OPTIMIZATION, LEARNING AND GENERATION FOR PROTEINS: DOCKING STRUCTURES AND MAPPING SEQUENCE–FUNCTION RELATIONSHIPS

A Thesis

by

YUE CAO

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Yang Shen
Committee Members,	Xiaoning Qian
	Krishna Narayanan
	Rui Tuo
Head of Department,	Miroslav M. Begovic

December 2021

Major Subject: Electrical Engineering

ABSTRACT

Proteins are the workhorse molecules of lives. Understanding how proteins function is one of the most fundamental problems in molecular biology, which can drive a plethora of biological and pharmaceutical applications. However, the experimental determination of protein mechanisms is expensive and time-consuming. Such a gap motivates developing computational methods for protein science. The goal of this thesis is to investigate to what extent machine learning can uncover the underlying mechanisms of proteins. We concentrate on two primitives: predicting the 3D structures of protein–protein interactions (called protein docking) and understanding the protein sequence–function relationships. Accordingly, we organize the thesis as follows:

First, we study protein docking. We introduce Bayesian Active Learning (BAL), the first optimization algorithm with uncertainty quantification (UQ) for protein docking. Extensive experiments demonstrated the superior performance of BAL against competitors on both optimization and UQ. In addition, we generalize BAL into the realm of meta-learning and propose LOIS: Learning to Optimize in Swarms. LOIS outperforms various optimization algorithms for general optimization tasks. Finally, we focus on the scoring problem in protein docking and introduce Energy-based Graph Convolutional Networks (EGCN) that directly learns energies from graph representations of docking models, which performed better than competitors.

Second, we focus on understanding the protein sequence–function relationship. We first study the forward protein function prediction and introduce TALE: Transformer-based protein function Annotation with joint sequence-Label Embedding. Combining TALE and a sequence similarity-based method, TALE+ outperformed competing methods when only sequence input is available. We also study the inverse design and describe our novel conditional autoregressive deep generative models. By learning the functional embeddings from Gene Ontology (GO) graph as conditional inputs, our conditional autoregressive models were able to model the distributions of protein sequences for given functions.

DEDICATION

To my girlfriend, my mother, my father and my grandmother.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Yang Shen, Professor Xiaoning Qian and Professor Krishna Narayanan of the Department of Electrical and Computer Engineering and Professor Rui Tuo of the Department of Industrial and Systems Engineering.

The experiments and results in Section 2.4.6 were developed, conducted, assessed and analyzed by members in Dr. Huiping Liu's group of the Feinberg School of Medicine at Northwestern University.

The attention mechanism in Section 3.3.5 was developed by Tianlong Chen of the Department of Electrical and Computer Engineering at the University of Texas at Austin.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

Graduate study was supported by

- National Science Foundation (NSF) CCF-1546278: "Dimension Reduction and Optimization Methods for Flexible Refinement of Protein Docking" PI: Yang Shen
- National Institutes of Health (NIH) R35GM124952: "Unraveling Molecular and System-level Mechanisms of Human Disease-Associated Protein Mutation" PI: Yang Shen

NOMENCLATURE

UQ	Uncertainty Quantification
BAL	Bayesian Active Learning
cNMA	complex-based Normal Mode Analysis
RMSD	Root-Mean-Square-Deviation
MCMC	Markov Chain Monte Carlo
RF	Random Forest
AUC	Area Under the Curve
ROC	Receiver Operating Characteristic
LOIS	Learning to Optimize in Swarms
LSTM	Long Short-Term Memory
EGCN	Energy-based Graph Convolutional Networks
TALE	Transformer-based protein function Annotation with joint sequence-Label Embedding
GO	Gene Ontology
DAG	Directed Acyclic Graph
CASP	Critical Assessment of protein Structure Prediction
CAPRI	Critical Assessment of PRedicted Interactions
CAFA	Critical Assessment of Function Annotation
MSI	Maximum Sequence Identity
PPI	Protein–Protein Interactions
HMM	Hidden Markov Models
MSA	Multiple Sequence Alignment

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Motivation

Proteins are the building blocks of lives. Proteins participate in almost every cellular activity [3]. For instance, enzyme proteins promote intracellular chemical reactions by providing intricate molecular surfaces. Membrane proteins transport ions or small molecules across the membrane. Signal proteins carry signals from one cell to another in order to control and coordinate physiological functions. There are many other proteins that act as antibodies, toxins, elastic fibers, to name a few. The variety of protein functions is directly attributed to a wide variety of their 3D structures, which are uniquely determined by their 1D sequences [4]. Therefore, understanding protein sequence-structure-function relationships can reveal deep insights into how proteins work and therefore, provide solid foundations for a number of biological and pharmaceutical applications.

However, as proteins are one of the most structurally complex and functionally sophisticated molecules, such relationships are of extreme difficulty to learn. First, although Anfinsen et al. well established the thermodynamics of protein folding that the native state of a protein corresponds to the system's lowest Gibbs free energy [4], solving the energy minimization problem theoretically is daunting due to non-convex energy minimization in the high-dimensional space of protein conformations (at least thousands of atoms). Secondly, the experimental determination of protein structures and functions is still low-throughput, which is significantly outpaced by the sequence determination, creating a huge knowledge gap between sequences and their properties. According to the newest release of UniProt [5] and Protein Data Bank [6], there are around 5.6×10^5 nonredundant sequences manually annotated in Swiss-Prot, and around 1.7×10^5 entries in RCSB Protein Data Bank, but over two orders of magnitude more (around 1.8×10^8 sequences) are awaiting full manual annotation in TrEMBL.

The weakness of theoretical and experimental approaches motivates the need of computa-

tional methods for deciphering the protein sequence-structure-function relationships. Over several decades, template-based methods have become the most popular and successful computational methods for both structure prediction and function annotation [7, 8, 9, 10], where a protein with similar sequence to the target and available structure/function is used as the template . The fundamental assumption is that similar inputs (sequences) imply similar outputs (structure, function). However, template-based methods are usually useful in the homologous region where a good homologous template can be found. When it comes to the "mid-night" zone or even "twilight" zone, where similarity to known annotated sequences is too low to sustain the assumption, the template-based methods often failed. Unfortunately, a large portion of sequences are falling in these two zones.

The aforementioned gap motivates us to predict protein properties directly from the scratch. Recently, artificial intelligence, especially machine learning, has revolutionized many fields. For instance, convolutional neural networks dominate the field of image recognition [11]. Reinforcement learning mastered the game of Go and beat human experts [12]. Attention mechanism [13] became the state-of-art method for natural language processing (NLP). The success of ML in the computer vision (CV) and NLP are due to the large-scale datasets and the smartly-designed ML algorithms/architectures. As protein data is also rich, it has witnessed a growing interests in applying existing ML algorithms to protein science [14, 15, 16, 17, 18]. Compared to images and texts, protein data have their unique properties. First, proteins have their unique "grammars", which are defined by the chemical and physical interactions within their structures and in the environment. Second, protein data are multi-modal. Depending on the specific goals, proteins can be represented as 1D sequences, 2D images, 3D point clouds or graphs. Such unique properties provide both challenges and opportunities for applying machine learning algorithms to studying protein science.

Based on the aforementioned vision, in this thesis, we set up one goal and focus on two primitives: **First**, our goal is to explore to what extent machine learning can uncover the protein sequence-structure-function relationship, through designing novel protein-specific machine

learning models. **Second**, towards the goal we concentrate on two primitives: predicting the 3D structures of protein complexes (called protein docking) and understanding the protein sequence–function relationship.

1.2 Summary of Contributions

In the end, we summarize our contributions:

- First, we make contributions in optimization for protein docking. We introduce a novel algorithm, Bayesian Active Learning (BAL) [19], for optimization and uncertainty quantification (UQ) of black-box functions with applications to flexible protein docking. BAL directly models the posterior distribution of the global optimum (i.e. native structures) with active sampling and posterior estimation iteratively feeding each other. BAL significantly improves against starting points from rigid docking and refinements by particle swarm optimization. Quality assessment empowered with UQ leads to tight quality intervals. BAL's estimated probability of a prediction being near-native achieves binary classification AUROC at 0.93 and AUPRC over 0.60 (compared to 0.50 and 0.14, respectively, by chance), which also improves ranking predictions. This study represents the first UQ solution for protein docking, with rigorous theoretical frameworks and comprehensive empirical assessments. Moreover, we demonstrate the power of BAL on several experimental applications. We have used BAL for modelling the homodimer surface of the CD44 glycoprotein on cancer stem cells [20], the inter cellular Adhesion Molecule 1 (ICAM1) [21], and the heterdimer interface between CD44 and the P21-activated kinase 2 (PAK2) [22].

- Second, we generalize the idea of BAL into meta-learning for general optimization problems. We propose Learning to Optimize in Swarms (LOIS) [23]. LOIS learns in the algorithmic space of both point-based and population-based optimization algorithms. The meta-optimizer targets at a meta-loss function consisting of both cumulative regret and entropy. Specifically, we learn and interpret the update formula through a population of LSTMs embedded with sample- and feature-level attentions. Meanwhile, we estimate the posterior

directly over the global optimum and use an uncertainty measure to help guide the learning process. Empirical results over non-convex test functions and the protein-docking application demonstrate that this new meta-optimizer outperforms existing competitors.

- Third, we study the second challenge in protein docking: scoring. We represent protein and complex structures as intra- and inter-molecular residue contact graphs with atom-resolution node and edge features. And we propose a novel graph convolutional kernel that aggregates interacting nodes' features through edges so that generalized interaction energies can be learned directly from 3D data. The resulting energy-based graph convolutional networks (EGCN) [24] with multihead attention are trained to predict intra- and inter-molecular energies, binding affinities, and quality measures for encounter complexes. Compared to a state-of-the-art scoring function for model ranking, EGCN improves ranking for a critical assessment of predicted interactions (CAPRI) test set involving homology docking.

- Fourth, we switch to the second primitive, and focus on sequence-based protein function prediction. We propose a novel deep learning model, named Transformer-based protein function Annotation through joint sequence–Label Embedding (TALE) [25]. TALE embed protein function labels (hierarchical GO terms on directed graphs) together with inputs/features (sequences) in a joint latent space. Combining TALE and a sequence similarity-based method, TALE+ outperformed competing methods when only sequence input is available. It even outperformed a state-of-the-art method using network information besides sequence, in two of the three gene ontologies. Furthermore, TALE and TALE+ showed superior generalizability to proteins of low homology and never/rarely annotated novel species or functions compared to training data, revealing deep insights into the protein sequence–function relationship.

- Lastly, we study the inverse function-to-sequence relationship. Specifically, we focus on designing functional protein sequences and predicting the effect of sequence variants. To achieve this, we first propose a novel graph embedding method for embedding the Gene On-

tology graphs using both node descriptions and graph topology. We then propose a conditional autoregressive model for predicting the distributions of protein sequences given functions. We first assess our model in mutational effect prediction. By fine-tuning on the dataset within each protein family, our model can beat the state-of-the-art alignment-free method in 35 out of 40 cases. We then assess the generated sequences of our fine-tuned model on a specific DNA binding domain. We found our model can not only generate natural-like, functional-relevant sequences, but also generate more diverse sequences than the natural ones, therefore, our model shows the potential of expanding the existing sequence space for given functions within the biological constraints of such functions.

1.3 Outline

The rest of the thesis is organized as follows:

- From Chapter 2 to Chapter 4, we will focus on our first primitive: protein docking. Specifically,
  - In Chapter 2, we will study the optimization and uncertainty quantification of protein docking. We will introduce the Bayesian Active Learning (BAL) for optimization and uncertainty quantification on protein docking.
  - In Chapter 3, we will extend the BAL into meta-learning and introduce Learning to Optimize in Swarms (LOIS).
  - In Chapter 4, we will study the scoring problem in protein docking and propose Energy-based Graph Convolutional Networks (EGCN).
- From Chapter 5 to Chapter 6, We will focus on our second primitive: protein sequence–function relationship. Specifically,
  - In Chapter 5, we will study the forward problem of sequence-to-function prediction and introduce TALE, short for Transformer-based protein function Annotation through joint sequence–Label Embedding.

  - In Chapter 6, We will study the inverse problem of function-to-sequence design and describe our conditional autoregressive generative model for predicting the effect of protein variants and generating protein sequences for specific functions.

- In Chapter 7, we will conclude the thesis and propose the future work. We will also talk about the broader impact of this thesis.

2. Optimization and Uncertainty Quantification in Protein Docking*

2.1 Preface

Protein-protein interactions underlie many cellular processes, which has been increasingly revealed by the quickly advancing high-throughput experimental methods. Structural information about these interactions often helps reveal their mechanisms, understand diseases, and develop therapeutics. However, compared to the binary information about which protein pairs interact, the structural knowledge about how proteins interact remains relatively scarce [26]. Protein docking helps close such a gap by computationally predicting the 3D structures of protein-protein complexes given individual proteins' 1D sequences or 3D structures [27].

We focus on *Ab initio* protein docking, where only two individual unbound structures/homologs are used without the complex template. Based on the thermodynamic theorem [4], the native structure of protein complex stays at the lowest Gibbs free energy. Therefore, *Ab initio* protein docking is often recast as an energy minimization problem. This poses a great challenge in protein docking: **optimization**, where the task is aiming at finding the native structure given the energy function.

In this chapter, we study the optimization challenge in protein docking. To the end, we propose a novel optimization algorithm with uncertainty quantification (UQ) for protein docking: Bayesian Active Learning (BAL).

2.2 Introduction

As discussed before, *Ab initio* protein docking can be regarded as an energy (or other objective functions) optimization problem. Usually, for the type of objective functions relevant to protein docking, neither the analytical form nor the gradient information would help global optimization as

*Reprinted with permission from "Bayesian Active Learning for Optimization and Uncertainty Quantification in Protein Docking" by Yue Cao and Yang Shen, 2020. J. Chem. Theory Comput, 16, 8, 5334–5347. COPYRIGHT 2020 AMERICAN CHEMICAL SOCIETY.

the functions are non-convex and extremely rugged thus their gradients are too local to inform the global landscapes. So the objective functions are often treated as *de facto* "black-box" functions for global optimization. Meanwhile, these functions are very expensive to evaluate. Various protein-docking methods, especially refinement-stage methods, have progressed to effectively sample the high-dimensional conformational space against the expensive functional evaluations [28, 29, 30, 31, 32, 33, 34].

While solving such optimization problems still remains a great challenge, quantifying the uncertainty of numerically-computed optima (docking solutions) is even more challenging and has not been addressed by any protein-docking method. Even though the uncertainty information is much needed by the end users, current protein-docking methods often generate a rank-ordered list of results without giving quality estimation with uncertainty to individual results and without providing the confidence in whether the entire list contains a quality result (for instance, a near-native protein-complex model with iRMSD $\leqslant 4$ Å).

Uncertainty quantification (UQ), if addressed, would lead to two benefits. First, for individual optimization trajectories, uncertainty awareness would improve the robustness of their optimization outcomes. Second, uncertainty of the solutions can be easily fed to any quality assessment tools for distributions rather than point estimates of some quality of interest (very often iRMSD); and comparing these distributions would provide more robust model ranking across trajectories.

In this study, we introduce a rigorous Bayesian framework to simultaneously perform function optimization and uncertainty quantification for expensive-to-evaluate black-box objective functions. To that end, our Bayesian active learning (BAL) iteratively and adaptively generates samples and updates posterior distributions of the global optimum. Specifically, we propose a posterior in the form of the Boltzmann distribution building upon a non-parametric kriging regressor and a novel adaptive-annealing schedule. The iteratively updated posterior carries the belief (and uncertainty as well) on where the global optimum is given historic samples and guides next-iteration sampling, which presents an efficient data-collection scheme for both optimization and UQ. Compared to typical Bayesian optimization methods [35] that first model the posterior of the objective

function and then optimize the resulting functional, our BAL framework directly models the posterior of the global optimum and overcomes the intensive computations in both steps of typical Bayesian optimization methods. Compared to another work [36] that also models the posterior of the global optimum, Nonparametric Conjugate Prior Distribution (or NCPD in short), we provide both theoretical and empirical results that our BAL has a consistent and unbiased estimator as well as a global uncertainty-aware and dimension-dependent annealing schedule.

We also make innovative contributions in the application domain of protein docking. Specifically, we design a machine learning-based objective function that estimates binding affinities for docked encounter complexes as well as assesses the quality of interest, iRMSD, for docking results. And we re-parameterize the search space for both external rigid-body motions [37] and internal flexibility [30], into a low-dimensional homogeneous and isotropic space suitable for high-dimensional optimization, using our (protein) complex normal modes (cNMA) [38]. Considering that protein docking refinement often starts with initial predictions representing separate conformational clusters/regions, we use estimated local posteriors over individual regions to construct local and global partition functions; and then calculate the probability that the prediction for each conformation, each conformational cluster, or the entire list of conformational clusters is near-native.

2.3 Bayesian Active Learning for Optimization and Uncertainty Quantification in Protein Docking

2.3.1 Mathematical Formulation

We consider a black-box function $f(\boldsymbol{x})$ (e.g. ΔG, the change in the Gibbs free energy upon protein-protein interaction) that can only be evaluated at any sample with an expensive yet noisy observation $y(\boldsymbol{x})$ (e.g. modeled energy difference or scoring function for conformation $\boldsymbol{x}$). Our goal in optimization is

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$$

(e.g. $\boldsymbol{x}^*$ denotes the native structure of a protein complex and $\mathcal{X}$ denotes the domain of sample space for conformations). And our goal in uncertainty quantification is the probability distribution of $\boldsymbol{x}^*$ around its prediction $\hat{\boldsymbol{x}}$, rather than the single point estimation $\hat{\boldsymbol{x}}$ itself (in which the distributions is equivalent to an impulse function at $\hat{\boldsymbol{x}}$).

Once the uncertainty of the solution is quantified, the uncertainty of the solution quality can be subsequently derived. A summary of the latter is simply the probability that the prediction $\hat{\boldsymbol{x}}$ falls in an interval $[lb, ub]$ of quality relative to $\boldsymbol{x}^*$:

$$P\left(lb \leqslant Q(\hat{\boldsymbol{x}}, \boldsymbol{x}^*) \leqslant ub\right) = 1 - \sigma,$$

where $1 - \sigma$ is the confidence level; $[lb, ub]$ is the confidence interval in the solution quality; and $Q(\cdot, \cdot)$, the quality of interest measuring some distance or dissimilarity, can be an Euclidean norm (as in our assessment for test functions), another distance metric, or other choices dependent on the user (for instance, iRMSD as in our assessment for protein docking with $ub = 4$ Å). Note that $Q(\cdot, \cdot)$ can be any quality assessment (QA) tool that does not assume the knowledge of $\boldsymbol{x}^*$ as well.

2.3.2 Bayesian active learning with a posterior of $\boldsymbol{x}^*$

We address the problem above in a Bayesian perspective: instead of treating $\boldsymbol{x}^*$ as a fixed point, we model $\boldsymbol{x}^*$ as a random variable and construct its probability distribution, $p(\boldsymbol{x}^*|\mathcal{D})$, given samples $\mathcal{D} = \{(\boldsymbol{x}, y)\}$. This probability distribution, carrying the belief and the uncertainty on the location of $\boldsymbol{x}^*$, is a prior when $\mathcal{D} = \varnothing$ (no sample) and a posterior otherwise. Considering the cost of function evaluation, we iteratively collect new samples in iteration t (where all samples collected by the end of the t-th iteration are denoted $\mathcal{D}^{(t)}$) based on the latest estimated posterior, $p(\boldsymbol{x}^*|\mathcal{D}^{(t-1)})$; and we update the posterior $p(\boldsymbol{x}^*|\mathcal{D}^{(t)})$ based on $\mathcal{D}^{(t)}$. An illustration of the iterative approach is given in Fig. 2.1.

For optimization, we set $\hat{\boldsymbol{x}}$ to be the best sample with the lowest y value given a computational budget (reflected in the number of samples or iterations). For UQ, given the posterior $p(\boldsymbol{x}^*|\mathcal{D}^{(t)})$ in the final iteration, one can propagate the inferred uncertainty in $\boldsymbol{x}^*$ forwardly to that in the quality

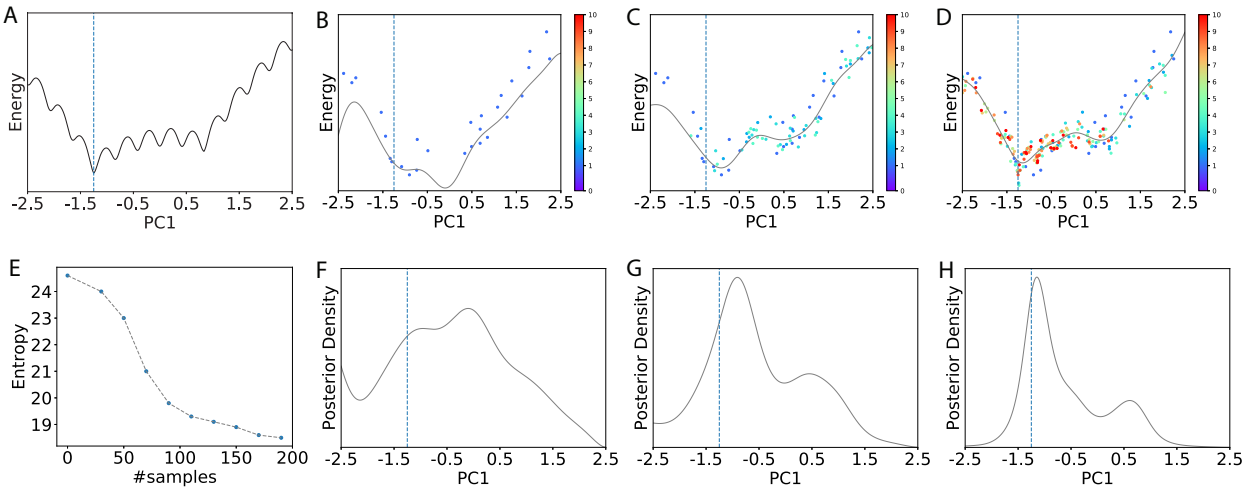


Figure 2.1: Illustration of the Bayesian active learning (BAL) algorithm. (A): A typical energy landscape projected onto the first principal component (PC1) just for visualization, using all samples collected. The dashed line indicates the location of the optimal solution. (B)-(D): The samples (dots) and the kriging regressors (light curves) in the 1st, 4th and 10th iteration, respectively. Samples are colored from cold to hot for increasing iteration indices and those in the same iteration have the same color. (E): The entropy (measuring uncertainty) of the posterior reduces as the number of samples increases. Its quick drop, as the number of samples increases from 30 to 100, corresponds to a drastic change of the kriging regressor, which suggests increasing exploitation in possible function basins. After 100 samples, the entropy goes down more slowly, echoing the smaller updates of the regressor. (F)-(H): The corresponding posterior distributions for (B)-(D), respectively.

of interest, $Q(\hat{\boldsymbol{x}}, \boldsymbol{x}^*)$, for the found $\hat{\boldsymbol{x}}$, using techniques such as Markov chain Monte Carlo.

2.3.3 Non-parametric posterior of $\boldsymbol{x}^*$

We propose to use the Boltzmann distribution to describe the posterior

$$p(\boldsymbol{x}^*|\mathcal{D}^{(t)}) \propto \exp(-\rho \cdot \hat{f}(\boldsymbol{x}))$$

where $\hat{f}(\boldsymbol{x})$ is an estimator for $f(\boldsymbol{x})$, and ρ is a parameter (sometimes $\frac{1}{RT}$ where R is the gas constant and T the temperature of the molecular system).

To iteratively guide the expensive sampling and balance between exploration and exploitation in a data efficient way, we choose ρ to follow an adaptive annealing schedule over iteration t:

$$\rho_t = \rho_0 \cdot \exp((h_p^{(t-1)})^{-1} n_t^{\frac{1}{d}})$$

where ρ_0, the initial ρ, is a parameter; $h_p^{(t-1)}$ is the (continuous) entropy of the last-iteration posterior, a shorthand notation for $h(p(\boldsymbol{x}^*|\mathcal{D}^{(t-1)}))$; $n_t = |\mathcal{D}^{(t)}|$ is the number of samples collected so far; and d is the dimensionality of the search space $\mathcal{X}$.

This annealing schedule is inspired by the adaptive simulated annealing (ASA) [39], especially the exponential form and the $n_t^{\frac{1}{d}}$ term. However, we use the $(h_p^{(t-1)})^{-1}$ term rather than a constant as in ASA so that we exploit all historic samples $\mathcal{D}^{(t)}$. In this way, as the uncertainty of $\boldsymbol{x}^*$ decreases, ρ_t increases and shifts the search toward exploitation.

The function estimator $\hat{f}(\boldsymbol{x})$ also updates iteratively according to the incrementally increasing n_t samples $\mathcal{D}^{(t)} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_t}$. We use a consistent and unbiased kriging regressor [40] which is known to be the best unbiased linear estimator (BLUE):

$$\hat{f}(\boldsymbol{x}) = f_0(\boldsymbol{x}) + (\boldsymbol{\kappa}^{(t)}(\boldsymbol{x}))^T (\boldsymbol{K}^{(t)} + \epsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{y}^{(t)} - \boldsymbol{f}_{\boldsymbol{0}}^{(t)})$$

where $f_0(\boldsymbol{x})$ is the prior for $E[f(\boldsymbol{x})]$; $\boldsymbol{\kappa}^{(t)}(\boldsymbol{x}) \in \mathcal{R}^{n_t}$ is the kernel vector with the ith element being the kernel, a measure of similarity, between $\boldsymbol{x}$ and $\boldsymbol{x}_i \in \mathcal{D}^{(t)}$; $\boldsymbol{K}^{(t)} \in \mathcal{R}^{n_t \times n_t}$ is the kernel

matrix with the (i, j) element being the kernel between $\boldsymbol{x}_i \in \mathcal{D}^{(t)}$ and $\boldsymbol{x}_j \in \mathcal{D}^{(t)}$; $\boldsymbol{y}^{(t)}$ and $\boldsymbol{f}_{\boldsymbol{0}}^{(t)}$ are the vector of $y_1, \ldots, y_{n_t}$ and $f_0(\boldsymbol{x}_1), \ldots, f_0(\boldsymbol{x}_{n_t})$, respectively; and ϵ reflects the noise in the observation and is estimated to be 2.1 as the prediction error for the training set.

We derive the kriging regressor in Appendix A.3. We will use the regressor to evaluate binding energy and estimate iRMSD for UQ over multiple regions in Sec. 2.3.2.9.

2.3.4 Adaptive sampling based on the latest posterior

For a sequential sampling policy that balances exploration and exploitation during the search for the optimum, we choose Thompson sampling [41] which samples a batch of points in the t-th iteration based on the latest posterior $p(\boldsymbol{x}^*|\mathcal{D}^{(t-1)})$. This seemingly simple policy has been found to be theoretically [42] and empirically [43] competitive compared to other updating policies such as Upper Confidence Bound [35]. In our case, it is actually straightforward to implement given the posterior on $\boldsymbol{x}^*$.

There are multiple reasons to collect in each iteration a batch of samples rather than a single one. First, given the high dimension of the search space, it is desired to collect adequate data before updating the posterior. Second, the batch sampling weakens the correlation among samples and make them more independent, which benefits the convergence rate of the kriging regressor. Last, parallel computing could be trivially applied for batch sampling, which would significantly improve the algorithm throughput.

Fig. 2.1 gives an illustration of the algorithm behavior. The initial samples drawn from a uniform distribution leads to a relatively flat posterior whose maximum is off the function optimum (Fig. 2.1F). As the iteration progresses, the uncertainty about the optimum gradually reduces(Fig. 2.1E) and newer samples are increasingly focused (Fig. 2.1C,D) as the posteriors are becoming narrower with peaks shifting toward the function optimum (Fig. 2.1G,H).

In our docking study, $d = 12$ for a homogeneous space spanned by complex normal modes (see Sec. 2.3.2.7). We construct a prior and collect 30 samples in the first iteration and 20 in each of the subsequent iterations. We limit the number of iterations (samples) to be 31 (630) for optimization and posteriors as a way to impose a computational budget (6–13 CPU hours for protein complexes

of typical sizes). The reason is that it costs minutes to locally minimize each conformational sample using CHARMM [44] and remove bond distortions in flexible perturbations, before energies can be evaluated. More samples, albeit more expensive, would improve the quality of energy minimization and posterior estimation. At the end of all iterations, an additional set of 1,000,000 samples will be generated according to the final posterior, for quality estimation and uncertain quantification of the final prediction. Note that these 1,000,000 samples do not drive the optimization process and do not need to be locally minimized anymore; and this stage of post-optimization UQ costs about a CPU hour.

2.3.5 Kernel with customized distance metric

The kernel in the kriging regressor for the posterior is a measure of similarity. For test functions defined in an Euclidean space, we use the radial basis function (RBF) kernel:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{2l^2}),$$

where $||\boldsymbol{x}_i - \boldsymbol{x}_j||$, a measure of dissimilarity, is the Euclidean distance and l, the bandwidth of the kernel, is set as $l = l_0 \cdot n_t^{\frac{1}{d}}$ following [45, 45]. l_0, dependent on search space, is set at 2.0 for docking without particular optimization.

For protein docking we replace the Euclidean distance in the RBF kernel with the interface RMSD (iRMSD) between two sample structures. iRMSD captures sample dissimlarity relevant to function-value dissimilarity and is independent of search-space parameterization. For this purpose, we also have to address two technical issues. First, protein interface information is determined by $\boldsymbol{x}^*$ and thus unknown. We instead use the putative interface seen in the samples. Each iRMSD is calculated using the same set of C_α atoms, the union of interface C_α atoms derived from 50 random perturbations of the starting structure (see more details in the **SI** Sec. 2.4 of [19]). Second, kernel calculation with iRMSD is time consuming. The time complexity of iRMSD calculation is $O(N)$ and that of regressor update is $O(Nn^2)$, where N, the number of interfacial atoms, can easily reach hundreds or thousands, and n, the number of samples, can also be large. To save computing time,

we develop a fast RMSD calculation method that reduces its time complexity from $O(N)$ down to $O(1)$ (see details in **SI** Sec. 2.1 of [19]).

2.3.6 Related methods

Current Bayesian optimization methods typically model the posterior distribution of $f(\boldsymbol{x})$ rather than that of $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$ directly. After modeling the posterior distribution over the functional space (a common non-parametric way is through Gaussian processes), they would subsequently sample the functional space and optimize sample functions. For instance, [46] used Monte Carlo sampling; and [47] discretized the functional space to approximate the sample paths of Gaussian processes using a finite number of basis functions then optimized each sample path to get one sample of $\boldsymbol{x}^*$. The two-step approach of current Bayesian optimization methods involve intensive sampling and (non-convex) optimization that is computationally intensive and not pragmatically useful for protein docking.

To our knowledge, [36] presented the only other study that directly models the posterior distribution over the optimum [36]. Both their method NCPD and our BAL fall in the general category of Bayesian optimization and use consistent non-parametric regressors. However, we prove in Sec. 1.2 of Appendix A that their regressor is biased whereas our kriging regressor is unbiased. We explain in Sec. 1.1 of the Appendix A that their annealing schedule (temperature control) only considers the pairwise distance between samples without location awareness and is independent of dimensionality d; whereas ours has a term involving location-aware global uncertainty and generalizes well to various dimensions. Beyond those theoretical comparisons, we also included empirical results to show the superior optimization and UQ performances of BAL.

The rest of the Materials and Methods section involve methods specific to the protein docking problem: parameterization, dimensionality reduction, and range reduction of the search space $\mathcal{X}$; machine learning model as $y(\boldsymbol{x})$, i.e., an energy model for encounter complexes; quality assessment with uncertainty quantification for a predicted structure or a list of predictions; and the use of such assessment metrics for scoring purposes: ranking predictions or classifying their nativeness.

2.3.7 Conformational Sampling in $\mathcal{X}$

In protein docking the full search space $\mathcal{X}$ captures the degrees of freedom for all atoms involved. Let one protein be receptor whose position is fixed and the other be ligand (the larger one is often chosen as the receptor, as done in a protein-docking benchmark set [48]). And let N_{R}, N_{L}, and N be the number of atoms for the receptor, the ligand, and the complex respectively. Then $\mathcal{X} = \mathbb{R}^{3N-6}$ is a Euclidean space whose dimension easily reaches 10^4 for a small protein complex without surrounding solvent molecules. If accuracy is sacrificed for speed, proteins can be (unrealistically) considered rigid and $\mathcal{X} = SE(3) = \mathbb{R}^3 \times SO(3)$ is a Riemannian manifold [49] of ligand translations and rotations. Docking methods fall in the spectrum between these two ends that are represented by all-atom molecular dynamics and FFT rigid docking, respectively. For instance, one can consider locally rigid pieces of a protein rather than a globally rigid protein, then $\mathcal{X}$ becomes the product of many $SE(3)$ for local rigidity [50]; or one can model individual proteins' internal flexible-body motions using normal modes on top of the ligand rigid-body motions, thus $\mathcal{X}$ becomes the product of $\mathbb{R}^K$ (where $K \ll N_{\mathrm{R/L}}$) and $SE(3)$ [29].

From the perspective of optimization and UQ, both the high-dimensionality of $\mathbb{R}^{3N-6}$ and the geometry of the lower-dimensional manifold present challenges. Almost all dimensionality reduction efforts in protein docking impose conditions (such as aforementioned local or global rigidity) in the full Euclidean space and lead to embedded manifolds difficult to (globally) optimize over. The challenge from the manifold has been either disregarded in protein docking or addressed by the local tangent space [49, 37, 50].

Could and how could the dimensionality of the conformational space be reduced while its geometry maintains homogeneity and isotropy of a Euclidean space and its basis vectors span conformational changes of proteins upon interactions? In this subsection we give a novel approach to answer this question for the first time. In contrast to common conformational sampling that separates internal flexible-body motions (often Euclidean) and external rigid-body motions (a manifold) [32], we re-parameterize the space into a Euclidean space spanned by complex normal modes [38] blending both flexible- and rigid-body motions. The mapping preserves distance metric in the

original full space. We further reduce the dimensionality and the range in the resulting space[51].

Complex normal modes blend flexible- and rigid-body motions We previously introduced complex normal mode analysis, cNMA [38], to model conformational changes of proteins during interactions. Using encounter complexes from rigid docking, cNMA extends anisotropic network model (ANM) to capture both conformational selection and induced fit effects. After the Hessian matrix is projected to remove the rigid-body motion of the receptor, its non-trivial eigenvectors $\boldsymbol{\mu}_j$ ($j = 1, \ldots, 3N-6$) form orthonormal basis vectors. We showed that $\boldsymbol{\mu}_j^{\mathrm{R}}$, the components of the complex normal modes, better capture the direction of individual proteins' conformational changes than conventional NMA did [38]. We also showed that the re-scaled eigenvalues for these components, $\lambda_j^{\mathrm{R}} = \frac{\lambda_j}{||\boldsymbol{\mu}_j^{\mathrm{R}}||^2}$, can be used to construct features for machine learning and predict the extent of the conformational changes.

Dimensionality reduction In this study we focus on the motions of a whole complex rather than individual proteins and develop sampling techniques for protein docking. Each complex normal mode [38] simultaneously captures concerted flexible-body motions of individual proteins (receptor and ligand) and rigid-body motion of the protein whose position is not fixed (ligand). Such modes together span a homogeneous and isotropic Euclidean space where the distance between two points is exactly the RMSD between corresponding complex structures. The Euclidean space is friendly to high-dimensional optimization. In this study, complex normal modes are pre-computed using the starting structure of each conformational cluster and not updated while sampling the cluster to save computational costs.

For dimensionality reduction in the resulting space, we choose the first K_1 non-trivial eigenvectors $\boldsymbol{\mu}_j$ ranked by increasing eigenvalues λ_j; and we additionally include K_2 $\boldsymbol{\mu}_j$ (not in the first K_1) ranked by increasing λ_j^R (λ_j rescaled using the receptor's contribution to this complex normal mode $\boldsymbol{\mu}_j$ [38]). In other words, we sample in a $(K_1 + K_2)$-dimensional Euclidean space spanned by complex normal modes and denote the set of basis vectors as $\mathcal{B}$. K_1 of these complex normal modes are the slowest for the whole complex (judging by λ_j) and the rest K_2 different ones are the slowest for the receptor portion of the complex (judging by λ_j^R). In this study K_1 and K_2 are set

at 9 and 3, respectively, leading to the dimension of the reduced space to $d = 12$. Empirically, we find that the first 9 non-trivial complex normal modes often contain six with dominant rigid-body motions of the ligand and three with dominant ligand flexibility; and the other 3 in the basis set are, by definition, with dominant receptor flexibility. .

Our framework of Bayesian active learning, using kernels in its unbiased kriging regressor, is applicable to any choice of the basis set $\mathcal{B}$. Naturally, it faces more challenge in optimization, let alone uncertainty quantification, as the dimensionality of $\mathcal{B}$ increases (see empirical results for test functions in Sec. 2.3.3.1). Although current basis vectors are low-frequency backbone flexibility derived from normal mode analysis, more vectors can be considered for the set $\mathcal{B}$, such as higher-frequency normal modes, local conformational rearrangements including loop and helix motions, and large conformational changes such as hinge motions. In this work, side-chain flexibility is considered by locally minimizing every conformational sample $\boldsymbol{x}$.

