Now What Sequence? Pre-trained Ensembles for Bayesian Optimization of Protein Sequences

Ziyue Yang,[1] Katarina A. Milas,[1] and Andrew D. White[1,*]

[1] *Department of Chemical Engineering, University of Rochester*

(Dated: August 5, 2022)

Pre-trained models have been transformative in natural language, computer vision, and now protein sequences by enabling accuracy with few training examples. We show how to use pre-trained sequence models in Bayesian optimization to design new protein sequences with minimal labels (i.e., few experiments). Pre-trained models give good predictive accuracy at low data and Bayesian optimization guides the choice of which sequences to test. Pre-trained sequence models also obviate the common requirement of finite pools. Any sequence can be considered. We show significantly fewer labeled sequences are required for many sequence design tasks, including creating novel peptide inhibitors with AlphaFold. This work should enable calibrated predictions with few examples and iterative design with low data (1-50).

I. INTRODUCTION

Sequence design is the construction of novel protein or peptide sequences which, when synthesized, will have chosen functional properties. Pre-trained sequence models like ESM,[1,2] UniRep,[3] and ProtTrans[4] have recently shown excellent performance on structure prediction tasks with minimal experimental data. Reducing experimental data needed for sequence design is especially useful when finding peptides that bind to specific proteins, where predicting binding requires solid-phase peptide synthesis followed by assay.[5] However, these pre-trained models are not able to guide experiments – only make predictions with few data. Bayesian optimization (BO) is becoming the standard approach for choosing which experiments to do, but is incompatible with pre-trained sequence models because BO requires accurate uncertainty predictions.[6] We show how to modify pre-trained sequence models to enable BO by modifying the encoder and readouts of the models. This marries the accuracy of pre-trained models with the ability to guide experiments of Bayesian optimization.

BO is a "black-box" optimization technique suited to expensive functions.[7] BO can optimize a function without access to its derivative or any other information (the "black-box") and optimizes with minimal evaluations of the function. BO has been successful in sequence, formulation, and molecular design problems where the "expensive function" is doing an experiment.[8–11] BO is a *Bayesian* method, so it requires construction of a probability distribution surrogate model that approximates the black-box function (i.e., the experiment). BO works by choosing experimental data points in a way that balances **exploring** sequence space to improve accuracy of the surrogate model and **exploiting** the surrogate model to maximize the black-box function. The surrogate model is nearly always a Gaussian process regression model,[6] which is not as empirically expressive as neural networks[12] and has limited ability to be pre-trained. We show here how to use pre-trained sequence models instead of Gaussian process regression. The advantage is that pre-trained sequence models are accurate with very few experiments and work well in the high-dimensional space of sequence design.

Current pre-trained models cannot be used as surrogate models in BO for two reasons. First, BO requires a gradient of the surrogate model with respect to its input, but sequence models have discrete integers as input. We use probabilistic reparameterization proposed in Linder and Seelig [13], which is similar to the Gumbel softmax-trick[14,15] to enable gradient ascent of a pre-trained sequence model. The second reason is that pre-trained models do not have uncertainties to compute probabilities in the BO algorithm. The classical solution to this problem is Bayesian neural networks,[16] but Izmailov *et al.* [17] recently showed that deep ensembles[18] are both competitive with integrating Bayesian neural network posteriors and more robust to noise in training data. We hypothesize that these two properties make deep ensembles a good approach in the low data regime targeted here.

Thus, our sequence design method is to modify pre-trained sequence models by replacing the discrete input with a categorical distribution and deep ensembling to allow BO sequence design. In contrast to previous work,[19] our method does not require a pool of known sequences and we can use any existing pre-trained sequence model without additional training. To test this method, we require the ability to label arbitrary new sequences (e.g., like you would in an experiment). We use here three tasks that mimic experiments to test our method: (1) designing a peptide that is hemolytic as evaluated by an RNN that is treated as a black-box;[20] (2) matching an unknown sequence by only receiving BLOSUM-scored distance;[21] and (3) designing a peptide that binds to a target protein as evaluated via an AlphaFold multimer calculation.[22,23]

* andrew.white@rochester.edu

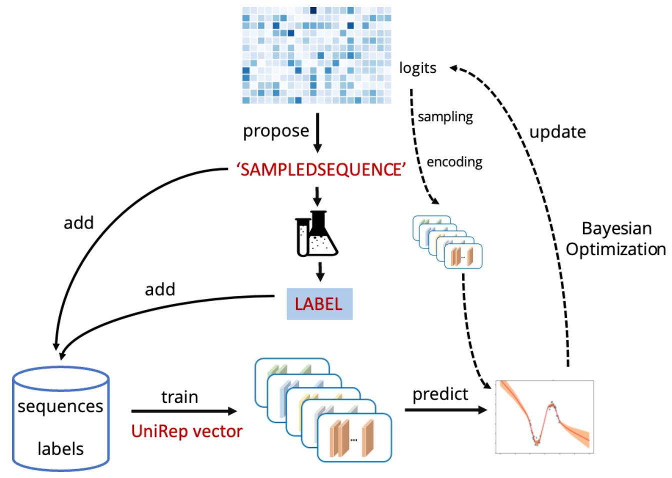

FIG. 1. An overview of the model and Bayesian optimization process. Sequences are defined by logits during BO acquisition function maximization and then labeled. The complete set of sequences and labels is then re-used to train the deep ensembled MLP. Finally, the MLP is used for the next round of BO.

A. Related Work

Sequence design (also known as protein engineering) is a broad category of tasks like enzyme design, peptide drug design, design of self-assembled structures, and de novo protein design. There are multiple approaches to designing new sequences that range from purely experimental to purely computational. A commonly used experimental method is directed evolution,[24] a process of optimizing fitness by stochastically mutating a wild-type sequence. Directed evolution typically requires a high-throughput assay that can select sequences. Directed evolution can be combined with computational approach.[25–27] Cheng *et al.*[28] proposed an efficient, experimental design-oriented closed-loop optimization framework for protein directed evolution, which employs a combination of novel low-dimensional protein encoding strategy and Bayesian optimization enhanced with search space prescreening via outlier detection. Harteveld *et al.*[29] proposed a framework based to automatically assemble structural templates with native-like features. Our approach is differentiated from directed evolution because (1) our approach requires only testing a few dozen sequences and (2) the assay does not need to involve screening.

