Conservative Objective Models for Biological Sequence Design

Sathvik Kolli

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-102
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-102.html

May 11, 2023

Acknowledgement

The research in this report was a collaborative effort with assistance and input from the following individuals: Xinyang Geng, Aviral Kumar, Amy X. Lu, Aniketh Janardhan Reddy, Michael H. Herschl, Sergey Ovchinnikov, Nilah M. Ioannidis, and Sergey Levine.

I would like to thank Professor Sergey Levine for giving me the opportunity to do research and for taking the time to provide guidance and supervision. I would also like to thank Xinyang Geng, Aviral Kumar, Amy X. Lu, and Aniketh Janardhan Reddy for mentoring me, answering my questions, and providing me with assistance that was integral to my progress. Finally, I would like to thank my family who supported and encouraged me throughout my academic journey.

Conservative Objective Models for Biological Sequence Design

by Sathvik Kolli

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Sergey Levine
Research Advisor

5/3/23

(Date)

* * * * * * *

Professor Nilah Ioannidis
Second Reader

5/6/23

(Date)

Conservative Objective Models for Biological Sequence Design

Sathvik Kolli
Dept. of Electrical Engineering and Computer Sciences, University of California Berkeley
sathkolli@berkeley.edu

Abstract

The success of deep learning in computational biology has been largely limited to prediction problems, such as protein structure prediction and gene expression prediction. Nevertheless, these successes serve as a testament to the ability of deep neural networks to extract useful insights from datasets of biological sequences, and this has recently motivated research into the applications of deep learning for biological sequence design problems. In this paper, we tackle two important synthetic biology problems: (1) the problem of designing promoter sequences that are differentially expressed and (2) the inverse-protein folding problem of recovering protein sequences from three-dimensional structure. We identify both problems as black-box computational design problems, and we adapt conservative objective models (COMs), a data-driven offline model-based optimization (MBO) technique that has been used successfully on a wide range of design problems, to design biological sequences in these settings. On both problems, we demonstrate that our approach significantly outperforms standard offline MBO techniques.

1 Introduction

Deep neural networks have been used extensively to predict important properties of biological sequences, such as fitness, structure, and expression, with a high degree of accuracy (Jumper et al., 2021; Baek et al., 2021; Avsec et al., 2021; Gligorijevic et al., 2019; Rives et al., 2019; Agarwal and Shendure, 2020). However, these models have rarely been employed effectively on the task of designing novel sequences.

The problem of designing biological sequences can be framed as a black-box computational design problem, given that the true objective function is unknown and expensive to query (Trabucco et al., 2021a). Data-driven model-based optimization (MBO), also known as model-guided exploration, is an established approach to solving this class of problems (Sinai and Kelsic, 2020; Hie and Yang, 2022; Brookes et al., 2019; Liao et al., 2019; Gómez-Bombarelli et al., 2018; Fannjiang and Listgarten, 2020; Kumar and Levine, 2019). However, the standard approach which optimizes designs against a learned proxy model of the ground truth objective is susceptible to producing out-of-distribution, invalid designs that "fool" the proxy model into outputting a high value (Trabucco et al., 2021b; Kumar and Levine, 2019; Zhu et al., 2016). This is especially true in the setting of biological sequence design where valid designs lie on a narrow manifold in a high-dimensional design space and where the learned proxy model is an overparameterized deep neural network.

Conservative objective models (COMs) have been shown to successfully address the aforementioned distribution shift problem (Trabucco et al., 2021b). These models mitigate overestimation errors in the learned proxy model by sampling adversarial inputs and penalizing the predictions on these inputs during training. While COMs have been effective on a wide array of problems, we adapt them to produce general design algorithms for two synthetic biology problems. We experiment with various modifications to the standard COMs procedure, and we use novel optimizers for adversarial sampling and design.

In this paper, we successfully apply our approach to two important biological sequence design problems: (1) the problem of designing promoter sequences that are differentially-expressed and (2) the problem of recovering protein sequences from structure. The first problem is critically important for gene therapy, which delivers therapeutic genetic cargo to disease-associated cells and tissue. In order to effectively target specific cells while mitigating side effects in other cells, the genetic cargo must be precisely expressed (Sayed et al., 2022; Shirley et al., 2020; Reddy et al., 2023). This is predominantly achieved via a regulatory DNA sequence called a promoter. However, only a handful of cell type-specific promoters are currently known. Consequently, a method for reliably designing novel promoter sequences with specific expression patterns would be tremendously impactful for therapeutic applications (Reddy et al., 2023). The second problem is a well-known inverse protein-folding problem. For proteins which fold into well-defined structures, Anfinsen's thermodynamic hypothesis of folding states that their folded shapes are determined through their primary sequence. This motivates the inverse protein-folding problem, which aims to recover the lowest-energy amino acid sequence that fits a desired structure.

For both problems, we demonstrate that our approach leads to significant improvements in design performance and that COMs-based design algorithms are a promising approach to biological sequence design.

2 Background

First, we begin by exploring what makes a design problem a black-box computational design problem and gaining a high-level understanding of offline model based optimization (MBO) and its distribution shift problem. Then, we provide an overview of conservative objective models (COMs), a successful new approach to tackling the distribution shift problem and significantly improving performance of offline MBO in a wide range of computational design problems.

2.1 Black-Box Computational Design

The problem of using deep learning to design proteins or genes is part of a more general class of problems known as black-box computational design problems. The universal characteristic of these problems is that they involve generating optimal designs where the objective function and constraints are unknown. Mathematically, we want to find the optimal design, x, that maximizes some unknown objective function, $f(x)$:

$$\arg\max_{x} \quad f(x) \tag{1}$$

Examples of black-box computational design problems include optimizing robot morphologies, biological sequences, computer chips, neural network architectures, or superconducting materials.

2.2 Offline Model-Based Optimization (MBO)

A promising approach to solving black-box optimization problems is data-driven MBO, where a proxy model of the unknown objective function is learned from empirically collected data and used to guide the design procedure (Snoek et al., 2012; Brookes et al., 2019; Kumar and Levine, 2019; Trabucco et al., 2021b).

In order to model the true objective function with high fidelity, it is often critical to actively collect additional data during the training procedure (Snoek et al., 2012). However, in many design problems, including the one of biological sequence design, active real-world data collection is expensive (e.g. requires synthesizing proteins in a wet lab) or dangerous (e.g. when optimizing over aircraft designs), and reliable simulations are infeasible. Thus, we focus instead on the more practical setting of offline MBO, where we are given a static dataset of designs and cannot make any queries to the ground truth (Kumar and Levine, 2019; Trabucco et al., 2021a;b).

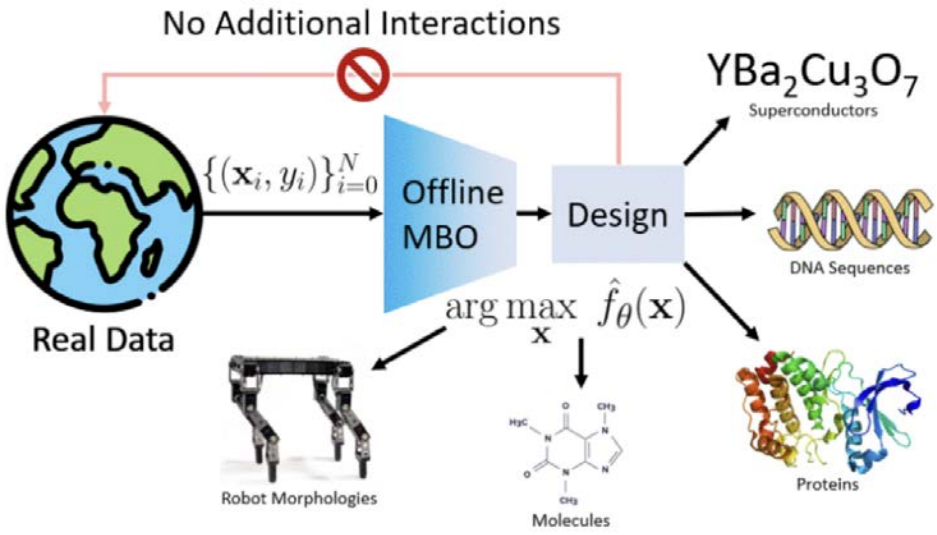

Figure 1: **A typical offline MBO workflow (Trabucco et al., 2021a).** We are given a static dataset of designs, which we use to learn a proxy model, $\hat{f}_\theta$, of the true objective. Then, our design procedure is guided by the learned proxy model.

In essence, when we use offline MBO to solve black-box optimization problems, we are trying to solve the optimization problem in equation (1) with two key assumptions:

1. Black-box assumption: $f(x)$ is an unknown function
2. Offline assumption: $f(x)$ is expensive to query

The general offline MBO workflow is illustrated in Figure 1.

While online exploration is possible when it comes to biological sequence design, focusing on the offline setting enables us to develop techniques that are considerably cheaper and faster. Furthermore, the problem of offline MBO is more accessible as it is a purely data-driven problem which enables practitioners without access to wet lab facilities to engage in research (Kolli et al., 2022).

2.3 MBO's Distribution Shift Problem

The most basic approach to offline, data-driven model-based optimization involves the following steps (Trabucco et al., 2021b):

1. We have a static dataset D of input designs and their corresponding objective values:

   $$\{(x_1, y_1), \ldots, (x_N, y_N)\}$$

   We assume this paired data was generated from a true, unknown objective function, $y = f(x)$.
2. Using the dataset D, we learn a proxy model $\hat{f}_\theta$ of the unknown objective function f via supervised regression.
3. Finally, we find an optimal generated design x^*, by optimizing some data point $x_0 \in D$ against the learned model $\hat{f}_\theta$. For example, we could use T gradient ascent/descent steps on the learned function:

   $$x_{k+1} \leftarrow x_k + \alpha \nabla_x \hat{f}_\theta(x)|_{x=x_k}, \text{ for } k \in [0, T-1]$$

The above approach generally does not perform well in high-dimensional input spaces, where the space of valid input designs lie on a narrow manifold, because overestimation errors in the proxy model $\hat{f}_\theta$ would cause the optimization procedure in step (3) to produce out-of-distribution, invalid, and low-valued designs (Trabucco et al., 2021b; Kumar and Levine, 2019). Consequently, for the above method to work, it is critical that we ensure that the proxy model $\hat{f}_\theta$ does not overestimate the objective value of out-of-distribution points.

Existing approaches to prevent overestimation of out-of-distribution inputs involve generative modeling, explicit density estimation, the use of ensemble proxy models, or regularization techniques that incentivize conservatism in regions with limited data. We will focus on the lattermost technique, which is known as conservative objective models (COMs).