Range reduction For range reduction in the dimension-reduced space, we perturb a starting complex structure $\vec{C}_0 \in \mathbb{R}^{3N-6}$ along aforementioned basis vectors to generate sample $\vec{C}$ while enforcing a prior on the scaling factor s in the first iteration. Specifically

$$\vec{C} = \vec{C}_0 + \sum_{j \in \mathcal{B}} r_j \frac{s}{\sqrt{\lambda_j}} \cdot \boldsymbol{\mu}_j$$

where r_j, the coefficient of the jth normal mode $\boldsymbol{\mu}_j$, is uniformly sampled on S^d, the surface of a d-dimensional standard sphere with a unit radius. The scaling factor s is given by

$$s = \frac{\tau_{\mathrm{R}}}{\frac{1}{\sqrt{N_{\mathrm{R}}}} \sum_{j \in \mathcal{B}} \frac{r_j}{\sqrt{\lambda_j}} \cdot \boldsymbol{\mu}_j^{\mathrm{R}}},$$

where τ_{R} is the estimated conformational change (measured by RMSD in all C_α atoms) between the unbound and the bound receptor. Note that vectors $\boldsymbol{\mu}_j^{\mathrm{R}}$ (the receptor portion of the jth complex normal mode) are not orthonormal to each other.

We previously predicted τ_{R} by a machine learning model giving $\widehat{\mathrm{RMSD}}_{\mathrm{R}}$, a single value for

each receptor [51]. Here we replace $\widehat{\text{RMSD}}_{\text{R}}$ with a predicted distribution by multiplying it to a truncated normal distribution $N(\mu = 0.99, \sigma^2 = 0.31^2)$ within [0, 2.5]. The latter distribution is derived by fitting the ratios between the actual and the predicted values, $\text{RMSD}_{\text{R}}/\widehat{\text{RMSD}}_{\text{R}}$, for 50 training protein complexes (see more details about datasets in Sec. 2.7 and those about distribution fitting in Sec. 2.2 of the **SI** of [19]). Therefore, our parameterization produces $\boldsymbol{x} = s \cdot \boldsymbol{r} \in \mathbb{R}^d$ whose prior is derived as above.

Since the ligand component of complex normal modes include simultaneous flexible- and rigid-body motions, conformational sampling could lead to severely distorted ligand geometry. We thus further restrict the ligand perturbation Δ_{L} (flexible- and rigid-body together) to be within $\overline{\Delta}_{\text{L}}$

$$\Delta_{\text{L}} = \sqrt{\frac{1}{N_{\text{L}}}} \sum_{j \in \mathcal{B}} r_j \frac{s}{\sqrt{\lambda_j}} \cdot \boldsymbol{\mu}_j^{\text{L}} \leqslant \overline{\Delta}_{\text{L}},$$

where $\boldsymbol{\mu}_j^{\text{L}}$ denotes the ligand portion of the jth complex normal mode.

We set $\overline{\Delta}_{\text{L}}$ at 6 Å according to the average size of binding energy attraction basins seen in conformational clusters [52]. For samples generated from the aforementioned prior or the updated posterior, we reject those violating the ligand perturbation limit. We discuss about the feasibility of the search region in **SI** Sec. 2.3 of [19].

Every conformational sample, generated through sampling the prior or the iteratively-updated posteriors of $\boldsymbol{x}^*$, is locally minimized through CHARMM[44] to remove possible bond distortions before energy evaluation. This setting could be changed in future, along with energy models, to reduce the cost of energy evaluation for each sample, allow for more samples, and improve energy minimization and posterior estimation.

2.3.8 Energy Model $y(\boldsymbol{x})$

We have so far introduced search strategies for functions defined in a Euclidean space or specifically for protein docking. Energy models $y(\boldsymbol{x})$ are at least as important as search strategies for protein docking. In fact, an improved search strategy might expose more deficiencies of energy models, such as false-positive energy wells. We therefore have developed a "funnel-like" energy

model to not only mitigate the issue but also to estimate model quality (iRMSD) of encounter complexes.

Binding affinity prediction for sampled encounter complexes We introduce a new energy model based on binding affinities $K'_d(\boldsymbol{x})$ of structure samples $\boldsymbol{x}$ that are often encounter complexes. The model assumes that K'_d correlates with K_d, the binding affinity of the native complex, and deteriorates with the increase of the sample's iRMSD (the encounter complex being less native-like):

$$K'_d(\boldsymbol{x}) = K_d \cdot \exp(\alpha \cdot (\text{iRMSD}(\boldsymbol{x}))^q),$$

where α and q are hyper-parameters optimized through cross-validation. In other words, we assume that the fraction of binding affinity loss is exponential in a polynomial of iRMSD. Therefore, the binding energy, a machine learning model $y(\boldsymbol{x}; \boldsymbol{w})$ of parameters $\boldsymbol{w}$ can be represented as

$$\begin{aligned} y(\boldsymbol{x}; \boldsymbol{w}) = &\ RT \ln (K'_d(\boldsymbol{x})) \\ = &\ RT \ln(K_d) + RT\alpha \cdot (\text{iRMSD}(\boldsymbol{x}))^q \end{aligned}$$

iRMSD prediction for sampled encounter complexes Given an observed or regressed $y(\boldsymbol{x})$ value for an encounter complex sample, one can estimate iRMSD$(\boldsymbol{x})$ with given K_d using the equation above inversely, which provides quality assessment (QA) without native structures.

Machine learning We train machine learning models, including ridge regression with linear or RBF kernel and random forest, for $y(\boldsymbol{x}; \boldsymbol{w})$. The 8 features include changes upon protein interaction in energy terms such as internal energies in bond, angle, dihedral and Urey-Bradley terms, van der Waals, non-polar contribution of the solvation energy based on solvent-accessible surface area (SASA), and electrostatics modeled by Generalized Born with a simple SWitching (GBSW), all of which are calculated in a CHARMM27 force field.

We use the same training set of 50 protein pairs (see details in **SI** Sec. 2.5 of [19]) as in predicting the extent of conformational change. From rigid docking and conformational sampling we generate 13,004 complex samples for 50 protein pairs, including 6,464 near-native and 6,540

worse examples in the training set. We balance the near-native and non-near native samples in order to make the binding-energy model, as well as the resulting posterior estimation and uncertainty quantification, focus on near-native or slightly worse encounter complexes as opposed to those with very high iRMSD (say, above 10 Å). Hyper-parameters of ridge regression with RBF kernel as well as random forest are optimized by cross-validation. And model parameters $\boldsymbol{w}$ are trained again over the entire training set with the best hyper-parameters. More details can be found in Sec. 2.6 of the **SI** of [19]. For the assessment, we use the test set **a** of 26 protein pairs (again in **SI** Sec. 2.5 of [19]) and generate 20 samples similarly for each of the 10 initial docking results for each pair, leading to 5,200 cases.

2.3.9 Quality Assessment with Uncertainty Quantification for Protein Docking

A unique challenge to protein docking refinement is that, instead of optimization and UQ in a single region $\mathcal{X}$, we may do so in K separate ones $\mathcal{X}_i$ ($i = 1, \ldots, K$) where each $\mathcal{X}_i$ is a promising conformational region/cluster represented by a initial docking result. This is often necessitated by the fact that the extremely rugged energy landscape is populated with low-energy basins separated by frequent high-energy peaks in a high-dimensional space, thus preferably searched over multiple stages [53].

One benefit of UQ for protein docking results is to determine, for each $\hat{\boldsymbol{x}}_i$ – the prediction in $\mathcal{X}_i$ (the ith structure model), its quality bounds [lb, ub] such that

$$P\left(lb \leqslant Q(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant ub\right) = 1 - \sigma,$$

where the quality of interest $Q(\hat{\boldsymbol{x}}, \boldsymbol{x}^*)$ here is iRMSD between a predicted and the native structure and $1 - \sigma$ is a desired confidence level. Again, $Q(\cdot, \cdot)$ can be any quality assessment (QA) function that does not necessarily need information about the native structure $\boldsymbol{x}^*$. We used our iRMSD predictor in this study.

To that end, we forwardly propagate the uncertainty from $\boldsymbol{x}^*$ (native structure) to iRMSD, given the final posterior $p(\boldsymbol{x}^*|\mathcal{D}^{(t)})$ in individual regions (local posteriors). Specifically, we gen-

erate 1,000,000 samples following the local posterior using Markov chain Monte Carlo, evaluate their binding energies using the kriging regressor, and estimate their iRMSD using our binding affinity prediction formula inversely (as described in Sec. 2.3.2.9). We then use these sample iRMSD values to determine confidence intervals $[lb, ub]$ for various confidence score $1-\sigma$ so that $P(\text{iRMSD} < lb) = P(\text{iRMSD} > ub) = \sigma/2$.

Confidence scores for near-nativeness We next calculate the probability that a prediction $\hat{\boldsymbol{x}}_i$ is near-native, i.e., $P(Q(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant 4$ Å) [54]. Calculating this quantity would demand the probability that the native structure lies in the ith conformational region / cluster, $P(\boldsymbol{x}^* \in \mathcal{X}_i)$ ($P(\mathcal{X}_i)$ in short) as well as that the probability that it lies in all the K regions, $P(\boldsymbol{x}^* \in \cup_{i=1}^K \mathcal{X}_i)$ ($P(U_K)$ in short). By following the chain rule we easily reach

$$\begin{aligned} & P(\text{iRMSD}(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant 4) \\ = \; & P(\text{iRMSD}(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant 4 | \mathcal{X}_i) P(\mathcal{X}_i | U_K) P(U_K) \end{aligned}$$

Here we use the fact that $\{\boldsymbol{x}^* \in \mathcal{X}_i\} \subset \{\boldsymbol{x}^* \in \cup_{i=1}^K \mathcal{X}_i\}$ and assume that $\{\text{iRMSD}(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant 4\} \subseteq \{\boldsymbol{x}^* \in \mathcal{X}_i\}$ (the range of conformational clusters in iRMSD is usually wider than 4 Å).

We discuss how to calculate each of the three terms for the product.

$\boldsymbol{P(\text{iRMSD}(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant 4 | \mathcal{X}_i)}$ If the native structure $\boldsymbol{x}^*$ (unknown) is contained in the ith region/cluster $\mathcal{X}_i$, what is the chance that the predicted structure $\hat{\boldsymbol{x}}_i$ (known) is within 4 Å? We again use forward uncertainty propagation starting with the local posterior $p(\boldsymbol{x}^* | \mathcal{D}_t)$ in $\mathcal{X}_i$. We sample 100,000 structures following the posterior with Markov chain Monte Carlo, calculate their iRMSD to the prediction $\hat{\boldsymbol{x}}_i$, and empirically determine the portion within 4 Å for the probability of interest here. Notice that the native interface is unknown thus the putative interface is used instead.

$\boldsymbol{P(\mathcal{X}_i|U_K)}$ If the native structure is contained in at least one of the K regions, what is the chance that it is in $\mathcal{X}_i$? Following statistical mechanics, we reach

$$\begin{aligned} P(\mathcal{X}_i|U_K) &= \frac{Z_i}{Z} \\ &= \frac{\int_{\boldsymbol{x}\in\mathcal{X}_i} \exp(-\frac{1}{RT}\hat{f}_i(\boldsymbol{x}))d\boldsymbol{x}}{\sum_{j=1}^{K}\int_{\boldsymbol{x}\in\mathcal{X}_j} \exp(-\frac{1}{RT}\hat{f}_j(\boldsymbol{x}))d\boldsymbol{x}} \end{aligned}$$

where Z_i and Z are local and global partition functions, respectively; and $\hat{f}_i(\boldsymbol{x})$ is the final kriging regressor. Different regions are assumed to be mutually exclusive. The integrals are calculated by Monte Carlo sampling.

Another approach is to replace $\frac{1}{RT}$ above with ρ_i, the final ρ for temperature control in $\mathcal{X}_i$. In practice we did not find significant performance difference between the two approaches, partly due to the fact that final ρ_i in various clusters / regions reached similar values for the same protein complex.

$\boldsymbol{P(U_K)}$ What is the chance that the native structure is within the union of the initial regions, i.e., at least one initial region or model is near-native? The way to calculate $P(U_K)$ is very similar to that in uncertainty quantification. Specifically, 100,000 structures are sampled following the posterior of each region $\mathcal{X}_i$, evaluated for binding energy using the kriging regressor $\hat{f}_i(\boldsymbol{x})$, and estimated with iRMSD using the binding affinity predictor formula inversely. We empirically calculate the portion q_i in which sample iRMSD values are above 4 Å. Assuming the independence among regions with regards to near-nativeness, we reach $P(U_K) = 1 - \prod_{i=1}^{K} q_i$. However, if conformational regions $\mathcal{X}_i$, presumed to be separate before search, are overlapping afterwards, $1 - \prod_{i=1}^{K} q_i$ would underestimate $P(U_K)$. One possible approach to address the issue, which could sacrifice optimization, is to introduce constraints on $\mathcal{X}_i$ and keep them separate during search.

2.3.10 Data sets

We use a comprehensive protein docking benchmark set 4.0 [48] of 176 protein pairs that diversely and representatively cover sequence and structure space, interaction types, and docking difficulty. We split them into a training set, test sets **a** and **b** with stratified sampling to preserve

the composition of difficulty levels in each set. The “training" set is not used for tuning BAL parameters. Rather, it is just for training energy model ($y(\boldsymbol{x}; \boldsymbol{w})$ in Sec. 2.3.2.8) and conformational-change extent prediction (τ_R in Sec. 2.3.2.7). The training and test **a** sets contain 50 and 26 pairs with known K_d values [55], respectively. And the test set **b** contain 100 pairs with K_d values predicted from sequence alone [56].

We also use a smaller yet more challenging CAPRI set of 15 recent CAPRI targets [51]. Unlike the benchmark set for unbound docking, the CAPRI set contains 11 cases of homology docking, 8 of which start with just sequences for both proteins and demand homology models of structures before protein docking. Compared to the benchmark test set of 86 (68%), 22 (18%) and 18 (14%) cases classified rigid, medium, and flexible, respectively; the corresponding statistics for the CAPRI set are 4 (27%), 5 (33%) and 6 (40%), respectively. Their K_d values are also predicted from sequence alone.

For each protein pair, we use 10 distinct encounter complexes as starting structures ($K = 10$). As reported previously [51], those for the benchmark sets are top-10 cluster representatives by ZDOCK , kindly provided by the Weng group; and those for the CAPRI set are top-10 models generated by the ZDOCK webserver.

2.4 Results

2.4.1 Optimization on Test Functions

We first tested our BAL algorithm on four non-convex test functions of various dimensions and compared it to particle swarm optimization (PSO) [57, 58], an advanced optimization algorithm behind a very successful protein-docking method SwarmDock [29].

For optimization we assess $||\hat{\boldsymbol{x}} - \boldsymbol{x}^*||$, the distance between the predicted and actual global optima, a measure of direct relevance to the quality of interest in protein docking – iRMSD.

Compared to PSO, BAL made predictions that are, on average, closer to the global optima with smaller standard deviations (except for 2D Griewank where BAL had larger standard deviation); and the improvement margins increased with the increasing dimensions (Fig. 2.2).

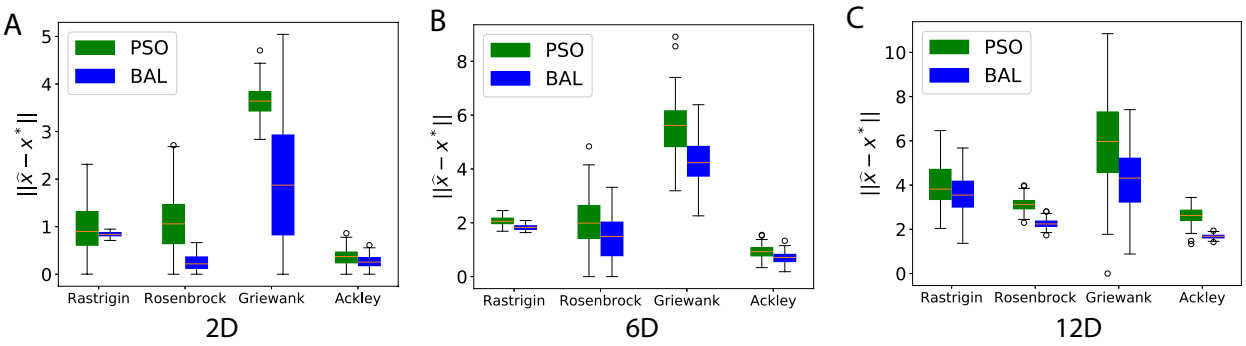


Figure 2.2: Optimization performances of PSO and BAL over four non-convex test functions in various dimensions. The performance metric is $||\hat{\boldsymbol{x}} - \boldsymbol{x}^*||$, the distance between the predicted and the actual global optima, and the box plot is generated using 100 optimization trajectories in each case.

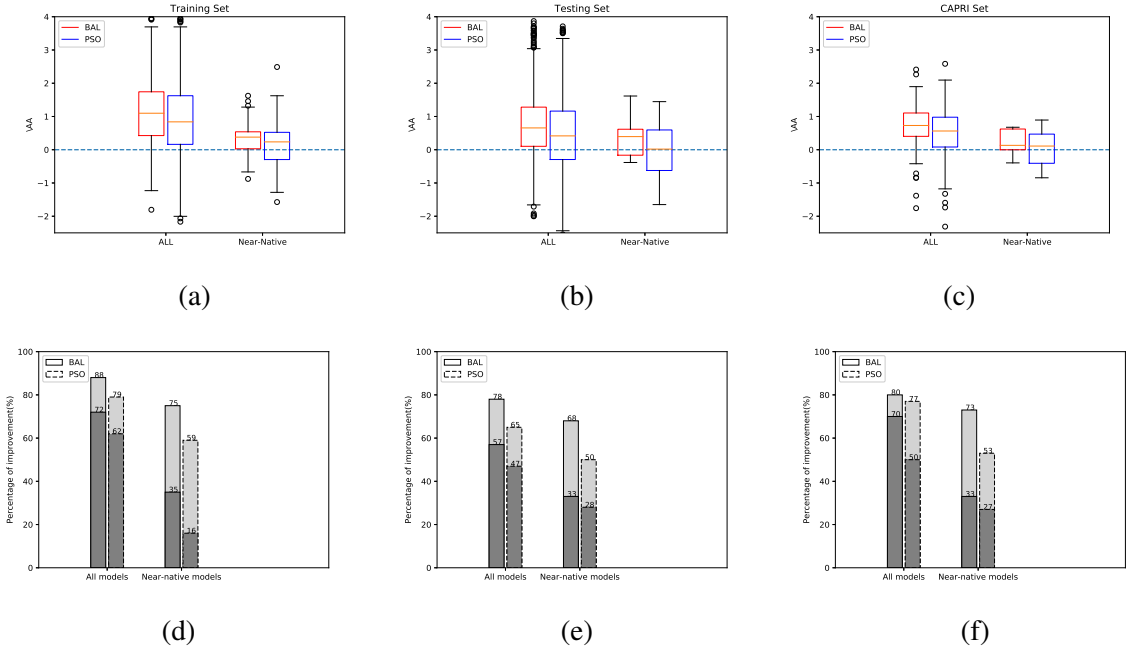


Figure 2.3: Box plots for the improvement in RMSD after PSO and BAL refinements for the training (a), test (b) and CAPRI (c) sets. Also reported are the percentages of BAL (solid bars) and PSO (dashed bars) refinement results with iRMSD improvement or with significant iRMSD improvement over 0.5Å(the darker portions) for the training (d), test (e) and CAPRI (f) sets.

2.4.2 Docking Performance: Optimization

We show the improvements in PSO and BAL solution quality (measured by the decrease of iRMSD) against the starting ZDOCK solutions in Fig. 2.3. Speaking of the amount of improvement, BAL improved iRMSD by 1.2 Å, 0.74 Å, and 0.76 Åfor the training, test, and CAPRI sets, respectively, outperforming PSO's corresponding measures of 0.82 Å, 0.45 Å, and 0.49 Å. It also outperformed PSO for the more challenging near-native cases (note that BAL's iRMSD improvement for the near-native test set or CAPRI set was almost neutral). Speaking of the portion with improvement, BAL improved iRMSD in the near-native cases for 75%, 68%, and 73% of the training, test, and CAPRI sets, respectively; whereas the corresponding statistics for PSO were 59%, 50%, and 53%, respectively.

We also compared BAL and PSO solutions head-to-head over subsets of varying difficulty levels for protein docking (Fig. 2.4). Overall, BAL's solutions are better (or significantly better by at least 0.5 Å) than those of PSO for 70%–80% (31%–45%) of the cases, which was relatively insensitive to the docking difficulty level.

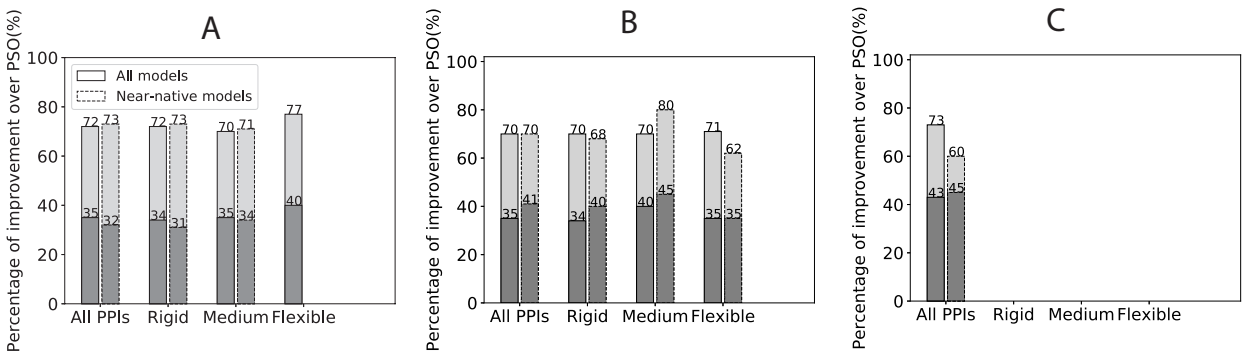


Figure 2.4: The percentage of BAL predictions with iRMSD improvement against PSO for A. the training set, B. the benchmark test set, and C. the CAPRI set. The CAPRI set is not further split because it only contains 15 targets and is predominantly in the flexible category. The darker gray portions correspond to significant improvement (over 0.5 Å in iRMSD) compared to corresponding PSO predictions.

2.4.3 Scoring Performance Empowered by UQ

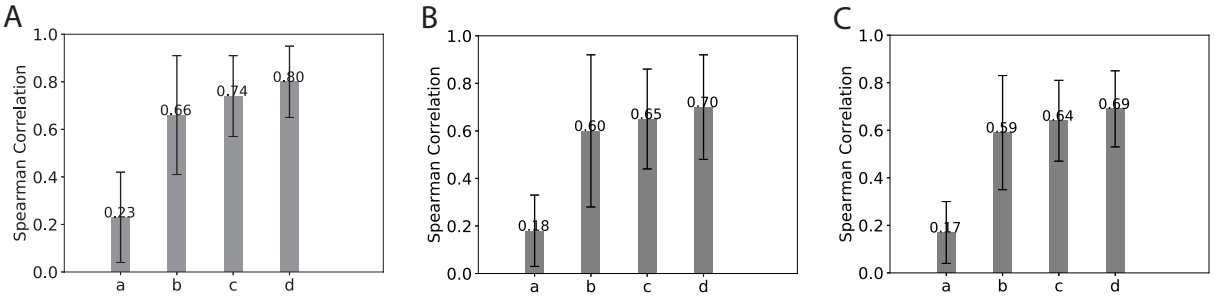


Figure 2.5: Ranking performance shown in the bar plot (with error bar in black) of Spearman's ρ for (A) Training set, (B) Test set and (C) CAPRI set, respectively. Scoring function a, b, c, and d in each figure correspond to the MM-GBSW model, our random-forest energy model, and confidence scores $P(\mathcal{X}_i|U_K)$, and $P(\text{iRMSD}(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant 4)$, respectively.

For scoring models or predictions, two metrics are used for assessing the performance. The first is Spearman's ρ for ranking protein-docking predictions (structure models) for each pair. The second is the area under the Precision Recall Curve (AUPRC), for the binary classification of each prediction being near-native or not. Considering that the near-natives are minorities among all predictions, AUPRC is a more meaningful measure than the more common AUROC.

With these two metrics we assess four scoring functions on predictions $\hat{\boldsymbol{x}}_i$: (1) $\Delta E(\hat{\boldsymbol{x}}_i)$, the MM-GBSW binding energy, i.e., the sum of the 8 features; (2) our random-forest energy model $y(\hat{\boldsymbol{x}}_i)$; (3) $P(\mathcal{X}_i|U_K)$, the conditional probability that the ith prediction's region is near-native give that there is at least such one in the top K predictions; and (4) $P(\text{iRMSD}(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant 4)$, the unconditional probability that the ith prediction is near-native.

For ranking assessment, from Fig. 2.5 we find that, whereas the original MM-GBSW model merely achieved merely 0.2 for Spearman's ρ, our energy model using the same 8 terms as features in random forest drastically improved the ranking performance with a Spearman's ρ around 0.6 for training, benchmark test, and CAPRI test sets. Furthermore, the confidence scores $P(\mathcal{X}_i|U_K)$ and $P(\text{iRMSD}(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant 4)$ further improved ranking. In particular, the unconditional probability for

	ZDOCK (Starting Point)				**PSO**				**BAL**			
Dataset (size)	N_1 (F_1)	N_3 (F_3)	N_5 (F_5)	N_{10} (F_{10})	N_1 (F_1)	N_3 (F_3)	N_5 (F_5)	N_{10} (F_{10})	N_1 (F_1)	N_3 (F_3)	N_5 (F_5)	N_{10} (F_{10})
Training (50)	6 (12%)	11 (22%)	13 (26%)	17 (34%)	9 (18%)	15 (30%)	17 (34%)	20 (40%)	14 (28%)	19 (38%)	20 (40%)	22 (44%)
Test (126)	20 (16%)	29 (23%)	33 (26%)	41 (33%)	26 (21%)	33 (26%)	39 (31%)	45 (36%)	32 (25%)	40 (32%)	42 (33%)	50 (40%)
CAPRI (15)	2(13%)	2 (13%)	3 (20%)	4 (27%)	3 (20%)	3 (20%)	4 (27%)	5 (33%)	3 (20%)	4 (27%)	5 (33%)	5 (33%)

Table 2.1: Summary of docking results measured by the number and the portion of targets in each set that have an acceptable near-native top-1, 3, 5, or 10 prediction.

a prediction to be near-native achieved around 0.70 in ρ even for the benchmark and the CAPRI test sets. Note that this probability, a confidence score on the prediction's near-nativeness, was derived from the posterior distribution of $\boldsymbol{x}^*$; thus it uses both enthalpic and entropic contributions.

2.4.4 Overall Docking Performance

We summarize our docking results (BAL predictions $\hat{\boldsymbol{x}}_i$ ranked by confidence scores on their nativeness $P(\text{iRMSD}(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant 4)$) in Table 2.1; and compare them to the ZDOCK starting results (ranked by cluster size roughly reflecting entropy) and the PSO refinement results (using the same energy model as BAL and ranked by the energy model). We use N_K to denote the number of targets with at least one near-native predictions in top K; and F_K the fraction of such targets among all in a given set (training, benchmark test, or CAPRI test set). Compared to the ZDOCK starting results and PSO refinements, BAL has improved the portion of acceptable targets with top-3 predictions from 23% and 26%, respectively, to 32% for the benchmark test set. Similar improvements were found for the CAPRI set. The portion for top 10 from BAL reached 40% compared to ZDOCK's 33% over the benchmark test set. Note that BAL only refined top-10 starting results from ZDOCK thus this improvement was purely from optimization (no ranking effect).

We further visualize the test-set performance along with iRMSD estimation and UQ-derived confidence scores ($P(\text{iRMSD}(\hat{\boldsymbol{x}}_i, \boldsymbol{x}^*) \leqslant 4)$). Fig. 2.6A shows the actual versus the estimated iRMSD of the top-1 prediction for each test target; and Fig. 2.6B shows the actual iRMSD versus the confidence score of each such prediction. These predictions are colored according to the quality (iRMSD) of the starting structure from ZDOCK. Predicted iRMSD values were positively correlated with actual values and rarely above 4 Å for near-native predictions. High confidence scores were almost exclusive to good predictions with low iRMSD values whereas low confidence

scores corresponded to a mixture of qualities.

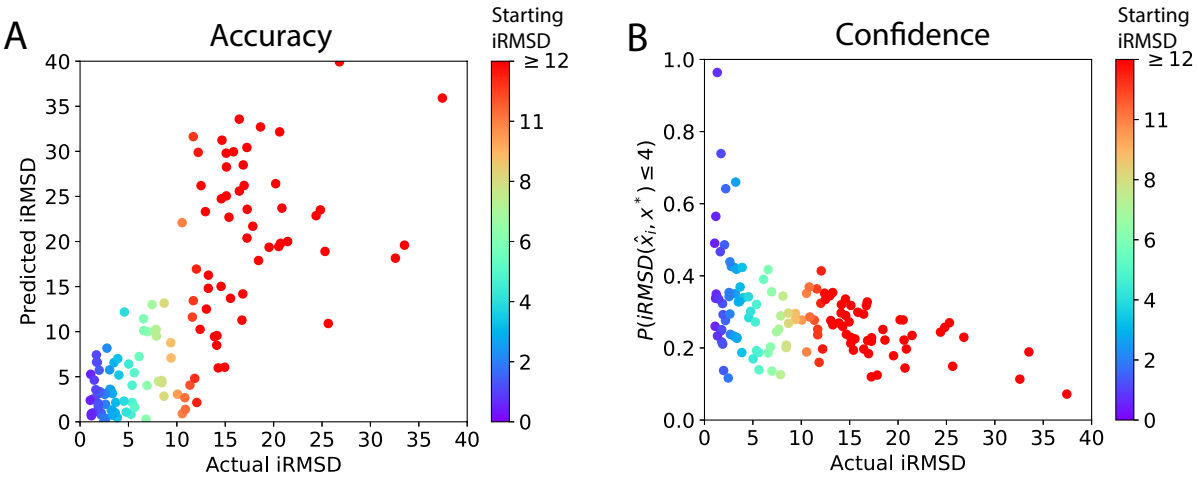


Figure 2.6: Actual iRMSD of the top-1 predictions versus predicted iRMSD or confidence scores in the benchmark test set. Each point representing the prediction for a target is colored according to the starting structure's quality (iRMSD value).

2.4.5 Case Studies

To examine the contributions of method components, such as energy model, quality estimation (iRMSD), optimization, and uncertainty quantification, we chose one successful and two failed cases for detailed analysis.

In the case of success (PDB: 1FFW, ZDOCK model 6), BAL improved prediction quality (iRMSD) from 3.4 Å to 2.2 Å and predicted a very close iRMSD of 2.5 Å. The 90% confidence interval (C. I.) in iRMSD, being [1.51 Å, 3.15 Å], contains the actual iRMSD of 2.2 Å. We project the 12D posterior $p(\boldsymbol{x}^*|\mathcal{D}^{(t)}))$ at the end of iteration t onto a 1D distribution in iRMSD$(\boldsymbol{x}^*, \hat{\boldsymbol{x}}^{(t)})$ where $\hat{\boldsymbol{x}}^{(t)}$ denotes the prediction at the end of iteration t. Fig. 2.7 shows that the predicted and the actual iRMSD values were both contained in the 90% confidence interval through iterations and both converged to the peak of the narrower posterior as iterations progressed.

In a failed case (PDB: 1QFW_IM:AB; ZDOCK model 6), BAL did improve docking quality by reducing actual iRMSD by 0.23Å compared to the starting ZDOCK model. The predicted and

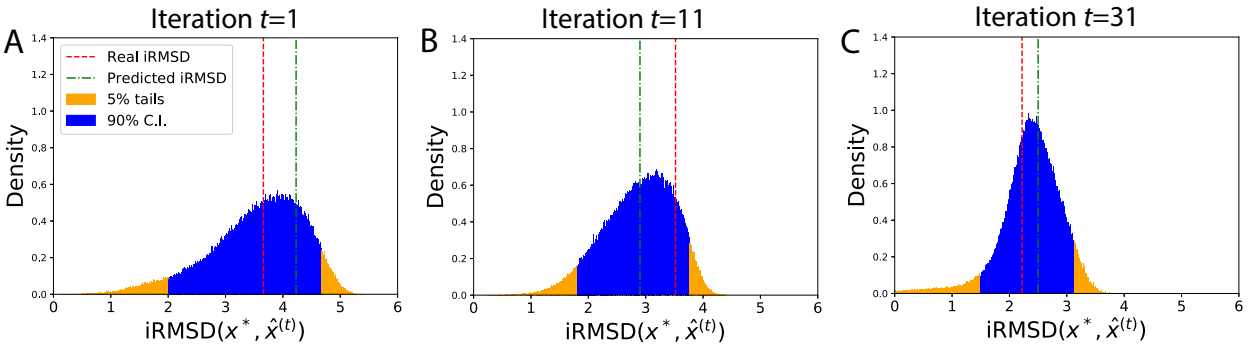


Figure 2.7: A successful case (PDB: 1FFW; ZDOCK model 6): The posterior distributions of the native structure $\boldsymbol{x}^*$ in its iRMSD to predictions $\hat{\boldsymbol{x}}^{(t)}$ over iterations t.

the actual iRMSD of iterative predictions were also close, as seen in Fig. 2.8. However, uncertainty quantification failed as the 90% confidence interval in iRMSD didn't encompass the predicted or the actual iRMSD. This failure is attributed to the energy model, as the lowest-energy prediction $\hat{\boldsymbol{x}}^{(t)}$ was often far from the most-probable conformation (where the peak of the posterior is) and the search drifted toward a non-native funnel. We also note that predictions were of worse quality (larger iRMSD) over iterations.

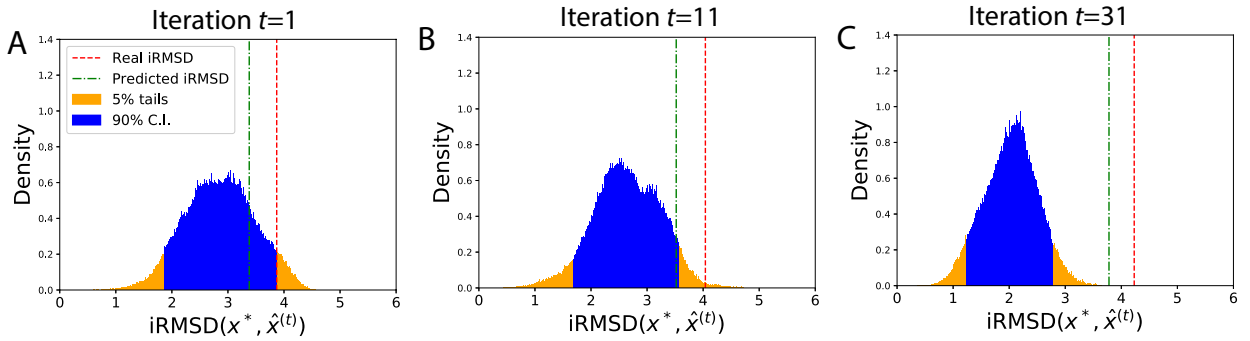


Figure 2.8: A case with failure in UQ (PDB: 1QFW_IM:AB; ZDOCK model 6). The posterior distributions of the native structure in iRMSD did not encompass actual or predicted iRMSD in their 90% confidence intervals. The predictions, although better than the starting ZDOCK model, actually became worse over iterations, which is likely driven by a low-energy non-native funnel in the energy model.

In another failed case (PDB: 2UUY; ZDOCK model 8), BAL failed to improve docking quality compared to the starting structure (iRMSD increased from 3.74 Å to 3.85Å). A close look at Fig. 2.9 suggests that the predictions actually improved over iterations (judging from the real iRMSD in red lines). However, even though the posteriors became narrower over iterations and predicted iRMSD values were close to the peaks of the posteriors, the actual iRMSD values were way off the predicted; and even outside the 90% confidence interval ([1.41 Å, 2.45 Å]). iRMSD prediction (quality estimation) is thus a major reason behind this failure.

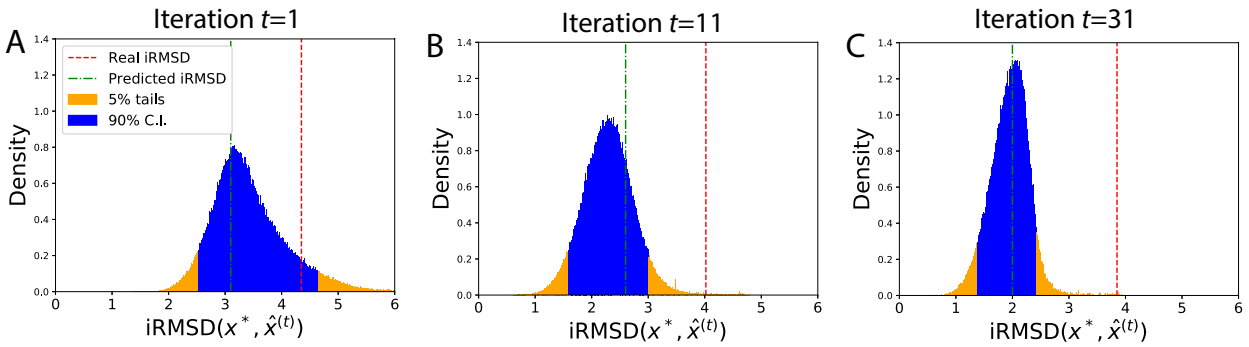


Figure 2.9: A case with failure in optimization and UQ (PDB: 2UUY; ZDOCK model 8). The posterior distributions of the native structure in iRMSD did encompass predicted iRMSD, but not actual iRMSD, in their 90% confidence intervals. The predictions actually became better over iterations but did not improve over the starting ZDOCK model.

