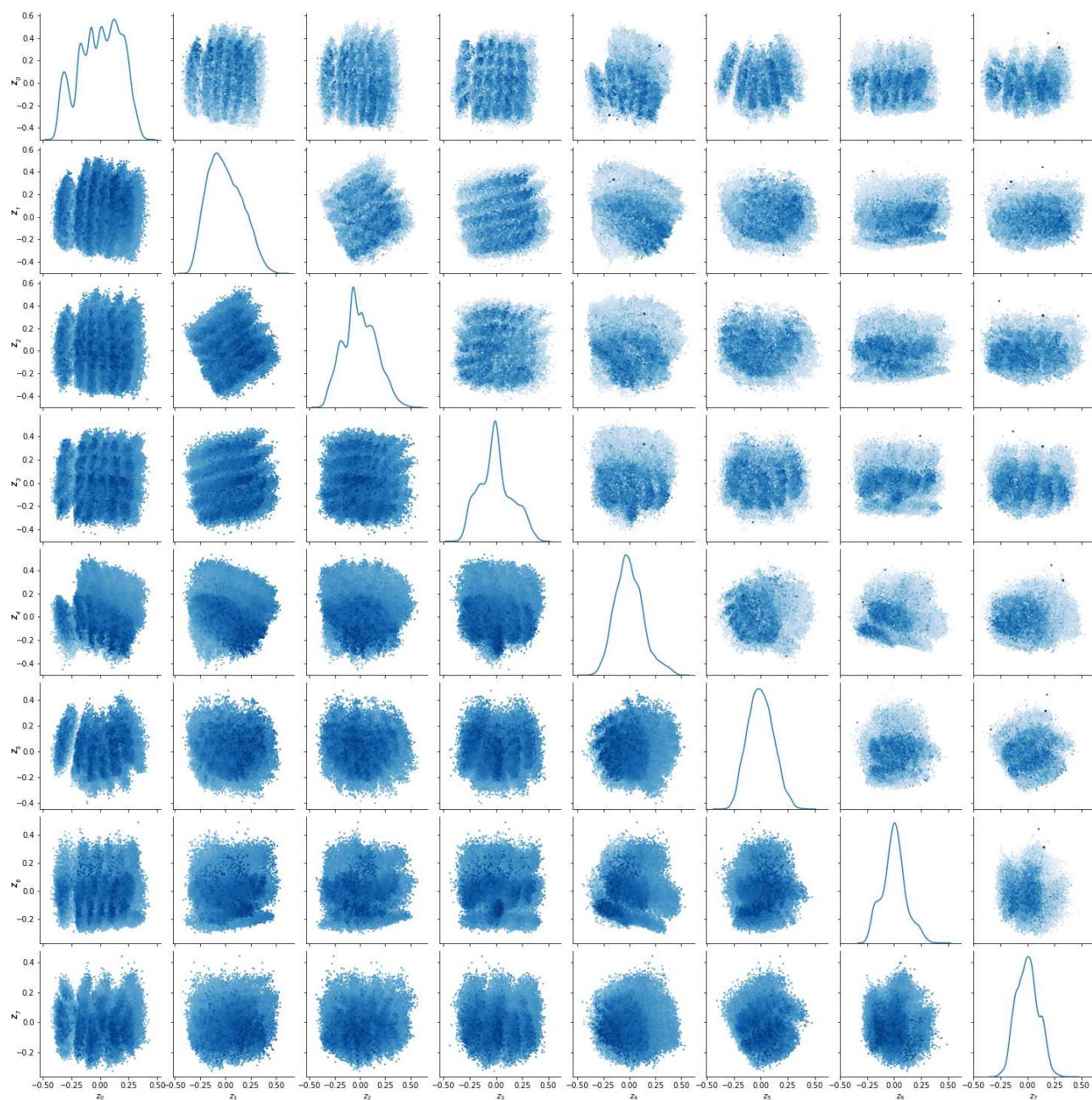


Figure S19: Predicted posterior mean μ and uncertainty σ from the terminal GPR projected onto the top 8 PCA coordinates ($\sim 81\%$ variance retained). The lower triangular half shows the predicted posterior mean μ , while the upper triangular half shows the predicted posterior standard deviations σ . The main diagonal shows the distribution of values for all embedded molecules within each PCA coordinate.

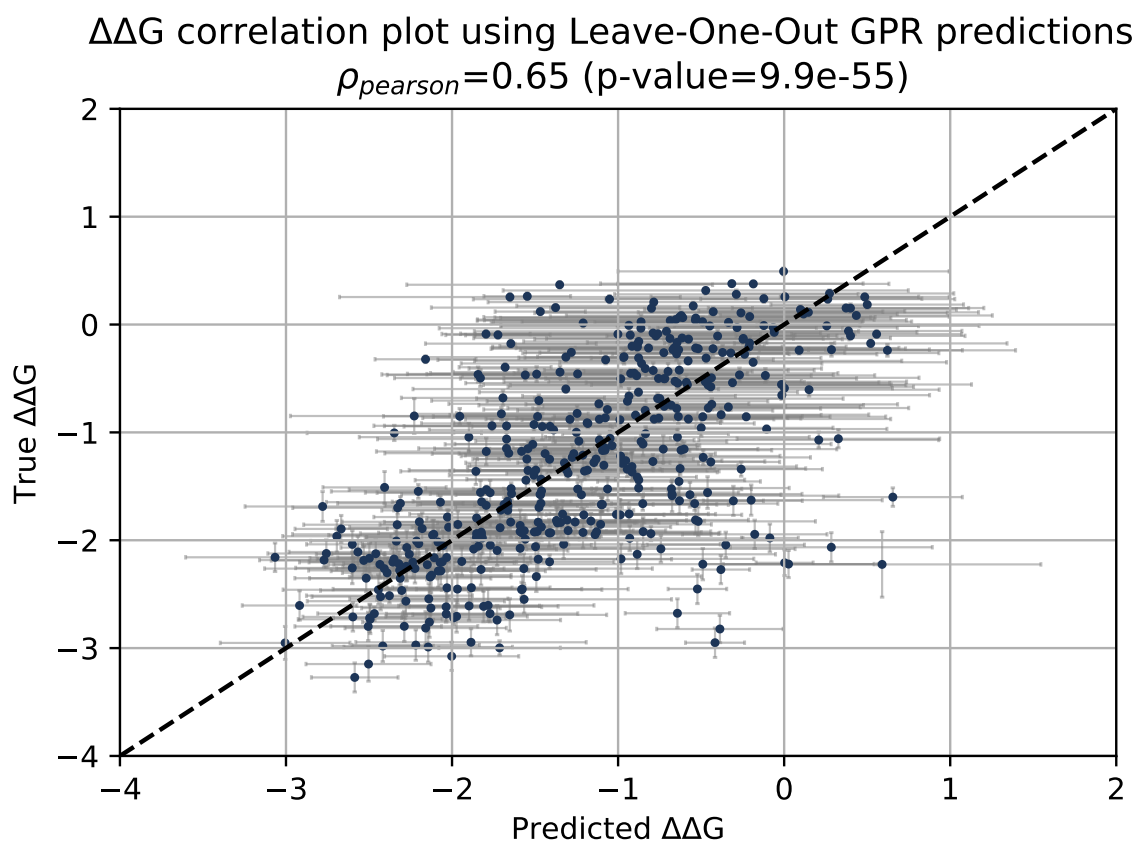


Figure S20: Correlation plot between calculated and predicted $\Delta\Delta G$ for all 439 interfacial molecules for which alchemical free energy calculations were conducted using leave-one-out cross-validation of the trained terminal GPT model. Error bars in the predicted $\Delta\Delta G$ correspond to uncertainties predicted by the GPR model, whereas those in the calculated $\Delta\Delta G$ denote the uncertainties in our free energies estimated by singular value decomposition included in the MBAR method.²¹

1.4.3 Bayesian optimization

Once we have fit a GPR surrogate model to predict the preferential insertion into CL-based membranes for each compound, we set out to navigate the design space by iteratively querying new coarse-grained compounds to simulate balancing both selecting candidates with high predicted performance (so called, exploitation) and potentially good candidates with high predicted uncertainty (exploration). Bayesian optimization (BO) provides a robust black-box optimization framework that purposes the posterior mean and uncertainties provided by the GPR predictions to suggest candidates that are most effective to test next which strike this explore-exploit balance.