2.4 Conservative Objective Models (COMs)

Conservative objective models (COMs) is a method that learns a proxy model of the ground-truth objective that is conservative in regions with limited data (Trabucco et al., 2021b). More specifically, with COMs, we learn a model that lower bounds the actual value of the ground-truth objective on out-of-distribution inputs. Thus, COMs directly addresses the distribution shift problem in MBO by ensuring that our proxy model does not overestimate the objective value of out-of-distribution designs.

With COMs, we modify step (2) in the standard MBO procedure described in Section 2.3. Rather than training the proxy model $\hat{f}_\theta$ using vanilla supervised regression, we train it via a regularized supervised regression procedure (Trabucco et al., 2021b):

1. Initialize $\hat{f}_\theta$. Pick α and an optimizer.
2. For each training step:
   (a) Sample $(x_0, y) \sim D$.
   (b) Using your optimizer of choice, optimize x_0 in order to obtain your adversarial sample x_{adv}. For example, if your optimizer performed T gradient ascent steps with learning rate η, your optimization procedure would be:

$$x_{k+1} \leftarrow x_k + \eta \nabla_x \hat{f}_\theta(x)|_{x=x_k}, \text{ for } k \in [0, T-1]$$

   and $x_{\text{adv}} = x_T$.
   (c) Minimize $\mathcal{L}(\theta; \alpha)$ with respect to θ.

$$\mathcal{L}(\theta; \alpha) = (\hat{f}_\theta(x_0) - y)^2 + \alpha(\hat{f}_\theta(x_{\text{adv}}) - \hat{f}_\theta(x_0))$$
$$\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}(\theta; \alpha)$$

At each training step, we sample an adversarial design x_{adv} by optimizing the input x_0 sampled from the training dataset. Conceptually, this adversarial input is a potentially invalid, out-of-distribution input that appears promising under the learned model $\hat{f}_\theta$.

Next, we update the model parameters θ to minimize a loss function that is a linear combination of both the standard supervised regression loss and a COMs regularizer term. The COMs regularizer term penalizes the value of the proxy model $\hat{f}_\theta$ on the adversarial input x_{adv}. In other words, we incentivize the proxy model to be conservative and to avoid overestimating the objective value of the adversarial input. Note that the COMs regularizer term also incentivizes maximizing the proxy model's prediction on the training dataset input, x_0. This ensures that we avoid systematic underestimation even for in-distribution points.

Mathematically, our training objective is given by the following equation, where μ denotes the distribution of all adversarial inputs found by the optimizer and α is a parameter that trades off conservatism for prediction performance (Trabucco et al., 2021b):

$$\min_\theta \quad \underbrace{\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[(\hat{f}_\theta(\mathbf{x}) - y)^2\right]}_{:= \text{ supervised loss}} + \alpha \underbrace{\left(\mathbb{E}_{\mathbf{x}\sim\mu}\left[\hat{f}_\theta(\mathbf{x})\right] - \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\hat{f}_\theta(\mathbf{x})\right]\right)}_{:= \text{ conservative regularizer}}. \tag{2}$$

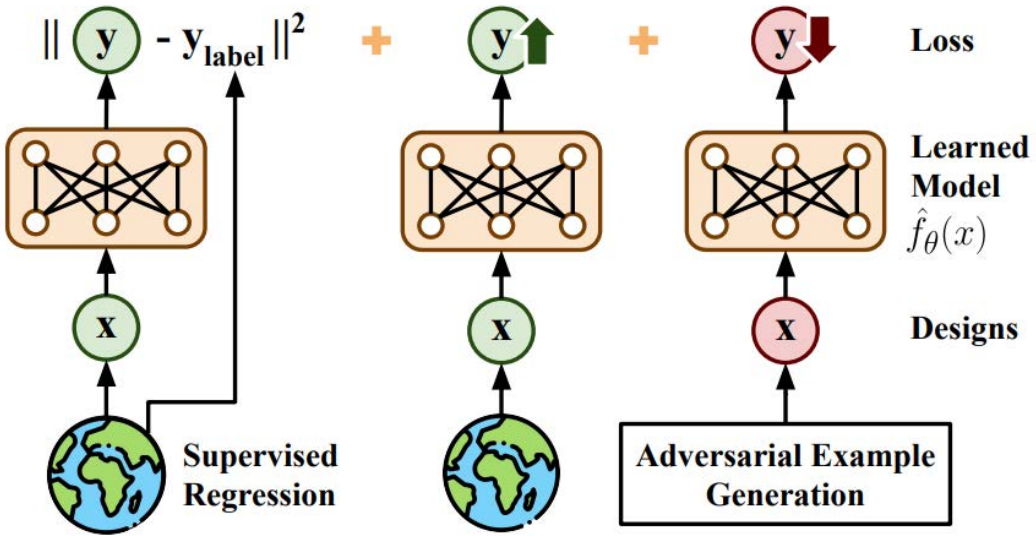

Figure 2: **Overview of COMs. (Trabucco et al., 2021b)** The COMs loss involves three terms: (a) the standard supervised regression loss, (b) a term that pushes up the model's predictions on in-dataset sequences, and (c) a term that pushes down the model's predictions on out-of-distribution sequences.

The general COMs approach is illustrated in Figure 2.

Overall, COMs provides us with a robust proxy model that when optimized against reliably produces high objective-value designs while avoiding producing out-of-distribution, low objective-value designs. In practice, COMs has outperformed other methods on a wide range of MBO problems (Trabucco et al., 2021b).

3 Applications

Recent successes in computational biology on supervised prediction tasks, such as protein structure prediction (Jumper et al., 2021; Baek et al., 2021; Yang et al., 2020), protein function prediction (Gligorijevic et al., 2019), and gene expression prediction (Avsec et al., 2021; Agarwal and Shendure, 2020; Reddy et al., 2023), have shown that deep neural networks can learn informative and complex features that are inaccessible to human domain experts from datasets of biological sequences. Naturally, this has inspired research into leveraging deep learning for biological sequence optimization and design. Success in computationally designing biological sequences can significantly reduce time and cost involved in experimentation and can accelerate innovation in the development of drugs and therapeutics.

In this paper, we frame biological sequence design as a black-box computational design problem, and we adapt COMs for and use it to tackle two important synthetic biology problems: (1) the problem of designing promoter sequences that are differentially-expressed and (2) the problem of recovering protein sequences from structure.

3.1 Differentially-Expressed Promoter Design

Gene therapy involves the treatment of disease through genetic modification of cells with missing or defective genetic material. In order to repair disease-associated cells and restore normal protein function, gene delivery technologies target affected cells with therapeutic genes, also known as transgenes.

There are two key challenges with current delivery methods: (a) the length of DNA that can be delivered is limited and (b) delivery methods usually don't perfectly target the desired

cell type, which can lead to dangerous side effects if the transgene is expressed in the wrong cell type (Sayed et al., 2022; Reddy et al., 2023).

Promoter design attempts to solve this problem. A promoter is a regulatory DNA sequence that determines where and when an adjacent gene is expressed. When designing gene therapies, promoters are included before the transgene in order to regulate expression. As described above, in order to design a transgene that effectively treats a disease, it is crucial that the transgene is compact and that it is only expressed in the target cell types (differential expression). Thus, effectively designing compact promoter sequences with specific expression profiles is critical for gene therapy (Reddy et al., 2023).

Recent research has found over 400 cell types in the human body; however, only a handful of cell type-specific promoters are known (Tabula Sapiens Consortium, 2022). Current approaches for engineering promoters with cell type specificity relies on domain knowledge approaches, such as tiling of cis-regulatory elements (CREs) or tandem repeats of transcription factor (TF) binding motifs (Miao et al., 2000; Nissim et al., 2017; Wu et al., 2019). While this has worked in some cell types, it is an unreliable and laborious process that can be significantly accelerated through the use of deep learning models and MBO techniques (Reddy et al., 2023).

3.2 Protein Sequence Recovery

The protein-folding problem is the problem of determining the three-dimensional atomic structure of a protein from its amino acid sequence. Protein sequence recovery, or protein design, is a well-known inverse protein-folding problem. More specifically, a successful protein design program should be able to recover wild-type protein sequences given their native backbone structures.

Protein sequences are combinations of 20 amino acid residues. Thus, there are 20^{100} or 2×10^{130} possible protein sequences of length 100. The vast majority of these sequences do not have well-defined structures. For proteins which fold into well-defined structures, Anfinsen's thermodynamic hypothesis of folding states that their structures are determined by their primary sequence. This motivates the inverse protein-folding problem, which aims to recover the lowest-energy amino acid sequence that fits a specific structural topology. This also extends to the important practical problem of designing de-novo proteins that fit a desired structure and function.

4 Design Choices & Methods

Applying data-driven offline MBO to any practical design problem requires answering several key design questions (Kolli et al., 2022):

1. **Dataset**: What dataset of designs are we going to use? What is the design space? What is the objective space? The latter questions are important because the design space and objective space do not necessarily directly correspond to the inputs x and targets y in the dataset and may involve additional processing.
2. **Proxy Model**: What proxy model are we going to use? What input will the model take? What will it output? The latter questions are important because the proxy model may not directly take in the designs and output the objectives. What model architecture will we use? What data from the dataset will the proxy model be trained on?
3. **Optimizer**: What optimizer will we use for adversarial sampling and the final design phase? What optimization hyperparameters will we use?
4. **Evaluation**: How can we assess hyperparameters, model training decisions, and generated sequences in the absence of wet lab evaluation? If we use oracle models, what architecture will we use? What data will we use to train the oracle models? And, how will we ensure that the oracle models are reliable and diverse?

4.1 Dataset

Promoter Design.

We use the gene expression dataset constructed by Reddy et al. (2023) by experimentally measuring promoter-driven expression of 20,000 promoters of length 250 base pairs (bp) in 3 immune cell lines: Jurkat, K-562, and THP-1.

The final dataset contains expression measurements for 17,104 promoters of length 250 base pairs (bp) for which there was adequate experimental data. Out of the 20,000 promoters whose expression was experimentally measured, approximately 50% of the promoters are derived from promoters of differentially expressed endogenous genes. Another $\sim 40\%$ are designed by tiling known and de-novo motifs that were discovered to be enriched in the promoters of differentially expressed endogenous genes by HOMER (Heinz et al., 2010), a motif detection tool. The final $\sim 10\%$ are derived from promoters of highly expressed endogenous genes that are not necessarily differentially expressed (Reddy et al., 2023).