2.4.6 Experimental Applications*

We demonstrate the power of BAL through several experimental applications.

First, we apply BAL to modelling the homophilic interactions of breast cancer stem cell marker CD44 [20]. CD44 molecule (CD44) is a well-known surface glycoprotein on tumor-initiating cells or cancer stem cells. However, its utility as a therapeutic target for managing metastases remains

*Reprinted with permission from "Homophilic CD44 Interactions Mediate Tumor Cell Aggregation and Polyclonal Metastasis in Patient-Derived Breast Cancer Models" by Liu *et al*, 2019. Cancer Discovery, 9 (1), 96-113. Copyright © 2019, American Association for Cancer Research.

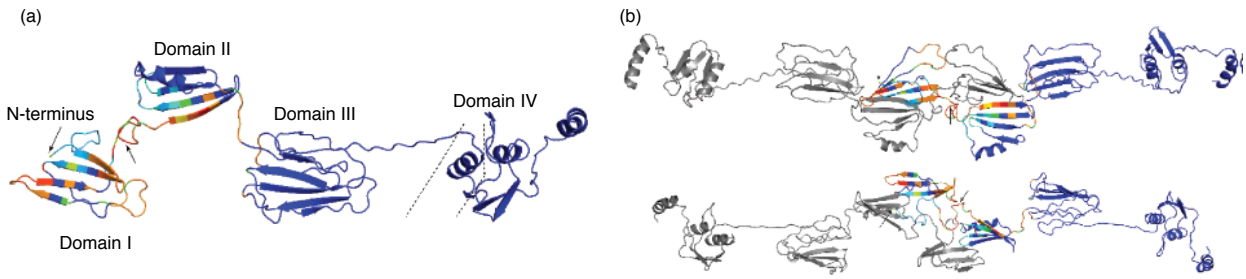


Figure 2.10: (a) Structure model of CD44s monomer with warmer colors indicating higher probabilities to be at the dimer interface, as predicted by BAL. (b) Two representative structure models (top and bottom) of predicted CD44 homodimers. The right monomer is color coded in the same way as in (a), whereas the left one is in gray for contrast.

to be fully evaluated. In order to determine the importance of CD44 homophilic interaction in cell aggregation and lung colonization, we dock two CD44 monomers through BAL. Our homodimer models (Fig 2.10) of CD44 suggest that the N-terminal domain I and domain II's first few residues are mainly responsible for the dimerization.

For experimental validation, our collaborators (Dr. Huiping Liu's group at Northwestern University) conducted two experiments. First, they truncated domain I for co-IP and cellular aggregation analyses. Upon transfection into HEK-293 cells, the deletion of domain I was deficient in forming intercellular complexes with CD44 and lost the capacity of mediating cell aggregation [20]. The N-terminal domain–dependent CD44 homophilic interaction was also blocked by an anti-CD44 neutralizing antibody when administered to aggregating cells in suspension [20]. Second, we picked 12 hotspot residues which are essential for CD44 dimerization based on our docking results, with most of them in domain I and a few in domain II. They mutated the hotspot residues into alanine for functional studies [22]. They found that the mutant CD44 disrupts transdimerization and homophilic interactions, blocks PAK2 activation and interferes with cell aggregation in vitro and lung colonization in vivo [22].

Second, we apply BAL to modelling the homophilic interactions of intercellular adhesion molecule 1 (ICAM1) [21]. The expression of ICAM1 is crucial for the lung colonization and metastasis. Based on BAL modelling, we determine the homodimer interface regions spanning

through domain II to domain IV. To verify the molecular basis of ICAM1 dimerization, our collaborators mutated several important residues on the binding site from domain II to domain IV, and found that all of the ICAM1 mutants lost their homophilic interactions with ICAM1-Flag in the co-immunoprecipitation assays, suggesting that these spanning regions of ICAM1 are indeed involved in the self-dimerization.

Those experimental results clearly show the power of BAL for protein-protein interaction predictions and its support for mutagenesis design. They also demonstrate that artificial intelligence, especially machine learning, is exponentially transforming the future technology and increasing the knowledge base in biological and biomedicine research.

2.5 Conclusion

We present BAL, the first optimization with uncertainty quantification (UQ) study for protein docking. This is accomplished by a rigorous Bayesian framework that actively samples a noisy and expensive black-box function (i.e., collecting data D) while updating a posterior distribution $p(\boldsymbol{x}^*|D)$ directly over the unknown global optimum $\boldsymbol{x}^*$. We demonstrate the superb performances of Bayesian active learning (BAL) on a protein docking benchmark set as well as a CAPRI set full of homology docking. Compared to the starting points from initial rigid docking as well as the refinement from PSO, BAL shows significant improvement, accomplishing a top-3 near-native prediction for about one-third of the benchmark and CAPRI sets. Its UQ results achieve tight uncertainty intervals whose radius is 25% of iRMSD with a 85% confidence level attested by empirical results. We further demonstrate the power of BAL through several experimental applications, by modelling the interface of homophilic CD44 and ICAM1.

3. Learning to Optimize in Swarms

3.1 Preface

In the previous chapter, we have shown that BAL can be applied to general optimization problems. As people do to the invention of every other hand-designed optimization algorithms, we spent a descent amount of time tuning and validating pipelines and architectures of BAL. Even by doing that, when facing a novel optimization problem, we need to manually tune the hyper-parameters. More importantly, optimization algorithms are usually only suitable for one specific kind of problems. For instance, in *ab initio* protein docking whose energy functions as objectives have extremely rugged landscapes and are expensive to evaluate, gradient-free algorithms are thus popular there, including BAL, Markov chain Monte Carlo (MCMC) [59] and Particle Swarm Optimization (PSO) [29]. In deep learning, there is a gallery of gradient-based algorithms specific to high-dimensional, non-convex objective functions, such as Stochastic Gradient Descent [60], RmsDrop [61], and Adam [62].

In this chapter, instead of hand-designing an optimization algorithm, we will focus on **learning** an optimization algorithm. To the end, we will propose a novel meta-learning algorithm called Learning to Optimization in Swarms (LOIS).

3.2 Introduction

To overcome the laborious manual design, an emerging approach of meta-learning (learning to learn) takes advantage of the knowledge learned from related tasks. In meta-learning, the goal is to learn a meta-*learner* that could solve a set of problems, where each sample in the training or test set is a particular problem. As in classical machine learning, the fundamental assumption of meta-learning is the generalizability from solving the training problems to solving the test ones. For optimization problems, a key to meta-learning is how to efficiently utilize the information in the objective function and explore the space of optimization algorithms.

In the field of learning to optimize, the optimization algorithm is usually parameterzied by a

neural network. The pioneer work [63] used a coordinate-wise structure of RNN which relieved the burden from the enormous number of parameters, so that the same update formula was used independently for each coordinate. [64] used the history of gradients and objective values as states and step vectors as actions in reinforcement learning. [65] also used RNN to train a meta-*learner* to optimize black-box functions, including Gaussian process bandits, simple control objectives, and hyper-parameter tuning tasks. Lately, [66] introduced a hierarchical RNN architecture, augmented with additional architectural features that mirror the known structure of optimization tasks.

The target applications of previous methods are mainly focused on training deep neural networks, except [65] focusing on optimizing black-box functions. There are three **limitations** of these methods. First, they learn in a limited algorithmic space, namely point-based optimization algorithms that use gradients or not (including SGD and Adam). So far there is no method in learning to learn that reflects population-based algorithms (such as evolutionary and swarm algorithms) proven powerful in many optimization tasks. Second, their learning is guided by a limited meta loss, often the cumulative regret in sampling history that primarily drives exploitation. One exception is the expected improvement (EI) used by [65] under Gaussian processes. Last but not the least, these methods do not interpret the process of learning update formula, despite the previous usage of attention mechanisms in [66].

To overcome aforementioned limitations of current learning-to-optimize methods, we present a novel framework in meta-learning, where we train a meta-optimizer that learns in the space of both point-based and population-based optimization algorithms for continuous optimization. To that end, we design a novel architecture where a population of RNNs (specifically, LSTMs) jointly learn iterative update formula for a population of samples (or a swarm of particles). To balance exploration and exploitation in search, we directly estimate the posterior over the optimum and include in the meta-loss function the differential entropy of the posterior. Furthermore, we embed feature- and sample-level attentions in our meta-optimizer to interpret the learned optimization strategies. To the end, we summarize our **contributions**:

- (*Where to learn*): We learn in an extended space of both point-based and population-based

optimization algorithms;

- (*How to learn*): We incorporate the posterior into meta-loss to guide the search in the algorithmic space and balance the exploitation-exploration trade-off.

- (*What more to learn*): We design a novel architecture where a population of LSTMs jointly learn iterative update formula for a population of samples and embedded sample- and feature-level attentions to explain the formula.

3.3 Learning to Optimize in Swarms

3.3.1 Background

Our goal is to solve the following optimization problem:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}).$$

Iterative optimization algorithms, either point-based or population-based, have the same generic update formula:

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t + \delta\boldsymbol{x}^t,$$

where $\boldsymbol{x}^t$ and $\delta\boldsymbol{x}^t$ are the sample (or a single sample called "particle" in swarm algorithms) and the update (a.k.a. step vector) at iteration t, respectively. The update is often a function $g(\cdot)$ of historic sample values, objective values, and gradients. For instance, in point-based gradient descent,

$$\delta\boldsymbol{x}^t = g(\{\boldsymbol{x}^\tau, f(\boldsymbol{x}^\tau), \nabla f(\boldsymbol{x}^\tau)\}_{\tau=1}^t) = -\alpha\nabla f(\boldsymbol{x}^t),$$

where α is the learning rate. In particle swarm optimization (PSO), assuming that there are k samples (particles), then for particle j, the update is determined by the entire population:

$$\delta\boldsymbol{x}_j^t = g(\{\{\boldsymbol{x}_j^\tau, f(\boldsymbol{x}_j^\tau), \nabla f(\boldsymbol{x}_j^\tau)\}_{j=1}^k\}_{\tau=1}^t) = w\delta\boldsymbol{x}_j^{t-1} + r_1(\boldsymbol{x}_j^t - \boldsymbol{x}_j^{t*}) + r_2(\boldsymbol{x}_j^t - \boldsymbol{x}^{t*}),$$

where $\boldsymbol{x}_j^{t*}$ and $\boldsymbol{x}^{t*}$ are the best position (with the smallest objective value) of particle j and among all particles, respectively, during the first t iterations; and w, r_1, r_2 are the hyper-parameters often randomly sampled from a fixed distribution (e.g. standard Gaussian distribution) during each iteration.

In most of the modern optimization algorithms, the update formula $g(\cdot)$ is analytically determined and fixed during the whole process. Unfortunately, similar to what the *No Free Lunch Theorem* suggests in machine learning, there is no single best algorithm for all kinds of optimization tasks. Every state-of-art algorithm has its own best-performing problem set or domain. Therefore, it makes sense to learn the optimal update formula $g(\cdot)$ from the data in the specific problem domain, which is called "learning to optimize". For instance, in [63], the function $g(\cdot)$ is parameterized by a recurrent neural network (RNN) with input $\nabla f(\boldsymbol{x}^t)$ and the hidden state from the last iteration: $g(\cdot) = \text{RNN}(\nabla f(\boldsymbol{x}^t), \boldsymbol{h}^{t-1})$. In [65], the inputs of RNN are $\boldsymbol{x}^t, f(\boldsymbol{x}^t)$ and the hidden state from the last iteration: $g(\cdot) = \text{RNN}(\boldsymbol{x}^t, f(\boldsymbol{x}^t), \boldsymbol{h}^{t-1})$.

3.3.2 Population-based learning to optimize with posterior estimation

We describe the details of our population-based meta-optimizer in this section. Compared to previous meta-optimizers, we employ k samples whose update formulae are learned from the population history and are individually customized, using attention mechanisms. Specifically, our update rule for particle i could be written as:

$$g_i(\cdot) = \text{RNN}_i\left(\alpha_i^{\text{inter}}(\{\alpha_j^{\text{intra}}(\{\boldsymbol{S}_j^\tau\}_{\tau=1}^t)\}_{j=1}^k), \boldsymbol{h}_i^{t-1}\right)$$

where $\boldsymbol{S}_j^t = (\boldsymbol{s}_{j1}^t, \boldsymbol{s}_{j2}^t, \boldsymbol{s}_{j3}^t, \boldsymbol{s}_{j4}^t)$ is a $n \times 4$ feature matrix for particle j at iteration t, $\alpha_j^{\text{intra}}(\cdot)$ is the intra-particle attention function for particle j, and $\alpha_i^{\text{inter}}(\cdot)$ is the i-th output of the inter-particle attention function. $\boldsymbol{h}_i^{t-1}$ is the hidden state of the ith LSTM at iteration $t-1$.

For typical population-based algorithms, the same update formula is adopted by all particles. We follow the convention to set $g_1(\cdot) = g_2(\cdot) = ... = g_k(\cdot)$, which suggests $\text{RNN}_i = \text{RNN}$ and $\alpha_j^{\text{intra}}(\cdot) = \alpha^{\text{intra}}(\cdot)$.

We will first introduce the feature matrix $\boldsymbol{S}_j^\tau$ and then describe the intra- and inter- attention modules.

3.3.3 Features from different types of algorithms

Considering the expressiveness and the searchability of the algorithmic space, we consider the update formulae of both point- and population-based algorithms and choose the following four features for particle i at iteration t:

- gradient: $\nabla f(\boldsymbol{x}_i^t)$
- momentum: $\boldsymbol{m}_i^t = \sum_{\tau=1}^{t}(1-\beta)\beta^{t-1}\nabla f(\boldsymbol{x}_i^\tau)$
- velocity: $\boldsymbol{v}_i^t = \boldsymbol{x}_i^t - \boldsymbol{x}_i^{t*}$
- attraction: $\frac{\sum_j\left(\exp(-\alpha d_{ij}^2)(\boldsymbol{x}_i^t-\boldsymbol{x}_j^t)\right)}{\sum_j \exp(-\alpha d_{ij}^2)}$, for all j that $f(\boldsymbol{x}_j^t) < f(\boldsymbol{x}_i^t)$. α is a hyperparameter and $d_{ij} = ||\boldsymbol{x}_i^t - \boldsymbol{x}_j^t||_2$.

These four features include two from point-based algorithms using gradients and the other two from population-based algorithms. Specifically, the first two are used in gradient descent and Adam. The third feature, velocity, comes from PSO, where $\boldsymbol{x}_i^{t*}$ is the best position (with the lowest objective value) of particle i in the first t iterations. The last feature, attraction, is from the Firefly algorithm [67]. The attraction toward particle i is the weighted average of $\boldsymbol{x}_i^t - \boldsymbol{x}_j^t$ over all j such that $f(\boldsymbol{x}_j^t) < f(\boldsymbol{x}_i^t)$; and the weight of particle j is the Gaussian similarity between particle i and j. For the particle of the smallest $f(\boldsymbol{x}_i^t)$, we simply set this feature vector to be zero. In this paper, we use $\beta = 0.9$ and $\alpha = 1$.

It is noteworthy that each feature vector is of dimension $n \times 1$, where n is the dimension of the search space. Besides, the update formula in each base-algorithm is linear w.r.t. its corresponding feature. To learn a better update formula, we will incorporate those features into our model of deep neural networks, which is described next.

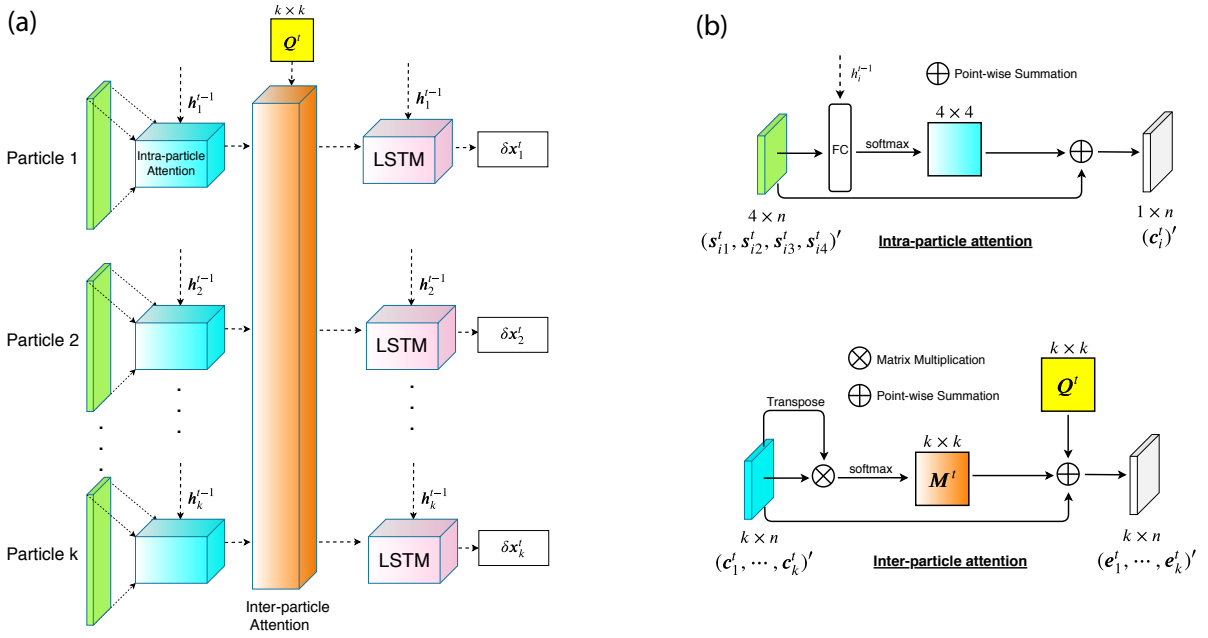


Figure 3.1: (a) The architecture of our meta-*optimizer* for one step. We have k particles here. For each particle, we have gradient, momentum, velocity and attraction as features. Features for each particle will be sent into an intra-particle (feature-level) attention module, together with the hidden state of the previous LSTM. The outputs of k intra-particle attention modules, together with a kernelized pairwise similarity matrix $\boldsymbol{Q}^t$ (yellow box in the figure), will be the input of an inter-particle (sample-level) attention module. The role of inter-particle attention module is to capture the cooperativeness of all particles in order to reweight features and send them into k LSTMs. The LSTM's outputs, $\delta \boldsymbol{x}$, will be used for generating new samples. (b) The architectures of intra- and inter-particle attention modules.

3.3.4 Overall model architecture

Fig. 3.1a depicts the overall architecture of our proposed model. We use a population of LSTMs and design two attention modules here: feature-level ("intra-particle") and sample-level ("inter-particle") attentions. For particle i at iteration t, the intra-particle attention module is to reweight each feature based on the context vector $\boldsymbol{h}_i^{t-1}$, which is the hidden state from the i-th LSTM in the last iteration. The reweight features of all particles are fed into an inter-particle attention module, together with a $k \times k$ distance similarity matrix. The inter-attention module is to learn the information from the rest $k-1$ particles and affect the update of particle i. The outputs of inter-particle attention module will be sent into k identical LSTMs for individual updates.

3.3.5 Attention mechanisms

For the intra-particle attention module, we use the idea from [68, 69, 70]. As shown in Fig. 3.1b, given that the jth input feature of the ith particle at iteration t is $\boldsymbol{s}_{ij}^t$, we have:

$$b_{ij}^t = \boldsymbol{v}_a^T \tanh\left(\boldsymbol{W}_a \boldsymbol{s}_{ij}^t + \boldsymbol{U}_a \boldsymbol{h}_{ij}^t\right), \quad p_{ij}^t = \frac{\exp(b_{ij}^t)}{\sum_{r=1}^4 \exp(b_{ir}^t)},$$

where $\boldsymbol{v}_a \in \mathbb{R}^n$, $\boldsymbol{W}_a \in \mathbb{R}^{n \times n}$ and $\boldsymbol{U}_a \in \mathbb{R}^{n \times n}$ are the weight matrices, $\boldsymbol{h}_i^{t-1} \in \mathbb{R}^n$ is the hidden state from the ith LSTM in iteration $t-1$, b_{ij}^t is the output of the fully-connected (FC) layer and p_{ij}^t is the output after the softmax layer. We then use p_{ij}^t to reweight our input features:

$$\boldsymbol{c}_i^t = \sum_{r=1}^4 p_{ir}^t \boldsymbol{s}_{ir}^t,$$

where $\boldsymbol{c}_i^t \in \mathbb{R}^n$ is the output of the intra-particle attention module for the ith particle at iteration t.

For inter-particle attention, we model $\delta \boldsymbol{x}_i^t$ for each particle i under the impacts of the rest $k-1$ particles. Specific considerations are as follows.

- The closer two particles are, the more they impact each other's update. Therefore, we construct a kernelized pairwise similarity matrix $\boldsymbol{Q}^t \in \mathbb{R}^{k \times k}$ (column-normalized) as the weight

matrix. Its element is $q_{ij}^t = \frac{\exp(-\frac{||\boldsymbol{x}_i^t - \boldsymbol{x}_j^t||^2}{2l})}{\sum_{r=1}^k \exp(-\frac{||\boldsymbol{x}_r^t - \boldsymbol{x}_j^t||^2}{2l})}$.

- The similar two particles are in their intra-particle attention outputs ($\boldsymbol{c}_i^t$, local suggestions for updates), the more they impact each other's update. Therefore, we introduce another weight matrix $\boldsymbol{M}^t \in \mathbb{R}^{k \times k}$ whose element is $m_{ij} = \frac{\exp\left((\boldsymbol{c}_i^t)' \boldsymbol{c}_j^t\right)}{\sum_{r=1}^k \exp\left((\boldsymbol{c}_r^t)' \boldsymbol{c}_j^t\right)}$ (normalized after column-wise softmax).

As shown in Fig. 3.1b, the output of the inter-particle module for the jth particle will be:

$$\boldsymbol{e}_j^t = \gamma \sum_{r=1}^{k} m_{rj}^t q_{rj}^t \boldsymbol{c}_r^t + \boldsymbol{c}_j^t,$$

where γ is a hyperparameter which controls the ratio of contribution of rest k-1 particles to the jth particle. In this paper, γ is set to be 1 without further optimization.

3.3.6 Loss function, posterior estimation, and model training

Cumulative regret is a common meta loss function: $L(\boldsymbol{\phi}) = \sum_{t=1}^{T} \sum_{j=1}^{k} f(\boldsymbol{x}_j^t)$. However, this loss function has two main drawbacks. First, the loss function does not reflect any exploration. If the search algorithm used for training the optimizer does not employ exploration, it can be easily trapped in the vicinity of a local minimum. Second, for population-based methods, this loss function tends to drag all the particles to quickly converge to the same point.

To balance the exploration-exploitation tradeoff, we bring the work from [71] — it built a Bayesian posterior distribution over the global optimum $\boldsymbol{x}^*$ as $p(\boldsymbol{x}^* | \bigcup_{t=1}^{T} D_t)$, where D_t denotes the samples at iteration t: $D_t = \left\{ \left(\boldsymbol{x}_j^t, f(\boldsymbol{x}_j^t) \right) \right\}_{j=1}^{k}$. We claim that, in order to reduce the uncertainty about the whereabouts of the global minimum, the best next sample can be chosen to minimize the entropy of the posterior, $h\left(p(\boldsymbol{x}^* | \bigcup_{t=1}^{T} D_t)\right)$. Therefore, we propose a loss function for function $f(\cdot)$ as:

$$\ell_f(\boldsymbol{\phi}) = \sum_{t=1}^{T} \sum_{j=1}^{k} f(\boldsymbol{x}_j^t) + \lambda h\left(p(\boldsymbol{x}^* | \bigcup_{t=1}^{T} D_t)\right),$$

where λ controls the balance between exploration and exploitation and $\boldsymbol{\phi}$ is a vector of model parameters.

Following [71], the posterior over the global optimum is modeled as a Boltzmann distribution:

$$p(\boldsymbol{x}^*|\bigcup_{t=1}^{T} D_t) \propto \exp(-\rho \hat{f}(\boldsymbol{x})),$$

where $\hat{f}(\boldsymbol{x})$ is a function estimator and ρ is the annealing constant. In the original work of [71], both $\hat{f}(\boldsymbol{x})$ and ρ are updated over iteration t for active sampling. In our work, they are fixed since the complete training sample paths are available at once.

Specifically, for a function estimator based on samples in $\bigcup_{t=1}^{T} D_t$, we use a Kriging regressor [40] which is known to be the best unbiased linear estimator (BLUE):

$$\hat{f}(\boldsymbol{x}) = f_0(\boldsymbol{x}) + (\boldsymbol{\kappa}(\boldsymbol{x}))' (\boldsymbol{K} + \epsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{f_0}),$$

where $f_0(\boldsymbol{x})$ is the prior for $\mathrm{E}[f(\boldsymbol{x})]$ (we use $f_0(\boldsymbol{x}) = 0$ in this study); $\boldsymbol{\kappa}(\boldsymbol{x})$ is the kernel vector with the ith element being the kernel, a measure of similarity, between $\boldsymbol{x}$ and $\boldsymbol{x}_i$; $\boldsymbol{K}$ is the kernel matrix with the (i, j)-th element being the kernel between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$; $\boldsymbol{y}$ and $\boldsymbol{f_0}$ are the vector consisting of $y_1, \ldots, y_{n_t}$ and $f_0(\boldsymbol{x}_1), \ldots, f_0(\boldsymbol{x}_{n_t})$, respectively; and ϵ reflects the noise in the observation and is often estimated to be the average training error (set at 2.1 in this study).

For ρ, we follow the annealing schedule in [71] with one-step update:

$$\rho = \rho_0 \cdot \exp\left((h_0)^{-1} \left| \bigcup_{t=1}^{T} D_t \right|^{\frac{1}{n}} \right),$$

where ρ_0 is the initial parameter of ρ ($\rho_0 = 1$ without further optimization here); h_0 is the initial entropy of the posterior with $\rho = \rho_0$; and n is the dimensionality of the search space.

In total, our meta loss for m functions $f_q(\cdot)$ ($q = 1, \ldots, m$) (analogous to m training examples) with L2 regularization is

$$L(\boldsymbol{\phi}) = \frac{1}{m} \sum_{q=1}^{m} \ell_{f_q}(\boldsymbol{\phi}) + C||\boldsymbol{\phi}||_2^2.$$

To train our model we use the optimizer Adam which requires gradients. The first-order gra-

dients are calculated numerically through TensorFlow following [63]. We use coordinate-wise LSTM to reduce the number of parameters. In our implementation the length of LSTM is set to be 20. For all experiments, the optimizer is trained for 10,000 epochs with 100 iterations in each epoch.

3.4 Results

We test our meta-optimizer through convex quadratic functions, non-convex test functions and an optimization-based application with extremely noisy and rugged landscapes: protein docking.

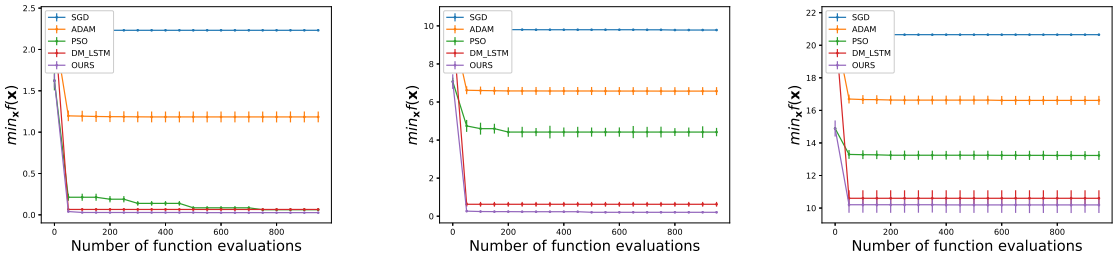
3.4.1 Learn to optimize convex quadratic functions

In this case, we are trying to minimize a convex quadratic function:

$$f(\boldsymbol{x}) = ||\boldsymbol{A}_q\boldsymbol{x} - \boldsymbol{b}_q||_2^2,$$

where $\boldsymbol{A}_q \in \mathbb{R}^{n\times n}$ and $\boldsymbol{b}_q \in \mathbb{R}^{n\times 1}$ are parameters, whose elements are sampled from i.i.d. normal distributions for the training set. We compare our algorithm with SGD, Adam, PSO and DeepMind's LSTM (DM_LSTM) [63]. Since different algorithms have different population sizes, for fair comparison we fix the total number of objective function evaluations (sample updates) to be 1,000 for all methods. The population size k of our meta-optimizer and PSO is set to be 4, 10 and 10 in the 2D, 10D and 20D cases, respectively. During the testing stage, we sample another 128 pairs of $\boldsymbol{A}_q$ and $\boldsymbol{b}_q$ and evaluate the current best function value at each step averaged over 128 functions. We repeat the procedure 100 times in order to obtain statistically significant results.

As seen in Fig. 3.2, our meta-optimizer performs better than DM_LSTM in the 2D, 10D, and 20D cases. Both meta-optimizers perform significantly better than the three baseline algorithms (except that PSO had similar convergence in 2D).



(a) 2D (b) 10D (c) 20D

Figure 3.2: The performance of different algorithms for quadratic functions in (a) 2D, (b) 10D, and (c) 20D. The mean and the standard deviation over 100 runs are evaluated every 50 function evaluations.

3.4.2 Learn to optimize non-convex Rastrigin functions

We then test the performance on a non-convex test function called Rastrigin function:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} \alpha \cos(2\pi x_i) + \alpha n,$$

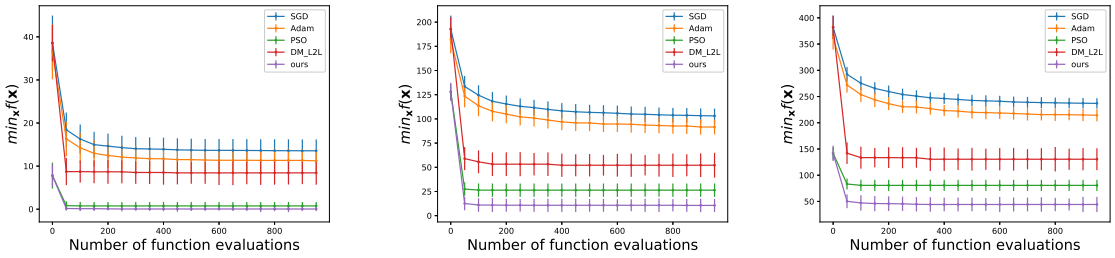
where $\alpha = 10$. We consider a broad family of similar functions $f_q(\boldsymbol{x})$ as the training set:

$$f_q(\boldsymbol{x}) = ||\boldsymbol{A}_q \boldsymbol{x} - \boldsymbol{b}_q||_2^2 - \alpha \boldsymbol{c}_q \cos(2\pi \boldsymbol{x}) + \alpha n, \tag{3.1}$$

where $\boldsymbol{A}_q \in \mathbb{R}^{n\times n}$, $\boldsymbol{b}_q \in \mathbb{R}^{n\times 1}$ and $\boldsymbol{c}_q \in \mathbb{R}^{n\times 1}$ are parameters whose elements are sampled from i.i.d. normal distributions. It is obvious that Rastrigin is a special case in this family with $\boldsymbol{A} = \boldsymbol{I}, \boldsymbol{b} = \{0, 0, \ldots, 0\}', \boldsymbol{c} = \{1, 1, \ldots, 1\}'$.

During the testing stage, 100 i.i.d. trajectories are generated in order to reach statistically significant conclusions. The population size k of our meta-optimizer and PSO is set to be 4, 10 and 10 for 2D, 10D and 20D, respectively. The results are shown in Fig. 3.3. In the 2D case, our meta-optimizer and PSO perform fairly the same while DM_LSTM performs much worse. In the 10D and 20D cases, our meta-optimizer outperforms all other algorithms. It is interesting that PSO is the second best among all algorithms, which indicates that population-based algorithms

have unique advantages here.



(a) 2D (b) 10D (c) 20D

Figure 3.3: The performance of different algorithms for a Rastrigin function in (a) 2D, (b) 10D, and (c) 20D. The mean and the standard deviation over 100 runs are evaluated every 50 function evaluations.

3.4.3 Interpretation of learned update formula

In an effort to rationalize the learned update formula, we choose the 2D Rastrigin test function to illustrate the interpretation analysis. We plot sample paths of our algorithm, PSO and Gradient Descent (GD) in Fig 3.4a. Our algorithm finally reaches the funnel (or valley) containing the global optimum ($\boldsymbol{x} = \mathbf{0}$), while PSO finally reaches a suboptimal funnel. At the beginning, samples of our meta-optimizer are more diverse due to the entropy control in the loss function. In contrast, GD is stuck in a local minimum which is close to its starting point after 80 samples.

To further show which factor contributes the most to each update, we plot the feature weight distribution over the first 20 iterations. Since for particle i at iteration t, the output of its intra-attention module is a weighted sum of its 4 features: $\boldsymbol{c}_i^t = \sum_{r=1}^{4} p_{ir}^t \boldsymbol{s}_{ir}^t$, we hereby sum p_{ir}^t for the r-th feature over all particles i. The final weight distribution (normalized) over 4 features reflecting the contribution of each feature at iteration t is shown in Fig. 3.4b. In the first 6 iterations, the population-based features contribute to the update most. Point-based features start to play an important role later.

Finally, we examine in the inter-particle attention module the level of particles working collab-

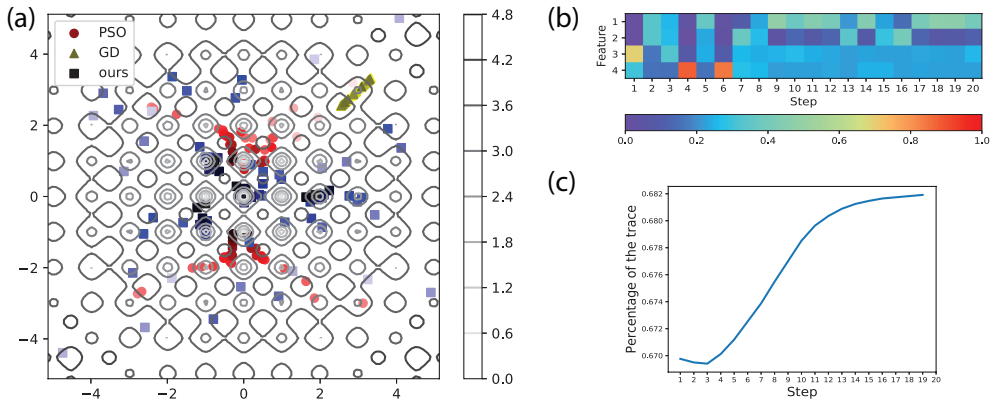


Figure 3.4: (a) Paths of the first 80 samples of our meta-optimizer, PSO and GD for 2D Rastrigin functions. Darker shades indicate newer samples. (b) The feature attention distribution over the first 20 iterations for our meta-optimizer. (c) The percentage of the trace of $\gamma \boldsymbol{Q}^t \odot \boldsymbol{M}^t + \boldsymbol{I}$ (reflecting self-impact on updates) over iteration t.

oratively or independently. In order to show this, we plot the percentage of the diagonal part of $\gamma \boldsymbol{Q}^t \odot \boldsymbol{M}^t + \boldsymbol{I}$: $\frac{\mathrm{tr}(\gamma \boldsymbol{Q}^t \odot \boldsymbol{M}^t + \boldsymbol{I})}{\sum \gamma \boldsymbol{Q}^t \odot \boldsymbol{M}^t + \boldsymbol{I}}$ ($\odot$ denotes element-wise product), as shown in Fig. 3.4c. It can be seen that, at the beginning, particles are working more collaboratively. With more iterations, particles become more independent. However, we note that the trace (reflecting self impacts) contributes 67%-69% over iterations and the off-diagonals (impacts from other particles) do above 30%, which demonstrates the importance of collaboration, a unique advantage of population-based algorithms.

3.4.4 Ablation study

How and why our algorithm outperforms DM_LSTM is both interesting and important to unveil the underlying mechanism of the algorithm. In order to deeply understand each part of our algorithms, we performed an ablation study to progressively show each part's contribution. Starting from the DM_LSTM baseline ($\mathbf{B}_0$), we incrementally crafted four algorithms: running DM_LSTM for k times under different initializations and choosing the best solution ($\mathbf{B}_1$); using k independent particles, each with the two point-based features, the intra-particle attention module, and the hidden state ($\mathbf{B}_2$); adding the two population-based features and the inter-particle attention module to $\mathbf{B}_2$ so as to convert k independent particles into a swarm ($\mathbf{B}_3$); and eventually, adding an entropy

term in meta loss to $\mathbf{B}_3$, resulting in our **Proposed** model.

We tested the five algorithms ($\mathbf{B}_0$–$\mathbf{B}_3$ and the **Proposed**) on 10D and 20D Rastrigin functions with the same settings as in Section 4.2. We compare the function minimum values returned by these algorithms in the table below (reported are means $\pm$ standard deviations over 100 runs, each using 1,000 function evaluations).