1.4.4 Acquisition function

Bayesian optimization is capable of performing optimization of expensive black-box functions by optimizing the surrogate objective defined by the GPR through an acquisition function which determines the next best candidates based on their predicted fitness. Of the various acquisition functions available³⁷, we elect to use the popular Expected Improvement (EI) acquisition function that balances exploitation and exploration to guide sampling towards candidates predicted to possess the maximum expected gains in fitness. Mathematically, the EI acquisition function takes the form,

$$EI(\mathbf{z}_*) = \begin{cases} (\mu(\mathbf{z}_*) - f(\mathbf{z}^+) - \xi)\Phi(Z) + \sigma(\mathbf{z}_*)\phi(Z) & \sigma(\mathbf{z}_*) > 0 \\ 0 & \sigma(\mathbf{z}_*) = 0 \end{cases}, \quad (15)$$

$$Z = \begin{cases} \frac{\mu(\mathbf{z}_*) - f(\mathbf{z}^+) - \xi}{\sigma(\mathbf{z}_*)} & \sigma(\mathbf{z}_*) > 0 \\ 0 & \sigma(\mathbf{z}_*) = 0 \end{cases}, \quad (16)$$

where $f(\mathbf{z}^+)$, $\mathbf{z}^+ \in \{\mathbf{z}^{(1)}, \mathbf{z}^{(1)}, \dots, \mathbf{z}^{(n)}\}$ is the highest fitness value of all candidate molecules sampled to date, Φ and ϕ are the standard normal distribution cumulative distribution function and probability density function, and ξ is a parameter that controls the trade-off between exploration and exploitation. Exploitation and exploration are controlled by the first and second terms of Eqn. 15, respectively: small values of ξ will encourage exploitation by selecting candidates with higher predicted mean, while exploration is driven by large values of ξ promoting selection of candidates with high posterior uncertainty³⁷. The candidate point \mathbf{z}^\dagger that maximizes the acquisition function $\mathbf{z}^\dagger = \operatorname{argmax}_k EI(\mathbf{z}^{(k)}; \mathbf{z}^{(k)} \notin \{\mathbf{z}^{(1)}, \mathbf{z}^{(1)}, \dots, \mathbf{z}^{(n)}\})$ is selected as the next best candidate to sample in the available design space. Throughout this work we adopt the fixed value of $\xi = 0.01$ as suggested in the literature.^{37,44}

1.4.5 Batched sampling

In typical sequential sampling at each BO iteration the EI acquisition function is evaluated on all untested candidates and the compound that maximizes the EI acquisition is selected for evaluation for the next round. A limitation of this strategy lies in only being able to evaluate one candidate each round. Batched sampling techniques enable us to utilize parallel compute resources by selecting multiple candidates in each round. From an information theoretic perspective this introduces inefficiency, since the model does not get the benefit of reincorporating new measurements on each newly sampled point in a sequential fashion. From a temporal perspective, however, we expedite sampling of the design space by considering multiple candidates per round. In this work, we elect to use the Kriging Believer⁴⁵ method as our batched sampling strategy.

Given the set of labeled data at any particular instant in the active learning process $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(n)}\}$, the Kriging Believer algorithm selects a batch of q candidates by first selecting the candidate $\mathbf{z}^{(n+1)} = \operatorname{argmax}_k EI(\mathbf{z}^{(k)}; \mathbf{z}^{(k)} \notin \{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(n)}\})$ that maximizes the EI acquisition function. The predicted posterior mean $\mu(\mathbf{z}^{(n+1)})$ and uncertainty $\sigma(\mathbf{z}^{(n+1)})$ of the selected point $\mathbf{z}^{(n+1)}$ is then appended to the training dataset and the surrogate model is refit to the now extended training dataset $\{\mathbf{z}^{(1)}, \mathbf{z}^{(1)}, \dots, \mathbf{z}^{(n)}, \mathbf{z}^{(n+1)}\}$. The next molecule is then selected as $\mathbf{z}^{(n+2)} = \operatorname{argmax}_k EI(\mathbf{z}^{(k)}; \mathbf{z}^{(k)} \notin \{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \dots, \mathbf{z}^{(n+1)}\})$. This process iteratively continues until all q compounds have been selected. All q candidates are then subjected to parallel screening as a batch by performing alchemical free energy

calculations. Once these calculations are complete, the labeled data set is augmented with the latent space representations and calculated fitness values $\{(\mathbf{z}^{(k)}, y^{(k)})\}_{k=1}^q$ of these q newly screened molecules. Based on our available parallel compute resources, we selected a batch size of $q = 60$.

1.4.6 Non-interfacial molecules

Each batch of compounds selected within each active learning cycle are passed onto testing through a series of molecular dynamics simulations and free-energy calculations. In an effort to expedite sampling, the free-energy workflow is terminated early for some compounds after the first stage if they are deemed unlikely to spontaneously insert into the lipid bilayer. Hence $\Delta\Delta G$ measurements for the preferential partitioning for these compounds are not available, however these compounds nevertheless are informative of which regions of latent space may be less profitable to explore. By assigning artificially unfavorable $\Delta\Delta G$ values to these non-interfacial compounds we can encourage the GPR to avoid assigning favorable posterior means and uncertainties predictions to these compounds and therefore better enable the EI acquisition function to select compounds in the vicinity of more profitable latent space regions. An artificially poor $\Delta\Delta G$ mean $\tilde{\mu}$ and uncertainty $\tilde{\sigma}$ are assigned to these non-interfacial compounds which are treated as hyperparameters during each active learning cycle. The non-interfacial uncertainty is fixed at $\tilde{\sigma} = 0.01$, chosen to be smaller than the smallest observed uncertainty of $\sigma_{min} \sim 0.012$. The assigned mean value $\tilde{\mu}$ is motivated by selecting a value larger than the largest observed $\Delta\Delta G$ value $\mu_{max} \sim 0.5$. By assigning artificially poor mean values $\tilde{\mu} > \mu_{max}$ we can discourage the acquisition function from selecting candidates near these non-interfacial compounds when operating in the exploitative regime, and correspondingly assigning artificially small uncertainties $\tilde{\sigma} < \sigma_{min}$ discourages selecting compounds due to exploration.

We show in Fig. S21 and Fig. S22 the latent space locations of all sampled interfacial and non-interfacial molecules simulated throughout our active learning screen projected into the leading PCA coordinates of the latent space embedding. Our active learning procedure effectively guides selection to regions of latent space dense with high-performing (low $\Delta\Delta G$) candidates, while deterring selection in regions of latent space consisting primarily of non-interfacial molecules.