Machine learning for sequence design is beneficial especially where assays are expensive or slow enough to outweigh the cost and time of sequencing and synthesis.[30,31] A recent review of *adaptive* machine learning approaches for sequence design can be found in Hie and Yang[32]. Biswas *et al.*[33] used machine learning to guide the sequence searching by learning a latent representation from a small number of mutants. Bayesian optimization has been used for sequence design previously (without pre-training) in Hughes *et al.*[11] and Greenhalgh *et al.*[34]. Khan *et al.*[35] recently showed how to approach sequence design as a combinatarial Bayesian optimization problem with feasible trust regions for antibody design. Das *et al.*[36] reported an efficient computational method for the generation of antimicrobials with desired attributes by leveraging guidance from classifiers trained on an informative latent space of molecules modeled using a deep generative autoencoder. Castro *et al.*[37] proposed a protein sequence design workflow by first training a deep Transformer-based autoencoder (ReLSO) to jointly generate protein sequences as well as predict fitness, following by fitness optimization over latent space and high fitness latent space sampling. Our work is similar to the previous Bayesian optimization approaches because it is iterative and calibrated, but different than approaches that are non-adaptive (train once on data).

Deep generative sequence models can be categorized into variational autoencoders (VAEs), generative adversarial networks (GANs) and language models, including RNNs and attention models. These are widely used to perform peptide generative with desired properies.[38–42] Linder *et al.*[43] developed a generative method which can explicitly control sequence diversity during training by penalizing any two generated sequences on the basis of similarity. Ferruz *et al.*[44] developed ProtGPT2, a Transformer-based generative model trained on UniRef50 which can generate de novo protein sequences following the principles of natural ones. Anand and Achim[45] introduced a fully data-driven denoising diffusion probabilistic model for protein structure, sequence, and rotamers that is able to generate highly realistic proteins across the full range of domains in the Protein Data Bank by using equivariant transformers. These generative models can propose new sequences, but not inside an iterative BO algorithm like proposed here.

Pre-trained models for sequence design are increasingly common. Alley *et al.*[3] used an mLSTM model to learn statistical representations of proteins from 24 million UniRef50[46] sequences. Rives *et al.*[1] obtained sequence representations containing information about biological properties by using unsupervised learning to train a transformer language model on 86 billion amino acids across 250 million protein sequences spanning evolutionary diversity. Detlefsen *et al.*[47] performed analysis on protein representations and demonstrated that pre-training representations can yield improved performance as well as significantly improve interpretability and let the models reveal biological information. Wang *et al.*[48] reconstructed single-sequence 3D protein structure by feeding the pre-trained sequence embedding into a multi-scale network which is able to predict the inter-residue 2D geometry. Transformer models equipped with self-attention mechanisms have shown to be particularly well-suited to capture dependency among sequence elements while being capable to scale vast amounts of model parameters.[49] Kaplan *et al.*[50], Hoffmann *et al.*[51] indicate pre-trained sequence models could be computeoptimally trained by balancing compute, observations, and model parameters. Ruffolo *et al.*[52] used antibody affinity maturation with language models and weakly supervised learning. Russ *et al.*[53] developed another sequence model for designing chorismate mutase enzymes. These promising results led to our choice of using pre-trained model.

Bayesian neural networks have been proposed as an improvement to neural networks that account for uncertainty in predictions.[16] The posteriors can be computed directly, with significant computational effort, via Hamiltonian Markov chain Monte Carlo (HMC).[54] There are frequent efforts to reduce the cost with approximations, such as Maddox *et al.*[55] and specifically for chemical systems in Soleimany *et al.*[56]. Deep ensembles[18] are a common baseline that seems to be in practice as good as HMC while being more robust.[17] Recent work has also proposed other ways of approximating uncertainty in transformers, the most common architecture for pre-trained sequence models.[57]

II. THEORY

The labeling of a sequence x with its property y is done with an expensive black-box function $f(x)$. $f(x)$ could require synthesis prior to an experiment, a molecular dynamics calculation, or an expensive calculation. Our goal is to find x^* that maximizes $f(x^*)$ through BO starting with zero labeled data points. We indicate iterations through the BO algorithm with index n (x_n)

We construct a pre-trained sequence model that embeds a sequence $x \in \{0,1\}^{L\times A}$ into a continuous vector space $u(x) = \vec{u} \in \mathbb{R}^D$, where L is the sequence length, A is the number of possible tokens in the sequence (alphabet), and D is the dimension of the sequence representation. We use UniRep for $u(x)$.[3] Unirep is a long short-term memory (LSTM) model trained to perform next amino acid prediction, as implemented in JAX.[58,59] Properties are predicted from $\vec{u}$ with a multi-layer perceptron (MLP) $g(\vec{u}) = \hat{y}$. To enable uncertainty predictions, we use deep ensembles.[18] Deep ensembles predict a normal distribution parameterized from M models $g_m(\vec{u})$. Finally, to enable optimization of the input sequence x, during BO we sample the sequence from a categorical distribution characterized by trainable logits, similar to the Gumbel-Softmax Trick.[14,15] These individual components are described in more details below and visually in Figure 1.