Dimension	$\mathbf{B}_0$	$\mathbf{B}_1$	$\mathbf{B}_2$	$\mathbf{B}_3$	**Proposed**
10	55.4±13.5	48.4±10.5	40.1±9.4	20.4±6.6	12.3±5.4
20	140.4±10.2	137.4±12.7	108.4±13.4	48.5±7.1	43.0 ±9.2

Our key observations are as follows. i) $\mathbf{B}_1$ v.s. $\mathbf{B}_0$: their performance gap is marginal, which proves that our performance gain is not simply due to having k independent runs; ii) $\mathbf{B}_2$ v.s. $\mathbf{B}_1$ and $\mathbf{B}_3$ v.s. $\mathbf{B}_2$: Whereas including intra-particle attention in $\mathbf{B}_2$ already notably improves the performance compared to $\mathbf{B}_1$, including population-based features and inter-particle attention in $\mathbf{B}_3$ results in the largest performance boost. This confirms that our algorithm majorly benefits from the attention mechanisms; iii) **Proposed** v.s. $\mathbf{B}_3$: adding entropy from the posterior gains further, thanks to its balancing exploration and exploitation during search.

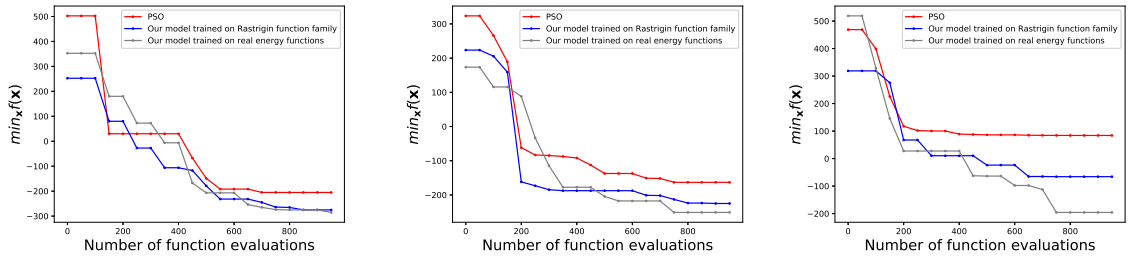
3.4.5 Application to protein docking

We bring our meta-*optimizer* into a challenging real-world application. In computational biology, the structural knowledge about how proteins interact each other is critical but remains relatively scarce [26]. Protein docking helps close such a gap by computationally predicting the 3D structures of protein-protein complexes given individual proteins' 3D structures or 1D sequences [27]. *Ab initio* protein docking represents a major challenge of optimizing a noisy and costly function in a high-dimensional conformational space [71].

Mathematically, the problem of *ab initio* protein docking can be formulated as optimizing an extremely rugged energy function: $f(\boldsymbol{x}) = \Delta G(\boldsymbol{x})$, the Gibbs binding free energy for conformation $\boldsymbol{x}$. We calculate the energy function in a CHARMM 19 force field as in [29] and shift it so that

$f(\boldsymbol{x}) = 0$ at the origin of the search space. And we parameterize the search space as $\mathbb{R}^{12}$ as in [71]. The resulting $f(\boldsymbol{x})$ is fully differentiable in the search space. For computational concern and batch training, we only consider 100 interface atoms. We choose a training set of 25 protein-protein complexes from the protein docking benchmark set 4.0 [48], each of which has 5 starting points (top-5 models from ZDOCK [72]). In total, our training set includes 125 instances. During testing, we choose 3 complexes (with 1 starting model each) of different levels of docking difficulty. For comparison, we also use the training set from Eq. 3.1 ($n = 12$). All methods including PSO and both versions of our meta-optimizer have $k = 10$ particles and 40 iterations in the testing stage.

As seen in Fig. 3.5, both meta-optimizers achieve lower-energy predictions than PSO and the performance gains increase as the docking difficulty level increases. The meta-optimizer trained on other protein-docking cases performs similarly as that trained on the Rastrigin family in the easy case and outperforms the latter in the difficult case.



(a) 1AY7_2 (b) 2HRK_1 (c) 2C0L_1

Figure 3.5: The performance of PSO, our meta-optimizer trained on Rastrigin function family and that trained on real energy functions for three different levels of docking cases: (a) rigid (easy), (b) medium, and (c) flexible (difficult).

3.5 Conclusion

In this chapter, we propose LOIS, a meta-learner that combines point-based meta-*optimizer* and population-based meta-*optimizer* through two-level attention mechanisms. In order to balance exploitation and exploration, we introduce the entropy of the posterior over the global optimum

into the meta loss, together with the cumulative regret, to guide the search of the meta-*optimizer*. Our experiments on quadratic functions, Rastrigin functions and protein docking demonstrate the superior performance of LOIS against its competitors. Ablation study shows that the performance improvement is directly attributable to our algorithmic innovations.

4. Scoring in Protein Docking*

4.1 Preface

The previous two chapters mainly focus on solving the optimization challenges with applications to protein docking. In their settings, the objective (energy) function is pre-defined. However, as the exact energy function is impossible to calculate, surrogate/approximated models are usually used as the approximation for the real energy functions. Those models are called "scoring functions" which are to score each candidate structure. This poses the second challenge in protein docking: **scoring**, where the task is aiming at identifying those near-natives among a large number of candidate decoys.

In this chapter, we focus on the second challenge of protein docking: **scoring**, and we will propose the first study of using graph neural networks for scoring protein docking decoys.

4.2 Introduction

Scoring, which aims at identifying those near-natives among a large number of candidate decoys is one of the most important challenge in protein docking. Current scoring functions for protein docking often aim at relative scoring, in other words, ranking near-natives high [73]. But they do not score the decoy/model quality directly (absolute scoring), which is more often known as quality estimation or quality assessment (without known native structures) in the protein-structure community of CASP [74]. Although quality estimation methods based on machine learning are emerging for single-protein structure prediction [75, 76, 77, 78, 79, 80], such methods are still rare for protein-complex structure prediction, that is, protein docking [19]. Second, state-of-the-art scoring functions (for relative scoring) in protein docking are often based on machine learning with hand-engineered features, such as physical-energy terms, statistical potentials, and graph kernels.

*Reprinted with permission from "Energy-based graph convolutional networks for scoring protein docking models" by Yue Cao and Yang Shen, 2020. Proteins: Structure, Function and Bioinformatics, 88 (8), 1091-1099. Copyright © 2020 Wiley Periodicals, Inc.

These features, heavily relying on domain expertise, are often not specifically tailored or optimized for scoring purposes. Recently, deep learning has achieved tremendous success in image recognition and natural language processing [68, 11], largely due to its automated feature/representation learning from raw inputs of image pixels or words. This trend has also rippled to structural bioinformatics and will be briefly reviewed later.

To fill the aforementioned gaps for scoring in protein docking, we propose a deep learning framework with automated feature learning to estimate the quality of protein-docking models (measured by inter-face RMSD or iRMSD) and to rank them accordingly. In other words,the framework simultaneously addresses both relative scoring (ranking) and absolute scoring (quality assessment) using features directly learned from data. To that end, our deep learning framework predicts binding free energy values of docking models (encounter complexes)whereas these values are correlated to known binding free energy values of native complexes according to model quality.

Technical challenges remain for the framework of deep learning: how to represent protein-complex structure data and learn from such data effectively? Current deep learning methods in structural bioinformatics often use 3D volumetric representations of molecular structures [81, 82, 83, 84]. However, learning features from volumetric data of molecular structures present several drawbacks. First, representing the volumetric input data as pixel data through tensors would require discretization, which may lose some biologically meaningful features while costing time. Second, such input data are often sparse, resulting in many convolutional operations for zero-valued pixels and thus low efficiency. Third, the convolutional operation is not rotation-invariant, which demands rotational augmentation of training data and increased computational cost.

To effectively learn features from and predict labels for protein-docking structure models, we represent proteins and protein-complexes. as intra- and inter-molecular residue contact graphs with atom-resolution node and edge features. Such a representation naturally captures the spatial relationship of protein-complex structures while over-coming the aforementioned drawbacks of learning from volumetric data. Moreover, we learn from such graph data by proposing a physics-

inspired graph convolutional kernel that pools interacting nodes' interactions through their node and edge features. We use two resulting energy-based graph convolutional networks (EGCN) of the same architecture but different parameters to predict intra- and inter-molecular energy potentials for protein-docking models (encounter complexes), which further predicts these encounter complexes' binding affinity and quality (iRMSD) values. Therefore, our EGCN is capable of both model ranking and quality estimation(or quality assessment without known native structures). We note that this is the first study of graph neural networks for protein docking.

4.3 Energy-based Graph Convolutional Neural Network for Scoring Protein Docking Models

4.3.1 Graphs and features

The structure of a protein or protein complex is represented as a graph G where each residue corresponds to a node and each intra- or inter-molecular residue-pair defines an edge. Such intra- and inter -molecular contact graphs can be united under bipartite graphs: The two sets of nodes correspond to the same protein for intramolecular graphs and they do to binding partners for intermolecular graphs. Atom resolution features are further introduced for nodes and edges of such bipartite contact graphs. In other words, to represent G, we have initialized two node feature matrices (denoted by $\boldsymbol{X}_A \in \mathcal{R}^{N_1 \times M}$ and $\boldsymbol{X}_B \in \mathcal{R}^{N_2 \times M}$) and an edge feature tensor (denoted by $A \in \mathcal{R}^{N_1 \times N_2 \times K}$, where N_1 or N_2 denote the number of residues for either protein A or B (which can be the same protein in the case of intramolecular contact graphs), M the number of features per node, and K the number of features per edge. As revealed later in our graph convolutional networks, node feature matrices will be learned, that is, updated layer after layer, by inter-acting with neighboring nodes' features through the fixed edge tensors.

For node-feature matrix $\boldsymbol{X}$, we use $M = 4$ features for each node. Each side chain is represented as a pseudo atom whose position is the geometric center of the side chain. And the first three node features are the side-chain pseudo atom's charge, nonbonded radii, and distance-to-C_a, as parameterized in the Rosetta coarse-grained energy model [85]. The last node feature is the sol-

vent accessible surface area (SASA) of the whole residue, as calculated by the FreeSASA program [86]. For edge-feature tensor A, we use K = 11 features for each edge. These features are related to pairwise atomic distances between the two corresponding residues. Specifically, each residue is a point cloud of six atoms, including five backbone atoms (N, HN, CA, C, and O as named in CHARMM27) and one side-chain pseudo atom (named SC). For numerical efficiency, we have picked 11 pairwise distances (including potential hydrogen bonds) out of the total 6 × 6 = 36 (see Table 4.1) and used their reciprocals for the edge features. We use a cutoff of 12 Å for pairwise atomic distances and set edge features at zero when corresponding distances are above the cutoff.

Index	Atom 1	Atom 2
1	SC	SC
2	SC	O
3	SC	N
4	O	SC
5	O	O
6	O	N
7	N	SC
8	N	O
9	N	N
10	HN	O
11	O	HN

Table 4.1: The inter-residue atom pairs whose distances are converted to edge features.

4.3.2 Energy-based Graph Convolutional Neural Network

Principle-driven Energy Model. To score protein-docking models, we try to use machine learning to model ΔG, the binding energy of encounter complexes, which can be written as:

$$\Delta G = G_C - G_{Ru} - G_{Lu} \tag{4.1}$$

where G_C, G_{Ru}, G_{Lu} denote the Gibbs free energies of the complex,unbound receptor and unbound ligand, respectively. For G_C, we can further decompose it into the intramolecular energies

within two individual proteins and the intermolecular energy between two proteins:

$$G_C = G_R + G_L + G_{RL} \tag{4.2}$$

where G_R, G_L, G_{RL} are the Gibbs free energies within the encountered receptor, within the encountered ligand and between them, respectively. Therefore, the binding energy in classical force fields can be written as:

$$\Delta G = G_R + G_L - G_{Ru} - G_{Lu} + G_{RL} \tag{4.3}$$

In flexible docking, unbound and encountered structures of the same protein are often different. Such protein conformational changes indicate that $G_R - G_{Ru} \neq 0$ and $G_L - G_{Lu} \neq 0$.

Extension to data-driven energy model. It is noteworthy that the expression of ΔG above consists of four terms measuring intramolecular free energies of individual proteins and one term measuring intermolecular free energies across two proteins. Therefore, we decide to use two machine-learning models f of the same neural-network architecture but different parameters to approximate the two types of energy terms as follows:

$$\begin{aligned}\Delta G \approx \Delta \hat{G} =& f_\theta(\boldsymbol{X}_R, \boldsymbol{X}_R, \boldsymbol{A}_{RR}) + f_\theta(\boldsymbol{X}_L, \boldsymbol{X}_L, \boldsymbol{A}_{LL}) - f_\theta(\boldsymbol{X}_{Ru}, \boldsymbol{X}_{Ru}, \boldsymbol{A}_{RuRu}) - \\ & f_\theta(\boldsymbol{X}_{Lu}, \boldsymbol{X}_{Lu}, \boldsymbol{A}_{LuLu}) + f_{\theta'}(\boldsymbol{X}_R, \boldsymbol{X}_L, \boldsymbol{A}_{RL})\end{aligned} \tag{4.4}$$

where f_θ and $f_{\theta'}$ are the intra- and inter-molecular energy models (graph convolutional networks here) parameterized by θ and θ', respectively. Subscripts are included for node feature matrices $\boldsymbol{X}$ or edge feature matrices $\boldsymbol{A}$ to indicate identities of molecules or molecular pairs. The parameters are to be learned from data specifically for the purpose of quality estimation.

Energy-based graph convolutional (EGC) layer. The node matrices ($\boldsymbol{X}_A$ and $\boldsymbol{X}_B$) and the edge tensor $\boldsymbol{A}$ of each contact graph are first fed to three consecutive graph convolutional layers inspired by physics. Specifically, energy potentials are often pairwise additive and those between atoms (i and j) are often of the form $\frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{r^a}$, where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are atom "features," r is the distance

between them, and a is an integer. For instance, $\boldsymbol{x}$ being a scalar charge and a being 1 lead to a Coulombic potential.

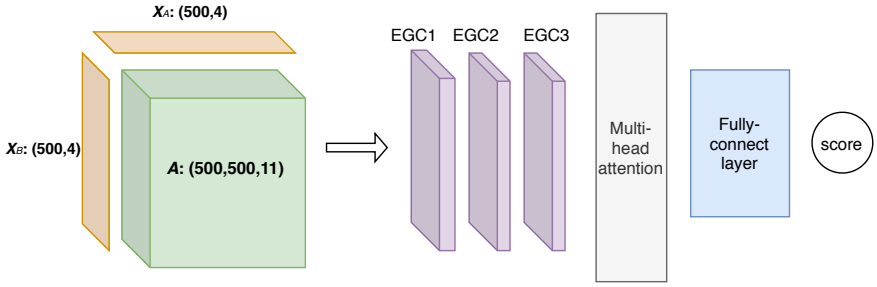


Figure 4.1: The architecture of the proposed graph convolutional network (GCN) models for intra- or inter-molecular energies. In our work, there are five types of such models together for predicting encounter-complex binding energy, including four intramolecular models with shared parameters for the unbound or encountered receptor or ligand as well as one intermolecular model for the encounter complex. In each type of model, the inputs (to the left of the arrow) include a pair of node-feature matrices ($\boldsymbol{X}_A$ and $\boldsymbol{X}_B$) for individual protein(s) and an edge-feature tensor, A, for intra- or inter-molecular contacts. And the inputs are fed through three layers of our energy-based graph convolution layers that learn from training data to aggregate and transform atomic interactions, followed by multihead attention module and fully connected layers for the output of intra- or inter-molecular energy

Inspired by the energy form, we propose a novel graph convolutional kernel that "pools" neighbor features to update protein A's node features layer by layer:

$$x_{ip}^{(l+1)} = \text{LeakyRelu}(\sum_{j\in B}\sum_{k=1}^{K}(\boldsymbol{W}_{kp}^{(l+1)}\boldsymbol{x}_i^{(l)})^T A_{ijk}(\boldsymbol{W}_{kp}^{l+1}\boldsymbol{x}_j^{(l)})) \tag{4.5}$$

where subscripts i and j are node indices for proteins A and B, respectively, p node-feature index, and k edge-feature index; superscripts (l) and (l+1) indicate layer indices. Accordingly, $\boldsymbol{x}_i^{(l)}$ of size $M^{(l)} \times 1$ is the node feature vector for the ith node of protein A in layer I, $x_{ip}^{(l+1)}$ is the pth node-feature for the ith node of protein A in layer (l+1) and $\boldsymbol{W}_{kp}^{(l+1)}$ of size $10 \times M^{(l)}$ is the trainable weight matrix for the kth edge feature and the pth node feature (p = 1,...,$M^{(l+1)}$) in the same layer $(l+1)$. In this way we sum all the interactions between node i in protein A and all neighboring nodes j in protein B through the $K = 11$ edge features. Similarly we update node

features for each node in protein B by following

$$x_{jp}^{(l+1)} = \text{LeakyReLU}(\sum_{i \in A} \sum_{k=1}^{K} (\boldsymbol{W}_{kp}^{(l+1)} \boldsymbol{x}_j^{(l)})^T \tilde{A}_{jik} (\boldsymbol{W}_{kp}^{(l+1)} \boldsymbol{x}_i^{(l)})), \tag{4.6}$$

where $\tilde{A}_{jik}$ is an element of $\tilde{\boldsymbol{A}}$, a permuted $\boldsymbol{A}$ with the first two dimensions swapped. When calculating intramolecular energy within a single protein, the second molecule's feature update can be skipped for numerical efficiency.

Multihead attention and fully connected layers. After the three EGC layers the output node feature matrices $\boldsymbol{X}_A^{(L)}$ and $\boldsymbol{X}_B^{(L)}$ (L=3 in this study) for both proteins (or two copies of the single matrix in the intramolecular case) are concatenated and fed into a multihead attention module [68], whose output subsequently goes through three fully connected (FC) layers with 128, 64, and 1 output, respectively. A dropout rate of 20% is applied to all but the last FC layer.

Label and loss function The label here is the binding energy ΔG although it is not available for encounter complexes. We estimate k_d', the binding affinity of an encounter complex, to weaken with the worsening quality of the encounter complex (measured by iRMSD):

$$k_d' = k_d \cdot \exp(\alpha \cdot (iRMSD)^q) \tag{4.7}$$

where α and q are hyperparameters optimized using the validation set. Specifically, α is searched between 0.5 and 10 with a step size of 0.5 and q among 0.25, 0.5, 0.75, 1, 1.5 and 2. The optimized α and q are 1.5 and 0.5, respectively. Therefore, for each sample (corresponding to an encounter complex or decoy), the predicted label is the previously defined $\Delta\hat{G}$, the actual label is $RTln(k_d') = RTln(k_d) + RT\alpha \cdot (iRMSD)^q$, and the error is simply the difference between the two.

We learn parameters θ and θ' of two graph convolutional networks of the same architecture, by minimizing the loss function of mean squared errors (MSE) over samples. Model parameters include $\boldsymbol{W}_{kp}^{(l)}$ in the EGC layers as well as those in the multihead attention module and FC layers.

One set of learned parameter values, θ are shared among all intramolecular energy models, and the other set, θ' are for the intermolecular energy model.

4.4 Results

In this section, we compare EGCN with two baseline methods, IRAD [87] and Random Forest (RF) [19]. IRAD is one of the top performing methods in CAPRI, with a linear combination of atom-based potentials and residue-based potentials. RF is a combination of eight potentials, including namely bond, angle, dihedral and Urey-Bradley terms of internal energy, van der Waals, nonpolar contribution of the solvation energy based on solvent-accessible surface area (SASA), and two solvation energy terms based on Generalized Born with a simple SWitching (GBSW).

4.4.1 Relative scoring (model ranking)

We first analyze the relative scoring performance of three models. We use the number of targets with at least acceptable docking-models (including that of medium or high-quality models according to the CAPRI iRMSD criteria [88]) when such models were among top 1, 5, or 10 ranked by a scoring function. To remove redundancy among decoys, we clustered all decoys of each target using the program FCC (Fraction of Common Contacts [89]) with default parameters and only passed those FCC retained cluster representatives to each scoring function.

For both the benchmark and CAPRI test sets, EGCN significantly outperformed IRAD. First, for the benchmark test set of 107 cases, EGCN generated 46 targets with at least acceptable top-5 predictions (the most possible number being 70 for the decoy set), representing more than 50% increase compared to 30 achieved by IRAD. The performance improvement for ranking medium and highquality predictions was even more impressive: EGCN generated 21 and 5 targets with medium and high-quality top-5 predictions, respectively, representing 62% and 150% increases compared to IRAD. Second, for the CAPRI test set, although the total number of targets (14) can be too few to provide statistical significance, EGCN again increased the number of at least acceptable, medium, and high quality top-5 models from IRAD's 4, 2, 0 to 6, 3, 2, respectively. Last, even for Score_set where test decoys were generated by diverse protocols different from EGCN training

		Top 1	Top 5	Top 10
Benchmark test set (107)	IRAD	10/0**/0***	30/13**/2***	40/18/7***
	RF	13/5**/0***	35/17**/5***	42/25**/10***
	EGCN	17/8**/2***	46/21**/5***	51/28**/11***
	Best possible	70/47**/20***	70/47**/20***	70/47**/20***
CAPRI test set (14)	IRAD	3/1**/0***	4/2**/0***	6/3**/1***
	RF	4/1**/1***	6/2**/1***	8/3/2***
	EGCN	5/1**/1***	6/3**/2***	7/4/2***
	Best possible	9/6**/3***	9/6**/3***	9/6**/3***
Score set (13)	IRAD	3/2**/0***	5/4**/1***	7/4**/2***
	RF	1/0**/0***	3/2**/0***	3/2**/1***
	EGCN	3/2**/0***	6/4**/1***	7/4**/1***
	Best possible	11/6**/3***	11/6**/3***	11/6**/3***

Table 4.2: Comparing relative scoring performance among IRAD, RF, and EGCN for three testing sets under the CAPRI criteria.

decoys, EGCN had comparable or slightly better ranking performances, producing six targets with at least acceptable top-5 predictions (including four targets with medium and one with high-quality predictions). We find the EGCN ranking performance particularly impressive considering that, unlike IRAD, the current EGCN was trained for absolute scoring and not optimized for relative scoring.

4.4.2 Absolute scoring (quality estimation)

We next analyze EGCN's absolute scoring performances and compare them to our previous RF model. IRAD, among many other scoring functions, is only for relative scoring or ranking and thus not compared here. EGCN significantly outperforms RF in quality estimation across all test sets. For the benchmark test set, EGCN estimates all docking models' interface RMSD values with an error of 1.32 Å, 1.43 Å, and 1.51 Å when the models' iRMSD values are within 4Å (acceptable), between 4 Å and 7 Å, and between 7 Å and 10 Å, respectively. These values represent 14%-22% improvement against RF's iRMSD prediction errors. Both EGCN and RF's prediction errors remain relatively flat when models are acceptable or close with iRMSD values within 10 Å whereas they rise out the range when precise quality estimation is no longer desired

for those far incorrect models.

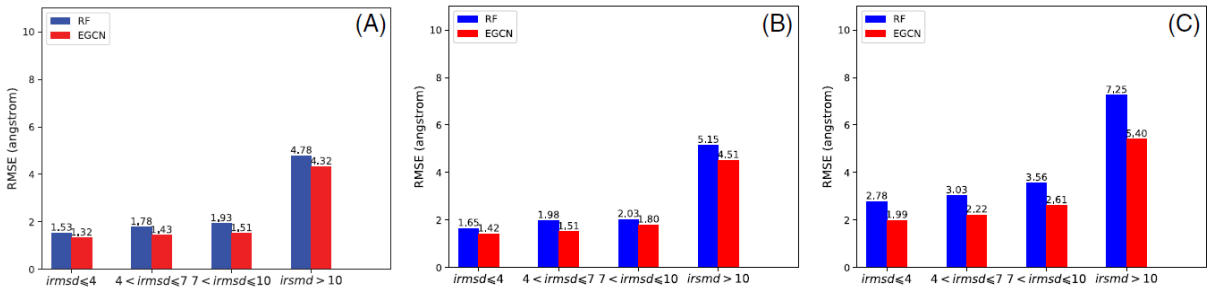


Figure 4.2: Comparing absolute scoring (quality estimation) performances among RF and EGCN. Reported are the RMSE of iRMSD predictions for A, benchmark test set, B, CAPRI test set, and C, Score_set, a CAPRI benchmark for scoring.

Very similar performances and trends are observed for the CAPRI test set and the CAPRI Score_set. EGCN's quality estimation performances only deteriorate slightly in the more challenging CAPRI test set involving homology docking: an error of 1.42 Å, 1.51 Å, and 1.80 Å when the models' iRMSD values are within 4 Å, between 4 Å and 7 Å, and between 7 Å and 10 Å, respectively. For Score_set involving diverse and distinct decoy generation protocols, EGCN's quality estimation performances further deteriorate slightly to an error of 1.99 Å, 2.22 Å, and 2.61 Å when the models' iRMSD values are within 4 Å, between 4 Å and 7 Å, and between 7 Å and 10 Å, respectively, which has shown even more improvements (27%-28%) relative to RF. It is noteworthy that these two test sets are more challenging than the benchmark test set for additional reasons. Unlike the case of relative scoring, binding affinities of native complexes are needed for absolute scoring. But their values have to be predicted for all but one target pairs in these two test sets (13 out of 14 or 12 out of 13), compared to just 34 out of 107 for the benchmark test set.

4.5 Conclusion

In this chapter, we propose EGCN, short for Energy-based Graph Convolutional Networks for scoring protein-docking models. EGCN represents the protein complex as a residue-level graph

with atom-level node and edge features. The key novel contribution of EGCN is the energy graph convolutional layer that can be interpreted as the calculating multiple energy terms. EGCN is found to perform better or equally well compared to a state-of-the-art method (IRAD) that has found great success in CAPRI as well as our previous RF model. For quality estimation that has seen few method developments in the field, EGCN is again found to outperform our previous RF model.

So far, we have finished the first part of this thesis: protein docking. From the next chapter, we will start the second part of this thesis: protein sequence-function relationship.

5. Forward Protein Sequence-to-Function Relationship*

5.1 Preface

The previous three chapters mainly studied protein–protein interactions. Besides understanding the protein–protein interactions, another fundamental aspect of biological information processing is the ubiquity of protein sequence–function relationships. The beauty and multiplicity of functions carried out by proteins arises from the huge number of different protein sequences. In the forward direction, the protein sequence encodes the functions. A detailed understanding of how these functions are encoded would allow us to more accurately reconstruct the tree of life, diagnose genetic diseases before they manifest symptoms, and design new proteins with useful properties. In the inverse direction, a protein function could map to a distribution over the protein sequence space. Although nature evolution produces the number of distinct proteins on the order of 10^{12}, a small protein of length 100 could have 20^{100} sequences. Therefore, understanding how sequences being designed from the functions would allow us to search for novel but functionally-relevant proteins in the vase majority of sequence space that is unexplored by nature.

Biologically, both mappings from protein sequence to function and from protein function to sequence are extraordinarily complex because they involve thousands of physical and chemical interactions that are dynamically coupled across multiple scales of length and time. Recently, with the data quickly accumulating on protein sequence and function, one critical question is to what extent data alone can reveal the complex sequence–function relationship. In this chapter, we study this question in-depth from the forward direction and propose a novel deep learning model: TALE: Transformer-based protein function Annotation with joint sequence-Label Embedding for predicting protein functions from sequences.

*Reprinted with permission from "TALE: Transformer-based protein function Annotation with joint sequence–Label Embedding" by Yue Cao and Yang Shen, 2021. Bioinformatics, 03, 1367-4803. Copyright © 2021 Oxford University Press.

5.2 Introduction

The explosive growth of protein sequence data in the past few decades, largely thanks to next-generation sequencing technologies, has provided enormous information and opportunities for biological and pharmaceutical research. In particular, the complex and intricate relationship between sequences, structures, and functions of proteins is fascinating. As experimental function annotation of proteins is often outpaced by sequence determination, computational alternatives have become both fundamental in exploring the sequence-function relationship and practical in predicting functions for growing un-annotated sequences (including *de novo* designs). According to the 2020_01 release of UniProt [90], there were around 5.6×10^5 nonredundant sequences manually annotated in Swiss-Prot but over two orders of magnitude more (around 1.8×10^8) sequences awaiting full manual annotation in TrEMBL.

Protein functions are usually described based on Gene Ontology (GO), the world's largest source of systematic representation of gene functions [91]. There are more than 40,000 GO terms across three domains: Molecular Function Ontology (MFO), Biological Process Ontology (BPO) and Cellular Component Ontology (CCO). Within each ontology, GO terms are structured hierarchically as a directed acyclic graph (DAG) with a root node. Each protein can be annotated with more than one GO term on three ontologies (thus a multi-label classification problem). If a protein is annotated with one GO term, then can also be annotated by all corresponding ancestral GO terms. Such a hierarchical constraint is present in many other ontologies as well, such as text ontology ([92]) and image ontology ([93]).

From the perspective of input type, computational methods for protein function annotation can be classified as sequence- [94, 17, 95], structure- [96], network- [97, 15], and literature-based [98, 99], whereas all but sequence-only methods have limited scope of usage due to data availability. Specifically, although structural information is important for understanding protein functions (e.g. [100, 101]), it is often not readily available: 0.01% of the TrEMBL sequences (UniProt 2020_01) have corresponding structural entries in Protein Data Bank (PDB), and this ratio increases to 1% if considering structural models in SWISS-MODEL Repository (SWR). Similarly, only around 7%

of the TrEMBL sequences have interaction entries in STRINGdb [102], not to mention that the network information can be noisy and incomplete. We note that structure, network, and literature data can be especially missing for novel sequences where computational function annotation is needed the most. We therefore focus on sequence-based methods in this study.

Predicting protein function from sequence alone is a challenging problem where each sequence can belong to multiple labels and labels are organized hierarchically. Critical Assessment of protein Function Annotation (CAFA) has provided an enabling platform for method development [103, 104, 105, 2] and witnessed still-limited power or scope of current methods. Sequence similarity-based methods [106, 10] leverage sequence homology, although their success is often limited to homologues and alignments to detect homology can be still costly. Recently, deep learning has emerged as a promising approach [15, 17] to improve the accuracy, where sequences are often inputs/features and GO terms are labels. However, as deep learning is a data-hungry technique, these methods often have to get rid of a large number of GO terms (labels) with few annotations, leading to narrow applicability. For instance, DeepGOPlus [17] only considered over 5,000 GO terms with at least 50 annotated sequences each, which only accounts for less than 12% of all GO terms.

We set out to overcome aforementioned barriers and boost the generalizability to sequences with low similarity as well as unseen or rarely seen functions (also known as tail labels) compared to the training data. To that end, we propose a novel approach named Transformer-based protein function Annotation through joint sequence–Label Embedding (TALE). Our contributions are as follows. First, TALE replaces previously-used convolutional neural networks (CNN) with self-attention-based transformers [13] which has made a major breakthrough in natural language processing and recently in protein sequence embedding [107, 108, 109]. Compared to CNN, transformers can deal with global dependencies within the sequence in just one layer, which helps detect global sequence patterns for function prediction much easier than CNN-based methods do. Second, TALE embeds sequence inputs/features and hierarchical function labels (GO terms) into a latent space. By considering similarities among function labels and sequence features, TALE can easily

deal with tail labels. Third, unlike previous methods that only consider GO terms as flat labels and enforce hierarchy *ad hoc* after training, TALE considers the hierarchy among labels through regularization during training. Last, we propose TALE+, by using an ensemble of top three TALE models and a sequence similarity-based method, DIAMOND [10], in convex combination (similar to DeepGOPlus) to reach the best of both worlds.

Over the CAFA3 test set, TALE+ outperformed six competing methods including GOLabeler[110], the top performer in CAFA3 for two of the three ontologies. Over a more recent and complex test set, TALE+ outperformed all other methods including NetGO [97], the upgraded version of GOLabeler [110] in all three ontologies when only sequence information is used. Importantly, TALE and TALE+ significantly improved against state-of-the-art methods in challenging cases where test proteins are of low similarity (in sequence, species, and label) to training examples. The results prove that our model can generalize well to novel sequences, novel species and novel functions.

The rest of the section is organized as follows. We will first discuss in Methods the data set used in the study. We will then introduce TALE and TALE+ models in details besides baselines and end the section with evaluation metrics. We will start the Results section with overall performance comparison. We will then delve into the analysis on model generalizability in sequence, species and function. Lastly, we will report an ablation study to delineate the major algorithm contributors to TALE's improved accuracy and improved generalizability.

5.3 TALE: Transformer-based protein function Annotation with joint sequence-Label Embedding

5.3.1 Datasets

In this study we consider two annotation time-split datasets: the standard CAFA3 dataset [2] and our curated dataset. The CAFA3 dataset is used for wider comparison with community performance. Our curated dataset, larger and more updated, is used for controlled comparison and in-depth analysis such as generalizability analysis and ablation study.

5.3.1.1 CAFA3 Dataset

We download the CAFA3 dataset, which includes a training set and a (no-knowledge) test set with experimental annotations published before September 2016 and between September 2016 and November 2017, respectively. For hyper-parameter tuning we randomly split 10% from the training set to be the validation set (thus training and validation sets are not time-split here). The sequence statistics of the CAFA3 dataset and its splits are provided in Table 5.1. Ontologies here include the molecular function ontology (MFO), biological process ontology (BPO) and cellular component ontology (CCO).

5.3.1.2 Our Dataset

We download the UniProtKB/Swiss-Prot dataset release-2015_12 (last modified on Jan. 20, 2016), release-2017_12 (last modified on Jan. 30, 2018), and the latest release-2020_05 (last modified on Dec. 2, 2020). Consistent with CAFA protocols [2], only sequences with high-quality function annotations are retained, i.e., those with at least one annotation within the following 8 experimental evidence codes: EXP, IDA, IPI, IMP, IGI, IEP, TAS, and IC. We use the remaining sequences annotated by Jan. 20, 2016, since Jan. 30, 2018, and between (but not including) the two dates as training, test, and validation sets, respectively. And we remove any test sequence that exists in the training or validation set (100% sequence identity). The sequence statistics of our curated dataset and its splits are also provided in Table 5.1.

For either dataset, we train all models on the corresponding training set and tune their hyper-parameters using the validation set. We then retrain models under their optimal hyperparameters, using both the training and the validation sets, and assess them on the corresponding test set.

5.3.1.3 Hierarchical Relationships between GO Terms

We download the identities and the hierarchical relationships between function labels (GO terms) from Gene Ontology [91] — "go-basic" release 2016-06-01 and 2020-12-08 for CAFA3 and our dataset, respectively. We consider 'is a' and 'part of' relationships in all three ontologies: Molecular Function Ontology (MFO), Biological Process Ontology (BPO) and Cellular Compo-

Dataset	Statistics	MFO	BPO	CCO
CAFA3	#Seq in Training Set	32499	48150	45537
	#Seq in Validation Set	3611	5350	5059
	#Seq in Train. + Val.	36110	53500	50596
	#Seq in Test Set	1137	2392	1265
	#GO terms	6239	19462	2434
Ours	#Seq in Training Set	34996	51345	49957
	#Seq in Validation Set	3327	2861	2300
	#Seq in Train. + Val.	38323	54206	52257
	#Seq in Test Set	1916	2836	2084
	#Seq in Test Set without net info.	1140	1743	1189
	#GO terms	6381	19939	2574

Table 5.1: Statistics of sequences and GO terms in various ontologies and splits.

nent Ontology (CCO); and do not consider cross-ontology relationships for now. In this way we obtain three separate ontologies, each of whose topology is a directed acyclic graph (DAG). For each annotation in each ontology, we additionally propagate annotations all the way from the corresponding GO term to the root node. Finally, those GO terms without a single annotated sequence are removed and not considered in this study. We note that other studies could remove GO terms with less than 10, 50, or even 250 annotated sequences thus consider much less GO terms than we do. The statistics of GO terms on the two datasets over three ontologies are shown in Table 5.1.

5.3.2 Model Overview

We will describe the details of our methods in this subsection. The overall architecture of TALE is shown in Fig. 5.1. The model has two inputs: a protein sequence and the label matrix (for capturing hierarchical relationships between all GO terms). It is worth mentioning that the label matrix is a constant matrix for a given ontology, thus being fixed during both stages of training and inference. The model itself consists of feature (sequence) embedding, label (function) embedding, joint similarity modules, as well as fully-connected and output (softmax) layers. We will introduce these components in the following subsections.