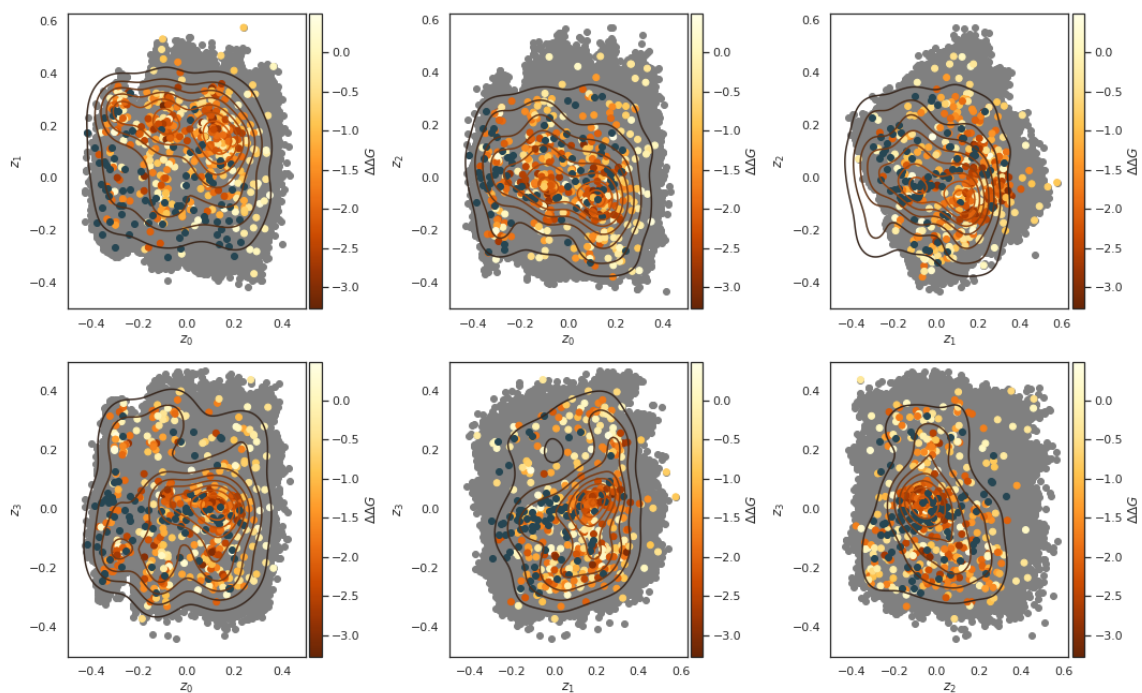


Figure S21: Locations of latent space representations of all sampled molecules projected into 4 dimensions using PCA and colored according to the measured $\Delta\Delta G$ values. Dark blue colored points represent molecules identified as non-interfacial. Select regions of latent space are more densely populated with high performing (i.e., low $\Delta\Delta G$) molecules.

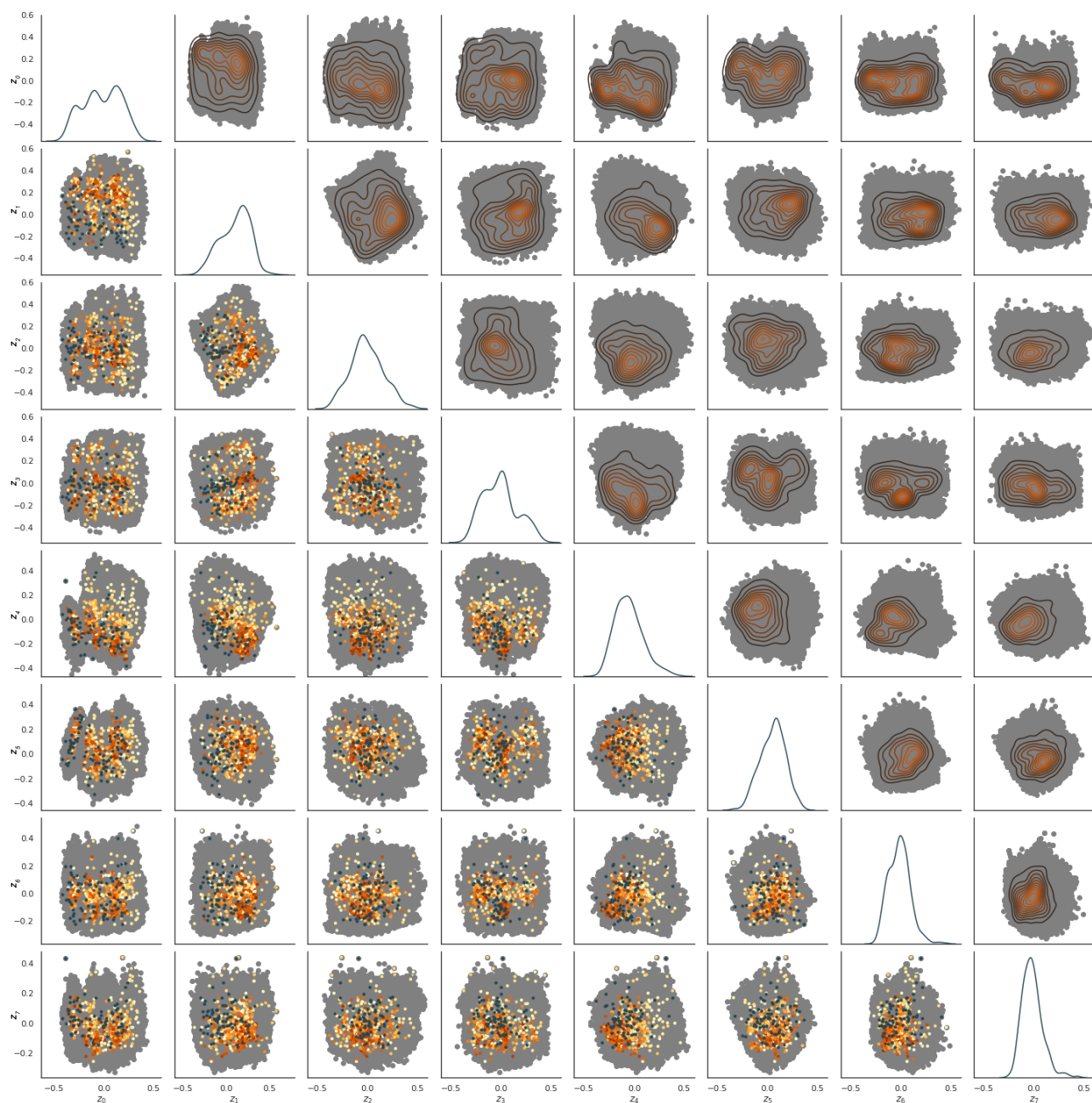


Figure S22: Locations of latent space representations of all sampled molecules projected into the top 8 PCA coordinates ($\sim 81\%$ variance retained). Dark blue points represent molecules identified as non-interfacial. The lower triangular half shows the individual latent space locations of all simulated candidate molecules, while the upper triangular half shows the density of all interfacial and non-interfacial molecules within the latent space. The main diagonal shows the distribution of values for only the interfacial and non-interfacial molecules along each latent space PCA coordinate.