A. Model

UniRep[3] is an LSTM model[60] trained on Uniref50[61] to perform next amino acid prediction by minimizing cross-entropy loss. By conducting this semi-supervised classification, the model learns how to internally represent protein sequences. Given a sequence of any length, UniRep returns a single fixed-length vector representation. We denote UniRep as $u(x) = \vec{u}$, where $u(\cdot)$ is the UniRep model, which takes sequence x as input and outputs the representation vector $\vec{u}$. The dimension of $\vec{u}$ is $N = 1900$ for UniRep. See Alley *et al.*[3] for complete details. We use a JAX implementation of UniRep.[58,59]

Uncertainty in predicted labels can be split into an epistemic uncertainty (EU) and aleatoric uncertainty (AU).

$$\sigma^2 = \sigma_e^2 + \sigma_a^2 \tag{1}$$

where σ_a^2 is AU and σ_e^2 is EU. g AU is also known as data uncertainty or statistical uncertainty. It is unavoidable noise from $f(x)$. For example, data collected from a laboratory will have uncertainty.

EU is also known as model uncertainty or system uncertainty. It arises from the lack of knowledge of the system in respect to quantities and processes within the system. It can be reduced through larger or better models. High EU arises in regions where there are few or no observations for training.

The output from each single MLP $g_m(x)$ is two numbers that characterize a normal distribution $\mathcal{N}(\hat{\mu}_m, \hat{\sigma}_m)$. We can use these to provide estimates of the AU and EU:[18]

$$\hat{\mu} = \frac{1}{M}\sum \hat{\mu}_m \tag{2}$$

$$\hat{\sigma}_e^2 = \frac{1}{M}\sum(\hat{\mu} - \hat{\mu}_m)^2,\ \hat{\sigma}_a^2 = \frac{1}{M}\sum_m \hat{\sigma}_m^2(x) \tag{3}$$

Where M is the ensemble size, m is the index of the MLP.

Likelihood describes the joint probability of the observed data as a function of the parameters of the MLPs. Negative log-likelihood minimization is a proxy problem to the problem of maximum likelihood estimation. We assume that the labels y follow a normal distribution of $P[f(x) = y]$ characterized by $\mu(x), \sigma(x)$. The negative log-likelihood of model parameters is the probability of observing the label y given sequence x:[18]

$$l(\theta_m, x, y) = -\log p(y|x) = \frac{\log \hat{\sigma}_m^2(x)}{2} + \frac{(y - \hat{\mu}_m(x))^2}{2\hat{\sigma}_m^2(x)} + C. \tag{4}$$

As proposed in Lakshminarayanan *et al.*[18], we also do adversarial training. After one step with the loss in Equation 4, we perform another step to smooth the model predictions:

$$x' = x + \epsilon\,\mathrm{sign}(\nabla_x l(\theta, x, y)) \tag{5}$$

where ϵ is a hyperparameter that controls the strength.

B. Bayesian Optimization

Bayesian optimization is a gradient-free global function optimization method which is constructed for expensive-to-evaluate functions.[6] The goal of Bayesian optimization is to both explore and exploit existing knowledge as expressed in an acquisition function. Namely, our next sequence to test is computed from

$$x_{n+1} = \arg\max_x \mathcal{A}(g, x) \tag{6}$$

where $\mathcal{A}(g, x)$ is our acquisition function that uses the model (g) and sequence (x). We use the simplest acquisition function for balancing exploration and exploitation called upper-confidence bound (UCB):

$$\mathcal{A}(g, x) = \hat{\mu} + \beta\hat{\sigma} \tag{7}$$

where β is a hyperparameter that balances exploration and exploitation that can be scheduled to increasingly exploit over the BO algorithm. UCB is not robust to label noise, and recent work has proposed different acquisition functions robust to label noise.[62–64] To alleviate label noise, we only use EU $(\hat{\sigma}_e)$ in Equation 7. We chose UCB due to its simplicity and robust performance across hyperparameter choices.

BO requires computing a gradient of the surrogate model – UniRep – with respect to input x to maximize the acquisition function.[65] This is problematic, since x is not continuous. We can redefine x as being a random categorical distribution parameterized by continuous logits $l \in \mathbb{R}^{L\times A}$. Then when a sequence is needed, x_i is computed from a random drawn categorical at the ith position (where i indexes positions in sequence and j indexes amino acid/word):

$$x(l)_{ij} = \mathbf{1}_{(Z_i=j)},\, Z_i \sim \sigma(l)_i \tag{8}$$

where $\sigma(l_i)$ is the categorical distribution parameterized by l_i This reparameterization makes the gradient accessible, but we must propagate a gradient through sampling from the distribution. We use a straight-through approximation

$$\frac{\partial\delta(l)_{ij}}{\partial l_{ik}} = \frac{\partial\sigma(l)_{ij}}{\partial l_{ik}} = \sigma(l)_{ik}\cdot(\mathbf{1}_{(i=k)} - \sigma(l)_{ij}) \tag{9}$$

However, the logits can drift close to zero so that the drawn sequences are highly variable or the logits can grow in magnitude and gradient updates no longer actually change the sequence. The standard solution to this is the Gumbel-Softmax Trick.[15] We used a slightly different approach introduced by Linder and Seelig [13], which is to simply add a trainable layer normalization that can trainably affect the mean and variance of logits.[66] Recently, Daulton *et al.* [67] showed that BO with probabilistic reparameterization ,like Equation 8, will converge to the true maximum of the acquisition function. Although, convergence of the $L \times A$ logit matrix is still a difficult high-dimensional optimization.

We considered using the continuous latent space as well. In Figure S1 we use the UniRep decoder to avoid optimizing the sequence directly. We optimized $g(\vec{u})$ by working with $\nabla g(\vec{u})$. We found that the decoder $x' = u^{-1}(\vec{u})$ gave a sequences back that were inconsistent enough with the forward label $g(u(x'))$ and prevented optimization past a certain point. Figure S1 shows optimization of latent space has continuous improvement, but after decoding to an actual sequence and evaluating $g(\vec{u})$ there is a plateau.

Variable sequence lengths were incorporated by maximizing the acquisition function over lengths $L-1$, L, and $L+1$ at each BO iteration and the length L was replaced by the best for the next iteration.