Notations. We use upper-case boldfaced letters to denote a matrix (e.g. $\boldsymbol{X}$), lower-case boldfaced letters to denote a vector (e.g. $\boldsymbol{x}$), and lower-case letters to denote a scalar (e.g. x). We use

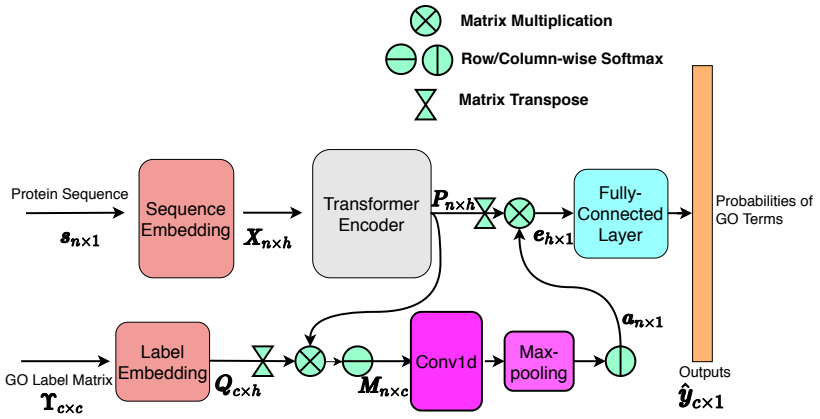


Figure 5.1: The architecture of TALE. Note that the GO label matrix is fixed for each ontology.

subscripts of a matrix to denote a specific row, column, or element (e.g. $\boldsymbol{X}_i$ for the ith row of $\boldsymbol{X}$; $\boldsymbol{X}_{,i}$ for the ith column; and $\boldsymbol{X}_{i,j}$ for the entry in the ith row and the j-th column of $\boldsymbol{X}$). We also use subscripts to denote scalar components of a vector (e.g. x_i for the ith entry of the vector $\boldsymbol{x}$). We use superscript T on a matrix to represent its transpose.

5.3.3 Sequence Embedding

Let $\mathcal{A}$ denote the set of 20 standard amino acids plus the padding symbol. For a given input protein sequence $\boldsymbol{s} \in \mathcal{A}^{n\times 1}$ of length n, we embed each character (residue) into an h-dimensional continuous latent space through a trainable lookup matrix $\boldsymbol{W}^{\text{seq}}$ and positional embedding, as described in [13]. The embedded matrix $\boldsymbol{X} \in \mathbb{R}^{n\times h}$ is fed through a transformer encoder that consist of multiple multi-head attention layers. In this study, we used 6 layers and 2 heads for each layer. The advantage of such "self-attention" layers compared to convolution layers is that self-attention can easily and quickly capture long-term dependencies within a whole sequence, whereas convolution can only capture dependencies of residues within neighborhoods determined by convolutional kernals. We denote the output matrix of the transformer encoder with $\boldsymbol{P} \in \mathbb{R}^{n\times h}$, where h is the hidden dimension.

5.3.4 Label Embedding.

For each given ontology represented as a directed acyclic graph (DAG), we first perform topological sorting of its nodes (GO terms or labels) and assign an index to each node based on its order in the sorted array. We then embed node i into a c-dimensional binary vector $\boldsymbol{\Upsilon}_i \in \{0,1\}^{1\times c}$ where c is the number of labels or GO terms. In this way we embed all nodes within the ontology with a label matrix $\boldsymbol{\Upsilon} \in \{0,1\}^{c\times c}$ where its ith row $\boldsymbol{\Upsilon}_i \in \{0,1\}^{1\times c}$ is the embedding vector for node i and its element $\boldsymbol{\Upsilon}_{i,j}$ is 1 if node j is an ancestor of node i and 0 otherwise: $\boldsymbol{\Upsilon}_{i,j} = 1\ (\forall j \in \mathrm{Anc}(i))$, where $\mathrm{Anc}(i)$ denotes the set containing all ancestors of node i (plus i itself). Similar to the sequence embedding, we use a trainable lookup matrix $\boldsymbol{W}^{\mathrm{label}} \in \mathbb{R}^{c\times h}$ to encode $\boldsymbol{\Upsilon}$ as $\boldsymbol{Q} \in \mathbb{R}^{c\times h}$, where

$$\boldsymbol{Q}_i = \boldsymbol{\Upsilon}_i \cdot \boldsymbol{W}^{\mathrm{label}}. \tag{5.1}$$

Unlike sequence input, the label matrix $\boldsymbol{\Upsilon}$ is fixed for each ontology and thus not needed to be further encoded using a transformer encoder.

5.3.5 Joint Sequence–Label Similarity

We inspect the contributions of individual amino acids to individual function labels, by calculating the matrix product between $\boldsymbol{P}$ and $\boldsymbol{Q}$ to measure the joint similarity between the sequence and the label:

$$\boldsymbol{M} = \mathrm{softmax}(\boldsymbol{P} \cdot \boldsymbol{Q}^T), \tag{5.2}$$

where the softmax is row-wised. $\boldsymbol{M}_{i,j}$ suggests the "closeness" or similarity score between amino acid i and label j. For each amino acid i, we further consider the contributions from other amino acids, by applying a 1D convolutional layer to $\boldsymbol{M}$ (along the row direction with the columns as channels), followed by a max-pooling layer. The output of the max-pooling layer is first normalized, and then used for weighting the sequence encoding matrix $\boldsymbol{P}$:

$$\boldsymbol{e} = \boldsymbol{P}^T \cdot \boldsymbol{a}, \tag{5.3}$$

where $\boldsymbol{a} \in \mathbb{R}^{n \times 1}$ is the output of the max-pooling after column-wise softmax.

5.3.6 Fully-connected and Output Layers

The output of the joint similarity module, $\boldsymbol{e} \in \mathbb{R}^{h \times 1}$, would go through two fully-connected (FC) layers, with the sigmoid activation function at the second FC layer. The output of the model $\hat{\boldsymbol{y}} \in \mathbb{R}^{c \times 1}$ is the predicted probabilities for individual GO terms in the ontology, where the ith component is the predicted probability of label i for a given input sequence.

5.3.7 Loss and Hierarchical Regularization

To train model parameters, we first consider the binary cross-entropy loss:

$$L' = -\frac{1}{c} \sum_{i=1}^{c} y_i \times \hat{y}_i + (1 - y_i) \times (1 - \hat{y}_i) \tag{5.4}$$

However, if we only use L', trained models may make predictions violating the hierarchical constraint of function annotation. For instance, the predicted score (probability) of a child GO term may be larger than the scores of ancestors. To mitigate such hierarchical violation, we then introduce an additional, hierarchical regularization term:

$$R = \frac{1}{|E|} \sum_{(i,j) \in E} \max(0, \hat{y}_j - \hat{y}_i) = \frac{1}{|E|} \sum_{(i,j) \in E} \mathrm{ReLU}(\hat{y}_j - \hat{y}_i), \tag{5.5}$$

where E is the set of all edges in the ontology graph, and (i, j) is one edge in E pointing from node i to j. Therefore, our overall loss function is a weighted sum of both terms: $L = L' + \lambda R$, where λ, the regularization constant to control the balance between the two terms, is treated as a hyper-parameter and tuned along with other hyper-parameters using the validation set.

5.3.8 Ensemble Model of TALE+

So far we have introduced all components of TALE. In order to reduce the variance of predicted scores and their generalization errors, we consider to first use the simple average of the outputs of top 3 models based on the validation set, as the final TALE predictions.

Similar to DeepGOPlus [17], we further use a convex combination of TALE (the simple average) and DIAMONDScore [10] as final outputs of TALE+:

$$\hat{\boldsymbol{y}}_{\text{TALE+}} = \alpha(\hat{\boldsymbol{y}}_{\text{TALE, 1}} + \hat{\boldsymbol{y}}_{\text{TALE, 2}} + \hat{\boldsymbol{y}}_{\text{TALE, 3}})/3 + (1-\alpha)\hat{\boldsymbol{y}}_{\text{DIAMOND}} \tag{5.6}$$

where $\hat{\boldsymbol{y}}_{\text{TALE},i}$ is from the ith best TALE model based on the validation set. After tuning on the validation set, the best αs for three ontologies were set to be 0.4 for MFO, 0.5 for BPO, and 0.7 for CCO. TALE can be regarded as a special case of TALE+ when $\alpha = 1$.

5.3.9 Experiment Details

We compare TALE and TALE+ to six competing methods, including baselines, latest published methods and top performers in CAFA. They include a naive approach of using background frequency of GO terms [2]; sequence similarity-based DIAMONDScore [10]; deep learning-based DeepGO (with network information) [15] and recent extensions DeepGOCNN and DeepGOPlus (DeepGOCNN + DIAMONDScore) [17]; and NetGO [97] that merges the network information and its earlier sequence-based GOLabeler [110]. For DeepGO, DeepGOCNN and DeepGOPlus, we used their codes published on GitGub and trained the models on our datasets.

5.3.10 Evaluation

For a test set D_{test}, we use two evaluation metrics: Fmax and AuPRC. Fmax is the official, protein-centric evaluation metric used in CAFA. It is the maximum score of the geometric average of averaged precision and recall over proteins for all thresholds:

$$\text{Fmax} = \max_t \left(\frac{2\overline{\text{Pre}}(t) \cdot \overline{\text{Rec}}(t)}{\overline{\text{Pre}}(t) + \overline{\text{Rec}}(t)}\right), \tag{5.7}$$

where $\overline{\text{Pre}}(t)$ and $\overline{\text{Rec}}(t)$ are the averaged precision and recall at threshold t. Specifically,

$$\begin{aligned}\overline{\text{Pre}}(t) &= \frac{1}{Q(t)} \sum_{k=1}^{Q(t)} \frac{\boldsymbol{y}_k \cdot \hat{\boldsymbol{y}}_i(t)}{|\hat{\boldsymbol{y}}_k(t)|_1} \\ \overline{\text{Rec}}(t) &= \frac{1}{|D_{\text{test}}|} \sum_{k=1}^{|D_{\text{test}}|} \frac{\boldsymbol{y}_k \cdot \hat{\boldsymbol{y}}_k(t)}{|\boldsymbol{y}_k|_1},\end{aligned} \tag{5.8}$$

where $Q(t)$ is the number of samples that have least one non-zero label in D_{test}; $\boldsymbol{y}_i$ is the true label vector of kth sample in D_{test}, and $\hat{\boldsymbol{y}}_k(t)$ is the predicted label vector of the kth sample in D_{test} at threshold t. For the calculation, we iterated t incrementally from 0 to 1 at a stepsize of 0.01.

AuPRC is a standard metric in machine learning for evaluating the binary classification performance, especially suitable for highly imbalance data, which is often the case in protein function annotation. In multi-label classification, we concatenate all the label vectors and use canonical AuPRC (single-label) to evaluate the performance.

5.4 Results

We perform comprehensive evaluation of our models from several perspectives. We will start with comparing them to aforementioned competing methods on various ontologies and test sets. We will proceed to assess the capability of all models to generalize to novel sequences, novel species, and novel functions relative to the training set. We will also conduct an ablation study for TALE and TALE+ to delineate the contributions of their various algorithmic components to overall performances and various generalizability.

5.4.1 Performance on the CAFA3 test set

We first compare TALE and TALE+ with competing methods over the CAFA3 testset. The results are shown in Table 5.2. Overall, TALE+ achieved the best performance on biological process (BPO) and cellular component (CCO); and was the second best (next to GOLabeler) on molecular function (MFO). Compared to GOLabeler (a top performer in the official assessment of CAFA3), TALE+ improved the Fmax from 0.400 to 0.431 (by 8%) on BPO and from 0.610 to 0.669 (by nearly 10%) on CCO while having a slightly worse Fmax on MFO (0.615 versus

	Fmax			AuPRC		
Ontology	MFO	BPO	CCO	MFO	BPO	CCO
Naive	0.331	0.253	0.541	0.312	0.173	0.483
DIAMONDScore	0.532	0.382	0.523	0.461	0.304	0.500
DeepGO	0.392	0.362	0.502	0.312	0.213	0.446
DeepGOCNN	0.411	0.388	0.582	0.402	0.213	0.523
DeepGOPlus	0.552	0.412	0.608	0.502	0.313	0.564
GOLabeler[1]	**0.620**	0.400	0.610	/	/	/
TALE	0.548	0.398	0.654	0.471	0.317	0.626
TALE+	0.615	**0.431**	**0.669**	**0.548**	**0.370**	**0.652**

Table 5.2: The performance of TALE and TALE+ against competing methods on the CAFA3 test set. Boldfaced are the top performance (Fmax or AuPRC) for each ontology [1]Top performer in CAFA3, GOLabeler's Fmax results (AuPRC being unavailable) are obtained from the official assessment [2]. Its upgraded version NetGO is only available in a webserver whose training set may include CAFA3 test sequences, thus not compared.

0.620). Speaking of TALE without the help of similarity-based DIAMONDScore, it tied with GOLabeler being the third best on BPO (next to TALE+ and DeepGOPlus, both of which use DIAMONDScore); and it achieved the second best on CCO (only next to TALE+; and improving Fmax from 0.610 to 0.654 by 7% compared to GOLabeler).

5.4.2 Performance on our test set

We then compare TALE and TALE+ with competing methods over our larger and newer test set and show the results in Table 5.3. Overall, TALE+ again achieved the best performance in BPO and CCO; and had the second best in molecular function (MFO). The best performer in MFO, NetGO was also the second best performer in BPO and the third in CCO. Specifically, compared to NetGO, TALE+ improved Fmax from 0.423 to 0.459 (by 9%) in BPO and from 0.636 to 0.677 (by 6%) in CCO; and it had worse Fmax (0.703 vs. 0.667) in MFO. TALE alone without adding similarity-based DIAMONDScore outperformed all other methods except our own TALE+ in CCO: compared to NetGO, TALE alone improved Fmax from 0.636 to 0.658 (by 4%). It is noteworthy that NetGO uses additional network information that is not used in TALE or TALE+. Such network information is often not available to proteins and, in such cases, TALE+

outperformed NetGO in all three ontologies including MFO, as shown in Table 5.4.

	Fmax			AuPRC		
Ontology	MFO	BPO	CCO	MFO	BPO	CCO
Naive	0.407	0.281	0.599	0.329	0.198	0.545
DIAMONDScore	0.582	0.359	0.548	0.505	0.207	0.448
DeepGO[2]	0.423	0.231	0.523	0.345	0.134	0.478
DeepGOCNN	0.476	0.266	0.616	0.419	0.162	0.563
DeepGOPlus	0.634	0.384	0.632	0.587	0.235	0.578
NetGO[2]	**0.703**	0.423	0.636	**0.634**	0.311	0.623
TALE	0.578	0.336	0.658	0.514	0.247	0.635
TALE+	0.667	**0.459**	**0.677**	0.604	**0.326**	**0.643**

Table 5.3: The performance of TALE and TALE+ against competing methods on our test set. [2]Note that both DeepGO and NetGO use network information besides sequence, whereas other methods including TALE and TALE+ use sequence alone.

The performance difference among the three ontologies, as observed in both datasets, can be attributed to the structure and complexity of the ontology as well as the available annotations [104]. Similarity-based DIAMONDScore performed much better than the naive approach (and did well among all methods) in MFO but worse in CCO. This observation echoes the hypothesis that sequence similarity may carry more information on basic biochemical annotations than cellular components. Interestingly, CCO is also the ontology where TALE and TALE+ did the best – even TALE alone was better than all methods other than TALE+ and adding similarity-based DIAMONDScore to TALE resulted in less help than it did in MFO and BPO.

5.4.3 Performance on our test set without network information

Protein-protein interaction (PPI) network information can be very useful in boosting the accuracy of computational protein function annotation. However, its availability can be limited and, when available, its quality can also be limited (noisy and incomplete). Table 5.1 shows that only round 40% of our test set has corresponding network information in STRINGdb, which is already biased considering that all test sequences have already been functionally annotated with experi-

ments until now. In reality, only 7% of TrEMBL sequences have corresponding STRINGdb entries available, regardless of the quality of the network information. The ratio can be even worse for *de novo* designed protein sequences. Therefore, a reliable sequence-only function annotation method is necessary especially when network information is not available.

We thus did performance analysis over the portion of our test set without network information, i.e, the test sequences whose network information cannot be found in STRINGdb (see statistics in Table 5.1). In this case alone we used the version of DeepGO codes without network features. As shown in Table 5.4, TALE+ significantly outperformed all competing methods in all three ontologies. Compared to NetGO, TALE+ improved Fmax from 0.643 to 0.661 (by 3%) in MFO, from 0.396 to 0.473 (by 19%) in BPO, and from 0.643 to 0.693 (by nearly 8%) in CCO.

	Fmax			AuPRC		
Ontology	MFO	BPO	CCO	MFO	BPO	CCO
Naive	0.416	0.283	0.597	0.324	0.198	0.548
DIAMONDScore	0.571	0.367	0.528	0.488	0.210	0.450
DeepGO	0.423	0.253	0.572	0.336	0.213	0.523
DeepGOCNN	0.461	0.273	0.625	0.391	0.167	0.576
DeepGOPlus	0.631	0.396	0.644	0.575	0.246	0.604
NetGO	0.643	0.396	0.643	**0.592**	0.222	0.544
TALE	0.539	0.346	0.669	0.485	**0.258**	0.649
TALE+	**0.661**	**0.473**	**0.693**	**0.591**	0.249	**0.669**

Table 5.4: The performance of TALE and TALE+ against competing methods on the portion of our test set that does not have network information available.

5.4.4 Generalizability from the training set to the test set

Despite the improved performances of our models, questions remain on their practical utility. Are they useful in cases where function annotation is needed the most? Sequence-based protein property prediction (e.g., fold, structure, and function) is needed the most in the "twilight" or even "midnight" zone where similarity to known annotated sequences is too low to sustain the assumption that similar inputs (sequences) imply similar outputs (aforementioned properties). Similarly, it

is also needed the most in the “twilight" or “midnight” zone in another sense, where examples for some specific function labels are never or rarely seen in known annotated sequences (thus referred to as tail labels).

Besides the practical questions on model applicability in the twilight or midnight zone, fundamental biological questions also remain. Have these machine learning models learned anything fundamental in sequence–function relationships? Or are they merely mimicking patterns in training data using a complicated function (such as a neural network) that could overfit?

To answer these questions, we examine our models and competing ones in their generalizability from the training set to the test set. Models considered include sequence-based Naive, DIAMOND, DeepGOCNN, DeepGOPlus, TALE and TALE+. As the generalizability analysis requires similarity measures between the training and test sets, we couldn't include the NetGO webserver for this analysis because its training set is neither accessible nor guaranteed to be identical to ours. Specifically, we examine the generalizability of these models from three perspectives: sequence, species, and label (function) as follows.

5.4.4.1 Sequence generalizability

To analyze the generalizability to novel sequences, we split our test set into bins of various sequence-identity levels compared to the training set and examine various models' accuracy (Fmax) over these bins. Specifically, sequence identity between a test sequence and the training set is measured by maximum sequence identity (MSI) [1] We partition the test sequences into 4 bins based on MSI: $[0, 0.2]$ (midnight zone), $(0.2, 0.3]$ (twilight zone), $(0.3, 0.4]$ and $(0.4, 1.0]$ (safe zone), and show the sequence counts (gray dots) and the Fmax scores (colored bars) in these bins (Fig. 5.2).

As shown in Fig. 5.2, sequence similarity-based method DIAMOND performed poorly compared to deep learning-based models and even the background frequency-based naive approach, when sequence identity is below 20% (midnight zone) or between 20% and 30% (twilight zone).

[1] Sequence identity is measured as the number of identical residues divided by the length of the alignment. Sequence alignment is performed with the Needleman–Wunsch algorithm [**?**], using the substitution matrix of BLOSUM62 and the gap penalty as the EBI webserver “EMBOSS Needle”.

Meanwhile, without a surprise, DIAMOND was the close-to-best performer in MFO, BPO, and CCO, when sequence identity is above 40% (safe zone). Between DeepGOCNN and TALE that do not use sequence-similarity scores from other sources, TALE outperformed DeepGOCNN in all bins. In fact TALE was even the best performer among all on MFO and CCO when sequence identity is below 20%. For all other combinations of the ontology and the bin, TALE+ had the best performance. As a convex combination of DIAMONDScore and TALE, TALE+ maintained the impressive performances of TALE in the midnight and twilight zones and significantly improved from TALE in the safe zone using the similarity-based information.

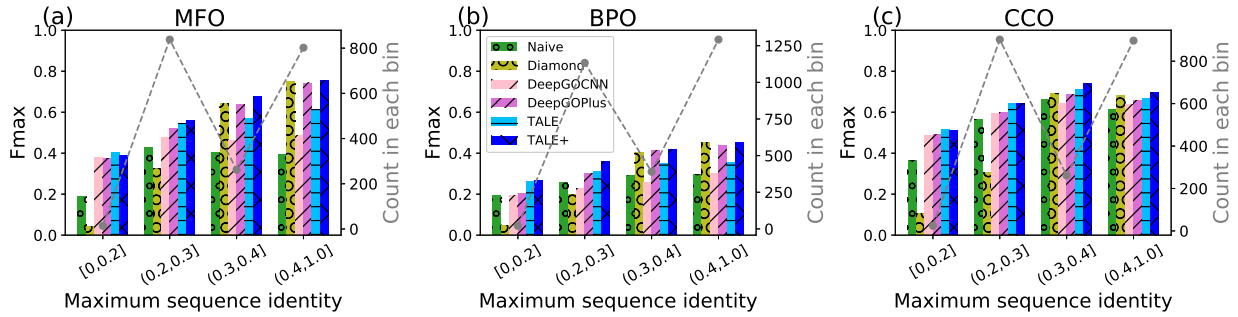


Figure 5.2: The Fmax performances of six models in three ontologies, over 4 bins of increasing sequence-identity ranges. Low sequence identity indicates low similarity between a test sequence and the training set. Sequence statistics over the bins (gray dots connected in dashed lines) are also provided.

5.4.4.2 Species generalizability

To analyze the generalizability to new species, we count for every test sequence the Number of training Samples of the Same Species (NSSS) and identify those in new species never seen in the training set (NSSS=0). We further remove "safe zone" test sequences whose maximum sequence identities (MSI) to the training set are above 40%. The remaining test sequences are thus in new species and low similarity compared to the training set.

Fig. 5.3 shows the counts (gray dots) and the Fmax scores of various models (colored bars) over prokaryotes (including bacteria and archaea) and eukaryotes. Again, TALE+ outperformed

all competing methods for all six combinations of clades and ontologies; and TALE remained the second best for all combinations except for prokaryotic CCO where DeepGOPlus, blending DeepGOCNN and similarity-based DIAMONDScore, edged TALE. Similarity-based DIAMOND performed very poorly while the naive approach was mediocre.

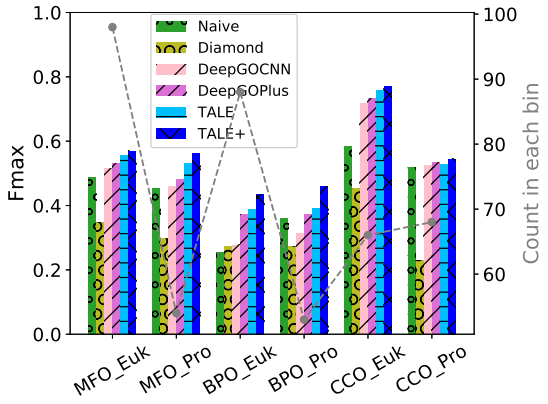


Figure 5.3: The Fmax performances of six models in three ontologies, over eukaryotes and prokaryotes with NSSS=0 (new species) and MSI $\leqslant$ 40% (low similarity).

Interestingly, all methods performed better for new eukaryotic species than for new prokaryotic species in MFO and CCO.

To understand this pattern, we compared the MSI between the sequences in new eukaryotic species and those in new prokaryotic species on each ontology (MSI below 40%). We found that, the MSIs for sequences from new eukaryotes are in distribution higher than those for sequences from new prokaryotes (P-values ranging from 3E-4 to 5E-8 according to one-sided Kolmogorov–Smirnov test), potentially posing an easier prediction task.

5.4.4.3 Function generalizability

To analyze the generalizability to new or rarely annotated functions (GO labels),

we first calculate the frequency of the ith label in the training set:

$$f(i) = \frac{1}{|D_{\text{train}}|} \sum_{j=1}^{|D_{\text{train}}|} y_{i,j}, \tag{5.9}$$

where $|D_{\text{train}}|$ is the number of training samples and the binary $y_{i,j}$ is for the ith label of the jth training sample. Then for the kth test sample, we calculate its average label frequency (ALF) in the training set as:

$$\text{ALF}(k) = \frac{1}{|\boldsymbol{y}_k|_1} \sum_{j=1}^{c} f(j) \cdot y_{k,j} \tag{5.10}$$

where c is the number of labels, the binary $y_{k,j}$ is for the jth label of the kth sample in the test set, and $\boldsymbol{y}_k$ is the stacked vector over all labels. We split the test set into 4 bins based on ALF: $[0, 0.2]$, $(0.2, 0.3]$, $(0.3, 0.4]$, and $(0.4, 1.0]$, and provide the histograms of the sequences over these bins in Fig. 5.4 (gray dots). We note that ALF is a protein centric measure that is a protein-specific average of individual GO term-centric frequencies.

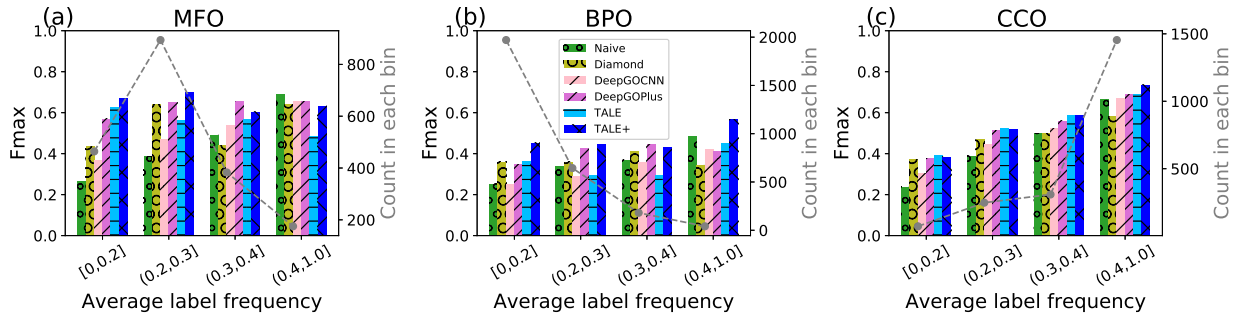


Figure 5.4: The Fmax performances of six models in three ontologies, over 4 bins of increasing function/label frequencies (for each test sequence, average label frequencies in the training set, measured by $\text{ALF}(\cdot)$). Low ALF bins indicate proteins with functions rarely annotated in the training set.

As shown in Fig. 5.4, the performances of both the naive approach and DeepGOCNN deteriorated noticeably as the average label frequency of a test sequence decreases, indicating that current

deep learning models for function annotation do not necessarily lead to better performances in such scenarios. Similarity-based method DIAMOND also had mediocre performance especially in low ALF ($\leqslant 0.3$). In all six combinations of 3 ontologies and 2 low ALF bins ($[0, 0.2]$ and $(0.2, 0.3]$), TALE (twice in CCO) or TALE+ (four times in MFO or BPO) had the best performance. In the other six combinations involving medium or high ALF, TALE+ led all other methods by being the best performer thrice, followed by DeepGOPlus (twice) and the naive approach (once). These results attest to the advantage of our models that target tail labels using joint sequence-label embedding. Interestingly, adding similarity-based DIAMONDScore to TALE did not always lead to a further improved TALE+, as seen in the two lowest-ALF bins in CCO.

In total, TALE and TALE+ outperformed all competing methods in all generalizability tests (low sequence similarity, new species, and rarely annotated functions), echoing our rationale that joint embedding of sequences and hierarchical function labels to address their similarities would significantly improve the performance for novel sequences and tail labels. In addition, combining TALE and similarity-based DIAMOND into TALE+ also enhanced the performances in some cases. In contrast, similarity-based DIAMOND did not generalize well to novel sequences in low similarity, new species and rarely annotated functions compared to the training ones, whereas the frequency-based naive approach had mediocre performance. Deep learning-based DeepGOCNN performed poorly for rarely annotated functions (tail labels), whereas DeepGOPlus combining DeepGOCNN and similarity-based DIAMONDScore improved the performance relative to DeepGOCNN.

5.4.5 Ablation Study

To rigorously delineate the contributions of algorithmic innovations that we have made in TALE and TALE+ to their improved performances and superior generalizability, we perform the following ablation study. Starting with DeepGOCNN, we incrementally add algorithmic components and introduce variants to eventually lead to TALE and TALE+:

- **B1**: replacing the convolutional layers in DeepGOCNN with the transformer encoder plus the input embeddings. Unlike Eq. (5.3) where $\boldsymbol{a}$ is the output from joint label embedding,

the output of the encoder here $\boldsymbol{P} \in \mathbb{R}^{n \times h}$ would be simply row-averaged to obtain $\boldsymbol{e}$;

- **B2**: replacing the DeepGOCNN post-training correction of hierarchical violations in **B1** with the additional loss term of hierarchical regularization (Eq. 5.5);
- **B3**: adding label embedding and joint similarity modules to **B2** for joint sequence–label embedding.

TALE is using the average of the top-3 **B3** models (based on the validation set) and TALE+ is a convex combination of TALE and DIAMONDScore.

The overall performances of the above models over the test set are summarized in Fig. 5.5(a). From DeepGOCNN to TALE+, both Fmax and AuPRC are gradually increasing over model variants. In MFO and BPO, the convex combination with DIAMONDScore had the largest single contribution to the overall Fmax or AuPRC increase, whereas the TALE innovations, including the transformers, hierarchical regularization, label embedding (jointly with sequence embedding) and model ensembling, together contributed about the same. In CCO, DIMAONDScore still contributed but the TALE innovations contributed much more in improving both measures.

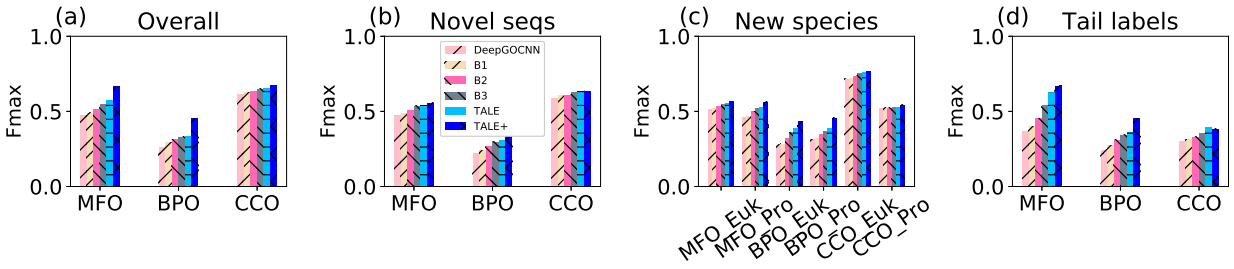


Figure 5.5: The Fmax performances of various models in the ablation study for (a) the overall test set as well as test sequences (b) with MSI$\leqslant$30%, (c) in new eukaryotic or prokaryotic species and with MSI$\leqslant$40%, or (d) average label frequency $\leqslant$ 20%.

Various generalizability of the above models is also assessed over test sequences with low similarity (MSI$\leqslant$30%), in new species (and with MSI$\leqslant$40%), or rarely annotated functions (ALF$\leqslant$20%),

as summarized in Fig. 5.5(b)–(d), respectively. Interestingly, similarity-based DIAMONDScore was no longer found the largest single contributor to Fmax improvements except in BPO, and sometimes found to hurt the performance. Instead, joint embedding, hierarchical regularization and sometimes transformers, were often found the largest contributors to many Fmax improvements among combinations of generalizability types and ontologies considered, which attests to the rationales of our algorithm designs.

5.5 Conclusion

In this chapter, we focus on the forward sequence-to-function prediction problem, and described a novel transformer-based deep learning model named TALE, with joint embedding of sequence inputs and hierarchical function labels. The transformer architecture could learn sequence embedding while considering the long-term dependency within the sequence, which could generalize better to sequences with low similarity to the training set. To further generalize to tail labels (functions never or rarely annotated in the training set), we learn the label embedding, jointly with the sequence embedding, and use their joint similarity to measure the contribution of each amino acid to each label. The similarity matrix is further used to reweigh the contributions of each amino acid toward final predictions. In addition, we use TALE+, a convex combination of TALE and a similarity-based method, DIAMOND, to further improve model performances and generalizability.

Our results on a time-split test set demonstrate that TALE+ outperformed all sequence-based methods in all three ontologies and outperformed the state-of-the-art hybrid method (using network information) in BPO and CCO. When network information is not available, TALE+ outperformed all competing methods in all ontologies. Importantly, both TALE and TALE+ showed superior generalizability to sequences of low homology (and in never-annotated new species) and never/rarely annotated functions, echoing the rationales of our algorithm development. Ablation studies indicate that our newly introduced algorithmic components, especially transformer encoders and joint sequence–label embedding, contributed the most to such sequence, species, and function generalizability, whereas sequence similarity-based DIAMONDScore sometimes helped.

TALE proposes a feasible way for solving the forward sequence-to-function prediction problem. In the next chapter, we will study the protein sequence-function relationship in the opposite direction: from function to sequences.

6. Inverse Protein Function-to-Sequence Relationship

6.1 Preface

The previous chapter studied the protein sequence–function relationship in the forward direction and proposed a novel algorithm for predicting the protein functions from protein sequences. In this chapter, we focus on the inverse direction: from functions to sequences. To the end, we propose Func2Seq: a novel conditional transformer-based autoregressive generator for functional protein sequence design.

6.2 Introduction

Designing novel proteins with desired functions can result in enormous impact on biomedical and pharmaceutical applications [111], including vaccination [112], immunotherapy [113], and biosensors [114]. Directed evolution[115, 116, 117] and physics-based force-field modelling [118, 119] are two main approaches in the past several decades. Directed evolution [115] relies on random mutations of known proteins with screening and selection, while physics-based modelling requires energy calculation for protein structures in every iteration of the algorithm. Despite great success, those methods not only are time-consuming, but also explore only a tiny fraction of the entire sequence space.

The challenges of functional protein engineering originate from two perspectives. First, the extremely large protein sequence space poses a direct barrier for any types of search algorithm. For instance, a small protein of length 100 could have 20^{100} sequence combinations. Second, the functional landscape in such a huge sequence space could lay in an irregular, non-smooth manifold. For instance, the neighbor of a functional sequence with just a single mutation could lead to the complete loss of function [120].

In order to overcome aforementioned barriers, a promising direction is to use data-driven methods. The data-driven methods, especially deep learning, have already revolutionized several fields outside of the realm of biology, such as computer vision [121] and natural language process-

ing [13]. They also show potential power for structure-based [122] , fold-based [123, 124] and function-based protein design [125, 126, 127]. More recently, AlphaFold2 [128], a deep learning model for protein structure prediction surpassed all competing methods with a big margin in the 14th Critical Assessment of Techniques for Protein Structure Prediction (CASP14) competition, which also motivates a strong confidence in utilizing deep learning for inverse design.

In the past few years, a number of data-driven methods have already been developed for learning the functional landscape in the sequence space. For example, Shin *et al* [125] used an autoregressive generative model for mutational effect prediction and functional protein design. Greener *et al* [126] developed a conditional variational autoencoder (VAE) for designing metalloproteins. Hawkins *et al* [129] also used VAE to generate novel, functional variants of the luxA bacterial luciferase. Lastly, Madani *et al* [127] trained a 1.2B-parameter language model conditioned on taxonomic and keyword tags.

Despite various eventual goals, the core of those methods (except Madani *et al*) is a unconditional generative model trained through a single family dataset. However, the knowledge of the functional sequences, which characterizes the functional landscape, should not be limited within the sequence family of desired functions, but shared across other functions and their corresponding sequence families. For instance, proteins for magnesium ion binding (Gene Ontology ID: GO:0000287) have the same predominant motif of secondary structures as proteins for manganese ion binding (Gene Ontology ID: GO:0030145). To utilize such knowledge, one must consider the relationships of different functions in the functional space. So a learned functional embedding for the generative model is necessary. In this sense, the one-hot tag input in Madani *et al*'s method is not suitable for learning the functional dependency when generating sequences.

In order to utilize the knowledge from other functions and their corresponding protein families, we develop a novel meta-training framework for functional protein engineering: 1) We combine the information of the node definition/description and the graph topology in the Molecular Function Ontology (MFO) to embed each node function into the latent space using a novel two-stage graph convolutional networks(GCN) + BioBERT[130]. We train such networks to obtain an em-

bedding vector for each node. 2) We then build a transformer-based autoregressive generative model conditioned on embedding vectors through the entire RP15 Pfam dataset [131].

We first train our GCN for embedding the MFO nodes into the latent space. Moreover, we combine two domain-to-function annotations datasets, Pfam2GO [132] and GODM [133] for annotating each Pfam domain by GO terms. We then pretrain our generative meta-model using the entire RP15 dataset with learned function embeddings as the inputs. Finally, for each downstream task for specific functions, we fine-tune our meta-pretrained model on each specific protein family.

Our results on mutational effect prediction show the improved performances in 35 out of 40 proteins against the state-of-the-art alignment-free method [1]. The ablation study further demonstrates the advantages of the meta-pretraining.

Moreover, we use our model to generate domain sequences for a DNA binding domain conditioned on the functions of that domain. Our results show that our model can learn the evolutionary information of domain sequences and, at the same time, design diverse and novel sequences with conserved functional motif. The results demonstrate the power of our model for designing functional sequences and expanding the current sequence space within the the allowed biological constraints of such functions.

6.3 Methods

In this section, we will describe our method in details.