1.5 Inference of design rules using LASSO regression

Having performed seven rounds of active learning we can proceed to interrogate our accumulated data to discover chemical motifs most predictive of $\Delta\Delta G$ using interpretable linear models employing graph representational learning. This technique was recently employed by Bhattacharjee *et al.*⁴⁶ to learn linear maps between small molecule thermochemical properties calculated at various levels of theory. Given our dataset of $N = 439$ interfacial compounds selected throughout our active learning procedure, we enumerate $k = 1608$ topologically unique subgraphs with 1 to 5 coarse-grained beads, which we find to be contained within these N compounds. Each coarse-grained structure n is then featurized according to the representation of different subgraphs $X_n \in \mathbb{R}^{k=1608}$, normalized to yield a per-compound subgraph frequency $\sum_{k=1}^{k=1608} X_{n,k} = 1$. However, because smaller subgraphs are contained within larger subgraphs the presence of larger subgraph motifs is correlated to the presence of smaller subgraphs. To account for these subgraph dependencies we borrow from Graphlet literature⁴⁷⁻⁵⁰ and reweight each subgraph frequency $X_{n,k}$ according to the weight $w_k = 1 - \frac{\log o_k}{\log 1608}$ where o_k is the number of subgraphs affected by subgraph k .⁴⁷ This re-weighted subgraph frequency $F_{n,k} = w_k X_{n,k}$ is finally once again normalized to unit length yielding a relative subgraph frequency $\sum_{k=1}^{k=1608} w_k X_{n,k} = \sum_{k=1}^{k=1608} F_{n,k} = 1$. These F_n features effectively capture the fundamental make-up of these coarse-grained topologies and provide a compact representation for downstream analysis and machine learning tasks.⁴⁶ Namely, we can use the features contained in the matrix $F \in \mathbb{R}^{n=439, k=1608}$ to train LASSO regression model on all our collected simulation data by minimizing the L1 regularized loss

$$L(\Delta\Delta G_{\text{predicted}}, \Delta\Delta G_{\text{true}}; \boldsymbol{\theta}, \alpha) = \frac{1}{2} (\Delta\Delta G_{\text{predicted}} - \Delta\Delta G_{\text{true}})^2 + \alpha \|\boldsymbol{\theta}\|_1, \quad (17)$$

$$\Delta\Delta G_{\text{predicted}}^{(n)} = \sum_{k=1}^{k=1608} \theta_k F_{n,k} + \theta_{\text{intercept}}. \quad (18)$$

where $\theta_{\text{intercept}}$ is the mean of all $N = 439$ $\Delta\Delta G$ values, α is the L1 regularization weight, and $\boldsymbol{\theta} \in \mathbb{R}^{k=1608}$ are the learned model parameters used to predict the transfer free energy for compound n . The L1 regularization parameter α controls the participation of nonzero elements in the weight vector $\boldsymbol{\theta}$. When $\alpha \rightarrow \infty$ the null model is returned where $\boldsymbol{\theta} = \mathbf{0}$ due to the predominance of the L1 regularization penalty in Eqn. 17. As α decreases in magnitude nonzero elements are progressively incorporated into the weight vector $\boldsymbol{\theta}$, meaning the participation of more features in F_n contributing to the regression task. The L1 loss in the LASSO regression algorithm prevents overfitting by retaining only a small number of the most generalizable features represented in our training dataset. Due to the linear structure of Eqn. 18 the sign of each nonzero weight θ_k can be interpreted as follows: for a learned weight $\theta_k < 0$, a high relative prevalence of the corresponding subgraph motif k within the feature vector F contributes to "more negative" $\Delta\Delta G$ predictions (favorable CL selectivity). Likewise, for a learned weight $\theta_k > 0$, a high relative prevalence of the corresponding subgraph motif k contributes to "more positive" $\Delta\Delta G$ predictions (unfavorable CL selectivity). Similarly, weights where $\theta_k \approx 0$ correspond to subgraph motifs that maintain relatively neutral cardiolipin selectivity. Because our goal is to minimize $\Delta\Delta G$ to better promote CL selectivity, identifying the subgraphs with negative learned weights θ_k corresponds to motifs most influential for good CL selectivity.

We determine the optimal number of features to retain in our model by performing cross-validation on the L1 regularization weight α presented in Fig. S23, suggesting that the lowest generalization error is achieved when $\alpha \approx 0.03$ corresponding to 126 nonzero coefficients and mean absolute error (MAE) for predicting $\Delta\Delta G$ of ~ 0.38 kcal/mol. The performance of this model is further quantified in Fig. S24 presenting a correlation plot between calculated and predicted $\Delta\Delta G$ using Leave One Out (LOO) cross-validation. The largest and smallest learned nonzero coefficient weights θ_k from this optimal model provide a rank-ordering for the subgraphs that correlate with motifs displaying favorable and unfavorable $\Delta\Delta G$ selectivity (Fig. 5). Our trained model can also be deployed to predict $\Delta\Delta G$ transfer free energies of coarse-grained compounds of potentially larger size than what was contained in our training dataset. Specifically, by constructing a feature vector for a target coarse-grained topology

based on representation of the $k = 1608$ enumerated subgraphs we can use Eqn. 18 to predict CL selectivity of coarse-grained compounds of arbitrary size.

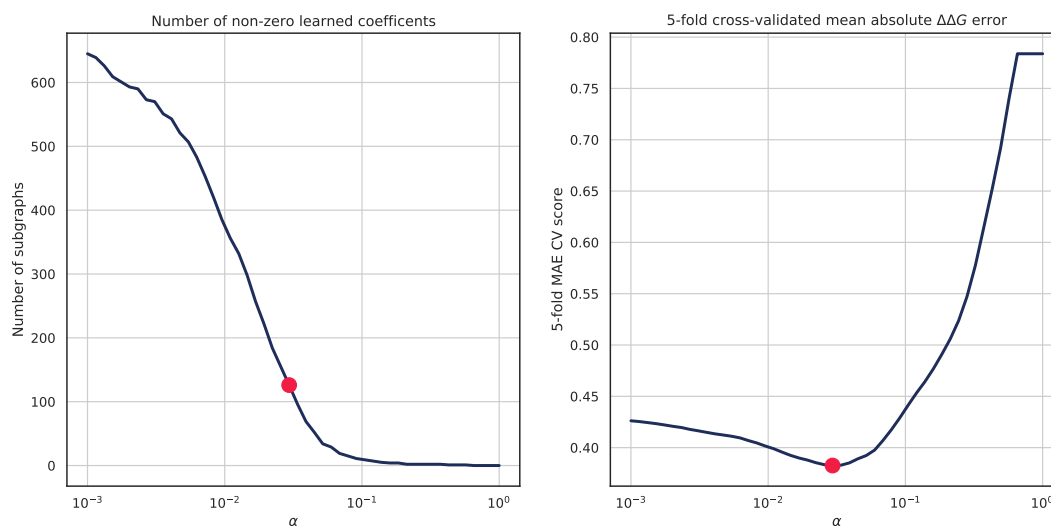


Figure S23: Performance of LASSO regression model as a function of the sparsity regularization parameter α . (left) The number of CG sub-graphs with non-zero learned coefficient values identified by training the LASSO regression model trained at each α value. (right) 5-fold cross validated MAE score of the LASSO regression model trained at each α value. The best generalization error of ~ 0.38 kcal/mol is achieved when $\alpha \approx 0.03$ where the model chooses to retain 126 nonzero coefficients.