III. METHODS

The deep ensembled MLPs are sensitive to mode collapse, where all $g_m(x)$ models converge to the same parameters by overfitting the training data.[68] This is common in our setting of only a few data points. We use three strategies to mitigate this. First, we use relatively few parameters in the MLPs to frustrate the loss landscape and make it sensitive to initial parameters. Second, we resample data so that the training data seen by each MLP is different. Lastly, we employ standard techniques to reduce overfitting like dropout and weight decay. There are more systematic approaches that could be used,[69] but we found these simple strategies give good calibrated uncertainties across the three tasks. Adversarial training was found to have minimal improvement, even when adding significant artificial label noise (Figure S3). All hyperparameters, including architecture and training parameters are given in Table I.

While training, we found resampling data to account for non-uniformity in label distribution is important – especially because BO targets high values that are rare. So while training the model, we resampled training data according to their labels by binning training data into 10 classes and sampling with replacement to get equal frequency. As discussed above, we repeated this process for each MLP $(g_m(x))$, so that they saw different training data frequencies.

Maximizing Equation 7 for BO is a non-convex problem[72] and so we started from 16 initial logit distributions and performed 200 gradient ascent steps with Adam[73] (Figure S3). This was repeated for each considered length - one higher and one lower than current length. Daulton *et al.* [67] recommended L-BFGS, although we found Adam works for our tasks as shown in Figure S3.

Hyperparameters	Choice
MLP shape	128, 32, 2
Activation	Swish[70,71]
Ensemble number	5
training epochs	100
training batch size	8
training learning rate	1e-4
training resampled classes	10
training weight decay (Adam)	0.1
training ϵ	0.001
dropout rate	0.2
BO batch size	16
BO iterations	200
BO β	2.0

TABLE I. Hyperparameters for work presented here. All were tuned on finding unknown target peptide sequence task.

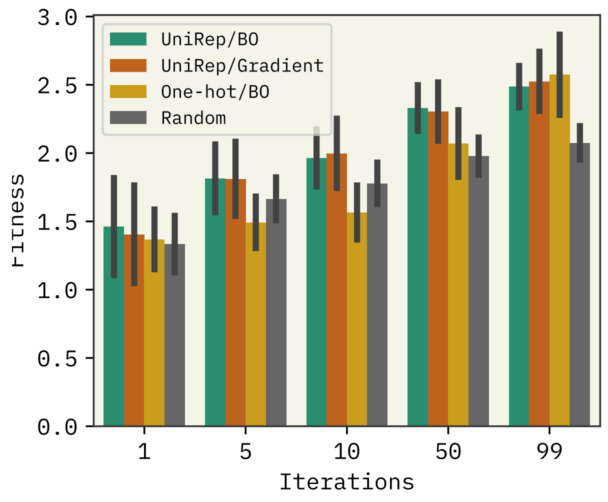

FIG. 2. Current best sequence on unknown target matching task along with ablations. BO is similar to direct gradient ascent (greedy) of surrogate model because this task is convex and has no noise. Pre-training helps at low iterations, but is surpassed by one-hot at later iterations. This is expected on this highly-specific task. Error bars are 95% confidence intervals from bootstraping 50 repeated BO runs.

IV. RESULTS

We tested our algorithm on three different tasks: designing a hemolytic peptide, finding an unknown target sequence, and designing a peptide that binds to Ras GTPase.[74] In the first two tasks, we also compared with ablations by removing the pre-trained model with one-hot encoding and/or using greedy optimization instead of BO.

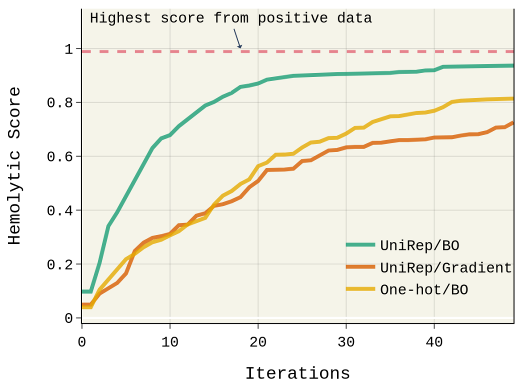

FIG. 3. Current best sequence during iterative optimization averaged across 50 runs. The different lines are the algorithm presented here (UniRep/BO) and ablations. Sequence length is fixed to thirteen to compare with one-hot encoding. Iterations is same as number of sequences observed.

A. Hemolytic peptide

The first task is to design a hemolytic peptide. In place of an experiment, we use a previously trained model as an stand-in for $f(x)$. This model is a bidirerctional LSTM[20,60] trained on data from Pirtskhalava *et al.*[75]. This task has variable sequence length and is initialized to a random peptide with random lengths from 10–20 uniformly sampled. Figure 5 shows the results of our algorithm averaged (mean) over 50 independent runs. Figure 4 shows corresponding distribution of sequence lengths over the course of the algorithm. It can find a likely hemolytic peptide within 5 iterations and nearly match the most hemolytic predicted peptide from the 9,316 peptides analyzed in Ansari and White[20] after 20 iterations.

Figure 3 compares against two ablations, showing the gain from adding pre-training and BO where the sequence length is not allowed to change due to the use of one-hot encoding. We find that indeed both components of our algorithm help in this task, and pre-training is better at all iterations.

