Supporting Information: Sequence determinants of protein phase separation and recognition by protein phase-separated condensates through molecular dynamics and active learning

Arya Changiarath,*[a] Aayush Arya[a], Vasileios Xenidis[c], Jan Padeken[b], and Lukas S. Stelzl[a,b]

I. METHODS

We study the mapping of a protein sequence to its property represented by a single numeric label $f(\vec{x})$. As we shall describe later, this $f(x)$ could be the second virial coefficient B_{22}, or a quantity that provides an estimate of the affinity of a sequence to bind to a condensate. Our scientific problem thus reduces to training a model to learn the mapping $\vec{x} \mapsto f(\vec{x})$, and simultaneously exploring the protein sequence space to maximize this label. We utilize the `wazy` package which provides the framework for performing such optimization tasks implemented here.

An overview of our workflow is shown in Fig. 1: We begin with an initial set of protein sequences, and run residue-level coarse-grained molecular dynamics simulations using HOOMD-blue[1] from which the quantity of interest $f(x)$ is to be extracted. We initialize a surrogate model with the protein sequence Bayesian Optimization (BO) package `wazy`[2], and provide this set of sequences and their labels for an initial calibration. In what follows, we will refer to this initial stage as Iteration 0. Next, the model is interrogated to suggest which training examples should be provided next. The Bayesian optimizer uses an *acquisition function* to rank proteins in the sequence space, in terms of their utility for the model. Coase-grained simulations of this new batch of protein sequences are again performed, which becomes input for the next cycle of learning. A schematic of this framework is shown in Figure 1. In what follows, we provide a detailed account of the training and optimization methodology.

Model Description

In Bayesian Optimization (BO) problems, the most commonly used choice of a surrogate model is a Gaussian Process (GP)[3]. In contrast, in our work we utilize the `wazy` package[2], which uses deep ensembles of feed-forward neural networks. This is desirable, as neural networks have greater expressive power and potential to be pre-trained than GPs.

For training the model for predicting the label $f(x)$, the sequence is first mapped to a numeric vector. In `wazy`, mapping from a FASTA protein sequence to a continuous vector $\vec{x}$ (called, *featurization*) is done using UniRep[4], which is a Long-Short Term Memory (LSTM) model designed for next amino acid prediction, trained on the UniRef50 dataset. In mapping sequence-property relationships, appropriate featurization is necessary for for similar sequences in the protein sequence space to be clustered together.

The desired numeric label is then predicted from from the feature vector $\vec{x}$ via a multi-layer perceptron. To allow uncertainty analysis, an ensemble of MLPs (in short, a *deep ensemble*) is used to simultaneously predict the same quantity in parallel. UniRep[4] parses the sequence into a fixed-length vector of dimensions $N = 1900$. As for network architecture, each single MLP takes an input of dimension 1900, followed by layers of 128, 32 and 2 neurons respectively. The final output layer thus provides two numbers μ_m and σ_m that characterize a normal distribution $\mathcal{N}(\hat{\mu_m}, \hat{\sigma_m})$. The mean over the ensemble of M multi-layer perceptrons

$$\hat{\mu} = \frac{1}{M}\sum_{m} \mu_m$$

is used as the final estimate. Estimates of model (epistemic) uncertainty

$hat\sigma_e$ are obtained from the dispersion in the predictions of different MLPs in the ensemble

$$\hat{\sigma}_e^2 = \frac{1}{M}\sum_m (\mu - \mu_m)^2$$

In addition, a statistical (aleatoric) uncertainty is estimated as

$$\hat{\sigma}_a^2 = \frac{1}{M}\sum_m \sigma_m^2$$

Throughout the discussion, we will refer to the combined total uncertainty, computed as $\sigma^2 = \sigma_e^2 + \sigma_a^2$

The state-of-the-art solution to uncertainty estimation is Bayesian Neural Networks. However, Bayesian NNs require non-trivial modifications of the training procedure, compared to non-Bayesian NNs. Deep ensembles use *model combinations* (in the present case, multi-layer perceptrons; MLPs) to provide predictive uncertainty estimates. Further, they are much simpler to implement, and require little hyperparameter tuning (see ref.[5] for a detailed discussion). For further details about `wazy`, we refer the reader to ref.[2].

Bayesian optimization works via ranking sequences using an acquisition function $\mathcal{A}$. We choose the "upper confidence bound" (UCB) acquisition function[3] which balances both exploitation and exploration:

$$\mathcal{A}(x;\lambda) = \mu(x) + \lambda\sigma(x) \tag{1}$$

Here, the exploitation term $\mu(x)$, is the current estimation of the value. And the exploration term, $\sigma(x)$, measures the uncertainty. The parameter λ acts as a control to adjust the balance between exploitation and exploration. Expected Improvement (EI) is another common acquisition function used in

Bayesian Optimization, along with UCB. The key idea behind EI is to select the next point to evaluate based on the current best result. The choice of the acquisition function sensitively affects which regions of sequence space get explored for, and thus the arrival at an optimal solution.