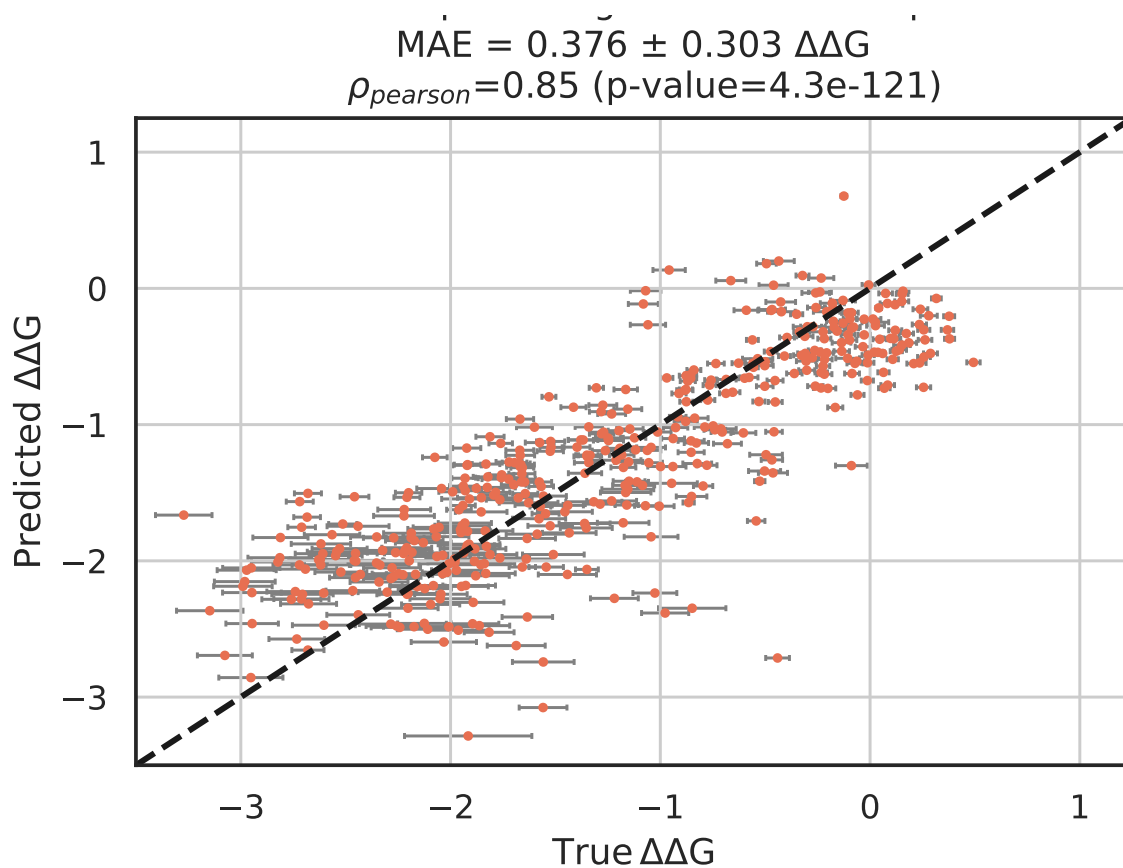


Figure S24: Correlation plot between true and predicted $\Delta\Delta G$ values for all 439 interfacial molecules for which alchemical free energy calculations were conducted using leave-one-out cross-validation predictions of the LASSO model at the optimal value of α chosen using 5-fold cross validation (cf. Fig. S23). Error bars in the predicted $\Delta\Delta G$ correspond to uncertainties predicted by the GPR model, whereas those in the calculated $\Delta\Delta G$ denote the uncertainties in our free energy calculations estimated by singular value decomposition.²¹

1.6 Functional group analysis

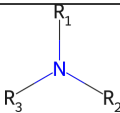
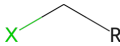

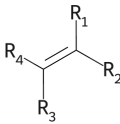
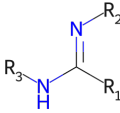
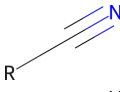
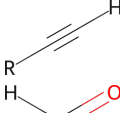
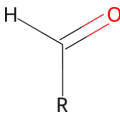
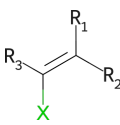
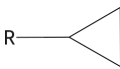
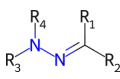
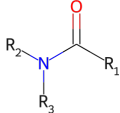
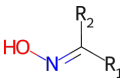
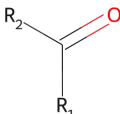
Coarse-grained beads represent an average over chemical and physical properties of contained substructures. To extract the encoded chemical information, we map functional groups identified on the atomistic level to CG beads attributed to the same positions in the small molecule. All-atom structures containing cycles require special treatment that we describe herein. Five-membered rings map to two *small* CG beads (S-beads), six-membered rings are modeled by three S-beads. For the data sets of molecules containing five- or six-membered rings, initially only Martini 2 parameterizations created with AutoMartini⁵¹ were present. For those cases, the Martini representations were transferred to the *5+0* model using a simple replacement scheme according to the water-octanol partitioning coefficients of the respective bead types, see Table 2. Ionizable groups were handled following the approach described in Sec. 1.6.1.

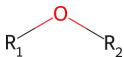
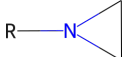
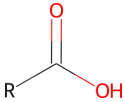
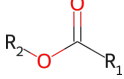
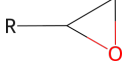
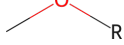
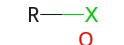
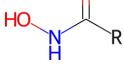
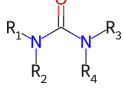
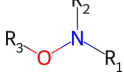
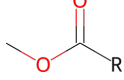
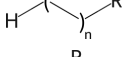
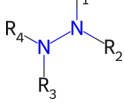

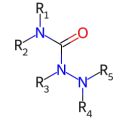
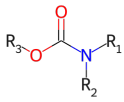
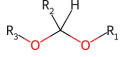
Table 2: Mapping Martini 2 bead types² to *5+0* bead types⁵ used to translate molecular representations present in the Martini force field to the *5+0* force field. Bead types part of the original publication of the *5+0* force field but not used in our work are set in cursive. Note the **N0** Martini bead type being mapped to a **T4** bead despite being considered nonpolar in Martini². This choice was made because the water-octanol partitioning coefficient for **N0** is given as $\Delta G_{W \rightarrow O1} = -1.00$ kcal/mol, which is closest to the $\Delta G_{W \rightarrow O1}$ of **T4** (-2.46 kcal/mol) in the subset of beads used here⁵.

Martini 2 Type	<i>5+0</i> Type	Polar/Nonpolar/Apolar, Donor/Acceptor
P5	T1	Polar
P4	T1	Polar
P3	T1	Polar
P2	T2	Polar
P1	T2	Polar
Nda	T3	Nonpolar Donor+Acceptor
<i>Nd</i>	<i>T3d</i>	<i>Nonpolar Donor</i>
<i>Na</i>	<i>T3a</i>	<i>Nonpolar Acceptor</i>
N0	T4	Apolar
C5	T4	Apolar
C4	T4	Apolar
C3	T5	Apolar
C2	T5	Apolar
C1	T5	Apolar

CG beads represent an average over the chemical and physical properties of all underlying heavy atoms. To extract the chemical information encoded by each bead type, we link functional groups automatically identified on the atomistic level using the Ertl algorithm⁵² to CG representations of small molecules. To account for the limited resolution of the CG model, the resolution of the identified functional groups was also deliberately reduced. For example primary, secondary and tertiary amines or alcohols were all labeled as *amine-* or *hydroxy-*groups. A list of the functional groups identified by this algorithm in our dataset is given in Table 3.