B. Unknown Target Matching

In the second task, the sequence length is fixed at thirteen residues. $f(x)$ is the similarity score between x and an unknown target. Similarity is measured by BLOSUM62 matrix,[21] which gives disagreement weighted by evolutionary data. This makes it so chemically similar side-chains disagreeing is less important. This task is extremely specific, so we would expect pre-training to be minimally effective or even prevent learning. This task

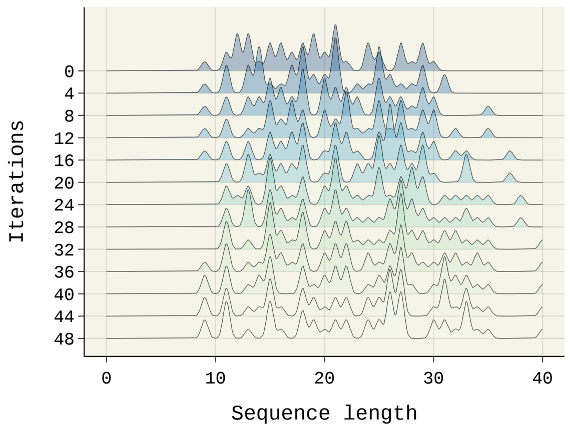

FIG. 4. Kernel density estimates of sequence length from Figure 5 algorithm as a function of iteration number. Initial sequence length is randomly sampled from a uniform distribution of [10, 30]

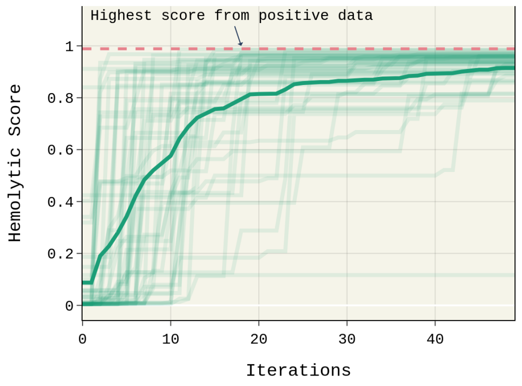

FIG. 5. Current best sequence during iterative optimization averaged across 50 runs. Different traces show individual runs. The solid line show the result averaged(mean) over 50 individual runs. Sequence length is allowed to change here. Iterations is same as number of sequences observed.

represents a worst-case for pre-training.

Model hyperparameters in this work have been tuned for scores that approximately range from 0-10 and so we transformed the score according to:

$$f(x) = \frac{b(x,t)}{A^2} - b_{min} \tag{10}$$

where b the BLOSUM62 score between x and the target t and b_{min} is the minimum possible score. The calibration (correctness of uncertainty) of the MLP is shown in Figure S2.

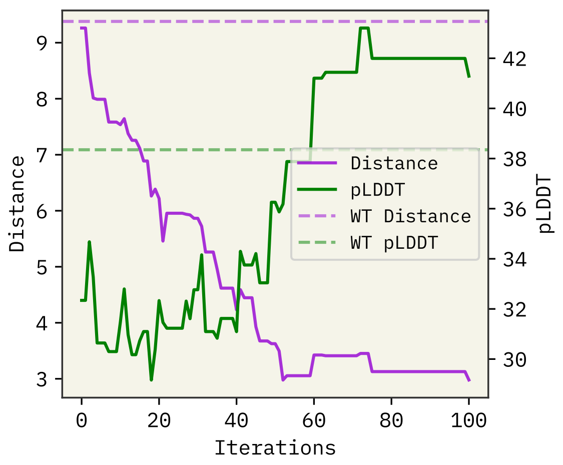

FIG. 6. Average of 10 runs of algorithm on AF2 task. The dashed line shows two scores of known binder to Ras GTPase. The optimization finds a peptide with a better score after about 50 iterations.

Figure 2 shows the results of the proposed algorithm and ablations. This task is convex (no local maximums) and has no noise, so BO is similar to direct gradient ascent of the surrogate model predictions. One-hot is direct use of the sequence in the MLP without pre-trained model. We expect this to be a better representation, since our task is highly specific, and indeed with enough data it eventually surpasses pre-training. Nevertheless, pre-training shows significant gains with fewer data points.

C. AlphaFold2 protein-peptide binding

This task is to identify a candidate peptide that binds to a target protein as evaluated with AlphaFold2.[22] The target protein is encoded by oncogene KRas G12C associated with development of cancer.[76] Margarit *et al.*[74] showed that activation of Ras GTPase is catalyzed by nucleotide exchange factor Son of Sevenless (SoS). As a result, an effective SoS inhibitor that would bind to the receptor-binding domain of the oncogenic system and preventing Ras overexpression is a therapeutic target.

We specifically used AlphaFold2-Multimer,[23] since this task is to predict simultaneously the Ras GTPase and bound peptide. Isak Johansson-Akhe[78] showed that AlphaFold2-Multimer has accuracy similar or better than other docking programs in predicting peptide-protein complexes from scores on known complexes. Using this result, Chang and Perez[79] showed a novel application of AF2-Multimer for competitive binding of different peptides to the same receptor. Following their work, we correlate good binding with the “confidence” score called predicted local distance difference test (pLDDT) output by AlphaFold2 and measuring distance to the binding site of SoS:

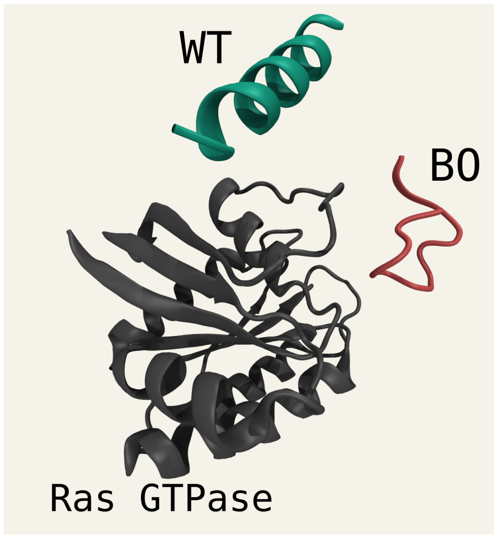

FIG. 7. Comparison of BO optimal peptide binder and the AlphaFold2-Multimer prediction of the Anupam Patgiri[77] (WT) binder. The BO sequence is KCEQCEGCDGDCD-CEAEDCEHHE.

$$f(x) = (10 - \text{RMSD}) \cdot \frac{\text{pLDDT}}{100} \tag{11}$$

where RMSD is the root mean squared distance of the peptide to the binding site evaluated at the closest backbone atom per residue pair between peptide and Ras GTPase. The transformation enables simultaneous minimization of RMSD and maximization of pLDDT, and keeps the score range between 0-10. There are more principled methods for multi-objective BO.[80] There are also other approaches to finding peptide binders that do not involve BO.[81,82]