Our design space is the discrete space of all promoters of length 250 base pairs (adenine, thymine, cytosine, and guanine). The targets y in our dataset are length-3 vectors representing the expression values in each of the three cell types, Jurkat, K-562, and THP-1. Since we want to optimize for differentially expressed promoters, we must add an additional transformation to this target to extract a score that represents the level of differential expression. We experimented with two such transformations:

1. Linear Combination: the expression in the cell type you want to upregulate expression in minus the average expression in the cell types you want to downregulate expresion in. For example, if we want to optimize for differential expression in Jurkat, the objective we optimize for would be:

$$y[\text{Jurkat}] - 0.5 \times y[\text{K-562}] - 0.5 \times y[\text{THP-1}]$$

2. Log-Fold Change: the logarithm of the ratio between the expression in the cell type you want to upregulate expression in and the average expression in the cell types you want to downregulate expresion in. For example, if we want to optimize for differential expression in Jurkat, the objective we optimize for would be:

$$\log_2 \left(\frac{y[\text{Jurkat}]}{0.5 \times y[\text{K-562}] + 0.5 \times y[\text{THP-1}]}) \right)$$

Protein Design.

We use the dataset curated by Yang et al. (2020) consisting of 15,051 protein chains collected from the Protein Data Bank (PDB). The dataset was constructed by collecting 94,962 X-ray entries with resolution ≤ 2.5Å (PDB snapshot as of 1 May 2018), extracting all protein chains with at least 40 residues, and finally removing redundancy at 30% sequence identity cutoff, resulting in a set of 16,047 protein changes with average length of 250 amino acids. Multiple sequence alignments (MSAs) were subsequently collected for all of the corresponding primary sequences using an iterative procedure, and only chains with at least 100 sequence homologs were included in the final dataset (Yang et al., 2020).

The inputs x in the dataset are MSAs, which are represented as matrices of amino acids plus one padding character with shape $[N, L]$ where N is the number of sequences in the MSA and L is the sequence length.

For each input, the targets y in the dataset consists of four tensors that collectively represent the interresidue geometries (distances and orientations) for all residue pairs (visualized in Figure 3) (Yang et al., 2020).

1. d ($C_\beta - C_\beta$ distances): a symmetric three-dimensional tensor of shape $[L, L, 37]$, where L is the length of the sequences in the MSA and the 37 represents the fact that the distance range (2 to 20 Å) is binned into 36 equally spaced segments, 0.5 Å each, plus one bin indicating that residues are not in contact.

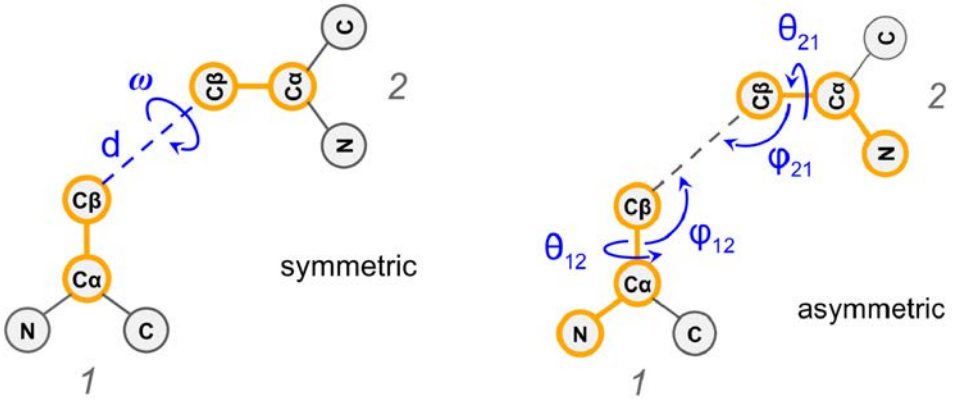

Figure 3: **Representation of the rigid-body transform from one residue to another using angles and distances (Yang et al., 2020)**

2. ω (rotation along the virtual axis connecting the C_β atoms of the 2 residues): a symmetric three-dimensional tensor of shape $[L, L, 25]$, where L is the length of the sequences in the MSA and the 25 represents the fact that the angle range (0° to 360°) is binned into 24 equally spaced segments, 15° each, plus one bin indicating that residues are not in contact.
3. θ (direction of the C_β atom of one residue in a reference frame centered on the other residue): an asymmetric three-dimensional tensor of shape $[L, L, 25]$, where L is the length of the sequences in the MSA and the 25 represents the fact that the angle range (0° to 360°) is binned into 24 equally spaced segments, 15° each, plus one bin indicating that residues are not in contact.
4. φ (direction of the C_β atom of one residue in a reference frame centered on the other residue): an asymmetric three-dimensional tensor of shape $[L, L, 13]$, where L is the length of the sequences in the MSA and the 13 represents the fact that the angle range (0° to 180°) is binned into 12 equally spaced segments, 15° each, plus one bin indicating that residues are not in contact.

When it comes to the sequence recovery problem, we will restrict our attention to optimizing the primary sequence of each MSA, so the design space is the discrete space of proteins of variable lengths. It is also possible to design entire MSAs, but this is more complex and less practically useful.

Since we want to design sequences that fold into a desired structure, we add an additional transformation on top of the dataset targets described above in order to define our objective function. More specifically, the objective is the negative of the sum of the cross-entropy loss of the four target tensors with the corresponding interresidue geometry tensors of the desired structure. Mathematically, if we want to design sequences that fold into a desired structure represented by interresidue geometry tensors $d^*, \omega^*, \theta^*, \varphi^*$, our objective would be:

$$\begin{aligned} -(&\text{CrossEntropy}(y[d], d^*) \\ &+ \text{CrossEntropy}(y[\omega], \omega^*) \\ &+ \text{CrossEntropy}(y[\theta], \theta^*) \\ &+ \text{CrossEntropy}(y[\varphi], \varphi^*)) \end{aligned}$$

4.2 Proxy Model

Promoter Design.

For the proxy model, we use the model architecture, proposed by Reddy et al. (2023), consisting of a 4-layer CNN, followed by 5 Transformer layers, and finally followed by three linear output heads, one for each target cell type. The architecture is illustrated in Figure 4.

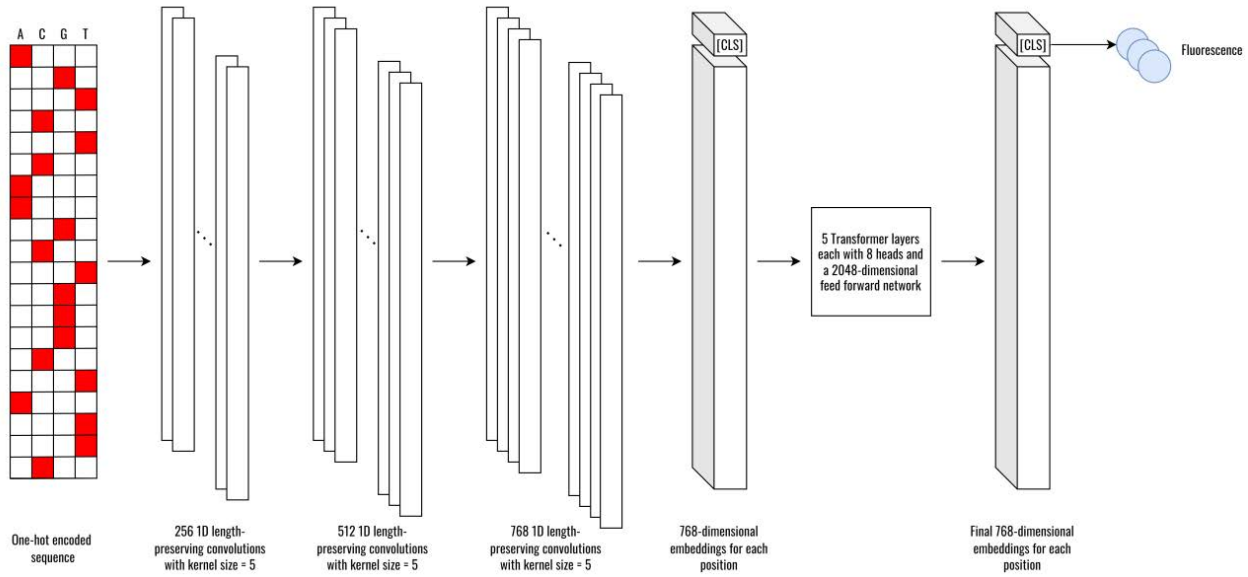

Figure 4: **The proxy model architecture for the promoter design problem (Reddy et al., 2023).**

The proxy model receives a promoter sequence as input and outputs a length-3 vector representing the expression values in each of the three target cell types, Jurkat, K-562, and THP-1. In other words, the model receives the inputs x from the dataset and outputs predicted targets $\hat{y}$ that match the format of the dataset's targets y.

The proxy model is initially pretrained on existing promoter-driven expression data from massively parallel reporter assays (MPRAs) in other cell types. This is followed by finetuning on the promoter expression dataset described in Section 4.1 in the target cell types. It has been shown that finetuning followed by pretraining improves prediction performance by 6 - 12% in all three target cell types when compared to directly training on the promoter expression dataset (Reddy et al., 2023).

The proxy model only undergoes COMs training during the finetuning phase.

Protein Design.

For the proxy model, we use a smaller version of the deep residual-convolutional network which was used successfully in trRosetta, an efficient algorithm for protein structure prediction, by Yang et al. (2020). The architecture is illustrated in Figure 5. In this paper, we use 31 residual blocks as opposed to 61.

The proxy model is capable of making predictions for MSAs or single protein sequences. However, the model does not directly receive the MSA or the primary protein sequence. Instead, the MSA or primary protein sequence undergoes several pre-processing and feature extraction steps, and in both cases, the output of the pre-processing and the input to the proxy model is a feature tensor of shape $[L, L, 526]$, where L is the sequence length. The network applies a sequence of 2D convolutions to this tensor to ultimately predict 4 histograms: 1 distance histogram (d) of shape $[L, L, 37]$ and 3 angle histograms (ω, θ, and φ) of shapes $[L, L, 25]$, $[L, L, 25]$, and $[L, L, 13]$, respectively.

More specifically, the proxy model first transforms the number of input features down to 64 (2D convolution with filter size 1) and subsequently applies 31 basic residual blocks with dilations. Each block consists of convolution operations using 64 3×3 filters and ELU activations. Dilations cycle through 1, 2, 3, 8, and 16. Finally, after the last residual block the network uses 4 heads, one per each objective, which consists of a single 2D convolution followed by softmax activation (Yang et al., 2020).