6.3.1 Data

6.3.1.1 Gene Ontology and Family Annotation

We download the identities and the hierarchical relationships between function labels (GO terms) from Gene Ontology [91] — "go-basic" release 2020-12-08. We consider 'is a' and 'part of' relationships in Molecular Function Ontology (MFO) and do not consider cross-ontology relationships for now. In this way we obtain one ontology which is a directed acyclic graph (DAG). As a result, our final ontology has 11163 nodes.

We then download the Protein family and GO association data from Pfam2GO and GODM.

We combine two datasets together and only consider the GO terms in the aforementioned MFO.

6.3.1.2 Pretraining

We download the RP15 dataset from Pfam website [131]. We exclude those sequences 1) that have non-natural amino acids, 2) whose family has no GO annotations, and 3) whose sequence lengths are more than 800. As a result, we have 8,772,255 sequences that will be used for pretraining a meta model.

6.3.1.3 Mutational Effect Prediction

We use the training and test datasets in Shin2021 *et al*[1] which are kindly provided by Marks's group. The dataset for fine-tuning includes 40 proteins, each of which has one sequence family training set and a mutation effect test set.

6.3.1.4 DNA binding domain

We consider the ARID domain (PF01388) in the Pfam for designing functional protein sequences. We choose this family due to that 1) it has very sufficient and diverse natural sequences, and 2) it has ground-truth motif assessed *in vitro*. We use Pfam UniProtKB which has 19,077 sequences. We filter out those sequences that either have non-natural amino acids (0.2% of all sequences), or are longer than 100 amino acids (1.5% of all sequences). As a result, this dataset for fine-tuning has 18,740 sequences.

6.3.2 Graph Ontology Embeddings

In this section, we describe how we embed the nodes in the Gene Ontology graph through a two-stages GCN plus BioBERT text embeddings.

6.3.2.1 Node Features by BioBERT.

We use BioBERT v1.1 [130] to embed both the definition phrase and the description sentence of each node to be its node features. Specifically, for each definition phrase with length n_{def} and each description sentence with length n_{def}, the BioBERT embeds them into two dense matrices with shape (n_{def}, h) and (n_{des}, h), respectively, where h is the hidden dimension and is set to be

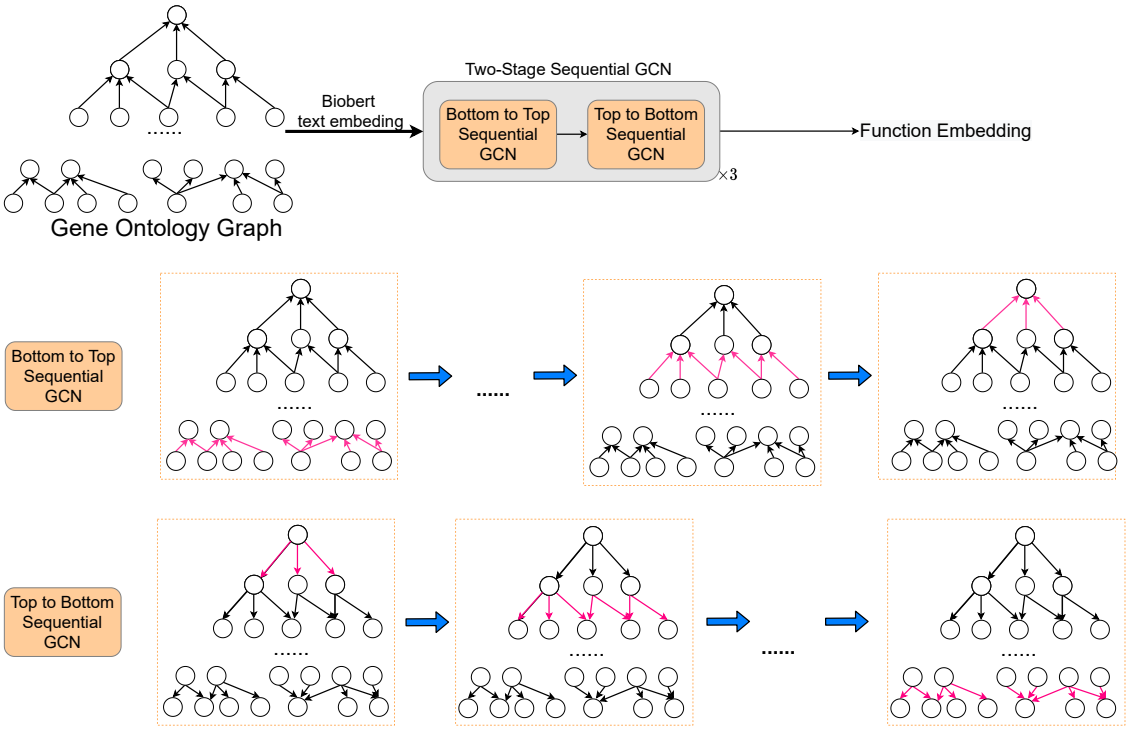


Figure 6.1: The architecture of graph ontology embedding.

768 by BioBERT. We then take the mean of each dense matrix along the sequence direction and then take the average over the two. As a result, we finally have a 768-length dense vector, which will be used as the node features during graph embedding.

6.3.2.2 Graph Ontology Embedding by TSGCN.

After obtaining the node features, we embed the whole ontology graph through a novel Two-stage Sequential Graph Convolutional Networks (TSGCN), as shown in Fig. 6.1. The key component of TSGCN is a sequential GCN (SGCN) module. Since each Gene Ontology graph is a directed acyclic graph (DAG), we consider the hierarchical relationship among the nodes. Therefore, unlike traditional graph convolutional networks, whose node features are aggregated isotropically and locally, we aggregate the node features anisotropically and globally in order to respect the hierarchical and directional relationships within the graph. Specifically, we aggregate the node features in two directions: bottom to top (B2T) (Fig. 6.1 middle) and top to bottom (T2B) (Fig. 6.1 bottom). The aggregation function is otherwise the same as in the original GCN paper. We

concatenate B2T-SGCN and T2B-SGCN sequentially as one TSGCN layer and we use 3 layers eventually for graph ontology embedding.

6.3.2.3 Self-supervised Loss Functions (Lower Bound of Mutual Information)

The goal of ontology embedding is to make semantically similar nodes similar in the latent space as well. To achieve this goal, we follow the ordinal contrast and metric learning framework and try to maximize the similarity between two semantically positive nodes and minimize the similarity between two negative nodes. For any two positive pair nodes (u, v), we want to maximize their mutual information between their latent representations $\boldsymbol{z}_u$ and $\boldsymbol{z}_v$. Inspired by the InfoNCE loss [134, 135], we construct the self-supervised loss as:

$$L(v) = -E_{P(u|v)}\text{SIM}(\boldsymbol{z}_v, \boldsymbol{z}_u) + \log N E_{P(u)} \exp \text{SIM}(\boldsymbol{z}_v, \boldsymbol{z}_u) \tag{6.1}$$

where $P(v)$, $P(u)$ and $P(u|v)$ are marginal distributions for u and v and the conditional distribution of u given v, respectively; $\text{SIM}(\boldsymbol{z}_v, \boldsymbol{z}_u)$ is the similarity measure between two latent vectors $\boldsymbol{z}_v, \boldsymbol{z}_u$ and N is the number of negative samples. Minimizing loss L equals to *maximizing the lower bound of the mutual information between $\boldsymbol{z}_v$ and $\boldsymbol{z}_u$*.

For $P(v)$ and $P(u)$, we choose the uniform distributions over all nodes in the graph. For $\text{SIM}(\boldsymbol{z}_v, \boldsymbol{z}_u)$, we follow the GraphSage paper and use $\text{SIM}(\boldsymbol{z}_v, \boldsymbol{z}_u) = \log(\sigma(\boldsymbol{z}_v, \boldsymbol{z}_u))$, where σ is the sigmoid function. The most important component is $P(u|v)$, as its reflects our prior knowledge about what should be a positive pair. In DeepWalk, Node2Vec and GraphSage, it is the distributions over the nodes that co-occurs in the fixed-length random walk starting from v. Therefore $P_{\text{random\_walk}}(u|v) = 1_v^T P_{\mathcal{G}}^k$, where 1_v is the one-hot vector over all nodes with node v equal to 1; $P_{\mathcal{G}}$ is the transition probability matrix, which is dependent on the topology of the graph $\mathcal{G}$, and k is the length of the random walk.

This $P(u|v)$ reflects the prior knowledge that the two near nodes should have similar embeddings, which is common in all general graph embeddings. However, we have more information besides that. The similarity of two nodes are positively related to the number of common ances-

tors. Moreover, it is related to the fraction of common ancestors over the union of their ancestors. Therefore, our $P(u|v)$ should reflect that. To do so, we assume a Boltzmann distribution over nodes as

$$P(u|v) \propto \exp(\alpha S_v(u)), \tag{6.2}$$

where $S_v(u)$ is the fraction of of common ancestors over the union of ancestors between node v and u, and α is the hyperparameter controlling the balance between the exploration and explotation. Large α means more local sampling while small α means more diverse sampling. After tuning on the validation set, we set $\alpha = 15$.

$$S_v(u) = \frac{|ancestor(u) \cap ancestor(v)|}{|ancestor(u) \cup ancestor(v)|}, \tag{6.3}$$

where $ancestor(u)$ is the set of all ancestors of node u.

During training, we sample 200 samples from $P_v(u)$ and $Q(u)$ respectively for every node v in the graph.

6.3.3 Transformer-based Generative Model

For a training seq $\boldsymbol{s}$, we denote its molecular function set as $\{v_s^1, v_s^2, ..., v_s^n\}$. Then its learned function embedding set is $\{z(v_s^1), z(v_s^2), ..., z(v_s^n)\}$. These n vectors will be concatenated together to form a 2D tensor and will be used as the input for the generative model. It is noteworthy that the transformer model is permutation invariant to the input sequence, which means that any order of input functions will output the same results. The architecture of our generative model is shown in Fig. 6.2. The main component is a transformer decoder, which includes 4 decoder layers. Each decoder layer starts with a self-attention module, followed by a cross-attention module and ended with a fully connected layer. The query and key inputs to the cross attention will be the functional embeddings.

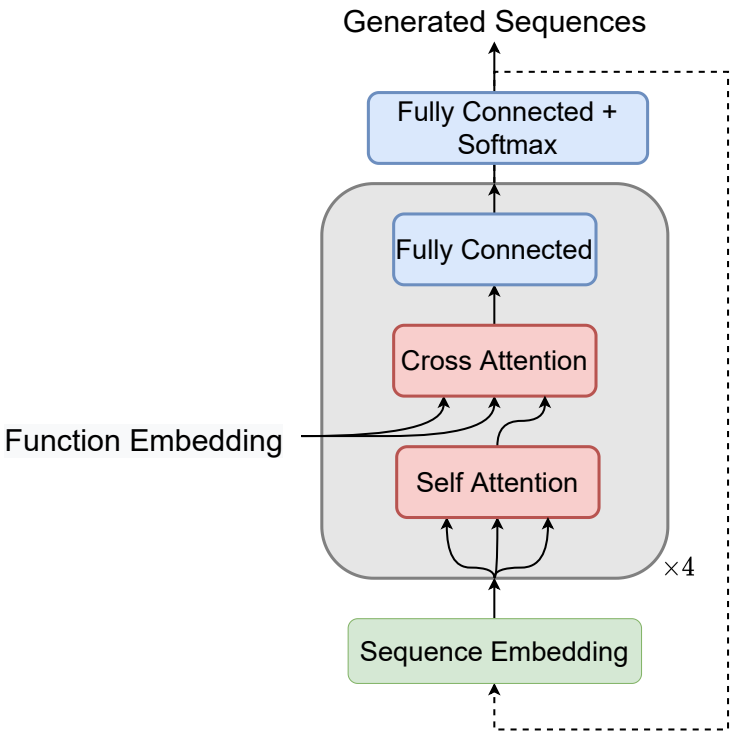


Figure 6.2: The architecture of the conditional autoregressive model.

6.3.4 Model Training and Hyperparameter Tuning

We train our model in two stages: meta-pretraining and fine tuning. In both stages, We use Adam optimizer with the learning rate as a hyperparameter. We also consider the weight decay as the regularization.

6.4 Results

We perform comprehensive experiments to analyze the performances of our model as well as competitors. Overall, we summarize our results into three parts:

- First, we justify the advantages of our learned function embeddings against others three two tasks:
  - We use learned embeddings to predict protein-protein interactions.
  - We assess the learned embeddings using metrics in language modelling.

- Second, we assess model's performances in mutational effect prediction:
  - We assess the correlation between the predicted effect against the fitness score of each mutation.
  - We assess the ability of each model for classifying disease and benign mutations through the disease-associated mutation dataset.
- Last, we choose one family for which we design functional protein sequences. We assess the generated sequences in-depth and compare them against the natural ones through sequence identity, sequence profile, sequence motif and tSNE visualization of the sequence space.

6.4.1 Predicting the Protein-Protein Interactions through GO embeddings

Before going into the generative model, we first assess whether our learned functional embeddings can actually capture the semantics of each GO node. One way of the assessment is to use embeddings for link prediction in the protein–protein interaction (PPI) networks, as neighbored proteins in the PPI network usually have similar functions. We follow the pipeline in GO2Vec [136] and obtain the PPI data from StringDB [102] for two organisms: Yeast (Saccharomyces cerevisiae) and Human (Homo sapiens). For each protein, we obtain their MFO annotation through Swiss-Prot with 8 CAFA evidence codes. We filter out those proteins that do not have any MFO annotations. As a result, we have 5,470 proteins and 5,596,512 interactions for Human and 2,156 proteins and 1,312,896 interactions for Yeast. We also randomly sample the same amount of non-connect protein pairs as the negative set. For similarity calculation, we also use the modified Hausdorff distance [137] as in the GO2Vec [136].

$$\begin{aligned}\text{Sim}(\boldsymbol{H}_a, \boldsymbol{H}_b) = \min(&\frac{1}{|\boldsymbol{H}_a|}\sum_{\boldsymbol{h}_a \in |\boldsymbol{H}_a|}\max_{\boldsymbol{h}_b \in \boldsymbol{H}_b}(cos(\boldsymbol{h}_a, \boldsymbol{h}_b)),\\ &\frac{1}{|\boldsymbol{H}_b|}\sum_{\boldsymbol{h}_b \in |\boldsymbol{H}_b|}\max_{\boldsymbol{h}_a \in \boldsymbol{H}_a}(cos(\boldsymbol{h}_a, \boldsymbol{h}_b)))\end{aligned} \tag{6.4}$$

where $\boldsymbol{H}_a = \{\boldsymbol{h}_{v1}, \boldsymbol{h}_{v2}, \boldsymbol{h}_{v3}...\}$ is the set of latent vectors for node $v1, v2, ..$, which are the GO annotations for protein a, and the same for protein b.

We report 5 different metrics for assessing the performances of link prediction: Area under the Receiver Operating Characteristic curve (AuROC), Area under the Precision and Recall Curve (AuPRC), F1-score, Precision and Recall, where F1-score, precision and recall are obtained when F1-score is the largest under varying thresholds.

We compare our BioBERT+TSGCN with 5 competitors: node2vec [138], DeepWalk [139], BioBERT [130], BioBERT+GAT and BioBERT+GCN. The performances are shown in Table 6.1. Overall, BioBERT+TSGCN has shown the best performances in 4 out of the 5 metrics, in which BioBERT+GCN is the second best performer. Specifically, BioBERT+TSGCN improved the AuROC by 0.017 and the AuPRC by 0.014 against the BioBERT+GCN in the Human test set, and AuROC by 0.024 and the AuPRC by 0.017 against the BioBERT+GCN in the Yeast test set. It is noteworthy that node2vec, DeepWalk only utilize the graph topology information, while BioBERT only uses the text information. This demonstrates the power of combining the text description with graph neural networks for embedding ontology nodes.

	Human					Yeast				
	AuROC	AuPRC	F1-score	Precision	Recall	AuROC	AuPRC	F1-score	Precision	Recall
node2vec	0.776	0.800	0.716	0.627	0.835	0.750	0.790	0.690	0.597	0.818
DeepWalk	0.771	0.800	0.710	**0.629**	0.815	0.738	0.785	0.682	0.572	0.843
BioBERT	0.740	0.760	0.698	0.600	0.833	0.737	0.762	0.694	0.589	0.843
BioBERT+GAT	0.778	0.798	0.720	0.625	0.849	0.750	0.788	0.692	**0.617**	0.786
BioBERT+GCN	0.800	0.822	0.734	0.624	0.889	0.777	0.811	0.711	0.610	0.851
BioBERT+TSGCN	**0.817**	**0.836**	**0.745**	0.616	**0.943**	**0.801**	**0.828**	**0.730**	0.577	**0.995**

Table 6.1: The performance of link prediction task in PPI networks.

6.4.2 Assessing the Generators as Language Models

In this section, we assess our learned functional embedding through its generalizability for generative models. Specifically, we split all Pfam families into 3 sets: a training set, a cross-domain (CD) set and a out-of-domain (OD) set. The CD set shares the individual GO terms with the training set, while the OD set has completely 'novel' GO terms against the training set. We further split 1% sequences from every family in the training set to form a in-domain (ID) test set.

The generalizability difficulty increases from ID test set to OD test.

Inspired from the language model assessment, we assess the perplexity per residue for each embedding method over 3 test sets. The performances are shown in Table 6.2. Overall, all methods perform much better in the ID test set than in the CD and OD test set. It demonstrates the difficulty of designing sequences for the new functions. Moreover, BioBERT+TSGCN performed on par with other methods on all three sets. In order to remove the effect of large family dominating the performances, we calculate the averaged perplexity per family. The results are shown in Table 6.3, from which we could obtain similar conclusions.

	ID Test Set	CD Test Set	OD Test Set
Random	21.00	21.00	21.00
node2vec	9.67	19.11	18.49
DeepWalk	9.61	19.13	18.52
BioBERT	9.60	19.41	18.86
BioBERT+GAT	9.98	19.05	18.70
BioBERT+GCN	10.09	19.33	18.80
BioBERT+TSGCN	10.04	19.20	18.56

Table 6.2: The perplexity of different methods for ID, CD and OD test sets, respectively.

	ID Test Set	CD Test Set	OD Test Set
Random	21.00 ± 0.00	21.00 ± 0.00	21.00 ± 0.00
node2vec	13.50 ± 3.98	20.32 ± 4.24	18.88 ± 1.67
DeepWalk	13.31 ± 3.99	20.15 ± 3.99	18.94 ± 1.48
BioBERT	13.32 ± 3.98	20.43 ± 4.40	19.19 ± 1.51
BioBERT+GAT	14.12 ± 3.98	20.16 ± 5.13	18.92 ± 1.39
BioBERT+GCN	14.24 ± 3.95	19.91 ± 3.95	19.06 ± 1.53
BioBERT+TSGCN	14.29 ± 3.99	20.05 ± 5.04	18.97 ± 1.45

Table 6.3: The averaged perplexity over families of different methods for ID, CD and OD test sets, respectively.

We would like to further assess the performances of the subsets of the CD or OD test sets which

share or does not share the clans with the training sets. Specifically, we split the CD(OD) test set into the in-clan CD(OD) test set and the out-clan CD(OD) test set, where families in the in-clan test set share the clans with at least one family in the training set, while families in the out-clan test set does not share clans with all families in the training set. As a result, we have 40 and 39 families for the CD in-clan and out-clan test sets, respectively, 81 and 120 families for the OD in-clan and out-clan test sets, respectively. The results are shown in Table 6.4 for overall perplexity and in Table 6.5 for family-averaged perplexity. Overall, all methods show slightly better performances in the in-clan CD test set than those in the out-clan CD test set. BioBERT+TSGCN has shown the best performance with least standard deviation in the in-clan test set on family-averaged perplexity. For OD test set, all methods show slightly better performances in the out-clan test set than those in the in-clan test set. It may be due to that the generalizability is so difficult for OD test set so that the in-clan and out-clan separation does not show too much difference.

	In-clan CD Test	Out-clan CD Test	In-clan OD Test	Out-clan OD Test
Random	21.00	21.00	21.00	21.00
node2vec	19.04	19.94	**18.83**	**18.22**
DeepWalk	**18.98**	20.19	**18.83**	18.31
BioBERT	19.39	20.23	19.04	18.76
BioBERT+GAT	**18.98**	19.86	19.15	18.34
BioBERT+GCN	19.25	**19.61**	19.12	18.57
BioBERT+TSGCN	19.03	19.87	18.86	18.40

Table 6.4: The perplexity of different methods for the in-clan, out-clan subsets of CD and OD test sets, respectively. (Boldfaced numbers are the best performance in each test set.)

6.4.3 Predict the Fitness-correlated Effect of Protein Variants

From this subsection, we will go into the second part of our results: predicting the effect of protein variants. In this section, we will assess the correlation between the predicted effect against the fitness score of each mutation using Shin *et al*'s [1] dataset.

Specifically, we followed the same pipeline as in Shin *et al*'s [1] for model fine-tuning and

	In-clan CD Test	Out-clan CD Test	In-clan OD Test	Out-clan OD Test
Random	21.00 ± 0.00	21.00 ± 0.00	21.00 ± 0.00	21.00 ± 0.00
node2vec	20.07 ± 3.58	20.57 ± 4.81	19.14 ± 1.79	$\mathbf{18.71 \pm 1.56}$
DeepWalk	19.61 ± 2.31	20.70 ± 5.12	19.13 ± 1.22	18.82 ± 1.62
BioBERT	19.80 ± 2.38	21.08 ± 5.70	19.26 ± 1.30	19.14 ± 1.63
BioBERT+GAT	19.53 ± 2.04	20.80 ± 6.95	19.16 ± 1.11	18.75 ± 1.53
BioBERT+GCN	19.54 ± 1.79	$\mathbf{20.28 \pm 5.29}$	19.24 ± 1.40	18.95 ± 1.60
BioBERT+TSGCN	$\mathbf{19.40 \pm 1.45}$	20.73 ± 6.95	$\mathbf{19.11 \pm 1.24}$	18.88 ± 1.57

Table 6.5: The averaged perplexity over families of different methods for the in-clan, out-clan subsets of CD and OD test sets, respectively. (Boldfaced numbers are the best performance in each test set.)

model ensemble. We calculate the Spearman correlation between our predicted effect against the ground-truth fitness score. We compare with Shin *et al* [1], an alignment-free method, and 4 alignment-based methods: DeepSequence [140], EVmutation [141], Independent and Hidden Markov Model [142] (HMM). The results of those 5 competitors are extracted from the Figure 2(a) in Shin *et al.* [1] through WebPlotDigitizer [143].

It can be seen from Figure 6.3, that our model outperformed Shin *et al* in 30 out of 35 proteins and even outperformed the best alignment-based method, DeepSequence, in 25 out of 35 proteins. On average, our model improves the Spearman correlation by 0.041 against Shin *et al*, and 0.053 against DeepSequence. To further justify the significance of those values, we perform one-sided t-test with null hypothesis: "The mean of our Spearman correlation is **not** larger than Shin *et al*'s or DeepSequence's", resulting in p-value = $6.2\text{E}-6$ and 0.0029, respectively, which demonstrates the advantages of our model comparing to both alignment-based and alignment-free methods.

Besides the overall performances, in order to figure out how much did pretraining, fine-tuning and ensemble contribute to the performance improvement, we also perform the following ablation study: we compare our final model against 3 ablated models: ours with only pretraining, ours with only fine-tuning (cold start), and ours with pretraining plus fine-tuning (without ensemble). In addition, we are interested in comparing our model without ensemble against Shin *et al* without ensemble.

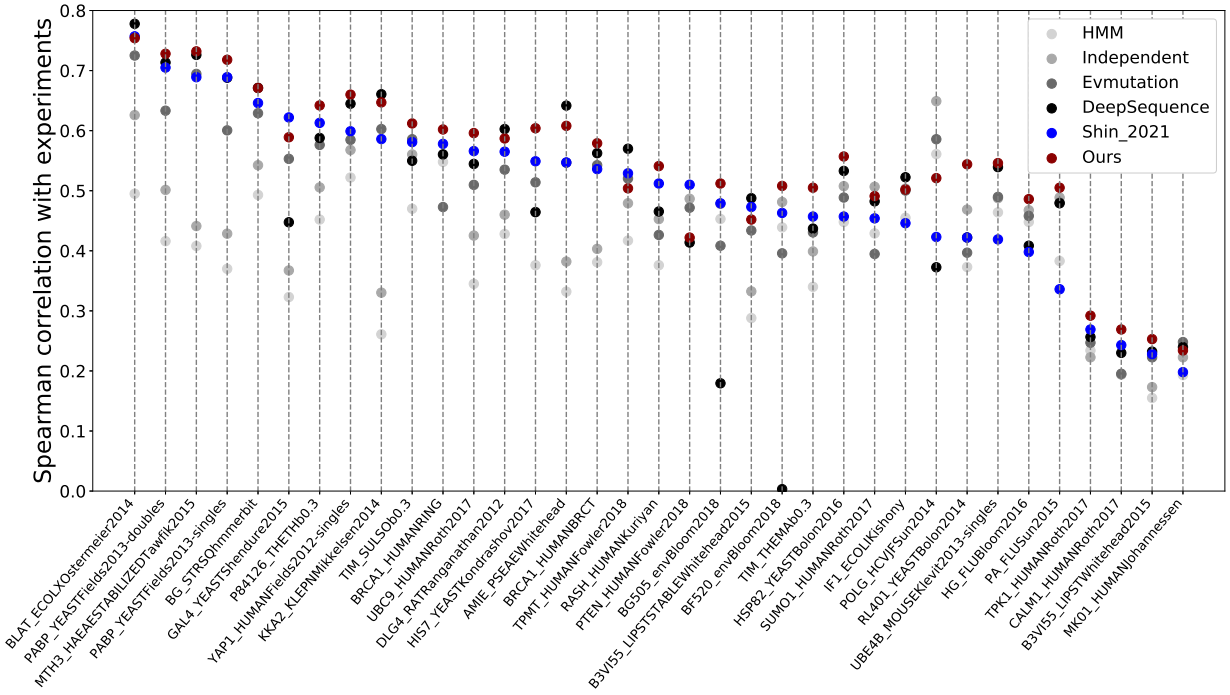


Figure 6.3: Performances of Mutational Effect Prediction in Shin *et al*'s [1] dataset.

The results are shown in Figure 6.4. Overall, Ours: Pretrain+Finetune(warm start) outperformed Shin2021 Finetune(cold start) in 30 out of 35 cases, with average Spearman correlation improved by 0.040. We perform the same aforementioned t-test, revealing a p-value of $4.2\text{E}-5$. This demonstrates our model without ensemble can still outperform Shin2021*et al* without ensemble.

Moreover, comparing Ours: Pretrain+Finetune (warm start) vs Ours: Finetune (cold start), the former one outperformed the latter one in 33 out of 35 cases, with average 0.044 improved Spearman correlation and p-value = $4.2\text{E}-8$. This demonstrates the advantages of pre-training.

Lastly, comparing Ours: Pretrain+Finetune(warm start) vs Ours: Pretrain, the former one outperformed the latter one in all 35 cases, with average Spearman correlation improved by 0.238 with p-value = $3.3\text{E}-13$.

In conclusion, both pretraining and fine-tuning contributes non-trivially to the performances improvement, with fine-tuning contributing more.

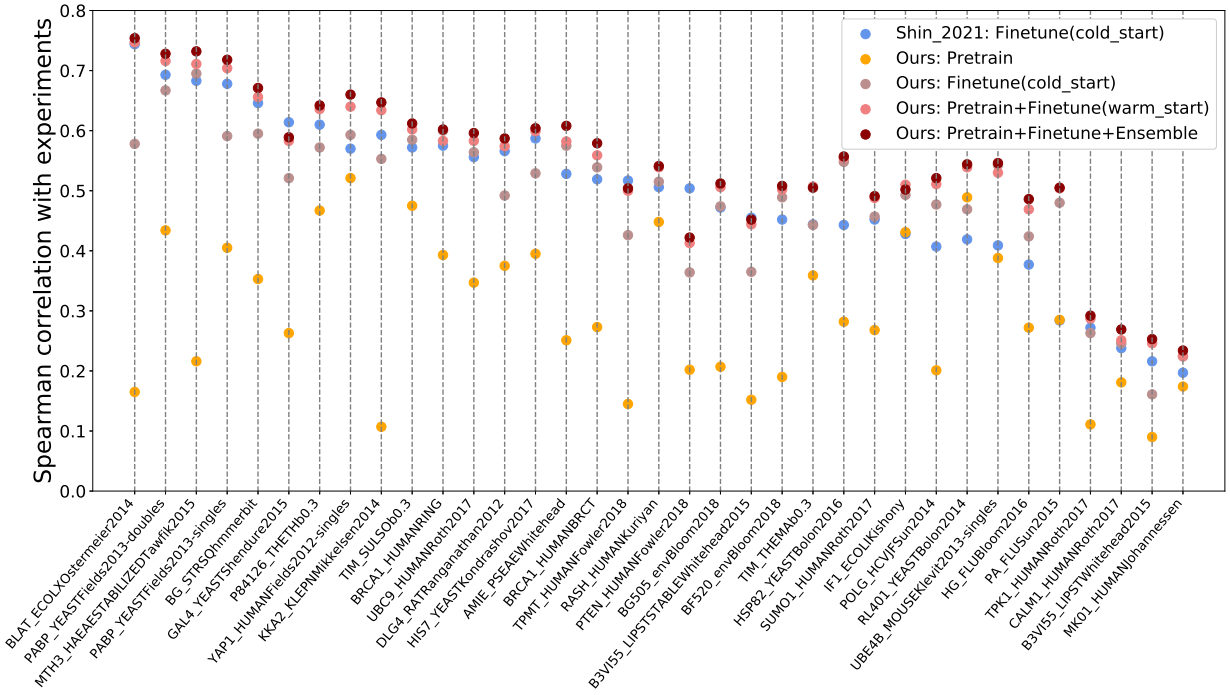


Figure 6.4: Ablation Study of Mutational Effect Prediction.

We also did the extra study of comparing Ours:pretrain against HMM. We found Ours:pretrain only outperforms HMM in 9 out of 35 cases. Moreover, Ours:pretrain worsens the Spearman correlation by 0.09 against HMM, with p-value = 0.99998, which indicates almost certainly that family-specific HMM performs better than the meta-model Ours:pretrain.

We further investigate the potential factors that impact the performance improvement from either Shin2021 or DeepSequence to our model. We consider three factors: the size of the family, the diversity of the family and the length of the test sequence. Specifically, we use the number of the training sequences as the size of the family, the number of the clusters after clustering all the training sequences as the diversity of the family. For each factor, we calculate the differences of Spearman correlations between ours and Shin2021 or DeepSequence respectively for all 35 test cases.

The results are shown in Fig. 6.5. Overall, the diversity (number of clusters) has most impact for the performance improvement of ours against Shin2021 (Pearson correlation=0.463), while the

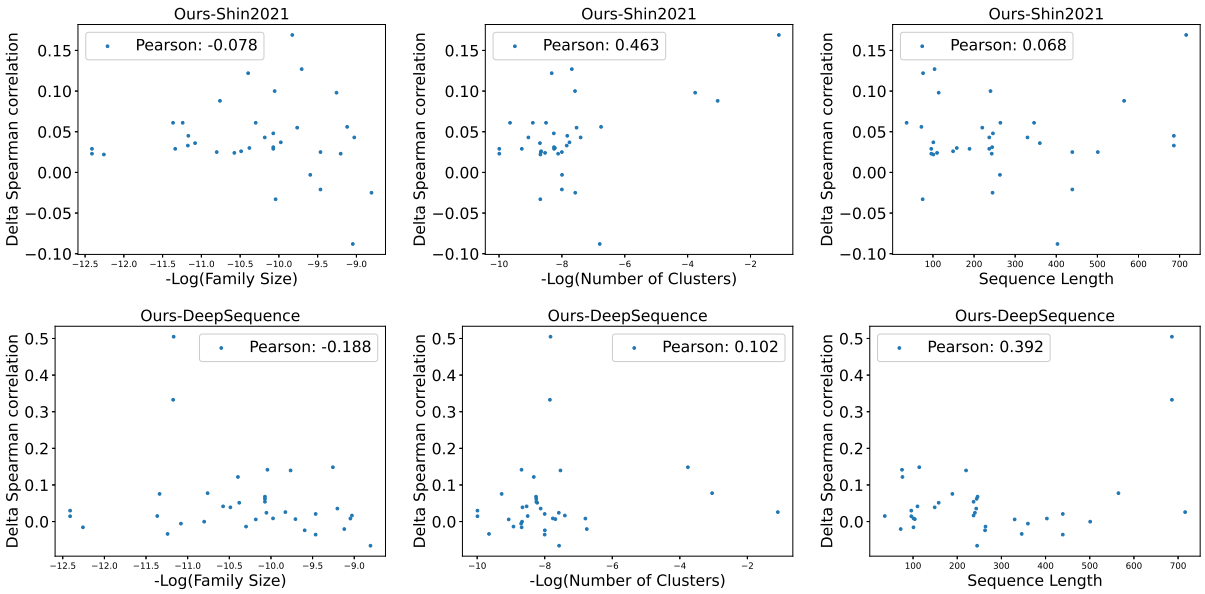


Figure 6.5: The delta Spearman correlation between ours and Shin2021 or DeepSequence of 35 test proteins against three factors.

sequence length has most impact for the performance improvement of ours against DeepSequence (Pearson correlation=0.392). As the data-driven, alignment-free method, Shin 2021 heavily relies on diversity of the training data. Although it utilizes the weight-balanced training, for low-diversity cases, our model significantly outperform it, which shows the unique advantage of our pretraining+finetuning training scheme. Moreover, DeepSequence is an alignment-based method which relies on the multiple sequence alignment (MSA). As MSA favours short, rich sequences, it is reasonable that our model has the biggest advantages in the region while the sequence length is long and the family size is small, which echoes the pattern shown in Fig 6.5.

6.4.4 Predict the Disease-associated Effect of Protein Variants

In this section, we will examine our models in disease-associated effect prediction of protein variants. Predicting such genotype–phenotype associations can provide enormous benefits for biomedical and pharmaceutical applications. Specifically, we use a disease-associated mutation effect dataset at Pfam level which was kindly provided by Savojardo *et al* [144]. The dataset contains in total 22,763 disease-related variations distributed in 1670 Pfam families. Each variation

has a binary ground-truth label which could be either "Disease" or "Benign".

We focus on predicting the binary ground-truth labels of all variations with in each Pfam family through totally unsupervised learning. We use Pfam UniprotKB dataset as the training set. In this study, we choose several Pfam families that are representative for different purposes:

- "Small family": Those families have small number of training sequences in Pfam UniprotKB. Therefore, training them only through fine-tuning (cold start) may not lead to good performances. We choose 5 small families based on the following criteria: We first rank all 1670 Pfam families based on the increasing size of their training sets. Then we choose the first 5 families which have 1) available GO annotations and 2) at least 20 variants in the test set.

- "Top disease-associated family": Families that are mostly associated with diseases: We choose the top 5 families that are mostly associated with diseases in Table 2 in Savojardo *et al*. [144].

- "Disease-Benign balanced family": Families have roughly same amount of disease and benign variants. We also choose 5 families in these case because we would like to investigate the model performances on balanced test sets.

The statistics of all 15 families are shown in Table 6.6. It can be seen that the number of training sequences for 5 Pfam families in the "Small familiy" set is much smaller than those in the other two sets. Moreover, we also report the ratio of disease variations in the test set. It can be seen that such ratio is very high in "Top disease-associated family", around 0.5 in "Disease-Benign balanced family" and random in "Small family".

For the final score used for classification, we choose the absolute value of the difference of the the log probability per residue between the mutant sequence and the wild type sequence:

$$\text{score} = |\frac{\log P(\boldsymbol{s}_{\text{mutant}})}{n_{\text{mutant}}} - \frac{\log P(\boldsymbol{s}_{\text{wildtype}})}{n_{\text{wildtype}}}| \tag{6.5}$$

where n_{mutant} and n_{wildtype} are the length of mutant and wild type sequences, respectively. In this study, due to all the variations are missense mutations, we have $n_{\text{mutant}} = n_{\text{wildtype}}$.

The reason why we use absolute value here is because for mutants that cause diseases would result in larger absolute differences between their functions and the wildetype's functions, no matter which genes they are on (disease-associated mutations on oncogenes or tumor suppressor genes will cause the gain of the functions or loss of the functions, respectively).

	Pfam	#Train seqs	#Test mutations	Disease ratio
Small family	PF02101	596	40	0.850
	PF00184	952	37	0.919
	PF02060	1128	20	0.650
	PF00377	1298	20	0.700
	PF09773	1768	32	0.781
Top disease-associated family	PF00105	46357	62	0.968
	PF00250	30349	91	0.967
	PF00010	112980	50	0.960
	PF00104	45655	207	0.908
	PF00307	109291	53	0.906
Disease-Benign balanced family	PF00028	358710	151	0.497
	PF00092	130462	131	0.458
	PF07679	496724	252	0.464
	PF02932	41273	119	0.487
	PF00084	169167	106	0.566

Table 6.6: The statistics of disease-associated dataset.

We report the performances using AuPRC, which is shown in Table 6.7-6.9. It can be seen that, in the "Small Family" set, pretrain is very useful. In the "Top disease-associated family", Our:pretrain+finetune(warm start) has 3 best out of 5 cases. In "Disease-Benign balanced family", Our:pretrain+finetune(warm start) also has the 3 best out of 5 cases. Moreover, in the 'Disease-Benign balanced family', we can see the performance improves a lot from Random to the final model, which demonstrate the usefulness of our model.