Figure 6 shows the BO of Equation 11 averaged across 10 runs with variable sequence length. The algorithm found better binders than the known binder from Anupam Patgiri[77] (FEGIYRLELLKAEEAN) based on wild type (WT) SoS protein. Of course, these are evaluated using our score function and not experimentally validated. Nevertheless, it shows that the method can optimize complex black-box functions like finding peptide inhibitors with 10-50 evaluations. Figure 7 shows a comparison of the output from a BO run vs the WT binder. It is clear that the peptide binder is closer – agreeing with the score. The sequence also features a number of cysteines, which AlphaFold2-Multimer predict strongly interact (via disulfide bridges), and likely improve the plDDT because of the reduction in flexibility. However, this may not translate to a better affinity in an experiment.

V. CONCLUSIONS

We have shown how to use pre-trained sequence models in a BO algorithm. Across three tasks, this process enables good optimization with only a few data points. Our strategy was deep ensembled MLPs to provide calibrated uncertainties and probability distributions over sequence space to enable end-to-end differentiation. We found the commonly proposed optimization in latent space followed by decoding does not work well with few examples.

ACKNOWLEDGMENTS

Research reported in this work was supported by the National Institute of General Medical Sciences of the National Institutes of Health under award number R35GM137966. We thank the Center for Integrated Research Computing (CIRC) at University of Rochester for providing computational resources and technical support.

CODE AVAILABILITY

Code is available at `https://github.com/ur-whitelab/wazy`. Hemolytic training data is available at `https://github.com/ur-whitelab/peptide-dashboard`.

[1] A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus, Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences, Proceedings of the National Academy of Sciences **118**, e2016239118 (2021).
[2] J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, and A. Rives, Language models enable zero-shot prediction of the effects of mutations on protein function, bioRxiv 10.1101/2021.07.09.450648 (2021).
[3] E. C. Alley, G. Khimulya, S. Biswas, M. AlQuraishi, and G. M. Church, Unified rational protein engineering with sequence-based deep representation learning, Nature Methods **16**, 1315 (2019).
[4] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rihawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, *et al.*, Prottrans: towards cracking the language of life's code through self-supervised deep learning and high performance computing, arXiv preprint arXiv:2007.06225 (2020).
[5] M. Pelay-Gimeno, A. Glas, O. Koch, and T. N. Grossmann, Structure-based design of inhibitors of protein–protein interactions: mimicking peptide binding epitopes, Angewandte Chemie International Edition **54**, 8896 (2015).

[6] P. I. Frazier, A tutorial on bayesian optimization, arXiv preprint arXiv:1807.02811 (2018).

[7] D. R. Jones, M. Schonlau, and W. J. Welch, Efficient global optimization of expensive black-box functions, J. Glob. Optim. **13**, 455 (1998).

[8] L. Tallorin, J. Wang, W. E. Kim, S. Sahu, N. M. Kosa, P. Yang, M. Thompson, M. K. Gilson, P. I. Frazier, M. D. Burkart, and N. C. Gianneschi, Discovering de novo peptide substrates for enzymes using machine learning, Nature Communications **9**, 10.1038/s41467-018-07717-6 (2018).

[9] T. Lookman, P. V. Balachandran, D. Xue, J. Hogden, and J. Theiler, Statistical inference and adaptive design for materials discovery, Curr. Opin. Solid State Mater. Sci. **21**, 121 (2017).

[10] K. Wang and A. W. Dowling, Bayesian optimization for chemical products and functional materials, Current Opinion in Chemical Engineering **36**, 100728 (2022).

[11] Z. E. Hughes, M. A. Nguyen, J. Wang, Y. Liu, M. T. Swihart, M. Poloczek, P. I. Frazier, M. R. Knecht, and T. R. Walsh, Tuning materials-binding peptide sequences toward gold-and silver-binding selectivity with bayesian optimization, ACS nano **15**, 18260 (2021).

[12] (although they are equivalent in theory[83]).

[13] J. Linder and G. Seelig, Fast activation maximization for molecular sequence design, BMC Bioinformatics **22**, 510 (2021).

[14] E. Jang, S. Gu, and B. Poole, Categorical reparameterization with gumbel-softmax, arXiv preprint arXiv:1611.01144 (2016).

[15] C. J. Maddison, D. Tarlow, and T. Minka, A* sampling, Advances in neural information processing systems **27** (2014).

[16] D. J. MacKay, Probable networks and plausible predictions-a review of practical bayesian methods for supervised neural networks, Network: computation in neural systems **6**, 469 (1995).

[17] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson, What are bayesian neural network posteriors really like?, in *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 139, edited by M. Meila and T. Zhang (PMLR, 2021) pp. 4629–4640.

[18] B. Lakshminarayanan, A. Pritzel, and C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17 (Curran Associates Inc., Red Hook, NY, USA, 2017) p. 6405–6416.

[19] R. Barrett and A. D. White, Investigating active learning and meta-learning for iterative peptide design, Journal of chemical information and modeling **61**, 95 (2020).

[20] M. Ansari and A. D. White, Serverless prediction of peptide properties with recurrent neural networks, bioRxiv (2022).

[21] S. Henikoff and J. G. Henikoff, Amino acid substitution matrices from protein blocks., Proceedings of the National Academy of Sciences **89**, 10915 (1992).

[22] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, *et al.*, Highly accurate protein structure prediction with alphafold, Nature **596**, 583 (2021).

[23] R. Evans, M. O'Neill, A. Pritzel, N. Antropova, A. W. Senior, T. Green, A. Žídek, R. Bates, S. Blackwell, J. Yim, *et al.*, Protein complex prediction with alphafold-multimer, BioRxiv (2021).