Training, Optimization, and Validation

For calibrating the model with sequence-label relationship as input, we chose an initial set of sequences drawn from ProtGPT2[6], a protein language model constructed for *de novo* protein design. We ensured that the initial set was diverse in its composition, and spanned a wide range of values for the numeric label of interest (here, B_{22})(SI fig). To track the learning of the model, we used in addition a set of sequences for validation that was never shown to the model for training. Following ref[2] we will refer to "optimization step" as the single training step in which the model is updated after each sequence and numeric label input. After each optimization, we predict the numeric label for validation set, and estimate the mean squared error from the residual between true (computed values, from simulations) and predicted values.

REFERENCES

[1]J. A. Anderson, J. Glaser and S. C. Glotzer, *Comput. Mater. Sci.*, 2020, **173**, 109363.

[2]Z. Yang, K. A. Milas and A. D. White, *bioRxiv*, 2022.

[3]P. I. Frazier, 2018.

[4]E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi and G. M. Church, *Nat. Methods*, 2019, **16**, 1315–1322.

[5]B. Lakshminarayanan, A. Pritzel and C. Blundell, *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 6405–6416.

[6]N. Ferruz, S. Schmidt and B. Höcker, *Nature Communications*, 2022, **13**, 4348.

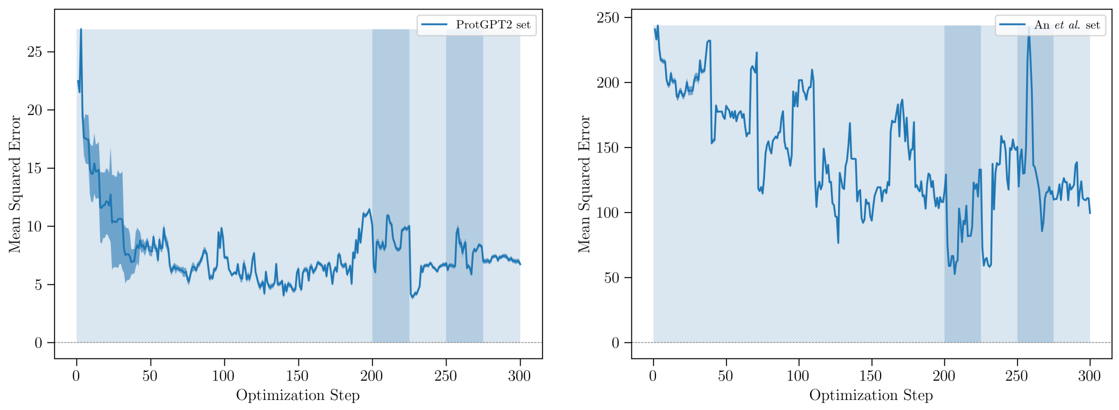

Figure S1. Training with an initial calibration set of 200 sequences, instead of 50 does not lead to improved outcomes (compare with Fig.1 in the main text).