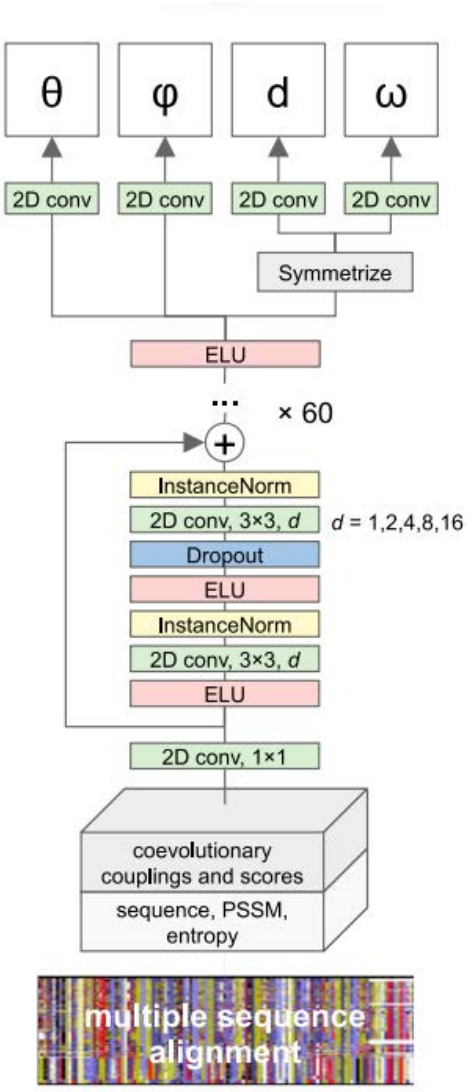

Figure 5: **The trRosetta model architecture (Yang et al., 2020).**

Similar to the promoter design problem, the proxy model undergoes two separate phases of training:

1. The first phase of training follows the protocol used for trRosetta in Yang et al. (2020). More specifically, the proxy model receives full MSAs as input and outputs the 4 interresidue geometry histograms (d, ω, θ, and φ). The model is trained to minimize the sum of the 4 individual categorical cross-entropy losses. The model is trained for 100 epochs with each epoch running through the whole training set.
2. The second phase of training involves performing COMs training on the proxy model with the model receiving only primary sequences, as opposed to MSAs, as input.

4.3 Optimizer & Modifications to COMs

When designing a suitable optimizer for COMs, it's important to note that both promoters and proteins are in discrete space. Consequently, a naive gradient ascent optimizer will not work. In this paper, we experimented with three different types of optimizers:

1. **Gradient Ascent Optimizer.** The general procedure to optimize a sequence x using the gradient ascent optimizer is:
   (a) Pick the number of outer rounds N_{outer}, the number of inner rounds N_{inner}, and a learning rate η.
   (b) Let $x^{(0)} = x$.
   (c) For each outer round i in (0, N_{outer}):
      i. Transform $x^{(i)}$ into a one-hot tensor $\tilde{x}^{(i)}$.
      ii. Perform N_{inner} gradient ascent steps on $\tilde{x}^{(i)}$ in continuous space:
      $$\tilde{x}^{(i)}_{k+1} \leftarrow \tilde{x}^{(i)}_{k} + \eta \nabla_x \hat{f}_\theta(x)|_{x=\tilde{x}^{(i)}_k}, \text{ for } k \in [0, N_{\text{inner}} - 1]$$
      iii. Map $\tilde{x}^{(i)}_{N_{\text{inner}}}$ back to discrete space, e.g. using an argmax.
      iv. Let $x^{(i+1)}$ equal the resulting tensor.
   (d) The final design is $x^{(N_{\text{outer}})}$.

   Note that the above procedure is deterministic.
2. **Single-Site Mutation Discrete Optimizer.** The general procedure to optimize a sequence x using the single-site mutation discrete optimizer is:
   (a) Pick the number of rounds N.
   (b) Let $x^{(0)} = x$.
   (c) For each round i in (0, N):
      i. Sample a random position in the sequence $x^{(i)}$ and sample a random monomer (amino acid or nucleotide).
      ii. Let $x^{(i)}_{\text{mut}}$ be the sequence $x^{(i)}$ with the mutation selected in the above step.
      iii. Query the proxy model $\hat{f}_\theta$ with $x^{(i)}_{\text{mut}}$ and accept the mutation if it improves the objective value of the sequence. Mathematically, if $\hat{f}_\theta(x^{(i)}_{\text{mut}}) > \hat{f}_\theta(x^{(i)})$, then we let $x^{(i+1)} = x^{(i)}_{\text{mut}}$; otherwise, we let $x^{(i+1)} = x^{(i)}$.

   Note that this optimizer uses a stochastic procedure.
3. **Rate-Based Discrete Optimizer.** Rather than selecting a single position in the sequence to mutate as we do in the single-site mutation discrete optimizer, we use a "mutation rate" to independently select positions to mutate in the sequence. In essence, each position in the sequence has a probability of being mutated based on the rate, and at each step of the optimizer, we could potentially mutate multiple positions simultaneously. Similar to the single-site mutation discrete optimizer, we select the mutations randomly, and at each optimization step, we only accept the new sequence if it performs better than the original sequence according to the proxy model.

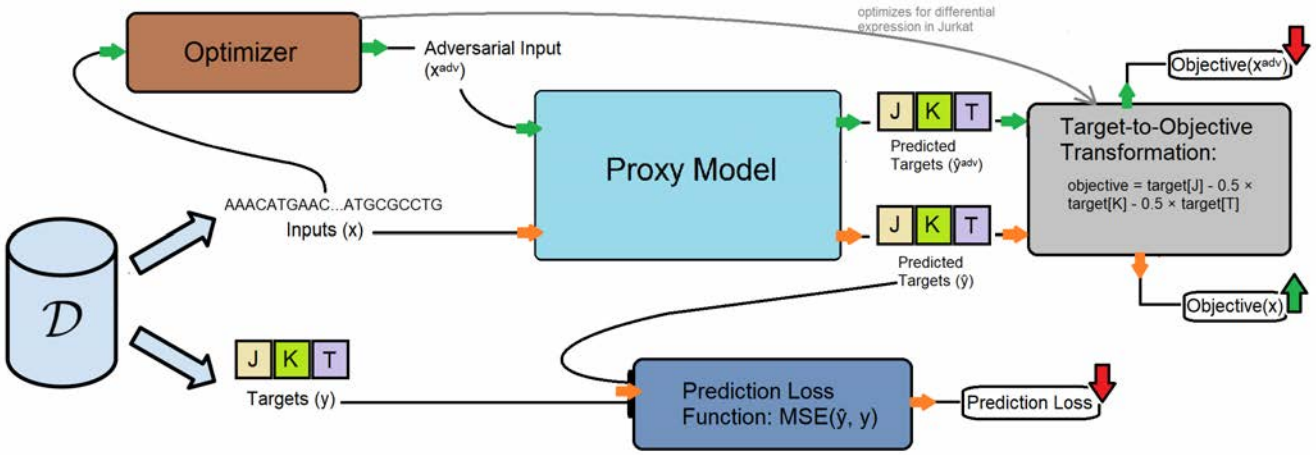

Figure 6: **The complete COMs workflow for the promoter design problem.** The dataset contains pairs of promoter sequences and length-3 vectors containing expression values for Jurkat, K-562, and THP-1. The design space is the space of promoter sequences, and the objectives are a linear combination of the length-3 vector of expression values. The model receives promoter sequences as input and outputs length-3 vectors with predicted expression values for Jurkat, K-562, and THP-1. Note that the above illustration only shows adversarial sampling and objective penalization for differential expression in Jurkat; in practice, we do this procedure three times, once for differential expression in each of the three cell types.

4. **Motif-Insertion Discrete Optimizer.** The motif-insertion discrete optimizer is similar to the single-site mutation discrete optimizer, with the main difference being that we sample motifs (rather than monomers) from a previously-collected set of motifs and mutate randomly selected contiguous subsequences (rather than single-sites) of the promoter or protein we are trying to optimize. For example, for the promoter design project, we may want to use transcription factor binding motifs.

We can design more complex optimizers that are geared towards specific problem settings using the four basic optimizers described above as building blocks.

Promoter Design.

For the promoter design problem, during training (for adversarial sampling), our optimizer interleaves rounds of the gradient ascent optimizer and rounds of a single-site mutation discrete optimizer. This provides an efficient stochastic optimization procedure that incorporates gradient information from the proxy model. During the design phase, we experiment with all four optimization procedures. The actual parameters we used for optimization during the training phase and the final design generation phase are outlined in Section 5.

The complete COMs promoter design workflow detailing the dataset, the proxy model, and the optimizer is illustrated in Figure 6.

Additionally, we make a slight modification to the traditional COMs workflow. Rather than sampling a single adversarial example as illustrated in Figure 6 where we only optimize for differential expression in Jurkat, we sample three adversarial examples, one for differential expression in each of the 3 cell types. Consequently, the final COMs regularizer term is an average of the 3 COMs losses for each cell type. Conceptually, by doing this, we are training a model that can reliably produce promoters that are differentially expressed in any of the 3 cell types.

Protein Design.

When it comes to the problem of protein sequence recovery, there are two general approaches to optimization and design: (a) start with a random sequence and optimize, (b) start with a known protein sequence that is only mutated in a few positions and optimize.

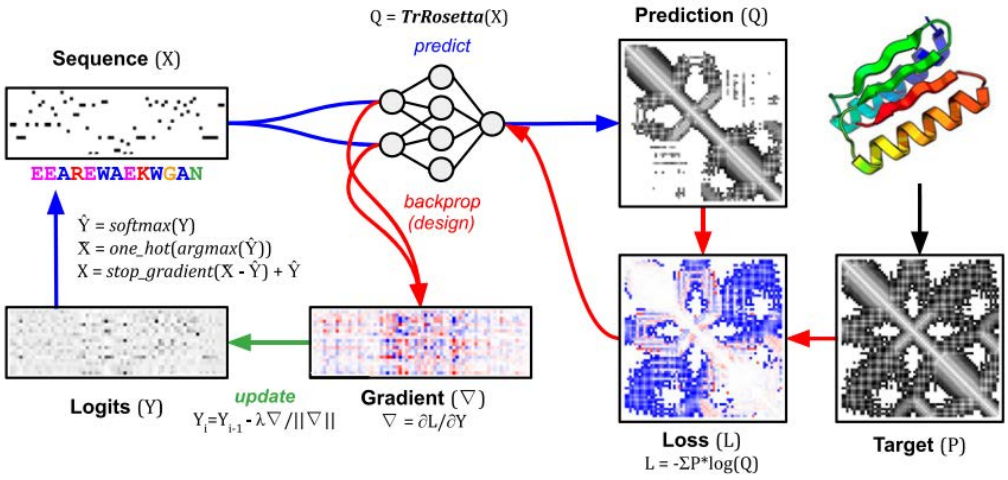

Figure 7: **Overview of protein sequence design using trRosetta as the proxy model and a simple gradient-based optimizer (Norn et al., 2021).**

Furthermore, unlike the standard COMs setting, we can optimize sequences to fold into a plethora of different structures. In other words, there are a wide range of different objectives we can optimize for.

In order to accommodate the above features of the protein design problem, we first use two distinct optimizers, one for "global" adversarial sampling, which starts with a random sequence and optimizes in order to design a sequence that folds into the desired structure, and one for "local" adversarial sampling, which starts with a slightly-mutated version of a known protein sequence and optimizes in order to design a sequence that folds into the desired structure. The resulting COMs regularizer term is the average of the COMs loss for the local adversarial sample and the COMs loss for the global adversarial sample. Second, during each training step, the desired structure we optimize for is just the target structure we sample from the dataset at that step. This enables us to develop a general proxy model that can be used to generate protein sequences for any target structure.