[24] F. H. Arnold, Design by directed evolution, Accounts of Chemical Research **31**, 125 (1998).

[25] C. N. Bedbrook, K. K. Yang, A. J. Rice, V. Gradinaru, and F. H. Arnold, Machine learning to design integral membrane channelrhodopsins for efficient eukaryotic expression and plasma membrane localization, PLoS computational biology **13**, e1005786 (2017).

[26] Z. Ren, J. Li, F. Ding, Y. Zhou, J. Ma, and J. Peng, Proximal exploration for model-guided protein sequence design, bioRxiv (2022).

[27] Y. Gumulya, J.-M. Baek, S.-J. Wun, R. E. S. Thomson, K. L. Harris, D. J. B. Hunter, J. B. Y. H. Behrendorff, J. Kulig, S. Zheng, X. Wu, B. Wu, J. E. Stok, J. J. De Voss, G. Schenk, U. Jurva, S. Andersson, E. M. Isin, M. Bodén, L. Guddat, and E. M. J. Gillam, Engineering highly functional thermostable proteins using ancestral sequence reconstruction, Nature Catalysis **1**, 878 (2018).

[28] L. Cheng, Z. Yang, B. Liao, C. Hsieh, and S. Zhang, Odbo: Bayesian optimization with search space prescreening for directed protein evolution (2022).

[29] Z. Harteveld, J. Bonet, S. Rosset, C. Yang, F. Sesterhenn, and B. E. Correia, A generic framework for hierarchical de novo protein design, bioRxiv (2022).

[30] K. K. Yang, Z. Wu, and F. H. Arnold, Machine-learning-guided directed evolution for protein engineering, Nature Methods **16**, 687 (2019).

[31] C. N. Bedbrook, K. K. Yang, J. E. Robinson, E. D. Mackey, V. Gradinaru, and F. H. Arnold, Machine learning-guided channelrhodopsin engineering enables minimally invasive optogenetics, Nature methods **16**, 1176 (2019).

[32] B. L. Hie and K. K. Yang, Adaptive machine learning for protein engineering, Current opinion in structural biology **72**, 145 (2022).

[33] S. Biswas, G. Khimulya, E. C. Alley, K. M. Esvelt, and G. M. Church, Low-n protein engineering with data-efficient deep learning, Nature Methods **18**, 389 (2021).

[34] J. C. Greenhalgh, S. A. Fahlberg, B. F. Pfleger, and P. A. Romero, Machine learning-guided acyl-acp reductase engineering for improved in vivo fatty alcohol production, Nature communications **12**, 1 (2021).

[35] A. Khan, A. I. Cowen-Rivers, D.-G.-X. Deik, A. Grosnit, K. Dreczkowski, P. A. Robert, V. Greiff, R. Tutunov, D. Bou-Ammar, J. Wang, *et al.*, Antbo: Towards real-world automated antibody design with combinatorial bayesian optimisation, arXiv preprint arXiv:2201.12570 (2022).

[36] P. Das, T. Sercu, K. Wadhawan, I. Padhi, S. Gehrmann, F. Cipcigan, V. Chenthamarakshan, H. Strobelt, C. dos Santos, P.-Y. Chen, Y. Y. Yang, J. P. K. Tan, J. Hedrick, J. Crain, and A. Mojsilovic, Accelerated antimicrobial discovery via deep generative models and molecular dynamics simulations, Nature Biomedical Engineering **5**, 613 (2021).

[37] E. Castro, A. Godavarthi, J. Rubinfien, K. B. Givechian, D. Bhaskar, and S. Krishnaswamy, Guided generative protein design using regularized transformers, CoRR **abs/2201.09948** (2022), 2201.09948.

[38] F. Wan, D. Kontogiorgos-Heintz, and C. de la Fuente-Nunez, Deep generative models for peptide design, Digi-

tal Discovery , (2022).

[39] J.-E. Shin, A. J. Riesselman, A. W. Kollasch, C. McMahon, E. Simon, C. Sander, A. Manglik, A. C. Kruse, and D. S. Marks, Protein design and variant prediction using autoregressive generative models, Nature communications **12**, 1 (2021).

[40] D. Repecka, V. Jauniskis, L. Karpus, E. Rembeza, I. Rokaitis, J. Zrimec, S. Poviloniene, A. Laurynenas, S. Viknander, W. Abuajwa, *et al.*, Expanding functional protein sequence spaces using generative adversarial networks, Nature Machine Intelligence **3**, 324 (2021).

[41] E. Nijkamp, J. Ruffolo, E. N. Weinstein, N. Naik, and A. Madani, Progen2: Exploring the boundaries of protein language models, arXiv preprint arXiv:2206.13517 (2022).

[42] D. Hesslow, N. Zanichelli, P. Notin, I. Poli, and D. Marks, Rita: a study on scaling up generative protein sequence models, arXiv preprint arXiv:2205.05789 (2022).

[43] J. Linder, N. Bogard, A. B. Rosenberg, and G. Seelig, A generative neural network for maximizing fitness and diversity of synthetic dna and protein sequences, Cell Systems **11**, 49 (2020).

[44] N. Ferruz, S. Schmidt, and B. Höcker, A deep unsupervised language model for protein design, bioRxiv (2022).

[45] N. Anand and T. Achim, Protein structure and sequence generation with equivariant denoising diffusion probabilistic models (2022).

[46] B. E. Suzek, Y. Wang, H. Huang, P. B. McGarvey, C. H. Wu, and the UniProt Consortium, UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches, Bioinformatics **31**, 926 (2014).

[47] N. S. Detlefsen, S. Hauberg, and W. Boomsma, Learning meaningful representations of protein sequences, Nature Communications **13**, 1914 (2022).

[48] W. Wang, Z. Peng, and J. Yang, Single-sequence protein structure prediction using supervised transformer protein language models, bioRxiv (2022).

[49] D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, CoRR **abs/1409.0473** (2015).

[50] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, Scaling laws for neural language models, CoRR **abs/2001.08361** (2020), 2001.08361.