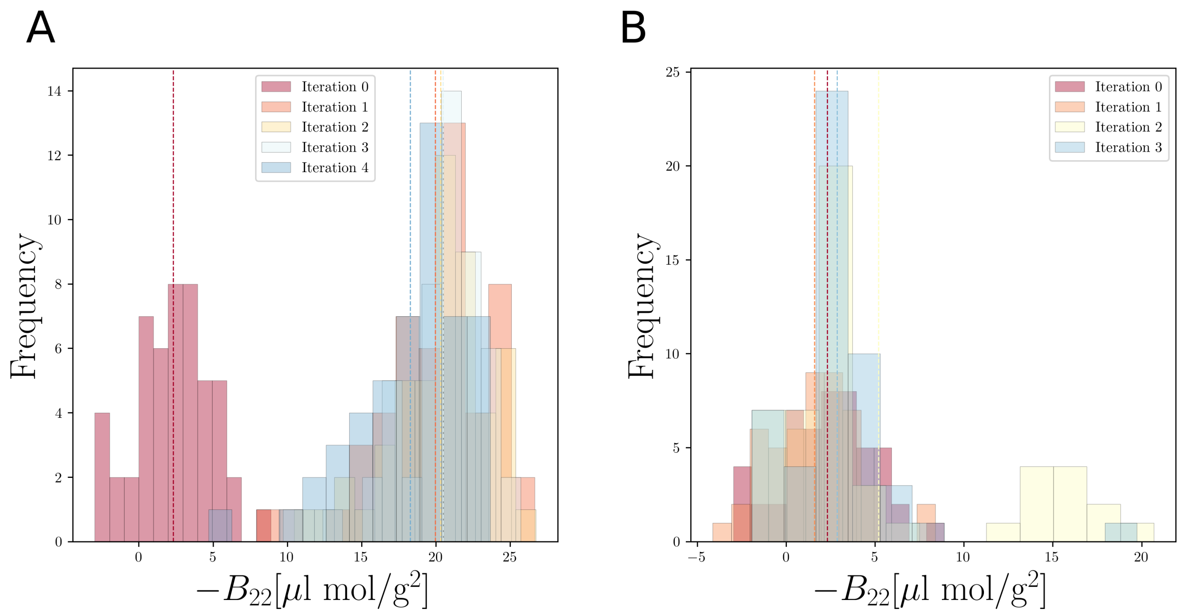

Figure S2. Comparison of B_{22} distribution of sequences suggested by `wazy` in each iteration. A) Optimisation using UCB. B) Optimisation using EI.

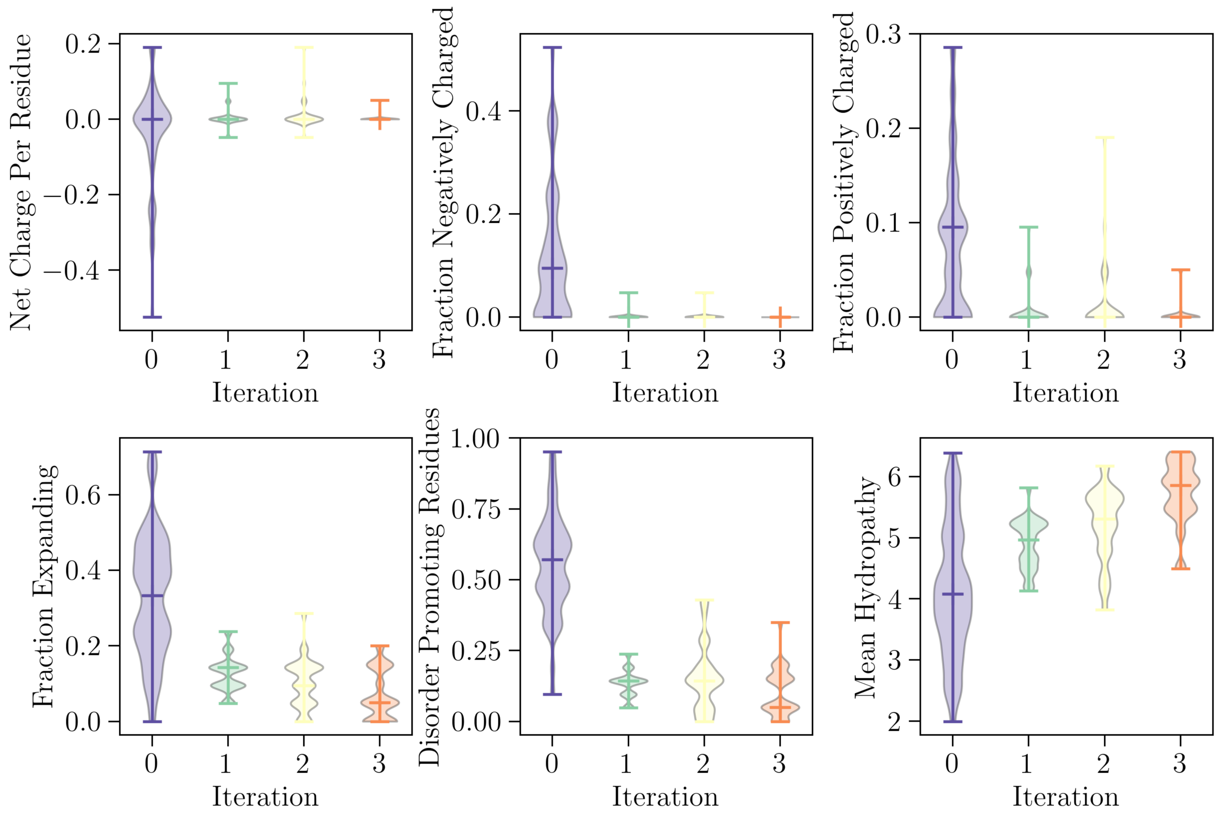

Figure S3. Sequence characteristics favoured by the model during the optimisation process for designing multiphasic condensates. The optimisation process favoured the enrichment of hydrophobic amino acid residues.