A simple gradient-based optimization procedure is illustrated in Figure 7. Our optimizers are more complex. At a high level, the "global" optimizer procedure consists of rounds of a gradient ascent optimizer followed by rounds of a single-site mutation discrete optimizer, and the "local" optimizer is a rate-based discrete optimizer. The actual parameters we use for both optimizers during the training phase and the final design generation phase depend on the specific task and are outlined in Section 5.

The complete COMs protein design workflow detailing the dataset, the proxy model, and the optimizer is illustrated in Figure 8.

4.4 Evaluation

Ideally, biological sequences that are computationally designed should be experimentally evaluated in the wet lab. However, there are purely-computational methods for evaluating designs when a wet lab is inaccessible, when we are trying to quickly sanity check algorithms and iterate, or when we want to filter designs to select a batch of the best ones to experimentally evaluate.

Promoter Design.

For the promoter design problem, we use an ensemble of learned models, which we call "oracle models", to evaluate our algorithm and the designs it produces. Our ensemble consists of 18 oracle models, all of which have the same architecture as the proxy model described in

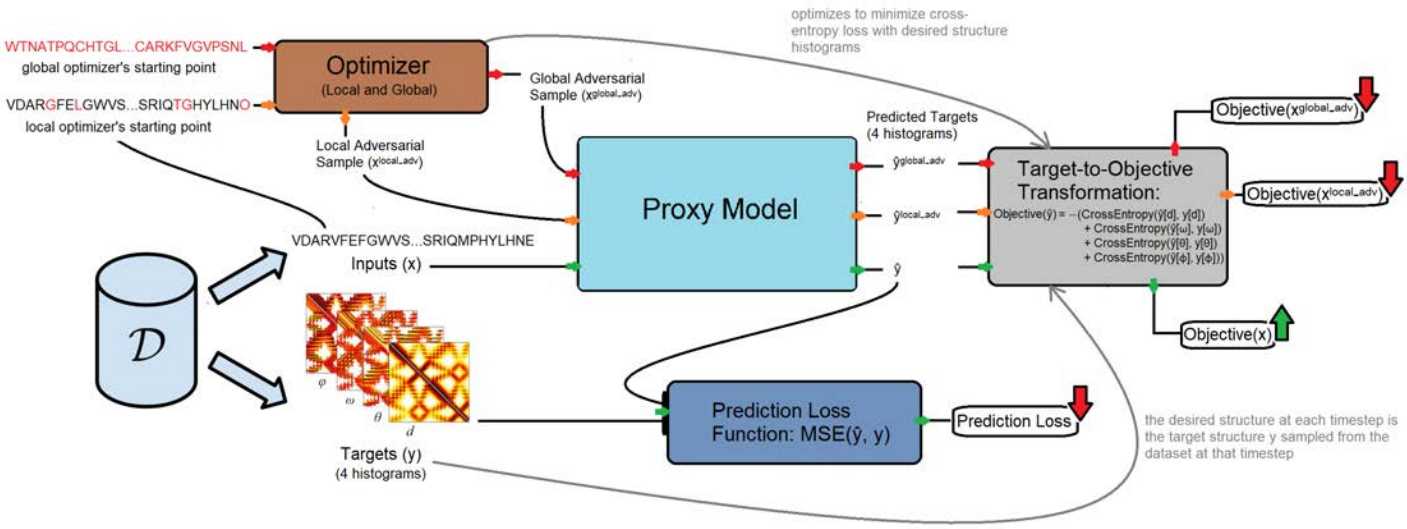

Figure 8: **The complete COMs workflow for the protein design problem.** The dataset contains pairs of protein sequences and 4 structure histograms (d, ω, θ, φ) that represent the interresidue geometries. The design space is the space of protein sequences, and the objective is the negative of the sum of the cross entropy losses of the predicted structure histograms and the desired structure histograms. The model receives protein sequences as input and outputs 4 structure histograms with predicted interresidue geometries.

Section 4.2 except that we modify the architecture of the three heads. More specifically, each head is a fully-connected network, and we try three different values for the number of layers (2, 4, and 8), three different values for the hidden dimension (512, 1024, and 2048), and two different activation functions (gelu and tanh). As was the case with the proxy model, the backbone of all 18 oracle models is initialized with weights from a pretraining phase which involves training on MPRA data, and the models are subsequently finetuned on the promoter expression dataset (Reddy et al., 2023).

Tuning of the proxy and oracle models' architecture, hyperparameters, and training decisions is done using the validation error on a held-out dataset with the goal of hitting the sweet spot between overfitting and underfitting. The main validation metrics we use to analyze prediction performance of the models is the Spearman rank correlation.

Our oracle models have two key desirable properties: (a) the oracle models are diverse as we can see in Table 1 and (b) the oracle ensemble performs significantly better than the individual oracle models or the proxy model on the prediction task as we can see in Table 2.

Protein Design.

The unique benefit of the protein sequence recovery task is that evaluation is straightforward and does not require any additional modeling. We have the wild-type sequence for all the desired structures we use as targets for design. Consequently, designs that were optimized for some desired structure are simply evaluated by measuring the sequence identity with its wild-type sequence. This metric is reported as a percentage and reflects how successful our algorithm is at recovering the optimal sequence for a provided structure.

5 Tasks & Results

For both the promoter design and protein design problems, we compare performance of COMs with varying levels of the conservatism parameter α and the standard offline MBO approach (outlined in Section 2.3), which trains a proxy model using standard supervised

Data Split	Cell Type	Average Variance	Average MSE	Oracle Ensemble MSE
Train	THP-1	0.284	0.451	0.167
Train	Jurkat	0.573	0.842	0.269
Train	K-562	0.577	0.767	0.189
Validation	THP-1	0.262	0.661	0.399
Validation	Jurkat	0.524	1.428	0.903
Validation	K-562	0.464	1.199	0.735
Test	THP-1	0.259	0.681	0.422
Test	Jurkat	0.508	1.322	0.814
Test	K-562	0.462	1.224	0.762

Table 1: **Diversity of oracle models.** The average variance is the average (across sequences) of the variance (across the oracle models) in model predictions. The average mean-squared error (MSE) is the average (across the oracle models) MSE metric. The oracle ensemble MSE is the MSE achieved by the oracle ensemble. Relative to the MSE's, the variance across models is significant, which suggests that there is a good deal of diversity in the oracle models.

Cell Type	Metric	Individual Oracle (Average)	Individual Oracle (Best)	Oracle Ensemble
THP-1	Spearman Correlation	0.520	0.553	**0.562**
THP-1	Pearson Correlation	0.585	0.624	**0.635**
THP-1	MSE	0.681	**0.403**	0.422
Jurkat	Spearman Correlation	0.656	0.676	**0.690**
Jurkat	Pearson Correlation	0.676	0.701	**0.716**
Jurkat	MSE	1.322	**0.803**	0.814
K-562	Spearman Correlation	0.673	0.694	**0.709**
K-562	Pearson Correlation	0.670	0.691	**0.708**
K-562	MSE	1.224	0.825	**0.762**

Table 2: **Prediction performance of the oracle ensemble in comparison to the individual oracle models.** All the above results are computed on a held-out test set. The "Individual Oracles" columns contains the average and best performance metric across the individual oracle models. The "Oracle Ensemble" column contains the performance metric achieved by the ensemble model. The oracle ensemble performs significantly better than the average of the individual oracles on every cell type and every metric and even performs better than the best individual oracle on all but two combinations of cell type and metric.

regression, on multiple problem-specific tasks. Note that the standard offline MBO approach can be viewed as COMs with α equal to zero.

5.1 Promoter Design

Once we train a proxy model on our data using the design algorithm of choice, we use our optimizer and the proxy model to generate a dataset of designs from the promoter expression dataset by producing a single design for each promoter in the dataset. In total, we actually generate three such datasets of designed sequences, one that optimizes for differential expression in each of the 3 cell types, Jurkat, K-562, and THP-1. We evaluate the designs using the oracle ensemble model, as described in Section 4.4.

Based on the oracle ensemble's predictions, we use two different tasks to evaluate the performance of our design algorithm:

1. The first task measures the ability of the algorithm to optimize the level of differential expression of any given promoter sequence. We use two performance metrics for this task: (a) "average improvement", which gives us that average difference in objective value (e.g. differential expression in Jurkat) between the designed sequence and

original sequence in the dataset that we started optimization from and (b) "success rate", which gives us the proportion of sequences from the dataset for which our design algorithm was able to improve the objective value.

2. The second task measures how good the top designed sequences according to the proxy model actually are. First, we begin by selecting the top $N = 128$ designs according to the proxy model's predicted objective values. Next, we report the 100th percentile ensemble model predicted objective value as well as the 50th percentile ensemble model predicted objective value on this set. This evaluation protocol is reasonable as it is usually followed in many real-world MBO problems, where a set of designs are produced and the best performing ones are used (Trabucco et al., 2021a;b).

The optimizer we use to generate adversarial designs during COMs training and to design the final designs for evaluation uses 5 rounds of optimization, each of which consists of 15 steps of the gradient-ascent optimizer followed by 15 steps of a single-site mutation discrete optimizer. Furthermore, in our experiments, we use the linear combination function outlined in Section 4.1 to compute the objective from the length-3 vector of expression values.

We report the performance of 4 COMs models with varying levels of the α hyperparameter on the first task in Table 3, and we report the performance of the same 4 proxy models on the second task in Table 4. We can also visualize the level of differential expression in our designs when compared to the original dataset in the scatter plots in Figure 9.

Cell Type	Alpha	Average Improvement	Success Rate (%)
THP-1	$\alpha = 0.0$	**0.570**	**98.097**
THP-1	$\alpha = 0.01$	0.552	97.116
THP-1	$\alpha = 0.03$	0.524	95.072
THP-1	$\alpha = 0.1$	0.469	91.124
Jurkat	$\alpha = 0.0$	0.414	89.603
Jurkat	$\alpha = 0.01$	**0.432**	**90.431**
Jurkat	$\alpha = 0.03$	0.426	88.910
Jurkat	$\alpha = 0.1$	0.382	85.556
K-562	$\alpha = 0.0$	0.520	88.146
K-562	$\alpha = 0.01$	0.563	90.090
K-562	$\alpha = 0.03$	0.649	92.951
K-562	$\alpha = 0.1$	**0.723**	**95.195**

Table 3: **Performance of 4 design algorithms on task 1 of the promoter design problem.** The cell type indicates which cell type the designs were optimized to maximize differential expression in. Note that COMs with α equal to zero is equivalent to the standard offline MBO approach.