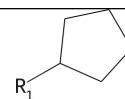
Table 3: Functional groups identified by the Ertl algorithm⁵² in the dataset of coarse-grained small molecules from the GDB.⁵³ To take into account the limited resolution of coarse-grained models, the functional groups and their chemical and physical properties are kept very generic. The section of molecules containing five- or six-membered rings were mainly used to investigate the influence of heteroatoms within the rings or substituents attached to rings on the choice of bead types in the coarse-graining process.

Functional group	Properties (R_x is any alkyl/aryl group)	Structure
Max. three-membered rings, structures represented by one CG bead		
amines	The basic nitrogen atom has one lone electron pair. Contains one or more substituents (R_x).	
alkyl halides	The carbon-halogen bond is polar with a partial positive charge on the carbon and a partial negative charge on the halogen.	
hydroxy group	Can form intermolecular hydrogen bonds. Can easily be deprotonated (high dipole moment between O and H).	
alkenes	C=C double bond. Apolar/hydrophobic, no rotation around double bond.	
amidines	Strong bases, but uncharged/ionized at physiological pH.	
cyano group	Polar (high dipole moment between C and N).	
alkynes (acetylenes)	Apolar/hydrophobic, C-C triple bond allows no bending or rotation.	
aldehydes	Polar (dipole moment between O and C). Can form hydrogen bonds.	
vinyl halide	Apolar/hydrophobic.	
cyclo-propyl	Apolar/hydrophobic, can interact with aromatic groups.	
hydrazones	Weak acids/bases, not ionized at physiological pH.	
amides	Polar (dipole moments between N and C and O and C). Weak base, not deprotonated under physiological pH. Hydrogen bond donors and acceptors.	
oximes	Neutral, hydrogen bond donor (N) and acceptor (O).	
ketones	Polar, hydrogen bond acceptor.	

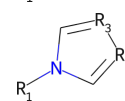
ethers	Nonpolar, hydrogen bond acceptor.	
aziridines	Polar, weak bases, not protonated at physiological pH. Hydrogen bond donor.	
carboxylic acids	Polar, hydrogen bond acceptors, deprotonated at physiological pH.	
carboxylic acid esters	Nonpolar, hydrogen bond acceptors. Acidic, not deprotonated around physiological pH.	
oxiranes	Hydrogen bond acceptors, polar.	
methoxy groups	Polar.	
halogen derivs.	Polar.	
hydroxamic acids	Polar, hydrogen bond donors and acceptors. Weak acid, not deprotonated at physiological pH.	
ureas	Polar. Acidic, not ionized at physiological pH. Hydrogen bond acceptors.	
hydroxylamines	Polar. Hydrogen bond donors and acceptors.	
methyl esters	Nonpolar. Hydrogen bond acceptor.	
alkanes	Apolar	
hydrazine derivs.	Polar. Forms hydrogen bonds. Basic, not protonated at physiological pH.	
carboxylic acid imides	Polar, acidic, hydrogen bond acceptor.	
semicarbazides	Polar. Hydrogen bond donors and acceptors.	
carbamic acid esters	Polar. Hydrogen bond donors and acceptors.	
acetals/ketals	Polar. Hydrogen bond acceptors.	

Five-membered rings, represented by two CG beads

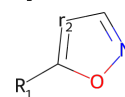
cyclopentyls

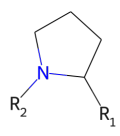
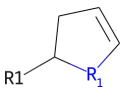
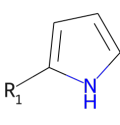
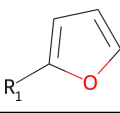
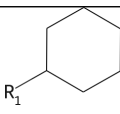
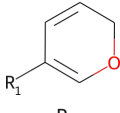
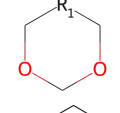
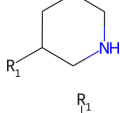
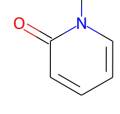
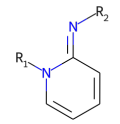
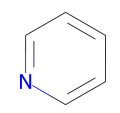

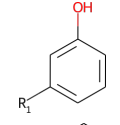
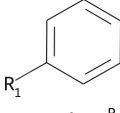
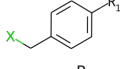
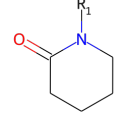


azoles (di- /triazoles)



oxazoles (oxadi- /-triaz.)



pyrrolidines	
pyrrolines	
pyrroles	
furans	
Six-membered rings, represented by three CG beads	
cyclo-hexyls	
pyrans	
oxanes (di- /trioxanes)	
piperidines	
oxohetarenes	
iminohetarenes	
pyridines	
di- /triazines	
phenols	
arenes (benzenes)	
benzyl halides	
δ -lactams	

1.6.1 Identification of ionizable functional groups

Chemical properties of molecules depend, among other characteristics, on their ionization state in a given environment. This is described by acidity or basicity following the Brønsted-Lowry theory, the tendency of a molecule to lose or gain a proton resulting in a net negative or positive charge (19). The probability to de-/protonate depends on the chemical structure of a molecule and the pH of the environment⁵⁴



The acid dissociation constant Ka (20) is expressed as the ratio of the molar concentrations (signified by brackets) of the dissociated and undissociated species⁵⁵

$$Ka = \frac{[\text{A}^-][\text{H}^+]}{[\text{HA}]} = \frac{[\text{conjugate base}]}{[\text{conjugate acid}]} \times [\text{H}^+]. \quad (20)$$

The dissociation constant Ka is commonly given in logarithmic form, following equation (21)

$$\text{pKa} = -\log_{10}(Ka) = \text{pH} + \log_{10} \frac{[\text{conjugate acid}]}{[\text{conjugate base}]}. \quad (21)$$

We follow the definition of acidity and basicity proposed by the *ChemAxon*⁵⁶ software. The apKa , signified by (a) in Eqn. 19 and defined in Eqn. 22, is equivalent to the standard definition of the pKa found in the literature. The bpKa (under (b) in Eqn. 19 and defined in Eqn. 23) however is expressed as the deprotonation of its conjugate acid instead of the protonation of the base.

$$(a) \text{apKa}(\text{HA}) = \text{pKa}(\text{HA}) = \text{pH} + \log_{10} \frac{[\text{HA}]}{[\text{A}^-]}, \quad (22)$$

$$(b) \text{bpKa}(\text{HA}) = \text{bKa}(\text{H}_2\text{A}^+) = \text{pH} + \log_{10} \frac{[\text{H}_2\text{A}^+]}{[\text{HA}]}. \quad (23)$$

For biomolecular systems, the relevant pH range is in the physiologically neutral spectrum around $\text{pH} \sim 7$. Following Eqn. 22, for compounds with an $\text{apKa} \leq 7$ this leads to a decrease of the acid ($[\text{HA}]$) concentration, while the concentration of the conjugate base ($[\text{A}^-]$) increases. Analogously, Eqn. 23 shows that for compounds with a $\text{bpKa} \geq 7$ the concentration of the base ($[\text{HA}]$) decreases and the concentration of the conjugate acid ($[\text{H}_2\text{A}^+]$) increases. This allows to predict if the charge-neutral compounds in the data set obtained from the GDB⁵³ are likely to be negatively or positively charged in the physiologically relevant pH range. Functional groups with a median $\text{apKa} \leq 7$ or with a median $\text{bpKa} \geq 7$ can be considered negatively or positively charged, respectively, under physiological conditions.