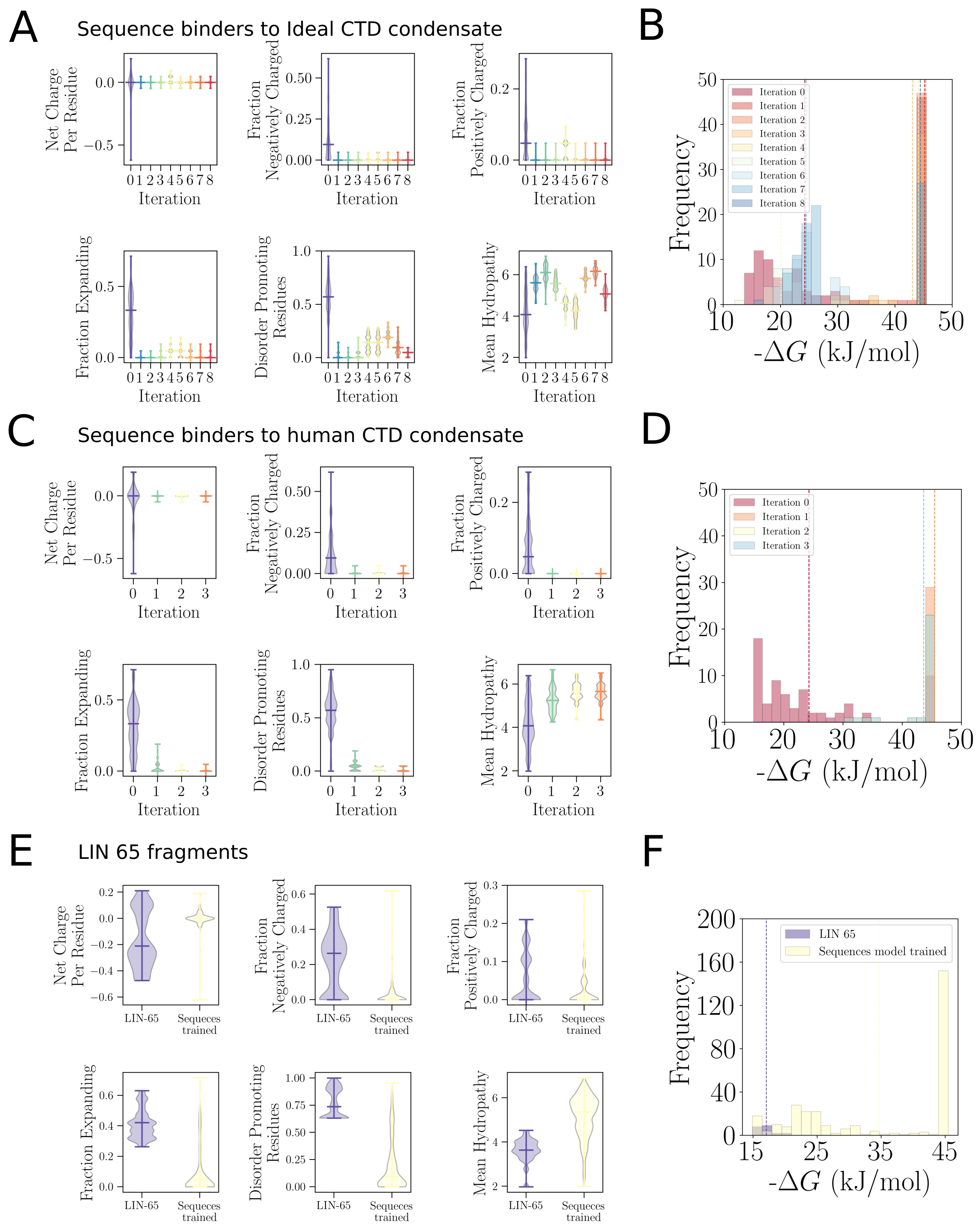

Figure S4. characterisation of peptides binding to phase-separated condensates of CTD and hCTD. A) Sequence characteristics of peptides that bind to ideal CTD suggested by `wazy` across iterations. Net charge per residue decreases with iteration, while hydropathy value generally increases, with a slight decrease after the 4th iteration. B) Distribution of ΔG shifts towards higher values after calibration. A slight shift towards lower values is observed for the 4th and 5th iterations, followed by a subsequent increase.

Figure S4. (Previous page.) C) Characteristics of peptides designed by `wazy` to bind to hCTD. Hydropathy values show an increase across iterations. D) ΔG distribution for hCTD-binding peptides shifts towards higher values after calibration. E) Comparison of sequence characteristics between LIN65 fragments and all peptides suggested and trained by `wazy`. F) Comparison of ΔG distributions, showing that LIN65 fragments represent a subset of less interacting sequences within the full range of designed peptides.