5.2 Protein Design

We evaluate our design algorithm on a small dataset of 11 de-novo protein sequences and their corresponding structures from PDB. The PDB id's of the proteins are: 6CZG, 6CZH, 6CZI, 6CZJ, 6D0T, 6DG5, 6DG6, 6MRR, 6MRS, 6MSP, and 6NUK. More specifically, for each de-novo protein sequence x and its corresponding structure y, we produce two designed sequences, one for each of the two tasks described below, that are optimized to fold to the structure y.

We evaluate performance of our design algorithm on two tasks:

1. The first task measures sequence recovery when given a slightly perturbed version of the protein sequence $\tilde{x}$ and the corresponding target structure y. In this case, we produce a design by starting optimization at $\tilde{x}$, and we use the "local" optimizer described in Section 4.3. In our experiments, we produce $\tilde{x}$ from x by independently mutating positions of x with probability 0.1 and randomly selecting the new amino acid to insert into the mutated positions.

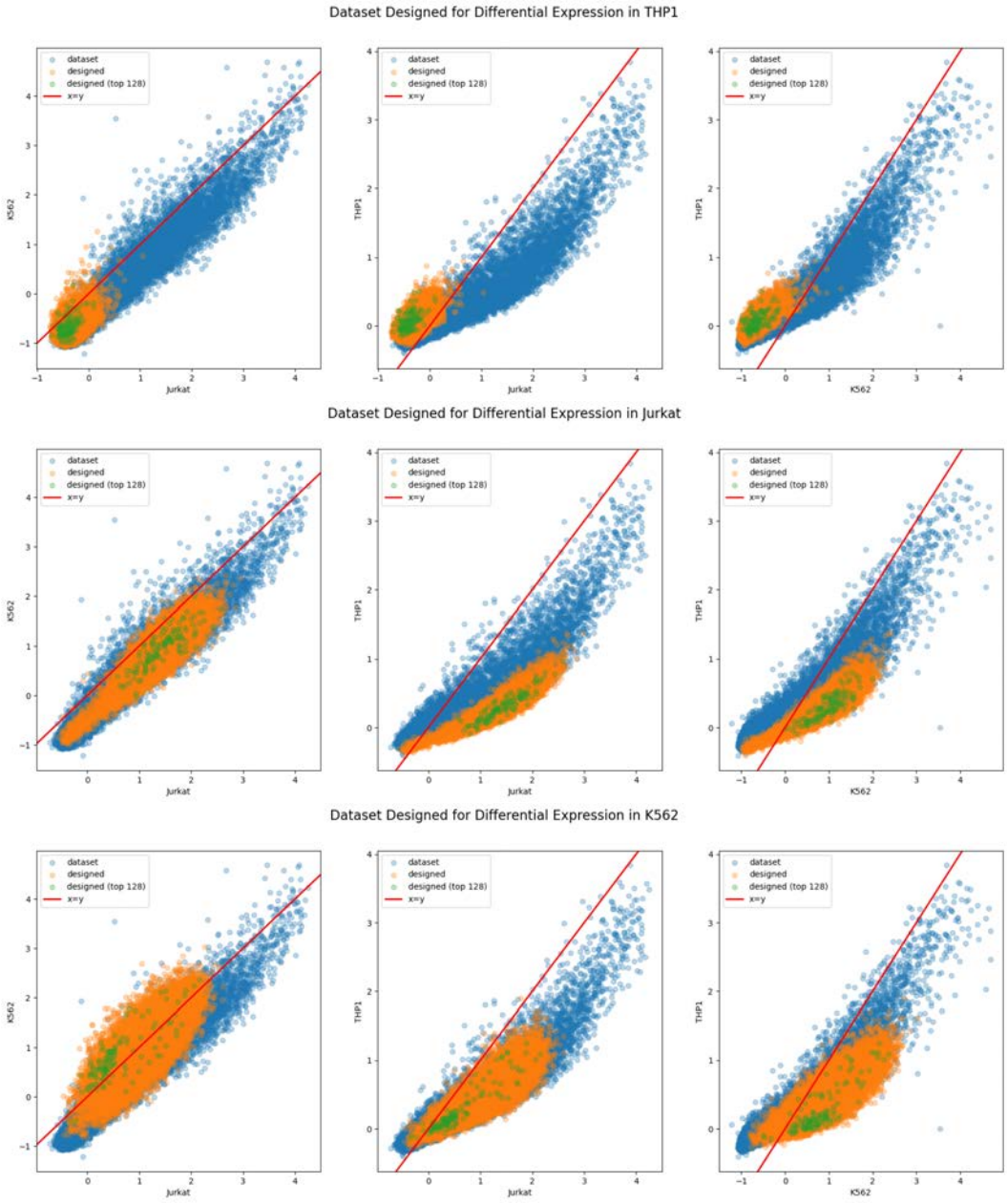

Figure 9: Scatter plot of the oracle ensemble's predictions on the promoters in the original dataset and the promoters in the three datasets which were designed using the COMs proxy model with α equal to 0.01. We also plot the oracle ensemble's predictions on the top 128 designed promoters in each designed dataset according to the proxy model.

Cell Type	Alpha	100th Percentile	50th Percentile
THP-1	$\alpha = 0.0$	0.786	**0.626**
THP-1	$\alpha = 0.01$	**0.876**	0.606
THP-1	$\alpha = 0.03$	0.777	0.602
THP-1	$\alpha = 0.1$	0.820	0.568
Jurkat	$\alpha = 0.0$	1.346	0.856
Jurkat	$\alpha = 0.01$	1.472	0.927
Jurkat	$\alpha = 0.03$	1.462	1.005
Jurkat	$\alpha = 0.1$	**1.496**	**1.031**
K-562	$\alpha = 0.0$	0.992	0.250
K-562	$\alpha = 0.01$	1.060	0.395
K-562	$\alpha = 0.03$	**1.578**	0.506
K-562	$\alpha = 0.1$	1.517	**0.514**

Table 4: **Performance of 4 design algorithms on task 2 of the promoter design problem.** The cell type indicates which cell type the designs were optimized to maximize differential expression in. Note that COMs with α equal to zero is equivalent to the standard offline MBO approach.

2. The second task measures sequence recovery when we are only given the target structure y. In this case, we produce a design by starting optimization at a random protein sequence, and we use the "global" optimizer described in Section 4.3.

For both tasks, we measure performance based on what percentage of the original wild-type protein sequence x we can recover in our design as described in Section 4.4.

In Table 5, we report performance of two proxy models on both tasks: (a) standard fully-trained trRosetta model with no COMs finetuning (i.e. just phase 1 of the training procedure described in Section 4.2) and (b) a model with both the standard trRosetta training and COMs finetuning, using local and global adversarial sampling with α equal to 0.5.

For the COMs proxy model, as described in Section 4.3, we use two optimizers during training, one that samples "local" adversarial inputs and one that samples "global" adversarial inputs. The "local" optimizer is a rate-based discrete optimizer that performs 150 rounds with a"mutation rate" of 5%. The "global" optimizer begins by running the gradient ascent optimizer with $N_{\text{outer}} = 1, N_{\text{inner}} = 100$, learning rate $\eta = 1$, and unit normalization of gradients followed with 100 rounds of a rate-based discrete optimizer with a "mutation rate" of 5%.

Furthermore, on task 1, the "local" optimizer that we use to generate protein sequences is a rate-based discrete optimizer that performs 200 rounds with a "mutation rate" of 5%. On task 2, the "global" optimizer that we use to generate designs is a gradient ascent optimizer with $N_{\text{outer}} = 1, N_{\text{inner}} = 300$, learning rate $\eta = 2.5$, and unit normalization of gradients.

Task	Model	Sequence Recovery (%)
Task 1: Local	Standard trRosetta	49.159
Task 1: Local	COMs ($\alpha = 0.5$)	**57.234**
Task 2: Global	Standard trRosetta	11.807
Task 2: Global	COMs ($\alpha = 0.5$)	**15.672**

Table 5: **Performance of 2 design algorithms on both protein design tasks.** The sequence recovery measurements are averages across 4 trials.

In Figure 10, we visualize some example optimization trajectories, which illustrate the objective described in Section 4.1 and the sequence recovery rate during each step of optimization, for task 1. Finally, in Figure 11, we visualize the predicted structure histograms for a few of the designs generated by the COMs proxy model in task 1 and the corresponding target structure histograms that the designs were optimized for.

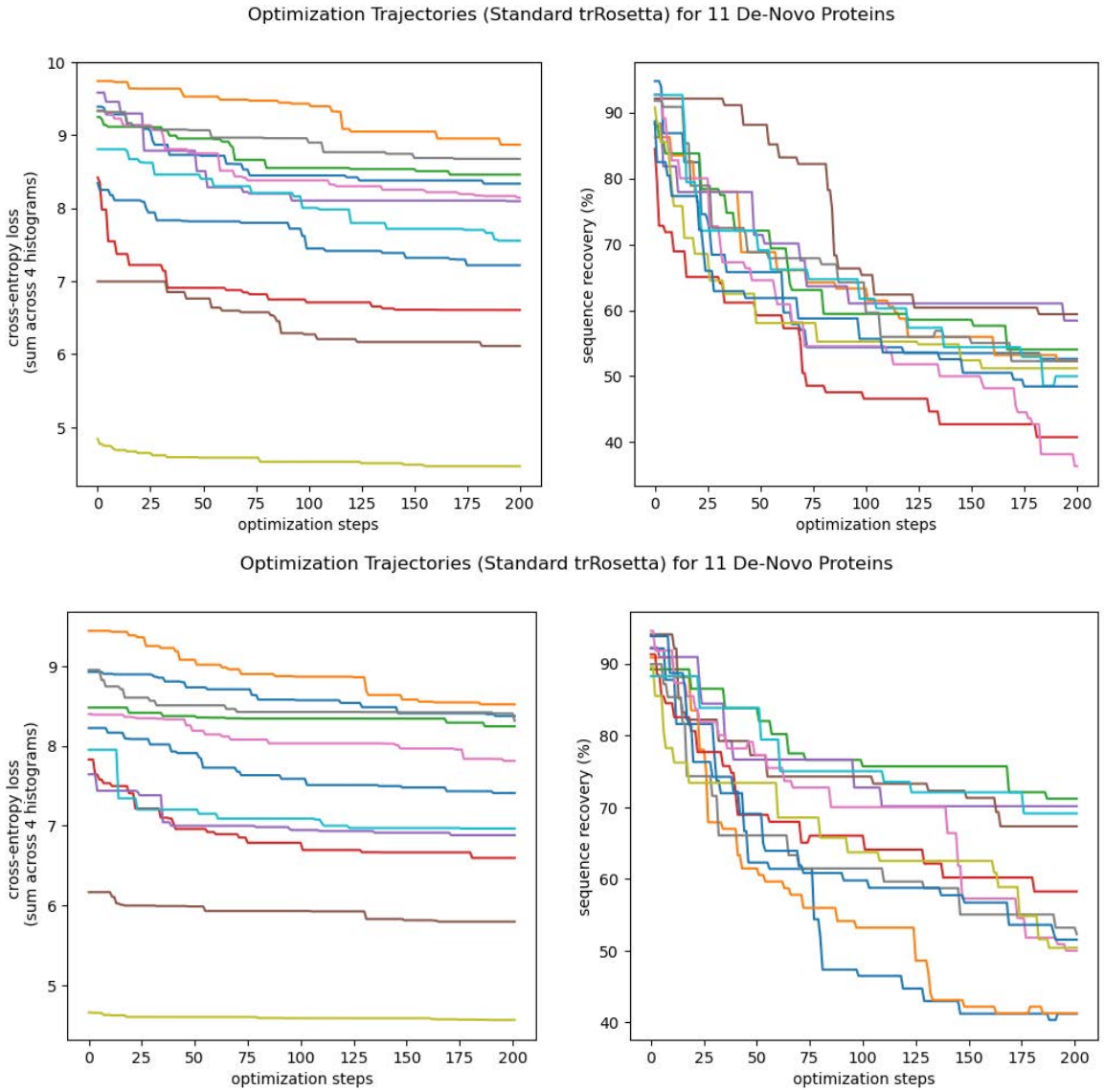

Figure 10: **Example optimization trajectories for the protein design algorithm's "local" optimizer.**