[51] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, *et al.*, Training compute-optimal large language models, arXiv preprint arXiv:2203.15556 (2022).

[52] J. A. Ruffolo, J. J. Gray, and J. Sulam, Deciphering antibody affinity maturation with language models and weakly supervised learning, arXiv preprint arXiv:2112.07782 (2021).

[53] W. P. Russ, M. Figliuzzi, C. Stocker, P. Barrat-Charlaix, M. Socolich, P. Kast, D. Hilvert, R. Monasson, S. Cocco, M. Weigt, and R. Ranganathan, An evolution-based model for designing chorismate mutase enzymes, Science **369**, 440 (2020).

[54] R. M. Neal *et al.*, Mcmc using hamiltonian dynamics, Handbook of markov chain monte carlo **2**, 2 (2011).

[55] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, A simple baseline for bayesian uncertainty in deep learning, Advances in Neural Information Processing Systems **32** (2019).

[56] A. P. Soleimany, A. Amini, S. Goldman, D. Rus, S. N. Bhatia, and C. W. Coley, Evidential deep learning for guided molecular property prediction and discovery, ACS central science **7**, 1356 (2021).

[57] T. Nguyen and A. Grover, Transformer neural processes: Uncertainty-aware meta learning via sequence modeling, in *International Conference on Machine Learning* (PMLR, 2022) pp. 16569–16594.

[58] E. J. Ma and A. Kummer, Reimplementing unirep in jax, bioRxiv (2020).

[59] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX: composable transformations of Python+NumPy programs (2018).

[60] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural computation **9**, 1735 (1997).

[61] B. E. Suzek, H. Huang, P. McGarvey, R. Mazumder, and C. H. Wu, Uniref: comprehensive and non-redundant uniprot reference clusters, Bioinformatics **23**, 1282 (2007).

[62] S. Daulton, S. Cakmak, M. Balandat, M. A. Osborne, E. Zhou, and E. Bakshy, Robust multi-objective bayesian optimization under input noise (2022).

[63] R. B. Gramacy and H. K. H. Lee, Optimization under unknown constraints (2010).

[64] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, Constrained Bayesian Optimization with Noisy Experiments, Bayesian Analysis **14**, 495 (2019).

[65] The gradient-free means it doesn't require gradients of experiments, but does of acquisition function/surrogate model.

[66] J. L. Ba, J. R. Kiros, and G. E. Hinton, Layer normalization (2016), arXiv:1607.06450 [stat.ML].

[67] S. Daulton, X. Wan, D. Eriksson, M. Balandat, M. A. Osborne, and E. Bakshy, Bayesian optimization over discrete and mixed spaces via probabilistic reparameterization, ICML2022 Workshop on Adaptive Experimental Design and Active Learning in the Real World (2022).

[68] S. Fort, H. Hu, and B. Lakshminarayanan, Deep ensembles: A loss landscape perspective, arXiv preprint arXiv:1912.02757 (2019).

[69] F. D'Angelo and V. Fortuin, Repulsive deep ensembles are bayesian, Advances in Neural Information Processing Systems **34**, 3451 (2021).

[70] P. Ramachandran, B. Zoph, and Q. V. Le, Searching for activation functions, arXiv preprint arXiv:1710.05941 (2017).

[71] S. Eger, P. Youssef, and I. Gurevych, Is it time to swish? comparing deep learning activation functions across nlp tasks, arXiv preprint arXiv:1901.02671 (2019).

[72] J. Wilson, F. Hutter, and M. Deisenroth, Maximizing acquisition functions for bayesian optimization, Advances in neural information processing systems **31** (2018).

[73] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[74] S. M. Margarit, H. Sondermann, B. E. Hall, B. Nagar, A. Hoelz, M. Pirruccello, D. Bar-Sagi, and J. Kuriyan, Structural evidence for feedback activation by ras· gtp of the ras-specific nucleotide exchange factor sos, Cell **112**, 685 (2003).

[75] M. Pirtskhalava, A. A. Amstrong, M. Grigolava, M. Chubinidze, E. Alimbarashvili, B. Vishnepolsky, A. Gabrielian, A. Rosenthal, D. E. Hurt, and

M. Tartakovsky, Dbaasp v3: database of antimicrobial/cytotoxic activity and structure of peptides as a resource for development of new therapeutics, Nucleic acids research **49**, D288 (2021).

[76] A. D. Cox, S. W. Fesik, A. C. Kimmelman, J. Luo, and C. J. Der, Drugging the undruggable ras: Mission possible?, Nature reviews Drug discovery **13**, 828 (2014).

[77] P. S. A. . D. B.-S. Anupam Patgiri, Kamlesh K Yadav, An orthosteric inhibitor of the ras-sos interaction, Nature Chemical Biology 10.1038/nchembio.612 (2011).

[78] B. W. Isak Johansson-Akhe, Benchmarking peptide-protein docking and interaction prediction with alphafold-multimer, bioRxiv 10.1101/2021.11.16.468810 (2021).

[79] L. Chang and A. Perez, Alphafold encodes the principles to identify high affinity peptide binders, bioRxiv (2022).

[80] S. Daulton, M. Balandat, and E. Bakshy, Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization, in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20 (Curran Associates Inc., Red Hook, NY, USA, 2020).

[81] C. Hsu, R. Verkuil, J. Liu, Z. Lin, B. Hie, T. Sercu, A. Lerer, and A. Rives, Learning inverse folding from millions of predicted structures, bioRxiv (2022).

[82] B. I. Wicky, L. F. Milles, A. Courbet, R. J. Ragotte, J. Dauparas, E. Kinfu, S. Tipps, R. D. Kibler, M. Baek, F. DiMaio, *et al.*, Hallucinating protein assemblies, bioRxiv (2022).

[83] A. Jacot, F. Gabriel, and C. Hongler, Neural tangent kernel: Convergence and generalization in neural networks, Advances in neural information processing systems **31** (2018).