2 Supporting Figures

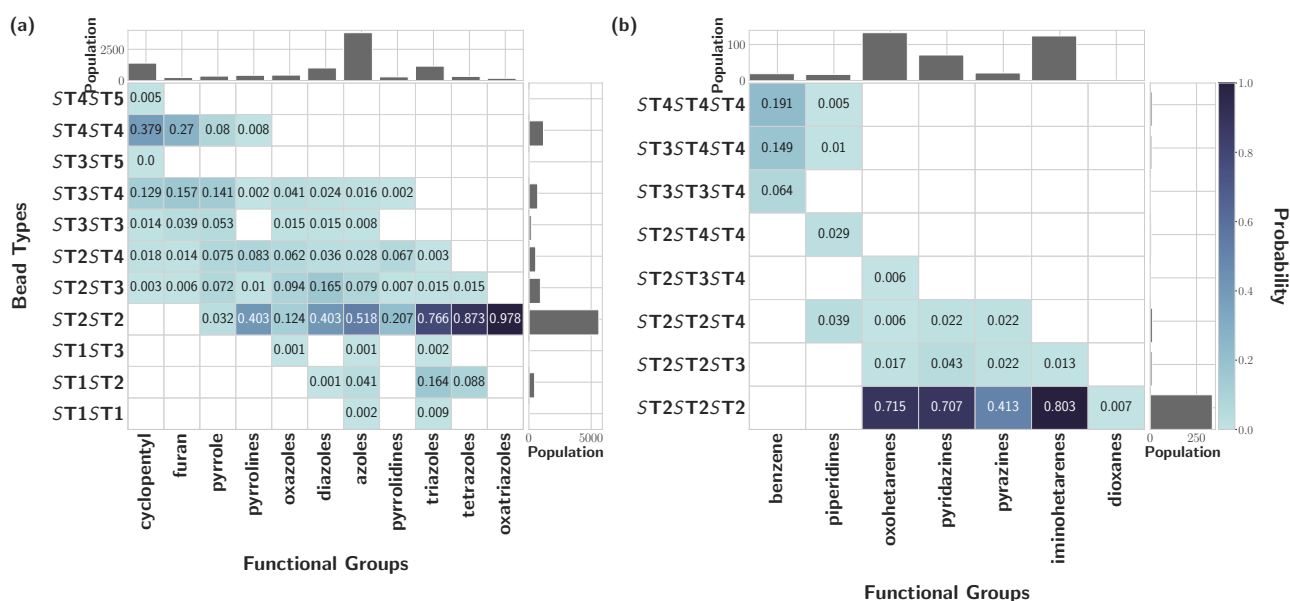


Figure S25: Ring structures identified in subsets of the GDB⁵³ mapping to (a) 5 beads or (b) 6 beads of the $5+0$ force field^{5,57}. The subsets were prescreened to remove molecules not containing ring systems. The prefix *S* signals the use of *small* variants of the beads used to model rings according to the Martini force field. Dark colors represent higher observation probabilities of a chemical structure for the corresponding bead sequence.

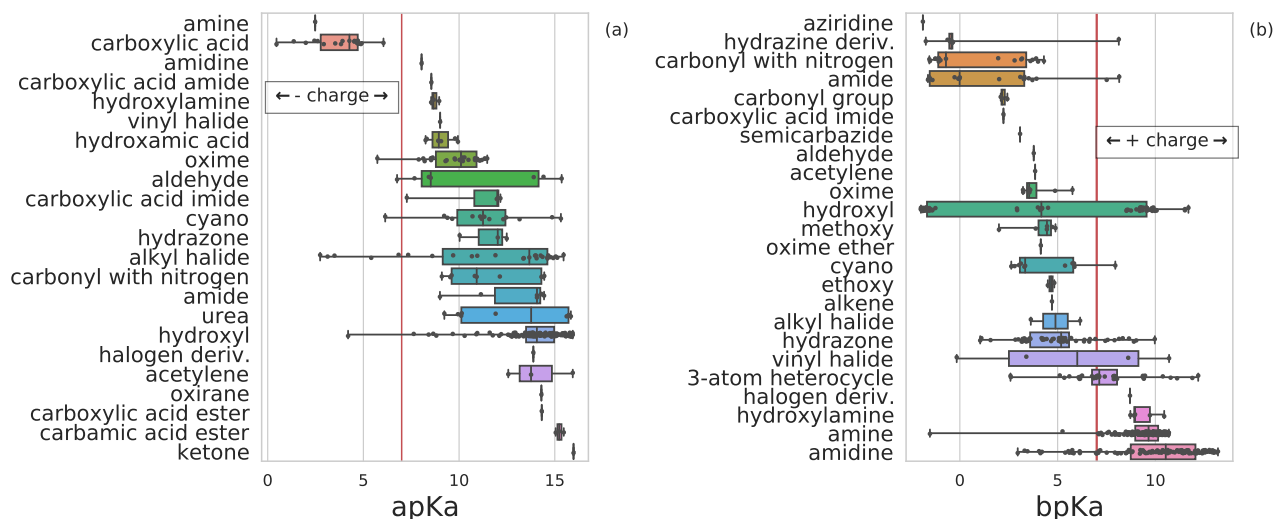


Figure S26: All dissociation constants calculated with the Calculator Plugin of Marvin⁵⁶ by ChemAxon. For the definition of the dissociation constant apK_a in (a) refer to Eqn. 22, for the bpK_a in (b) see Eqn. 23. Only the functional groups with (a) $apK_a \leq 7$ will be negatively charged at physiological pH ~ 7 . The opposite holds for (b), only the functional groups with $apK_a \geq 7$ will carry a positive charge around pH 7.

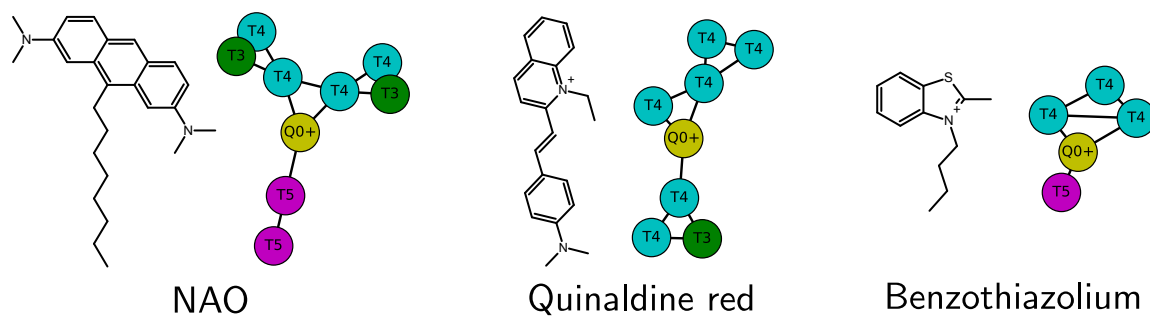


Figure S27: CG representations for NAO, Quinaldine red and Benzothiazolium used for experimental validation. The probabilistic relationship of functional groups and substructures of small molecules containing rings in Fig. S25 was used to generate the CG representations.