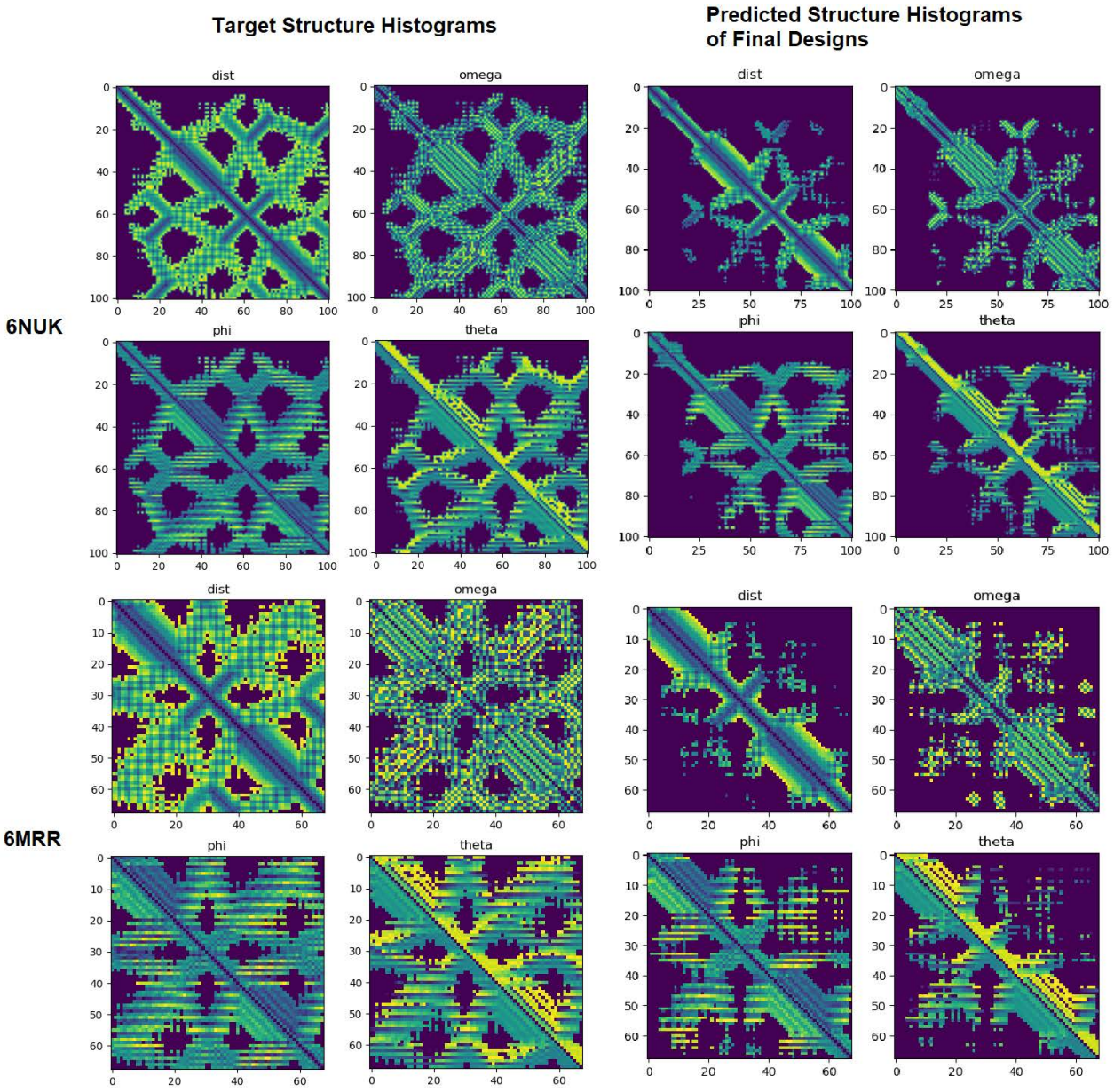

Figure 11: Target structure histograms and corresponding predicted structure histograms for designed sequences from task 1 (start with a slightly-perturbed version of the wildtype sequence and optimize to fold to the target structure using the "local" optimizer) of the de-novo proteins with PDB ID's 6NUK and 6MRR.

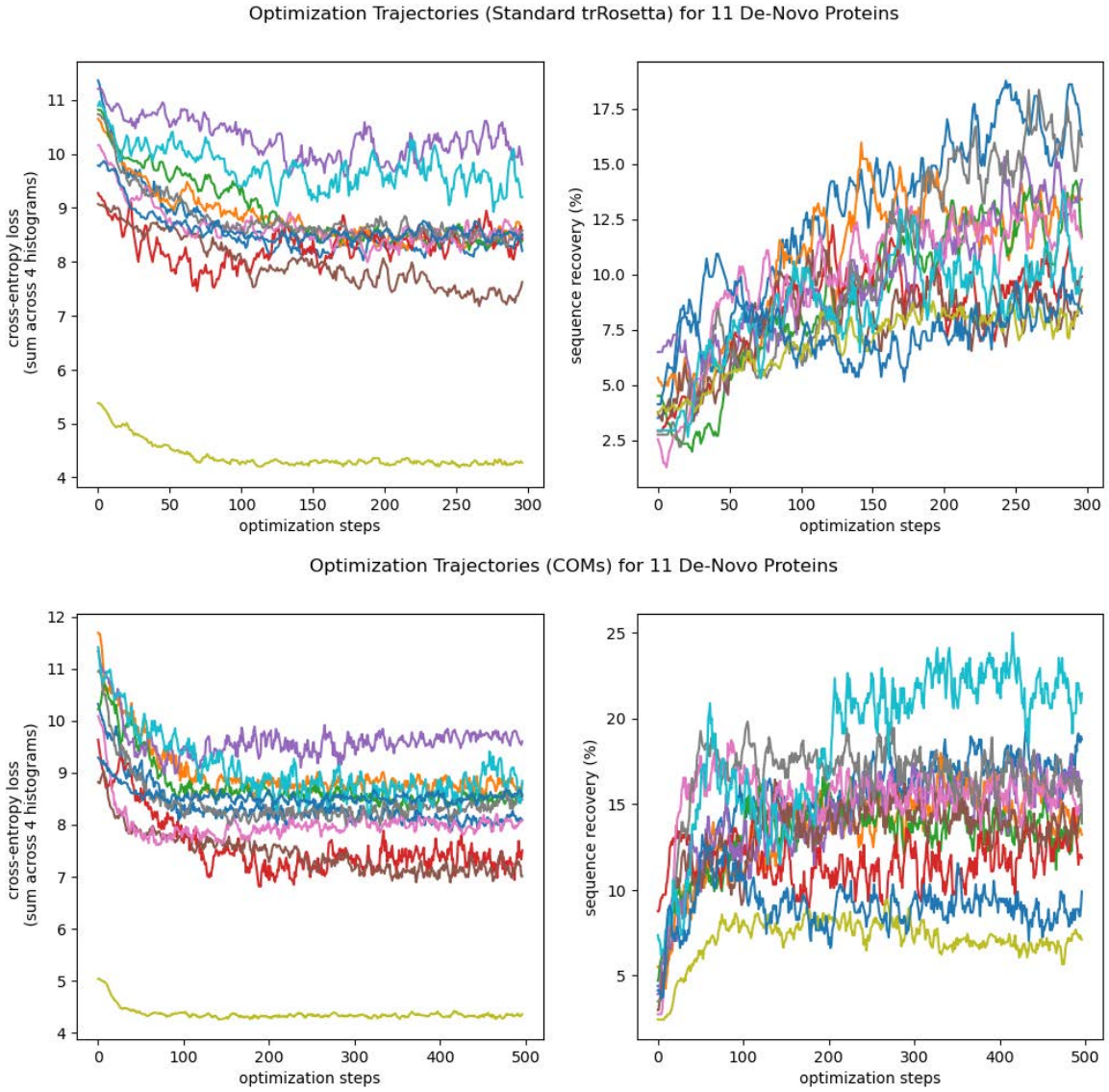

Figure 12: **Example optimization trajectories for the protein design algorithm's "global" optimizer.** The average final sequence recovery for the Standard trRosetta model in the trial visualized above is 11.678%, and the average final loss was 8.484. The average final sequence recovery for the COMs model in the trial visualized above is 14.313%, and the average final loss was 8.116.

In Figure 12, we visualize some example optimization trajectories, which illustrate the objective described in Section 4.1 and the sequence recovery rate during each step of optimization, for task 2. Finally, in Figure 13, we visualize the predicted structure histograms for a few of the designs generated by the COMs proxy model in task 2 and the corresponding target structure histograms that the designs were optimized for.