	PF02101	PF00184	PF02060	PF00377	PF09773
Random	0.850	0.919	0.650	0.700	0.781
Shin 2021	**0.943**	0.979	**0.931**	0.871	0.960
Our:pretrain	0.889	**0.993**	0.847	0.739	0.849
Our:finetune(cold start)	0.857	0.983	0.783	0.936	0.960
Our:pretrain+finetune(warm start)	**0.943**	0.982	0.814	**0.941**	**0.973**

Table 6.7: Small family.

	PF00105	PF00250	PF00010	PF00104	PF00307
Random	**0.968**	0.967	0.960	0.908	0.906
Shin 2021	0.941	0.990	0.998	0.982	0.964
Our:pretrain	0.949	0.987	0.991	0.949	0.917
Our:finetune(cold start)	0.940	**0.997**	0.995	0.971	**0.982**
Our:pretrain+finetune(warm start)	**0.965**	0.994	**0.999**	**0.986**	0.974

Table 6.8: Top disease-associated family.

	PF00028	PF00092	PF07679	PF02932	PF00084
Random	0.497	0.458	0.464	0.487	0.566
Shin 2021	0.767	0.749	0.647	0.901	0.854
Our:pretrain	0.702	0.685	0.692	**0.908**	0.866
Our:finetune(cold start)	0.775	0.733	**0.709**	0.903	0.869
Our:pretrain+finetune(warm start)	**0.792**	**0.772**	0.660	0.896	**0.876**

Table 6.9: Disease-Benign balanced family

6.4.5 Designing Functional Proteins for a DNA binding domain

In this section, we will go into the last part of our results. We will generate sequences through our conditional generator and analyze the design ability of our model for a DNA binding domain: ARID domain (AT-rich interaction domain; also known as BRIGHT domain). Protein–DNA interactions play central roles in many cellular processes. ARID-encoding genes are involved in a variety of biological processes including embryonic development, cell lineage gene regulation and cell cycle control.

Starting from our pretrained model, we fine-tune it using the Pfam UniprotKB dataset of this

family. To assess the model's performance during training, we generated 64 sequences every 5000 iterations through ancestral sampling. We analyze the maximum sequence identity between each generated sequences to the training set. The performance is shown in Fig 6.6 (Top). It can be seen that, the MSI reaches a plateau around 60%. We then use the model trained at 90,000 iterations and generate 1,000,000 sequences. We then use Diamond to filter out the generated sequences compared to the training set, which cut around 23% of the sequences. We randomly choose the same amount of the sequences as the training set for further analysis.

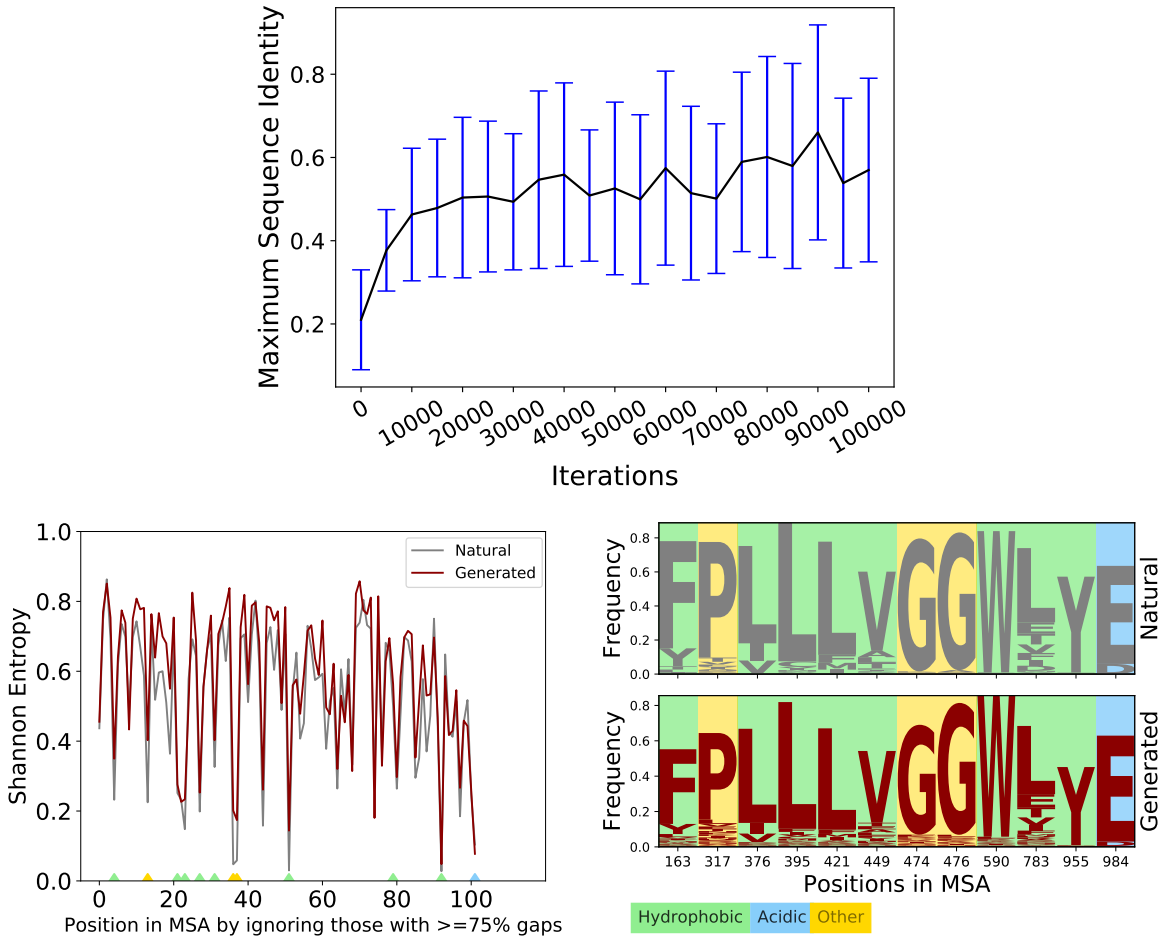


Figure 6.6: (Top) The mean(std) of the maximum sequence identity between generated sequences and the training sequences against the number of training iterations. (Left). The entropy of natural and generated sequences at each position. (Right). The frequency of amino acids at sequence motif positions for both natural and generated sequences.

We first examine whether our generated sequences can learn the intrinsic pattern within the natural sequences. To achieve that, we first did multiple sequence alignment by combing the natural and generated sequences and then calculate the Shannon entropy at each position for both natural and generated sequences, respectively. The results are shown in Fig 6.6 (Left). It can be seen that the two curves well overlapped with each other, especially for the low entropy part. Also we can see for the high entropy region, the generated curve is not overlapped well with the natural ones, showing the diversity of our generated sequences. We then plot the amino acid distributions at the motif positions for both natural and generated sequences. As shown in Figure 6.6 (Right), the distributions between natural and generated are almost identical at every motif position, which demonstrate that our model indeed learns the intrinsic relationship within the natural sequences.

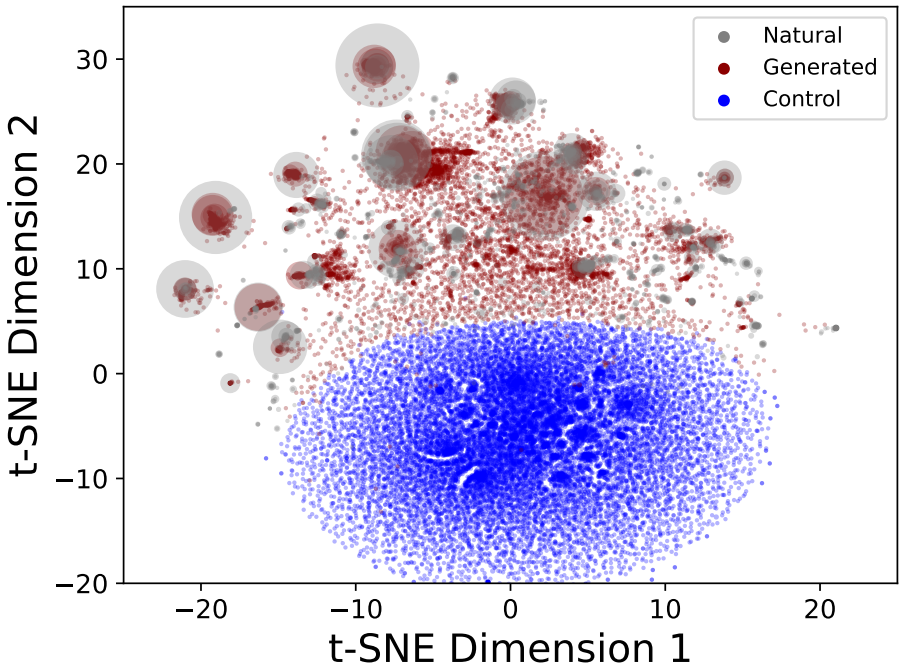


Figure 6.7: The tSNE plot of natural, generated and control sets.

Then, we investigate whether our generated sequences can expand the natural sequence space.

To set a control set, we also sample the same amount of sequences outside of the clan of the target family. Then for each set of sequences (Natural, Generated and Control), we did clustering through 70% cutoff through MMSeq2. We then calculate the pairwise distance among each cluster head through Clustal Omega. We plot the tSNE figures of these cluster heads as shown in Figure 6.7. It can be clearly seen that 1) our generated sequences are overlapped with the natural sequences, while the control sequences are almost exclusive with both natural and generated ones and 2) the generated sequences are obviously more diverse and cover more space compared to the natural sequences. Therefore, we can claim that our generated sequences could expand the functional space of natural sequences while preserving the biological constraints within such family.

6.5 Conclusion

In this chapter, we investigate the inverse function-to-sequence relationship by studying designing functional protein sequences and predicting the effect of sequence variants. To achieve this, we first propose a novel graph embedding method for embedding the Gene Ontology graph using both node description information and graph topology. We then propose a conditional autoregressive model for broadly learning the relationship between protein functions and protein sequences. We first assess our model in mutational effect prediction. By fine-tuning on the dataset within each protein family, our model can beat the state-of-the-art alignment-free method on 35 out of 40 proteins. We then assess the generated sequences of our fine-tuned model on a specific DNA binding domain. We found our model can not only generate natural-like, functional-relevant sequences, but also generate more diverse sequences than the natural ones, so that, our model shows the potential of expanding the existing sequence space for a given functions within the the allowed biological constraints of such functions.

7. Conclusion and Future Work

In this thesis, we mainly focus on answering this question: *To what extent can machine learning uncover the protein sequence-structure-function relationships?* To answer the question, we focus on the optimization and scoring in protein docking (Chapter 2-4) and modelling the protein sequence–function relationships (Chapter 5-6). We also investigate the extension of our algorithms into meta-learning for solving a broader range of optimization problems (Chapter 3).

As a result, we propose Bayesian Active Learning (BAL) for optimization and uncertainty quantification in protein docking, Energy-based Graph Convolutional Networks (EGCN) for scoring protein docking models, Transformer-based protein function Annotation with joint Sequence-Label embedding (TALE) for predicting protein functions and conditional autoregressive generative model with graph ontology embedding for functional protein design and mutational effect prediction. We also generalize BAL into meta-learning and propose LOIS, a meta-learner for automatically solving general optimization problems. Our results show that those models significantly outperform the competing models or even the state-of-the art methods. They demonstrate that machine learning can actually be a powerful tool for uncovering the protein mechanisms to some extent, and such a tool could be generalised to other fields.

However, our results also show that there is still a decent space for future improvement. Looking ahead, we hereby list several potential directions for improvement :

- The parameterization of protein docking search space is highly approximate, which is obtained by modelling the internal energy as the Hooke's potential. Such parameterization may not represent the real rigid-body motion and the conformational change when two proteins encounter with each other. Instead, we can learn the representation of protein motions during its interaction with other proteins, through a large corpus of MD or NMA simulations. Such learned representation could better capture the direction and extension of protein position and conformational changes.

- Inspired from AlphaFold2 [128], another promising direction for protein docking is through end-to-end training. One could integrate parameterization, scoring and optimization together in one model, with protein sequences or homology structures as input and the predicted complex structures as the output. Similarly, one can use Multiple Sequence Alignment (MSA) and contact prediction as additional data to combine them with attention mechanisms.

- Both BAL and LOIS consider the uncertainty within the data and the models while doing optimization, However, no attention was ever paid to the uncertainty arising from the optimizer that is directly responsible for deriving the end solutions with given data and models. The optimizer is usually pre-defined and fixed in the optimization algorithm space. The uncertainty in the optimizer is intrinsically defined over the optimizer space and important to the optimization and UQ solutions. However, such uncertainty is unwittingly ignored when the optimizer is treated as a fixed sample in the space. Therefore, a potential direction is to quantify this new form of uncertainty, that lies in the optimization algorithm (optimizer), besides the classical data- or model- based uncertainties (also known as epistemic and aleatoric uncertainties).

- Utilizing information other than sequences could be another direction for the future work of protein function prediction. Many proteins have available information like structures and networks that could significantly boost the performance. To integrate those information with sequences through joint embedding learning is a promising direction for protein function prediction.

- Although our Func2Seq model is demonstrated to have the ability to design functional sequences, such functions are still seen in our training set or in the nature. An ambitious goal is to design protein sequences with novel functions that have never been seen in the nature, which could result in huge impact for biomedical and pharmaceutical applications.

In a broader view, the fields of artificial intelligence and biological science have often developed their own specific methods and approaches historically. But in the past few decades, they

have witnessed significantly growing interests in applying artificial intelligence technology to biological problems. Whether AI could reshape the whole biological world, like it did for images, languages and games, is still a question. Our BAL, EGCN and TALE, like the recent breakthrough from DeepMind for protein structure prediction are advancing to give a positive answer.

On one hand, we believe that biology will continue being impacted by AI. Some grand biological problems will be eventually solved with the help of AI. On the other hand, we also believe that AI will benefit a lot from biology. Biology provides a huge amount of data with diverse types, representations and challenges. Those data will motivate the invention for new AI technology, and our LOIS is one of the examples. In the future, we believe there will be a trend of developing novel AI technology specially for biology and such trend will eventually lead AI to a new era.

REFERENCES

[1] 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*, vol. 12, no. 1, pp. 1–11, 2021.

[2] N. Zhou, Y. Jiang, T. R. Bergquist, A. J. Lee, B. Z. Kacsoh, A. W. Crocker, K. A. Lewis, G. Georghiou, H. N. Nguyen, M. N. Hamid, *et al.*, "The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens," *Genome biology*, vol. 20, no. 1, pp. 1–23, 2019.

[3] B. Alberts, D. Bray, K. Hopkin, A. D. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Essential cell biology*. Garland Science, 2013.

[4] C. B. Anfinsen, M. Sela, and J. P. Cooke, "The reversible reduction of disulfide bonds in polyalanyl ribonuclease," *Journal of Biological Chemistry*, vol. 237, no. 6, pp. 1825–1831, 1962.

[5] U. Consortium, "UniProt: a worldwide hub of protein knowledge," *Nucleic acids research*, vol. 47, no. D1, pp. D506–D515, 2019.

[6] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic acids research*, vol. 28, no. 1, pp. 235–242, 2000.

[7] A. Fiser, "Template-based protein structure modeling," in *Computational biology*, pp. 73–94, Springer, 2010.

[8] M. Källberg, H. Wang, S. Wang, J. Peng, Z. Wang, H. Lu, and J. Xu, "Template-based protein structure modeling using the raptorx web server," *Nature protocols*, vol. 7, no. 8, pp. 1511–1522, 2012.

[9] D. Petrey, T. S. Chen, L. Deng, J. I. Garzon, H. Hwang, G. Lasso, H. Lee, A. Silkov, and B. Honig, "Template-based prediction of protein function," *Current opinion in structural biology*, vol. 32, pp. 33–38, 2015.

[10] B. Buchfink, C. Xie, and D. H. Huson, "Fast and sensitive protein alignment using diamond," *Nature methods*, vol. 12, no. 1, pp. 59–60, 2015.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[12] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[14] M. Karimi, D. Wu, Z. Wang, and Y. Shen, "DeepAffinity: interpretable deep learning of compound–protein affinity through unified recurrent and convolutional neural networks," *Bioinformatics*, vol. 35, no. 18, pp. 3329–3338, 2019.

[15] M. Kulmanov, M. A. Khan, and R. Hoehndorf, "DeepGO: predicting protein functions from sequence and interactions using a deep ontology-aware classifier," *Bioinformatics*, vol. 34, no. 4, pp. 660–668, 2018.

[16] S. Wang, S. Sun, Z. Li, R. Zhang, and J. Xu, "Accurate de novo prediction of protein contact map by ultra-deep learning model," *PLoS computational biology*, vol. 13, no. 1, p. e1005324, 2017.

[17] M. Kulmanov and R. Hoehndorf, "DeepGOPlus: improved protein function prediction from sequence," *Bioinformatics*, vol. 36, no. 2, pp. 422–429, 2020.

[18] R. Rao, N. Bhattacharya, N. Thomas, Y. Duan, P. Chen, J. Canny, P. Abbeel, and Y. Song, "Evaluating protein transfer learning with tape," in *Advances in Neural Information Processing Systems*, pp. 9689–9701, 2019.

[19] Y. Cao and Y. Shen, "Bayesian active learning for optimization and uncertainty quantification in protein docking," *Journal of chemical theory and computation*, vol. 16, no. 8, pp. 5334–5347, 2020.

[20] X. Liu, R. Taftaf, M. Kawaguchi, Y.-F. Chang, W. Chen, D. Entenberg, Y. Zhang, L. Gerratana, S. Huang, D. B. Patel, *et al.*, "Homophilic CD44 interactions mediate tumor cell aggregation and polyclonal metastasis in patient-derived breast cancer models," *Cancer discovery*, vol. 9, no. 1, pp. 96–113, 2019.

[21] R. Taftaf, X. Liu, S. Singh, Y. Jia, N. K. Dashzeveg, A. D. Hoffmann, L. El-Shennawy, E. K. Ramos, V. Adorno-Cruz, E. J. Schuster, *et al.*, "Icam1 initiates ctc cluster formation and trans-endothelial migration in lung metastasis of breast cancer," *Nature communications*, vol. 12, no. 1, pp. 1–15, 2021.

[22] M. Kawaguchi, N. Dashzeveg, Y. Cao, Y. Jia, X. Liu, Y. Shen, and H. Liu, "Extracellular domains i and ii of cell-surface glycoprotein CD44 mediate its trans-homophilic dimerization and tumor cluster aggregation," *Journal of Biological Chemistry*, vol. 295, no. 9, pp. 2640–2649, 2020.

[23] Y. Cao, T. Chen, Z. Wang, and Y. Shen, "Learning to optimize in swarms," in *Advances in Neural Information Processing Systems*, pp. 15044–15054, 2019.

[24] Y. Cao and Y. Shen, "Energy-based graph convolutional networks for scoring protein docking models," *Proteins: Structure, Function, and Bioinformatics*, 2020.

[25] Y. Cao and Y. Shen, "TALE: Transformer-based protein function annotation with joint sequence-label embedding," *bioRxiv*, 2020.

[26] R. Mosca, A. Céol, and P. Aloy, "Interactome3d: adding structural details to protein networks," *Nat. Methods*, vol. 10, no. 1, p. 47, 2013.

[27] G. R. Smith and M. J. Sternberg, "Prediction of protein–protein interactions by docking methods," *Curr. Opin. Struct. Biol.*, vol. 12, no. 1, pp. 28–35, 2002.

[28] J. J. Gray, S. Moughon, C. Wang, O. Schueler-Furman, B. Kuhlman, C. A. Rohl, and D. Baker, "Protein–protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations," *J. Mol. Biol.*, vol. 331, no. 1, pp. 281–299, 2003.

[29] I. H. Moal and P. A. Bates, "SwarmDock and the Use of Normal Modes in Protein-Protein Docking," *Int. J. Mol. Sci.*, vol. 11, pp. 3623–3648, Sept. 2010.

[30] Y. Shen, "Improved flexible refinement of protein docking in capri rounds 22–27," *Proteins: Struct., Funct., Bioinf.*, vol. 81, no. 12, pp. 2129–2136, 2013.

[31] B. Jiménez-García, J. Roel-Touris, M. Romero-Durana, M. Vidal, D. Jiménez-González, and J. Fernández-Recio, "Lightdock: a new multi-scale approach to protein–protein docking," *Bioinformatics*, vol. 34, no. 1, pp. 49–55, 2017.

[32] N. A. Marze, S. S. Roy Burman, W. Sheffler, and J. J. Gray, "Efficient flexible backbone protein–protein docking for challenging targets," *Bioinformatics*, vol. 34, no. 20, pp. 3461–3469, 2018.

[33] E. Pfeiffenberger and P. A. Bates, "Refinement of protein-protein complexes in contact map space with metadynamics simulations," *Proteins: Struct., Funct., Bioinf.*, vol. 87, no. 1, pp. 12–22, 2019.

[34] L. S. P. Rudden and M. T. Degiacomi, "Protein Docking Using a Single Representation for Protein Surface, Electrostatics, and Local Dynamics," *J Chem Theory Comput*, vol. 15, pp. 5135–5143, Sep 2019.

[35] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. d. Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, vol. 104, pp. 148–175, Jan. 2016.

[36] P. Ortega, J. Grau-Moya, T. Genewein, D. Balduzzi, and D. Braun, "A nonparametric conjugate prior distribution for the maximizing argument of a noisy function," in *Advances in Neural Information Processing Systems*, pp. 3005–3013, Lake Tahoe, 2012.

[37] Y. Shen, I. C. Paschalidis, P. Vakili, and S. Vajda, "Protein docking by the underestimation of free energy funnels in the space of encounter complexes," *PLoS Comput. Biol.*, vol. 4, no. 10, p. e1000191, 2008.

[38] T. Oliwa and Y. Shen, "cNMA: a framework of encounter complex-based normal mode analysis to model conformational changes in protein interactions," *Bioinformatics*, vol. 31, pp. i151–i160, June 2015.

[39] L. Ingber, "Adaptive simulated annealing (ASA): lessons learned," *CoRR*, vol. cs.MS/0001018, 2000.

[40] J.-P. Chilès and P. Delfiner, *Geostatistics: Modeling Spatial Uncertainty, 2nd Edition*. 2012.

[41] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, Z. Wen, *et al.*, "A tutorial on thompson sampling," *Fou. and Tre. in Mach. Learn.*, vol. 11, no. 1, pp. 1–96, 2018.

[42] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," in *Conference on learning theory*, pp. 39–1, Edinburgh, 2012.

[43] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Advances in Neural Information Processing Systems.*, pp. 2249–2257, Granada, 2011.

[44] B. R. Brooks, C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus, "CHARMM: the biomolecular simulation program," *J. Comput. Chem.*, vol. 30, pp. 1545–1614, July 2009.

[45] L. Györfi, M. Kohler, A. Krzyzak, and H. Walk, *A Distribution-Free Theory of Nonparametric Regression*. Springer Series in Statistics, New York: Springer-Verlag, 2002.

[46] J. Villemonteix, E. Vazquez, and E. Walter, "An informational approach to the global optimization of expensive-to-evaluate functions," *Journal of Global Optimization*, vol. 44, no. 4, p. 509, 2009.

[47] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, "Predictive entropy search for efficient global optimization of black-box functions," in *Advances in Neural Information Processing Systems*, pp. 918–926, Montreal, 2014.

[48] H. Hwang, T. Vreven, J. Janin, and Z. Weng, "Protein-Protein Docking Benchmark Version 4.0," *Proteins*, vol. 78, pp. 3111–3114, Nov. 2010.

[49] Y. Shen, P. Vakili, S. Vajda, and I. C. Paschalidis, "Optimizing noisy funnel-like functions on the euclidean group with applications to protein docking," in *IEEE Conference on Decision and Control*, pp. 4545–4550, IEEE, New Orleans, 2007.

[50] H. Mirzaei, S. Zarbafian, E. Villar, S. Mottarella, D. Beglov, S. Vajda, I. C. Paschalidis, P. Vakili, and D. Kozakov, "Energy minimization on manifolds for docking flexible molecules," *J. Chem. Theory Comput.*, vol. 11, no. 3, pp. 1063–1076, 2015.

[51] H. Chen, Y. Sun, and Y. Shen, "Predicting protein conformational changes for unbound and homology docking: learning from intrinsic and induced flexibility," *Proteins: Struct., Funct., Bioinf.*, vol. 85, no. 3, pp. 544–556, 2017.

[52] D. Kozakov, K. H. Clodfelter, S. Vajda, and C. J. Camacho, "Optimal clustering for detecting near-native conformations in protein docking," *Biophys. J.*, vol. 89, no. 2, pp. 867–875, 2005.

[53] S. Vajda and D. Kozakov, "Convergence and combination of methods in protein–protein docking," *Curr. Opin. Struct. Biol.*, vol. 19, no. 2, pp. 164–170, 2009.

[54] R. Méndez, R. Leplae, M. F. Lensink, and S. J. Wodak, "Assessment of CAPRI predictions in rounds 3–5 shows progress in docking procedures," *Proteins: Struct., Funct., Bioinf.*, vol. 60, pp. 150–169, Aug. 2005.

[55] P. L. Kastritis and A. M. J. J. Bonvin, "Are Scoring Functions in Protein-Protein Docking Ready To Predict Interactomes? Clues from a Novel Binding Affinity Benchmark," *J. Proteome Res.*, vol. 9, pp. 2216–2225, May 2010.

[56] K. Yugandhar and M. M. Gromiha, "Protein–protein binding affinity prediction from amino acid sequence," *Bioinformatics*, vol. 30, no. 24, pp. 3583–3589, 2014.

[57] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, Perth, 1995.

[58] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[59] J. J. Gray, S. Moughon, C. Wang, O. Schueler-Furman, B. Kuhlman, C. A. Rohl, and D. Baker, "Protein–Protein Docking with Simultaneous Optimization of Rigid-body Displacement and Side-chain Conformations," *J. Mol. Biol.*, vol. 331, pp. 281–299, Aug. 2003.

[60] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[61] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, 2012.

[62] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[63] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, 2016.

[64] K. Li and J. Malik, "Learning to optimize," *arXiv preprint arXiv:1606.01885*, 2016.

[65] Y. Chen, M. W. Hoffman, S. G. Colmenarejo, M. Denil, T. P. Lillicrap, M. Botvinick, and N. de Freitas, "Learning to learn without gradient descent by gradient descent," in *Proceed-*

ings of the 34th International Conference on Machine Learning-Volume 70, pp. 748–756, JMLR. org, 2017.

[66] O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. de Freitas, and J. Sohl-Dickstein, "Learned optimizers that scale and generalize," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3751–3760, JMLR. org, 2017.

[67] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications*, SAGA'09, (Berlin, Heidelberg), pp. 169–178, Springer-Verlag, 2009.

[68] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[69] N. Bansal, X. Chen, and Z. Wang, "Can we gain more from orthogonality regularizations in training deep networks?," in *Advances in Neural Information Processing Systems*, pp. 4261–4271, 2018.

[70] T. Chen, S. Ding, J. Xie, Y. Yuan, W. Chen, Y. Yang, Z. Ren, and Z. Wang, "Abd-net: Attentive but diverse person re-identification," *ICCV*, 2019.

[71] Y. Cao and Y. Shen, "Bayesian active learning for optimization and uncertainty quantification in protein docking," *arXiv preprint arXiv:1902.00067*, 2019.

[72] B. G. Pierce, K. Wiehe, H. Hwang, B.-H. Kim, T. Vreven, and Z. Weng, "ZDOCK server: interactive docking prediction of protein–protein complexes and symmetric multimers," *Bioinformatics*, vol. 30, pp. 1771–1773, 02 2014.

[73] K. A. Porter, I. Desta, D. Kozakov, and S. Vajda, "What method to use for protein–protein docking?," *Current opinion in structural biology*, vol. 55, pp. 1–7, 2019.

[74] A. Kryshtafovych, A. Barbato, K. Fidelis, B. Monastyrskyy, T. Schwede, and A. Tramontano, "Assessment of the assessment: evaluation of the model quality estimates in casp10," *Proteins: Structure, Function, and Bioinformatics*, vol. 82, pp. 112–126, 2014.

[75] R. Cao, D. Bhattacharya, J. Hou, and J. Cheng, "Deepqa: improving the estimation of single protein model quality with deep belief networks," *BMC bioinformatics*, vol. 17, no. 1, p. 495, 2016.

[76] B. Manavalan, J. Lee, and J. Lee, "Random forest-based protein model quality assessment (rfmqa) using structural features and potential energy terms," *PloS one*, vol. 9, no. 9, p. e106542, 2014.

[77] L. J. McGuffin, "The modfold server for the quality assessment of protein structural models," *Bioinformatics*, vol. 24, no. 4, pp. 586–587, 2008.

[78] P. Popov and S. Grudinin, "Knowledge of native protein–protein interfaces is sufficient to construct predictive models for the selection of binding candidates," *Journal of chemical information and modeling*, vol. 55, no. 10, pp. 2242–2255, 2015.

[79] J. Qiu, W. Sheffler, D. Baker, and W. S. Noble, "Ranking predicted protein structures with support vector regression," *Proteins: Structure, Function, and Bioinformatics*, vol. 71, no. 3, pp. 1175–1182, 2008.

[80] A. Ray, E. Lindahl, and B. Wallner, "Improved model quality assessment using proq2," *BMC bioinformatics*, vol. 13, no. 1, p. 224, 2012.

[81] G. Pagès, B. Charmettant, and S. Grudinin, "Protein model quality assessment using 3d oriented convolutional neural networks," *Bioinformatics*, vol. 35, no. 18, pp. 3313–3319, 2019.

[82] G. Derevyanko, S. Grudinin, Y. Bengio, and G. Lamoureux, "Deep convolutional networks for quality assessment of protein folds," *Bioinformatics*, vol. 34, no. 23, pp. 4046–4053, 2018.

[83] J. Gomes, B. Ramsundar, E. N. Feinberg, and V. S. Pande, "Atomic convolutional networks for predicting protein-ligand binding affinity," *arXiv preprint arXiv:1703.10603*, 2017.

[84] J. Li, W. Zhu, J. Wang, W. Li, S. Gong, J. Zhang, and W. Wang, "Rna3dcnn: Local and global quality assessments of rna 3d structures using 3d deep convolutional neural networks," *PLoS computational biology*, vol. 14, no. 11, p. e1006514, 2018.

[85] R. Das and D. Baker, "Macromolecular modeling with rosetta," *Annu. Rev. Biochem.*, vol. 77, pp. 363–382, 2008.

[86] S. Mitternacht, "Freesasa: An open source c library for solvent accessible surface area calculations," *F1000Research*, vol. 5, 2016.

[87] T. Vreven, H. Hwang, and Z. Weng, "Integrating atom-based and residue-based scoring functions for protein–protein docking," *Protein Science*, vol. 20, no. 9, pp. 1576–1586, 2011.

[88] M. F. Lensink, G. Brysbaert, N. Nadzirin, S. Velankar, R. A. Chaleil, T. Gerguri, P. A. Bates, E. Laine, A. Carbone, S. Grudinin, *et al.*, "Blind prediction of homo-and hetero-protein complexes: The casp13-capri experiment," *Proteins: Structure, Function, and Bioinformatics*, vol. 87, no. 12, pp. 1200–1221, 2019.

[89] J. P. Rodrigues, M. Trellet, C. Schmitz, P. Kastritis, E. Karaca, A. S. Melquiond, and A. M. Bonvin, "Clustering biomolecular complexes by residue contacts similarity," *Proteins: Structure, Function, and Bioinformatics*, vol. 80, no. 7, pp. 1810–1817, 2012.

[90] UniProtConsortium, "UniProt: a worldwide hub of protein knowledge," *Nucleic Acids Res.*, vol. 47, pp. D506–D515, Jan 2019.

[91] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, *et al.*, "Gene ontology: tool for the unification of biology," *Nature genetics*, vol. 25, no. 1, pp. 25–29, 2000.

[92] S. Baker and A.-L. Korhonen, "Initializing neural networks for hierarchical multi-label text classification," Association for Computational Linguistics, 2017.

[93] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[94] R. Fa, D. Cozzetto, C. Wan, and D. T. Jones, "Predicting human protein function with multi-task deep neural networks," *PloS one*, vol. 13, no. 6, 2018.

[95] G. Zhou, J. Wang, X. Zhang, and G. Yu, "DeepGOA: Predicting gene ontology annotations of proteins via graph convolutional network," in *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pp. 1836–1841, IEEE, 2019.

[96] J. Yang, R. Yan, A. Roy, D. Xu, J. Poisson, and Y. Zhang, "The i-tasser suite: protein structure and function prediction," *Nature methods*, vol. 12, no. 1, p. 7, 2015.

[97] R. You, S. Yao, Y. Xiong, X. Huang, F. Sun, H. Mamitsuka, and S. Zhu, "NetGO: improving large-scale protein function prediction with massive network information," *Nucleic acids research*, vol. 47, no. W1, pp. W379–W387, 2019.

[98] I. Kahanda and A. Ben-Hur, "Gostruct 2.0: Automated protein function prediction for annotated proteins," in *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pp. 60–66, 2017.

[99] R. You, X. Huang, and S. Zhu, "DeepText2Go: Improving large-scale protein function prediction with deep semantic text representation," *Methods*, vol. 145, pp. 82–90, 2018.

[100] M. Stewart, H. M. Kent, and A. J. McCoy, "Structural basis for molecular recognition between nuclear transport factor 2 (NTF2) and the GDP-bound form of the Ras-family GTPase Ran," *Journal of molecular biology*, vol. 277, no. 3, pp. 635–646, 1998.

[101] D. Wrapp, N. Wang, K. S. Corbett, J. A. Goldsmith, C.-L. Hsieh, O. Abiona, B. S. Graham, and J. S. McLellan, "Cryo-EM structure of the 2019-ncov spike in the prefusion conformation," *Science*, vol. 367, no. 6483, pp. 1260–1263, 2020.

[102] D. Szklarczyk, J. H. Morris, H. Cook, M. Kuhn, S. Wyder, M. Simonovic, A. Santos, N. T. Doncheva, A. Roth, P. Bork, *et al.*, "The STRING database in 2017: quality-controlled

protein–protein association networks, made broadly accessible," *Nucleic acids research*, p. gkw937, 2016.

[103] P. Radivojac, W. T. Clark, T. R. Oron, A. M. Schnoes, T. Wittkop, A. Sokolov, K. Graim, C. Funk, K. Verspoor, A. Ben-Hur, G. Pandey, J. M. Yunes, A. S. Talwalkar, S. Repo, M. L. Souza, D. Piovesan, R. Casadio, Z. Wang, J. Cheng, H. Fang, J. Gough, P. Koskinen, P. T?r?nen, J. Nokso-Koivisto, L. Holm, D. Cozzetto, D. W. Buchan, K. Bryson, D. T. Jones, B. Limaye, H. Inamdar, A. Datta, S. K. Manjari, R. Joshi, M. Chitale, D. Kihara, A. M. Lisewski, S. Erdin, E. Venner, O. Lichtarge, R. Rentzsch, H. Yang, A. E. Romero, P. Bhat, A. Paccanaro, T. Hamp, R. Ka?ner, S. Seemayer, E. Vicedo, C. Schaefer, D. Achten, F. Auer, A. Boehm, T. Braun, M. Hecht, M. Heron, P. H?nigschmid, T. A. Hopf, S. Kaufmann, M. Kiening, D. Krompass, C. Landerer, Y. Mahlich, M. Roos, J. Bj?rne, T. Salakoski, A. Wong, H. Shatkay, F. Gatzmann, I. Sommer, M. N. Wass, M. J. Sternberg, N. ?kunca, F. Supek, M. Bo?njak, P. Panov, S. D?eroski, T. ?muc, Y. A. Kourmpetis, A. D. van Dijk, C. J. ter Braak, Y. Zhou, Q. Gong, X. Dong, W. Tian, M. Falda, P. Fontana, E. Lavezzo, B. Di Camillo, S. Toppo, L. Lan, N. Djuric, Y. Guo, S. Vucetic, A. Bairoch, M. Linial, P. C. Babbitt, S. E. Brenner, C. Orengo, B. Rost, S. D. Mooney, and I. Friedberg, "A large-scale evaluation of computational protein function prediction," *Nat. Methods*, vol. 10, pp. 221–227, Mar 2013.