References

- [1] M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess and E. Lindahl, *SoftwareX*, 2015, **1**, 19–25.
- [2] S. J. Marrink, H. J. Risselada, S. Yefimov, D. P. Tieleman and A. H. De Vries, *The Journal of Physical Chemistry B*, 2007, **111**, 7812–7824.
- [3] J. Michalowsky, L. V. Schäfer, C. Holm and J. Smiatek, *The Journal of Chemical Physics*, 2017, **146**, 054501.
- [4] J. Michalowsky, J. Zeman, C. Holm and J. Smiatek, *The Journal of Chemical Physics*, 2018, **149**, 163319.
- [5] K. H. Kanekal and T. Bereau, *The Journal of Chemical Physics*, 2019, **151**, 164106.
- [6] D. H. De Jong, S. Baoukina, H. I. Ingólfsson and S. J. Marrink, *Computer Physics Communications*, 2016, **199**, 1–7.
- [7] N. Goga, A. Rzepiela, A. De Vries, S. Marrink and H. Berendsen, *Journal of Chemical Theory and Computation*, 2012, **8**, 3637–3649.
- [8] M. Parrinello and A. Rahman, *Physical Review Letters*, 1980, **45**, 1196.
- [9] M. Parrinello and A. Rahman, *Journal of Applied Physics*, 1981, **52**, 7182–7190.
- [10] T. Darden, D. York and L. Pedersen, *The Journal of Chemical Physics*, 1993, **98**, 10089–10092.
- [11] T. T. Pham and M. R. Shirts, *The Journal of Chemical Physics*, 2011, **135**, 034114.
- [12] T. T. Pham and M. R. Shirts, *The Journal of Chemical Physics*, 2012, **136**, 124120.
- [13] J. W. Pitera and W. F. van Gunsteren, *Molecular Simulation*, 2002, **28**, 45–65.
- [14] T. Steinbrecher, D. L. Mobley and D. A. Case, *The Journal of Chemical Physics*, 2007, **127**, 214108.
- [15] Y. Qi, H. I. Ingólfsson, X. Cheng, J. Lee, S. J. Marrink and W. Im, *Journal of Chemical Theory and Computation*, 2015, **11**, 4486–4494.
- [16] C. Chipot and A. Pohorille, *Free Energy Calculations*, Springer, 2007.
- [17] G. J. Rocklin, D. L. Mobley, K. A. Dill and P. H. Hünenberger, *The Journal of Chemical Physics*, 2013, **139**, 11B606_1.
- [18] M. Aldeghi, V. Gapsys and B. L. de Groot, *ACS Central Science*, 2018, **4**, 1708–1718.
- [19] B. Davies, *Integral transforms and their applications*, Springer Science & Business Media, 2002, vol. 41.
- [20] B. Mohr, K. Shmilovich, I. Kleinwächter, D. Schneider, A. L. Ferguson and T. Bereau, *Supporting data for: "Data-driven discovery of cardiolipin-selective small molecules by computational active learning"*, 2021, <https://doi.org/10.5281/zenodo.5507577>.
- [21] M. R. Shirts and J. D. Chodera, *The Journal of Chemical Physics*, 2008, **129**, 124105.
- [22] Beauchamp, Kyle A and Chodera, John D and Naden, Levi N and Shirts, Michael R, *Python implementation of the multistate Bennett acceptance ratio (MBAR)*, <https://github.com/choderalab/pymba>, Published under the MIT license.

- [23] R. Menichetti, K. H. Kanekal, K. Kremer and T. Berau, *The Journal of Chemical Physics*, 2017, **147**, 125101.
- [24] 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 Central Science*, 2018, **4**, 268–276.
- [25] P. Ghosh, M. S. Sajjadi, A. Vergari, M. Black and B. Schölkopf, *arXiv preprint arXiv:1903.12436*, 2019.
- [26] D. P. Kingma and M. Welling, *arXiv preprint arXiv:1312.6114*, 2013.
- [27] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, *arXiv preprint arXiv:1806.01261*, 2018.
- [28] M. Simonovsky and N. Komodakis, International Conference on Artificial Neural Networks, 2018, pp. 412–422.
- [29] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals and G. E. Dahl, International conference on machine learning, 2017, pp. 1263–1272.
- [30] J. Chung, C. Gulcehre, K. Cho and Y. Bengio, *arXiv preprint arXiv:1412.3555*, 2014.
- [31] Y. Li, D. Tarlow, M. Brockschmidt and R. Zemel, *arXiv preprint arXiv:1511.05493*, 2015.
- [32] O. Vinyals, S. Bengio and M. Kudlur, *arXiv preprint arXiv:1511.06391*, 2015.
- [33] D. P. Kingma and J. Ba, *arXiv preprint arXiv:1412.6980*, 2014.
- [34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai and S. Chintala, *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035.
- [35] K. Shmilovich, R. A. Mansbach, H. Sidky, O. E. Dunne, S. S. Panda, J. D. Tovar and A. L. Ferguson, *The Journal of Physical Chemistry B*, 2020, **124**, 3873–3891.
- [36] K. D. Konze, P. H. Bos, M. K. Dahlgren, K. Leswing, I. Tubert-Brohman, A. Bortolato, B. Robbason, R. Abel and S. Bhat, *Journal of Chemical Information and Modeling*, 2019, **59**, 3782–3793.
- [37] E. Brochu, V. M. Cora and N. De Freitas, *arXiv preprint arXiv:1012.2599*, 2010.
- [38] D. Arthur and S. Vassilvitskii, *k-means++: The advantages of careful seeding*, Stanford technical report, 2006.
- [39] D. Sivia and J. Skilling, *Data analysis: A Bayesian tutorial*, Oxford University Press, 2006.
- [40] M. Ebden, *arXiv preprint arXiv:1505.02965*, 2015.
- [41] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning (adaptive computation and machine learning)*, The MIT Press, 2005.
- [42] 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.
- [43] H. Mohammadi, R. L. Riche, N. Durrande, E. Touboul and X. Bay, *arXiv preprint arXiv:1602.00853*, 2016.

- [44] D. J. Lizotte and D. James, *Practical Bayesian optimization*, Library and Archives Canada = Bibliothèque et Archives Canada, 2009.
- [45] D. Ginsbourger, R. Le Riche and L. Carraro, *Computational intelligence in expensive optimization problems*, Springer, 2010, pp. 131–162.
- [46] H. Bhattacharjee and D. G. Vlachos, *Journal of Chemical Information and Modeling*, 2020, **60**, 4673–4683.
- [47] T. Milenković and N. Pržulj, *Cancer Informatics*, 2008, **6**, CIN-S680.
- [48] N. Pržulj, D. G. Corneil and I. Jurisica, *Bioinformatics*, 2004, **20**, 3508–3515.
- [49] N. Pržulj, *Bioinformatics*, 2007, **23**, e177–e183.
- [50] T. Hočevár and J. Demšar, *Bioinformatics*, 2014, **30**, 559–565.
- [51] T. Berau and K. Kremer, *Journal of Chemical Theory and Computation*, 2015, **11**, 2783–2791.
- [52] P. Ertl, *Journal of Cheminformatics*, 2017, **9**, 1–7.
- [53] T. Fink and J.-L. Reymond, *Journal of Chemical Information and Modeling*, 2007, **47**, 342–353.
- [54] J. L. MacCallum, W. D. Bennett and D. P. Tieleman, *Biophysical Journal*, 2008, **94**, 3393–3404.
- [55] P. Bruice, *Organic Chemistry*, Pearson/Prentice Hall, 2004.
- [56] ChemAxon, *Calculator Plugin of Marvin 17.28.0*, 2017, <https://www.chemaxon.com>.
- [57] R. Menichetti, K. H. Kanekal and T. Berau, *ACS Central Science*, 2019, **5**, 290–298.