[104] Y. Jiang, T. R. Oron, W. T. Clark, A. R. Bankapur, D. D'Andrea, R. Lepore, C. S. Funk, I. Kahanda, K. M. Verspoor, A. Ben-Hur, d. a. C. E. Koo, D. Penfold-Brown, D. Shasha, N. Youngs, R. Bonneau, A. Lin, S. M. Sahraeian, P. L. Martelli, G. Profiti, R. Casadio, R. Cao, Z. Zhong, J. Cheng, A. Altenhoff, N. Skunca, C. Dessimoz, T. Dogan, K. Hakala, S. Kaewphan, F. Mehryary, T. Salakoski, F. Ginter, H. Fang, B. Smithers, M. Oates, J. Gough, P. T?r?nen, P. Koskinen, L. Holm, C. T. Chen, W. L. Hsu, K. Bryson, D. Cozzetto, F. Minneci, D. T. Jones, S. Chapman, D. Bkc, I. K. Khan, D. Kihara, D. Ofer, N. Rappoport, A. Stern, E. Cibrian-Uhalte, P. Denny, R. E. Foulger, R. Hieta, D. Legge, R. C. Lovering, M. Magrane, A. N. Melidoni, P. Mutowo-Meullenet, K. Pichler, A. Shypitsyna,

B. Li, P. Zakeri, S. ElShal, L. C. Tranchevent, S. Das, N. L. Dawson, D. Lee, J. G. Lees, I. Sillitoe, P. Bhat, T. Nepusz, A. E. Romero, R. Sasidharan, H. Yang, A. Paccanaro, J. Gillis, A. E. Sede?o-Cort?s, P. Pavlidis, S. Feng, J. M. Cejuela, T. Goldberg, T. Hamp, L. Richter, A. Salamov, T. Gabaldon, M. Marcet-Houben, F. Supek, Q. Gong, W. Ning, Y. Zhou, W. Tian, M. Falda, P. Fontana, E. Lavezzo, S. Toppo, C. Ferrari, M. Giollo, D. Piovesan, S. C. Tosatto, A. Del Pozo, J. M. Fern?ndez, P. Maietta, A. Valencia, M. L. Tress, A. Benso, S. Di Carlo, G. Politano, A. Savino, H. U. Rehman, M. Re, M. Mesiti, G. Valentini, J. W. Bargsten, A. D. van Dijk, B. Gemovic, S. Glisic, V. Perovic, V. Veljkovic, N. Veljkovic, D. C. Almeida-E-Silva, R. Z. Vencio, M. Sharan, J. Vogel, L. Kansakar, S. Zhang, S. Vucetic, Z. Wang, M. J. Sternberg, M. N. Wass, R. P. Huntley, M. J. Martin, C. O'Donovan, P. N. Robinson, Y. Moreau, A. Tramontano, P. C. Babbitt, S. E. Brenner, M. Linial, C. A. Orengo, B. Rost, C. S. Greene, S. D. Mooney, I. Friedberg, and P. Radivojac, "An expanded evaluation of protein function prediction methods shows an improvement in accuracy," *Genome Biol.*, vol. 17, p. 184, 09 2016.

[105] I. Friedberg and P. Radivojac, "Community-Wide Evaluation of Computational Function Prediction," *Methods Mol. Biol.*, vol. 1446, pp. 133–146, 2017.

[106] C. E. Jones, U. Baumann, and A. L. Brown, "Automated methods of predicting the function of biological sequences using go and blast," *BMC bioinformatics*, vol. 6, no. 1, p. 272, 2005.

[107] A. Rives, S. Goyal, J. Meier, 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," *bioRxiv*, p. 622803, 2019.

[108] D. B. Duong, L. Gai, A. Uppunda, D. Le, E. Eskin, J. J. Li, and K.-W. Chang, "Annotating gene ontology terms for protein sequences with the transformer model," *bioRxiv*, 2020.

[109] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rihawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, D. Bhowmik, *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.

[110] R. You, Z. Zhang, Y. Xiong, F. Sun, H. Mamitsuka, and S. Zhu, "GOLabeler: improving sequence-based large-scale protein function prediction by learning to rank," *Bioinformatics*, vol. 34, no. 14, pp. 2465–2473, 2018.

[111] P.-S. Huang, S. E. Boyken, and D. Baker, "The coming of age of de novo protein design," *Nature*, vol. 537, no. 7620, pp. 320–327, 2016.

[112] B. E. Correia, J. T. Bates, R. J. Loomis, G. Baneyx, C. Carrico, J. G. Jardine, P. Rupert, C. Correnti, O. Kalyuzhniy, V. Vittal, *et al.*, "Proof of principle for epitope-focused vaccine design," *Nature*, vol. 507, no. 7491, pp. 201–206, 2014.

[113] D.-A. Silva, S. Yu, U. Y. Ulge, J. B. Spangler, K. M. Jude, C. Labão-Almeida, L. R. Ali, A. Quijano-Rubio, M. Ruterbusch, I. Leung, *et al.*, "De novo design of potent and selective mimics of il-2 and il-15," *Nature*, vol. 565, no. 7738, pp. 186–191, 2019.

[114] A. Ibraheem and R. E. Campbell, "Designs and applications of fluorescent protein-based biosensors," *Current opinion in chemical biology*, vol. 14, no. 1, pp. 30–36, 2010.

[115] F. H. Arnold, "Design by directed evolution," *Accounts of chemical research*, vol. 31, no. 3, pp. 125–131, 1998.

[116] P. A. Romero and F. H. Arnold, "Exploring protein fitness landscapes by directed evolution," *Nature reviews Molecular cell biology*, vol. 10, no. 12, pp. 866–876, 2009.

[117] M. J. Dougherty and F. H. Arnold, "Directed evolution: new parts and optimized function," *Current opinion in biotechnology*, vol. 20, no. 4, pp. 486–491, 2009.

[118] N. Koga, R. Tatsumi-Koga, G. Liu, R. Xiao, T. B. Acton, G. T. Montelione, and D. Baker, "Principles for designing ideal protein structures," *Nature*, vol. 491, no. 7423, pp. 222–227, 2012.

[119] P.-S. Huang, Y.-E. A. Ban, F. Richter, I. Andre, R. Vernon, W. R. Schief, and D. Baker, "Rosettaremodel: a generalized framework for flexible backbone protein design," *PloS one*, vol. 6, no. 8, p. e24109, 2011.

[120] D. M. Fowler and S. Fields, "Deep mutational scanning: a new style of protein science," *Nature methods*, vol. 11, no. 8, pp. 801–807, 2014.

[121] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[122] J. Ingraham, V. Garg, R. Barzilay, and T. Jaakkola, "Generative models for graph-based protein design," in *Advances in Neural Information Processing Systems*, pp. 15820–15831, 2019.

[123] Y. Cao, P. Das, V. Chenthamarakshan, P.-Y. Chen, I. Melnyk, and Y. Shen, "Fold2seq: A joint sequence (1d)-fold (3d) embedding-based generative model for protein design," in *International Conference on Machine Learning*, pp. 1261–1271, PMLR, 2021.

[124] M. Karimi, S. Zhu, Y. Cao, and Y. Shen, "De novo protein design for novel folds using guided conditional wasserstein generative adversarial networks (gcwgan)," *bioRxiv*, p. 769919, 2019.

[125] A. J. Riesselman, J.-E. Shin, A. W. Kollasch, C. McMahon, E. Simon, C. Sander, A. Manglik, A. C. Kruse, and D. S. Marks, "Accelerating protein design using autoregressive generative models," *bioRxiv*, p. 757252, 2019.

[126] J. G. Greener, L. Moffat, and D. T. Jones, "Design of metalloproteins and novel protein folds using variational autoencoders," *Scientific reports*, vol. 8, no. 1, pp. 1–12, 2018.

[127] A. Madani, B. McCann, N. Naik, N. S. Keskar, N. Anand, R. R. Eguchi, P.-S. Huang, and R. Socher, "Progen: Language modeling for protein generation," *arXiv preprint arXiv:2004.03497*, 2020.

[128] 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*, pp. 1–11, 2021.

[129] A. Hawkins-Hooker, F. Depardieu, S. Baur, G. Couairon, A. Chen, and D. Bikard, "Generating functional protein variants with variational autoencoders," *PLOS Computational Biology*, vol. 17, no. 2, p. e1008736, 2021.

[130] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[131] S. El-Gebali, J. Mistry, A. Bateman, S. R. Eddy, A. Luciani, S. C. Potter, M. Qureshi, L. J. Richardson, G. A. Salazar, A. Smart, *et al.*, "The pfam protein families database in 2019," *Nucleic acids research*, vol. 47, no. D1, pp. D427–D432, 2019.

[132] S. Burge, E. Kelly, D. Lonsdale, P. Mutowo-Muellenet, C. McAnulla, A. Mitchell, A. Sangrador-Vegas, S.-Y. Yong, N. Mulder, and S. Hunter, "Manual go annotation of predictive protein signatures: the interpro approach to go curation," *Database*, vol. 2012, 2012.

[133] S. Z. Alborzi, D. W. Ritchie, and M.-D. Devignes, "Computational discovery of direct associations between go terms and protein domains," *BMC bioinformatics*, vol. 19, no. 14, pp. 53–66, 2018.

[134] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[135] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic, "On mutual information maximization for representation learning," *arXiv preprint arXiv:1907.13625*, 2019.

[136] X. Zhong, R. Kaalia, and J. C. Rajapakse, "Go2vec: transforming go terms and proteins to vector representations via graph embeddings," *BMC genomics*, vol. 20, no. 9, pp. 1–10, 2019.

[137] M.-P. Dubuisson and A. K. Jain, "A modified hausdorff distance for object matching," in *Proceedings of 12th international conference on pattern recognition*, vol. 1, pp. 566–568, IEEE, 1994.

[138] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

[139] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

[140] A. J. Riesselman, J. B. Ingraham, and D. S. Marks, "Deep generative models of genetic variation capture the effects of mutations," *Nature methods*, vol. 15, no. 10, pp. 816–822, 2018.

[141] T. A. Hopf, J. B. Ingraham, F. J. Poelwijk, C. P. Schärfe, M. Springer, C. Sander, and D. S. Marks, "Mutation effects predicted from sequence co-variation," *Nature biotechnology*, vol. 35, no. 2, pp. 128–135, 2017.

[142] S. R. Eddy, "Accelerated profile hmm searches," *PLoS computational biology*, vol. 7, no. 10, p. e1002195, 2011.

[143] A. Rohatgi, "Webplotdigitizer: Version 4.5," 2021.

[144] C. Savojardo, G. Babbi, P. L. Martelli, and R. Casadio, "Mapping omim disease–related variations on protein domains reveals an association among variation type, pfam models, and disease classes," *Frontiers in molecular biosciences*, vol. 8, p. 318, 2021.

[145] G. Matheron, "Principles of geostatistics," *Economic geology*, vol. 58, no. 8, pp. 1246–1266, 1963.

[146] S. Yakowitz and F. Szidarovszky, "A comparison of kriging with nonparametric regression methods," *Journal of Multivariate Analysis*, vol. 16, no. 1, pp. 21–53, 1985.

APPENDIX A

BAL' THEORY

We first theoretically and empirically compare our Bayesian active learning (BAL) to [36] (NCPD for Nonparametric Conjugate Prior Distribution) that also models the posterior of the global optimum directly. During the comparison, we establish BAL's advantages in theory, namely (1) the annealing schedule balancing exploration and exploitation is aware of the global uncertainty and dependent on dimensionality of the search space; and (2) the Kriging regressor is consistent and unbiased. We also establish BAL's advantage in practice through empirical comparison over test functions.

A.1 Unlike NCPD, BAL's annealing schedule is global uncertainty-aware and dimension-dependent

In Nonparametric Conjugate Prior Distribution, the temperature constant ρ was estimated proportional to the effective number of data points:

$$\rho_{\text{NCPD}} = \rho_0(\xi + n \cdot \frac{\sum_i K(x_i, x_i)}{\sum_i \sum_j K(x_i, x_j)})$$

where ρ_0 is the initial value for ρ, and ξ is the effective number of data points in the prior distribution and n is the number of data points, which is penalized by $\frac{\sum_i K(x_i,x_i)}{\sum_i \sum_j K(x_i,x_j)}$ to become the effective number of sample points. The rational is that the effective number of samples somehow measures the uncertainty in the system.

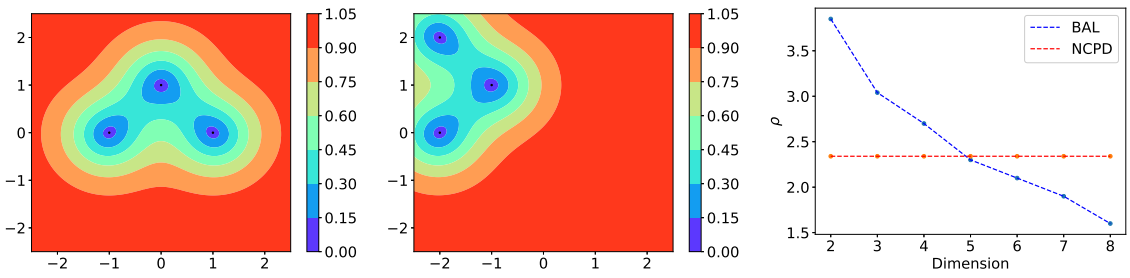
However, as our problem has a constraint for the search space, only considering the pairwise distance between samples is obviously insufficient. The location of samples in the search space also contributes to the global uncertainty in the system. Considering the two cases in Fig. A.1a and Fig. A.1b. Here we have three data points which have the exact same pairwise distance between

each other. The figure shows the standard deviation within the square search space. Obviously, the second situation has more uncertainties than the first one because the points in the second figure are closer to the boundary, which makes the large region of the right bottom untouched and high variance. But the ρ used in NPCD remains the same for both cases.

In contrast, the ρ in our BAL is defined as

$$\rho_{\text{BAL}} = \rho_0 \cdot \exp((h_p^{(t-1)})^{-1} n_t^{\frac{1}{d}})$$

Here we use $h_p^{(t-1)}$, the continuous entropy for the latest posterior distribution p, which is a global measure of uncertainty. In other words, we consider not only the internal structure between the samples but also the location of samples within the search space. Obviously our ρ for the case in Fig. A.1a is bigger than that for Fig. A.1b, which makes much more sense.



(a) (b) (c)

Figure A.1: (a): The contour plot for the standard deviation within [-2.5,2.5]×[-2.5,2.5] with three data points at [-1,0], [1,0], [0,1]. (b) The contour plot for the standard deviation within [-2.5,2.5]×[-2.5,2.5] with three data points at [-2,0], [-2,2], [-1,1]. (c) The ρ keeps unchanged for NCPD(red line) as dimension goes higher, while it decreases quickly in BAL (blue line).

Moreover, a good temperature constant should be generalizable for different dimensions. However, NCPD is bad-generalized for different dimensions. For instance, we remain the same pairwise distance between any two samples as shown Fig. A.1a and then we extends the dimension from 2 to 8. It could be seen in Fig. A.1c. The ρ in NCPD remains the same while in BAL it decreases

rapidly as the dimension goes larger. For the same set of data, as the dimension goes higher, the uncertainty of the system must decrease, which means ρ must decrease at the same time. Therefore, compared to NCPD our BAL matches the rationale and could be generalizable for various dimensions.

Lastly, we found that the ρ in NCPD decreases as the number of samples increases in some situations. This is totally controversial to our rational for ρ. As we are getting more samples, our knowledge about the system is increasing (not dropping the old samples). In adaptive simulated annealing, this means our system is getting cooler and cooler as the annealing procedure goes forward. Therefore, there is a monotonous-positive relationship between ρ and the number of samples. However, considering the simple example below, we have two data points $\boldsymbol{x_1} = [1, 0]$ and $\boldsymbol{x_2} = [-1, 0]$. We considered a kernel that $K(\boldsymbol{x_1}, \boldsymbol{x_2}) \approx 0$ and $K(\boldsymbol{x_i}, \boldsymbol{x_i}) = 1$(e.g. RBF kernel with bandwith $l << 1$). The effective number of location for the sample is

$$t_2 = 2 \cdot \frac{\sum_i K(x_i, x_i)}{\sum_i \sum_j K(x_i, x_j)} = 2 \cdot \frac{2}{2} = 2$$

Then we add the third sample $\boldsymbol{x_3} = [-1, 0 + \epsilon]$, where ϵ is a tiny positive number, so that $K(\boldsymbol{x_1}, \boldsymbol{x_3}) \approx 0$ and $K(\boldsymbol{x_1}, \boldsymbol{x_2}) \approx 1$. Then the effective number of location will become:

$$t_3 = 3 \cdot \frac{\sum_i K(x_i, x_i)}{\sum_i \sum_j K(x_i, x_j)} = 3 \cdot \frac{3}{5} = 1.8$$

We have $t_3 < t_2$! That means when collecting a new sample $\boldsymbol{x_3}$, the system's uncertainty is becoming larger. Although the new sample is located very closed to the old one, it is obviously controversial to our understanding and rationale for ρ. The situation for ρ decreasing frequently happens when n becomes large, resulting from the new samples having a large chance to be closed to the old ones. By contrast, in BAL, our ρ is in positive relation to n and the negative relation to H, while n is getting larger and H is getting smaller over iterations. Therefore, our ρ strictly increased when the system is getting more samples.

A.2 Unlike NCPD, BAL's Kriging regressor is unbiased

In this section we prove the regressor used in NCPD is biased. Then we follow [145] and briefly derive our Kriging results.

A.2.0.1 The regressor in NCPD is biased

We first consider the one-dimensional case, and then extend to multi-dimensional cases. Suppose we have two groups of random variables X_i, Y_i which are $i.i.d.$ with joint pdf (probability distribution function) $p(x, y)$. The mean function $f(x) = E[Y|X = x]$ is the true function that we want to predict. Suppose the marginal pdf of X has the form

$$p(x) \propto e^{\frac{f(x)}{\beta}}$$

The estimator in NCPD for $f(x)$ is

$$\widehat{f}(x) = \frac{\sum_{i=1}^{n} K(\frac{x-x_i}{h}) y_i + k_0(x) * y_0(x)}{\sum_{i=1}^{n} K(\frac{x-x_i}{h}) + k_0(x)}$$

For a given x_i, the random variable y_i can be written as

$$y_i = f(x_i) + \epsilon_i$$

where ϵ_i is a zero-mean noise. In our case, it could be regarded as the system error and the difference between our supposed energy function and the true energy function.

Therefore, we have

$$\sum_{i=1}^{n} K(\frac{x - x_i}{h}) y_i = \sum_{i=1}^{n} K(\frac{x - x_i}{h}) f(x_i) + \sum_{i=1}^{n} K(\frac{x - x_i}{h}) \epsilon_i$$

add prior function to both sides, and divide them by $\sum_{i=1}^{n} K(\frac{x-x_i}{h}) + k_0(x)$, we reach

$$\frac{\sum_{i=1}^{n} K(\frac{x-x_i}{h}) y_i + k_0(x) * y_0(x)}{\sum_{i=1}^{n} K(\frac{x-x_i}{h}) + k_0(x)} = \frac{\sum_{i=1}^{n} K(\frac{x-x_i}{h}) f(x_i) + \sum_{i=1}^{n} K(\frac{x-x_i}{h}) \epsilon_i + k_0(x) * y_0(x)}{\sum_{i=1}^{n} K(\frac{x-x_i}{h}) + k_0(x)}$$

Note that the left side is just $\widehat{f}(x)$. So we have

$$\widehat{f}(x) - f(x) = \frac{\sum_{i=1}^{n} K(\frac{x-x_i}{h}) f(x_i) + \sum_{i=1}^{n} K(\frac{x-x_i}{h}) \epsilon_i + k_0(x) * y_0(x)}{\sum_{i=1}^{n} K(\frac{x-x_i}{h}) + k_0(x)} - f(x)$$

Multiply both the nominator and the denominator by $\frac{1}{nh}$, we have

$$\widehat{f}(x) - f(x) = \frac{\frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h}) f(x_i) + \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h}) \epsilon_i + \frac{1}{nh} k_0(x) * y_0(x)}{\frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h}) + \frac{1}{nh} k_0(x)} - f(x)$$

Note the the first term in the denominator is the kernel density estimator for $p(x)$

$$\widehat{p}(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - x_i}{h})$$

We can rewrite the whole equation

$$\widehat{f}(x) - f(x) = \frac{\frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h})(f(x_i) - f(x)) + \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x-x_i}{h}) \epsilon_i + \frac{1}{nh} k_0(x) * (y_0(x) - f(x))}{\widehat{p}(x) + \frac{1}{nh} k_0(x)}$$

Define

$$m_1(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - x_i}{h})(f(x_i) - f(x))$$

$$m_2(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - x_i}{h}) \epsilon_i$$

We can simplify the equation above as

$$\widehat{f}(x) - f(x) = \frac{m_1(x)}{\widehat{p}(x) + \frac{1}{nh}k_0(x)} + \frac{m_2(x)}{\widehat{p}(x) + \frac{1}{nh}k_0(x)} + \frac{1}{nh}\frac{k_0(x) * (y_0(x) - f(x))}{\widehat{p}(x) + \frac{1}{nh}k_0(x)}$$

By fixing x, we start to analyze each term in the right side of the equation.

(1) We start with the simplest term among the three, the second term. We first calculate the expectation and variance of $m_2(x)$.

Since we know $E[\epsilon_i|X_i] = 0$

Then

$$E[m_2(x)] = E_{x_i}[E_{\epsilon_i|x_i}[\frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h})\epsilon_i]] = E_{x_i}[0] = 0$$

Because (X_i, Y_i) are i.i.d, we have

$$Var[m_2(x)] = \frac{1}{n^2h^2}\sum_{i=1}^{n} Var[K(\frac{x - x_i}{h})\epsilon_i] = \frac{1}{nh^2}E[K^2(\frac{x - x_i}{h})\epsilon_i^2]$$

Define $\sigma^2(x) = E[\epsilon_i^2|X_i]$. We have

$$Var[m_2(x)] = \frac{1}{nh^2}E[K^2(\frac{x - x_i}{h})\sigma^2(x)] = \frac{1}{nh^2}\int K^2(\frac{x - x_i}{h})\sigma^2(x_i)p(x_i)dx_i$$

By setting $u = \frac{x_i - x}{h}$, we have

$$Var[m_2(x)] = \frac{1}{nh}\int K^2(u)\sigma^2(hu + x)p(hu + x)du$$

We use Taylor expansion for $\sigma^2(hu + x)$ and $p(hu + x)$ up to $o(h)$, and get

$$Var[m_2(x)] = \frac{1}{nh}\int K^2(u)\sigma^2(x)p(x)du + o(\frac{1}{n})$$

Define $\mu = \int K^2(u)du$. We obtain

$$Var[m_2(x)] = \frac{\mu\sigma^2(x)p(x)}{nh} + o(\frac{1}{n})$$

By applying Central Limit Theorem, we get the asymptotic result for $m_2(x)$

$$\lim_{\substack{h\to 0 \\ nh\to\infty}} \sqrt{nh}m_2(x) \longrightarrow_d N(0, \mu\sigma^2(x)p(x))$$

(2) Second, we work on the first term and calculate the expectation and variance of $m_1(x)$. Because $X_i s$ are $i.i.d$, we have

$$E[m_1(x)] = \frac{1}{h}E[K(\frac{x-x_i}{h})(f(x_i) - f(x))] = \frac{1}{h}\int K(\frac{x-x_i}{h})(f(x_i) - f(x))p(x_i)dx_i$$

Let $u = \frac{x_i - x}{h}$,

$$E[m_1(x)] = \int K(u)(f(hu+x) - f(x))p(hu+x)du$$

Similar to the work above, we expand $(f(hu+x) - f(x))$ and $p(hu+x)$ up to $o(h^2)$, and obtain

$$\begin{aligned} E[m_1(x)] &= \int K(u)(huf^{'}(x) + \frac{h^2u^2}{2}f^{''}(x))(p(x) + hup^{'}(x))du + o(h^3) \\ &= hf^{'}(x)p(x)\int K(u)udu + h^2(f^{'}(x)p^{'}(x) + \frac{1}{2}f^{''}(x)p(x))\int K(u)u^2du + o(h^3) \end{aligned}$$

Let $\kappa_2 = \int K(u)u^2du$. Since $\int K(u)udu = 0$, we have

$$E[m_1(x)] = h^2\kappa_2(f^{'}(x)p^{'}(x) + \frac{1}{2}f^{''}(x)p(x)) + o(h^3)$$

The same method can obtain

$$Var[m_1(x)] = o(\frac{h^2}{nh})$$

Therefore, we have

$$\lim_{\substack{h\to 0\\ nh\to\infty}} \sqrt{nh}(m_1(x) - h^2\kappa_2(f^{'}(x)p^{'}(x) + \frac{1}{2}f^{''}(x)p(x))) \longrightarrow_p 0$$

The kernel density estimator $\widehat{p}(x)$ has the property

$$\lim_{\substack{h\to 0\\ nh\to\infty}} \widehat{p}(x) \longrightarrow_p p(x)$$

By using Slutsky's theorem, and let $B(x) = f^{'}(x)p^{'}(x)p^{-1}(x) + \frac{1}{2}f^{''}(x)$, we get

$$\lim_{\substack{h\to 0\\ nh\to\infty}} \sqrt{nh}(\frac{m_1(x)}{\widehat{p}(x) + \frac{1}{nh}k_0(x)} + \frac{m_2(x)}{\widehat{p}(x) + \frac{1}{nh}k_0(x)} - h^2\kappa_2 B(x)) = \lim_{\substack{h\to 0\\ nh\to\infty}} \sqrt{nh}(\frac{m_1(x) + m_2(x)}{p(x)})$$

$$\longrightarrow_d N(0, \frac{\mu\sigma^2(x)}{p(x)})$$

(3) Third, we calculate the last term in $\widehat{f}(x) - f(x)$. Note that $\widehat{p}(x)$ is the only part including random variables of $\frac{1}{nh}\frac{k_0(x)*(y_0(x)-f(x))}{p(x)+\frac{1}{nh}k_0(x)}$. So we can easily conclude

$$\lim_{\substack{h\to 0\\ nh\to\infty}} \frac{k_0(x) * (y_0(x) - f(x))}{\widehat{p}(x) + \frac{1}{nh}k_0(x)} \longrightarrow_p \frac{k_0(x) * (y_0(x) - f(x))}{p(x)}$$

so that

$$\lim_{\substack{h\to 0\\ nh\to\infty}} \sqrt{nh}(\frac{1}{nh}\frac{k_0(x) * (y_0(x) - f(x))}{\widehat{p}(x) + \frac{1}{nh}k_0(x)}) \longrightarrow_p 0$$

Therefore, in summary, we obtain

$$\lim_{\substack{h\to 0\\ nh\to\infty}} \sqrt{nh}(\widehat{f}(x) - f(x) - h^2\kappa_2 B(x)) \longrightarrow_d N(0, \frac{\mu\sigma^2(x)}{p(x)})$$

It is easy to extend the result to multi-variable cases: For d dimensions, we have

$$\lim_{\substack{H\to 0\\ n|H|\to\infty}} \sqrt{n|H|}(\widehat{f}(\mathbf{x}) - f(\mathbf{x}) - \kappa_2 \sum_{i=1}^{i=d} h_i^2 B_i(\mathbf{x})) \longrightarrow_d N(0, \frac{\mu^d\sigma^2(\mathbf{x})}{p(\mathbf{x})})$$

where H is the bandwidth matrix. In the Gaussian kernel, it is the covariance matrix.

In total, we have proved that the regressor in NCPD is biased and converges to the normal distribution with mean equal to $\kappa_2 \sum_{i=1}^{i=d} h_i^2 B_i(\mathbf{x})$.

A.3 Derivation of BAL's Kriging regressor

We briefly derive the Kriging regressor following [40]. Let $F(\boldsymbol{x})$ be a random function with mean equal to $f(\boldsymbol{x})$, and $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ be our observed data with a noise ϵ. We are trying to find an unbiased linear estimator $\hat{f}(\boldsymbol{x}) = \sum_i \lambda_i(\boldsymbol{x}) y_i$ for $f(\boldsymbol{x})$ with the smallest variance:

$$\text{Minimize } Var[(\hat{f}(\boldsymbol{x}) - F(\boldsymbol{x}))^2]$$
$$\text{Subject to } E[\hat{f}(\boldsymbol{x})] = E[F(\boldsymbol{x})]$$

We define the covariance between $f(\boldsymbol{x}_1)$ and $f(\boldsymbol{x}_2)$ as $cov(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\boldsymbol{x}_1, \boldsymbol{x}_2)$. Therefore, we have $cov(y_1, y_2) = k(\boldsymbol{x}_1, \boldsymbol{x}_2) + \epsilon^2$. We then expand our objective function as

$$Var[(\hat{f}(\boldsymbol{x}) - F(\boldsymbol{x}))^2] = \kappa(\boldsymbol{x}, \boldsymbol{x}) + \boldsymbol{\lambda}(\boldsymbol{\kappa}(\boldsymbol{x}))^T(\boldsymbol{K} + \epsilon^2\boldsymbol{I})\boldsymbol{\lambda}(\boldsymbol{x}) - 2\boldsymbol{\lambda}(\boldsymbol{x})^T\boldsymbol{k}$$

where $\boldsymbol{K}$ is the covariance matrix with $[\boldsymbol{K}_n]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$; $\boldsymbol{\kappa}(\boldsymbol{x})$ is the covariance vector between

$\boldsymbol{x}$ and $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n$, and $\boldsymbol{\lambda}(\boldsymbol{x})$ is the vector of $\lambda_1(\boldsymbol{x}), \lambda_2(\boldsymbol{x}), ..., \lambda_n(\boldsymbol{x})$.

In [145], it assumes that $f(\boldsymbol{x})$ consists of a linear combination of finite low-degree functions:

$$f(\boldsymbol{x}) = \sum_{i=1}^{l} \beta_i f_i(\boldsymbol{x})$$

where the coefficient vector $\boldsymbol{\beta}$ is unknown. Then we could expand the unbiased constraint as:

$$\sum_{i}^{l} \sum_{j}^{n} \beta_i \lambda_j(\boldsymbol{x}) f_i(\boldsymbol{x}_j) = \sum_{i}^{l} \beta_i f_i(\boldsymbol{x})$$

Because the above equation should hold for any arbitrary $\boldsymbol{\beta}$, we obtain:

$$\sum_{j}^{l} \lambda_j(\boldsymbol{x}) f_i(\boldsymbol{x}_j) = f_i(\boldsymbol{x}) \text{ for all i}$$

We write it into the matrix form: $\boldsymbol{G\lambda}(\boldsymbol{x}) = \boldsymbol{f}$, where $\boldsymbol{G}_{l\times n} = \begin{bmatrix} f_1(\boldsymbol{x}_1) & f_1(\boldsymbol{x}_2) & ... \\ f_2(\boldsymbol{x}_1) & f_2(\boldsymbol{x}_2) & ... \\ ... & & \end{bmatrix}$, and $\boldsymbol{f}$ is the vector of $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ...$

We let $\boldsymbol{\gamma}$ be the vector of the Lagrangian multiplier and write the Lagrangian formula:

$$L(\boldsymbol{\lambda}(\boldsymbol{x}), \boldsymbol{\gamma}) = \kappa(\boldsymbol{x}, \boldsymbol{x}) + \boldsymbol{\lambda}(\boldsymbol{x})^T(\boldsymbol{K} + \epsilon^2\boldsymbol{I})\boldsymbol{\lambda}(\boldsymbol{x}) - 2\boldsymbol{\lambda}(\boldsymbol{x})^T\boldsymbol{\kappa}(\boldsymbol{x}) + 2\boldsymbol{\gamma}^T(\boldsymbol{G\lambda}(\boldsymbol{x}) - \boldsymbol{f})$$

We take the partial derivatives with respect to $\lambda(\boldsymbol{x})$ and $\boldsymbol{\gamma}$ and let them equal to 0:

$$\frac{\partial L}{\partial \boldsymbol{\lambda}(\boldsymbol{x})} = 2(\boldsymbol{K} + \epsilon^2\boldsymbol{I})\lambda(\boldsymbol{x}) - 2\boldsymbol{\kappa}(\boldsymbol{x}) + 2\boldsymbol{G}^T\boldsymbol{\gamma} = \boldsymbol{0}$$
$$\frac{\partial L}{\partial \boldsymbol{\gamma}} = 2(\boldsymbol{G\lambda}(\boldsymbol{x}) - \boldsymbol{f}) = \boldsymbol{0}$$

The above two equations form the general linear system for Kriging. In practice, we usually assume a prior estimator $f_0(\boldsymbol{x})$ for $f(\boldsymbol{x})$. We thus shift the mean away to consider a zero-mean

case:

$$P(\boldsymbol{x}) = F(\boldsymbol{x}) - f(\boldsymbol{x}) \approx F(\boldsymbol{x}) - f_0(\boldsymbol{x})$$

We solve the linear system for $P(\boldsymbol{x})$. As the mean of $P(\boldsymbol{x})$ equal to 0, the matrix $\boldsymbol{G}$ is a zero matrix. It is straightforward to get: $\boldsymbol{\lambda}(\boldsymbol{x}) = (\boldsymbol{K} + \epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{\kappa}(\boldsymbol{x})$. Remember the observed y_i for $P(\boldsymbol{x})$ should be also shifted by $f_0(\boldsymbol{x_i})$. Therefore, we get the estimator for $E[P(\boldsymbol{x})]$:

$$\hat{p}(\boldsymbol{x}) = \boldsymbol{\kappa}(\boldsymbol{x})^T(\boldsymbol{K} + \epsilon^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{f_0})$$

where $\boldsymbol{y}$ and $\boldsymbol{f_0}$ are the vector of $y_1, y_2, ..., y_n$ and $f_0(\boldsymbol{x}_1), f_0(\boldsymbol{x}_2), ..., f_0(\boldsymbol{x}_n)$, respectively. We finally add the prior back to the equation, and get our final estimator for $f(\boldsymbol{x})$:

$$\hat{f}(\boldsymbol{x}) = \boldsymbol{\kappa}(\boldsymbol{x})^T(\boldsymbol{K} + \epsilon^2 \boldsymbol{I})^{-1}(\boldsymbol{y} - \boldsymbol{f_0}) + f_0(\boldsymbol{x})$$

So far we have proved the regressor used in NCPD is biased and then derived our Kriging regressor. The unbiased property of Kriging regressor could let the estimated function capture the location of the optimal funnel more accurate. Moreover, if the variogram is known, the expected square error of the kriging regressor is no greater than that of the NPR estimator [146]. Lastly, NCPD regressor suffers mostly from its high biasness at the boundary of the search space, because of the asymmetry of the kernel weights in such regions. This will cause the posterior may still remain a high probability value at the boundary, while the probability values of the outside regions are regarded as 0.

A.4 Empirical Comparison

In order to be more rigorous, we put the empirical comparison between the NCPD and BAL here. We use the same testing functions as in the main text and the same method for posterior analysis to get the optimization and uncertainty quantification results. For the optimization, in Table A.1, our BAL outperforms NCPD for every test function with every tested dimensionality. For UQ, in Table A.2, the tightness of BAL is much lower then NCPD. At the same time, in

	NCPD			BAL		
Dimension	2	6	12	2	6	12
Rastrigin	1.04(0.36)	1.96(0.34)	5.35(1.23)	0.84(0.05)	1.83(0.11)	3.58(0.87)
Rosenbrock	0.78(0.73)	1.96(0.86)	4.00(0.56)	0.23(0.19)	1.43(0.75)	2.25(0.21)
Griewank	2.73(0.51)	4.45(1.05)	6.01(2.24)	1.65(1.42)	4.35(1.01)	4.11(1.28)
Ackley	0.33(0.31)	0.91(0.25)	2.63(0.31)	0.26(0.14)	0.69(0.23)	1.69(0.11)

Table A.1: The means and the standard deviations (in the parenthesis) of $||\hat{\boldsymbol{x}} - \boldsymbol{x}^*||_2$ for four test functions on three dimensions.

		NCPD			BAL		
Dim		d	por	T	d	por	T
4*2	Rastrigin	1.84(1.03)	0.78	0.78(0.23)	1.37(1.18)	0.91	0.54(0.13)
	Rosenbrock	1.31(0.32)	0.89	0.57(0.15)	0.63(0.14)	0.98	0.55(0.06)
	Griewank	3.91(1.88)	0.80	0.65(0.22)	2.85(1.45)	0.86	0.62(0.20)
	Ackley	0.51(0.17)	0.99	0.53(0.21)	0.41(0.20)	0.99	0.43(0.02)
4*6	Rastrigin	2.68(1.54)	0.72	0.33(0.12)	2.25(1.02)	0.91	0.22(0.11)
	Rosenbrock	3.04(1.98)	0.90	0.51(0.16)	2.16(1.43)	0.92	0.50(0.25)
	Griewank	5.21(0.87)	0.68	0.14(0.03)	4.53(0.78)	0.76	0.04(0.02)
	Ackley	1.13(0.22)	0.89	0.33(0.30)	0.77(0.34)	0.96	0.20(0.08)
4*12	Rastrigin	6.53(1.98)	0.70	0.23(0.11)	3.62(0.36)	0.73	0.01(0.01)
	Rosenbrock	4.50(1.30)	0.83	0.12(0.10)	2.22(1.18)	0.87	0.03(0.01)
	Griewank	7.05(2.23)	0.67	0.11(0.03)	4.25(1.01)	0.63	0.05(0.03)
	Ackley	4.89(1.64)	0.95	0.88(0.69)	2.89(0.49)	0.93	0.66(0.23)

Table A.2: Uncertainty Quantification for the test functions.

majority of the cases, the portion of BAL is closer to 0.9. This means we have more tighter confidence interval but the accuracy of the interval is increasing at the same time. This is because our ρ captured the global uncertainty in the system and is also dimensional independent, and we use a mini-batch sampling which could not only explore more on the search space, but also make the data more i.i.d. which will benefit the convergence of the regressor.

A.5 Summary

We compare the posteriors of NCPD and our BAL from both theoretical and empirical prospectives. We claim three drawbacks of the ρ in NCPD and how our BAL overcomes it. We shows

the regressor we used is the best linear unbiased regressor. The empirical results show our method outperforms theirs in both optimization and uncertainty quantification.