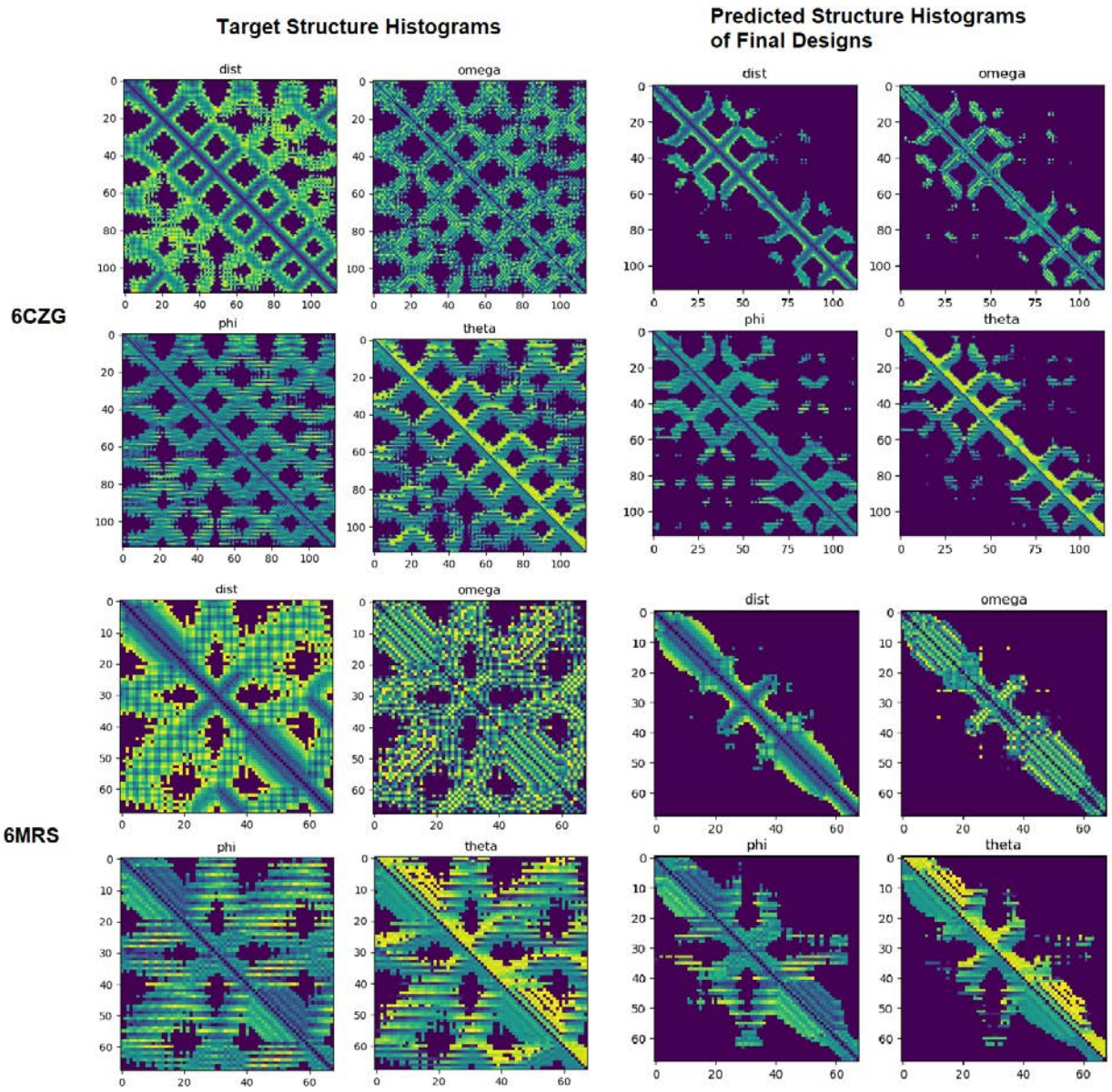

Figure 13: Target structure histograms and corresponding predicted structure histograms for designed sequences from task 2 (start with a random sequence and optimize to fold to the target structure using the "global" optimizer) of the de-novo proteins with PDB ID's 6CZG and 6MRS.

6 Discussion & Conclusion

6.1 Promoter Design

As we can see from the results on both task 1 (Figure 3) and task 2 (Figure 4), conservatism certainly helps. In fact, when we designed promoters for differential expression in K-562, the most conservative proxy model performed the best on both tasks, and when we designed promoters for differential expression in Jurkat, COMs models performed better than the standard offline MBO approach on both tasks. By contrast, when it came to the promoters designed for differential expression in THP-1, the standard offline MBO approach (i.e. $\alpha = 0.0$) seemed to perform better. However, based on the scatter plot visualizations in Figure 9, there are two possible explanations for this phenomenon that could both be the topic of future work:

1. The first explanation is that this is an unfortunate consequence of the promoter expression dataset we used. From a qualitative analysis of the scatter plots, the dataset appears to lack many examples of promoter sequences that are differentially expressed in THP-1. This potentially explains why the optimizer ostensibly struggles to design sequences that simultaneously upregulate expression in THP-1 while downregulating expression in Jurkat and K-562. The result is that the designed sequences are all clustered close to the origin. In this case, conservatism will obviously not help and will only hinder an already-challenging optimization task. Better datasets with more diverse data may fix this problem.
2. Another potential explanation for this phenomenon is that we need a more powerful optimizer or a different transformation function that maps from the individual expression values to the scalar objective used for optimization. In our experiments, we used the same optimization parameters for differential expression in all 3 cell types. Better cell-type specific parameter tuning may be necessary.

Overall, however, based on the results in task 1 and task 2, COMs appear to provide better performance on a wide range of promoter design scenarios. It can be used to effectively improve any given promoter sequence, as illustrated by task 1, and it can also be used to reliably generate a batch of high-quality designed promoters from a larger dataset of promoters, as illustrated in task 2. This can be practically useful in real-world promoter design.

6.2 Protein Design

As we can see in Table 5, on both task 1 and task 2, sequence recovery performance improves significantly when we subject the proxy model to additional COMs finetuning. Furthermore, in Figure 13 and Figure 11, we can see that the designed sequences seem to capture some of the features of the target structure relatively well, although not perfectly.

An other interesting observation is that the adapted COMs training, which involved optimizing a different objective function (i.e. optimizing for a different target structure) at each timestep, provided a proxy model that generalizes to designing sequences for any target structure, even that of de-novo proteins that the model has never seen before.

Overall, these results suggest that further research into COMs and its applications to the inverse protein folding problem would be worthwhile.

References

Vikram Agarwal and Jay Shendure. Predicting mrna abundance directly from genomic sequence using deep convolutional neural networks. *Cell reports*, 31(7):107663, 2020.

Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R. Ledsam, Agnieszka Grabska-Barwinska, Kyle R. Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R. Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, 18:1196–1203, 2021.

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, pages 773–782. PMLR, 2019.

Clara Fannjiang and Jennifer Listgarten. Autofocused oracles for model-based design. *arXiv preprint arXiv:2006.08052*, 2020.

Vladimir Gligorijevic, P Douglas Renfrew, Tomasz Kosciolek, Julia Koehler Leman, Kyunghyun Cho, Tommi Vatanen, Daniel Berenberg, Bryn C Taylor, Ian M Fisk, Ramnik J Xavier, et al. Structure-based function prediction using graph convolutional networks. *bioRxiv*, page 786236, 2019.

Rafael Gómez-Bombarelli, David Duvenaud, José Miguel Hernández-Lobato, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. In *ACS central science*, 2018.

Sven Heinz, Christopher Benner, Nathanael Spann, Eric Bertolino, Yin C Lin, Peter Laslo, Jason X Cheng, Cornelis Murre, Harinder Singh, and Christopher K Glass. Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and b cell identities. *Molecular cell*, 38(4):576–589, 2010.

Brian L Hie and Kevin K Yang. Adaptive machine learning for protein engineering. *Current opinion in structural biology*, 72:145–152, 2022.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873): 583–589, 2021.

Sathvik Kolli, Amy X. Lu, Xinyang Geng, Aviral Kumar, and Sergey Levine. Data-driven optimization for protein design: Workflows, algorithms and metrics. In *ICLR2022 Machine Learning for Drug Discovery*, 2022. URL `https://openreview.net/forum?id=Dc5J-bcEGW5`.

Aviral Kumar and Sergey Levine. Model inversion networks for model-based optimization. *NeurIPS*, 2019.

Thomas Liao, Grant Wang, Brian Yang, Rene Lee, Kristofer Pister, Sergey Levine, and Roberto Calandra. Data-efficient learning of morphology and controller for a microrobot. In *2019 IEEE International Conference on Robotics and Automation*, 2019. URL `https://arxiv.org/abs/1905.01334`.

Carol H Miao, Kazuo Ohashi, Gijsbert A Patijn, Leonard Meuse, Xin Ye, Arthur R Thompson, and Mark A Kay. Inclusion of the hepatic locus control region, an intron, and untranslated region increases and stabilizes hepatic factor ix gene expression in vivo but not in vitro. *Molecular Therapy*, 1(6):522–532, 2000.

Lior Nissim, Ming-Ru Wu, Erez Pery, Adina Binder-Nissim, Hiroshi I Suzuki, Doron Stupp, Claudia Wehrspaun, Yuval Tabach, Phillip A Sharp, and Timothy K Lu. Synthetic rna-based immunomodulatory gene circuits for cancer immunotherapy. *Cell*, 171(5):1138–1150, 2017.

Christoffer Norn, Basile I. M. Wicky, David Juergens, Sirui Liu, David Kim, Doug Tischer, Brian Koepnick, Ivan Anishchenko, Foldit Players, David Baker, and Sergey Ovchinnikov. Protein sequence design by conformational landscape optimization. *Proceedings of the National Academy of Sciences of the United States of America*, 118(11), 2021.

Aniketh Janardhan Reddy, Michael H. Herschl, Sathvik Kolli, Amy X. Lu, Xinyang Geng, Aviral Kumar, Patrick D. Hsu, Sergey Levine, and Nilah M. Ioannidis. Pretraining strategies for effective promoter-driven gene expression prediction. *bioRxiv*, 2023. doi: 10.1101/2023.02.24.529941. URL `https://www.biorxiv.org/content/early/2023/02/27/2023.02.24.529941`.

Alexander Rives, Siddharth Goyal, Joshua Meier, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, page 622803, 2019.

Nilofer Sayed, Prince Allawadhi, Amit Khurana, Vishakha Singh, Umashanker Navik, Sravan Kumar Pasumarthi, Isha Khurana, Anil Kumar Banothu, Ralf Weiskirchen, and Kala Kumar Bharani. Gene therapy: Comprehensive overview and therapeutic applications. *Life sciences*, page 120375, 2022.

Jamie L Shirley, Ype P de Jong, Cox Terhorst, and Roland W Herzog. Immune responses to viral gene therapy vectors. *Molecular Therapy*, 28(3):709–722, 2020.

Sam Sinai and Eric D Kelsic. A primer on model-guided exploration of fitness landscapes for biological sequence design. *arXiv preprint arXiv:2010.10614*, 2020.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, 2012. URL `http://dl.acm.org/citation.cfm?id=2999325.2999464`.

Tabula Sapiens Consortium. The tabula sapiens: A multiple-organ, single-cell transcriptomic atlas of humans. *Science*, 376(6594):eabl4896, 2022.

Brandon Trabucco, Xinyang Geng, Aviral Kumar, and Sergey Levine. Design-bench: Benchmarks for data-driven offline model-based optimization, 2021a. URL `https://github.com/brandontrabucco/design-bench`.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In *International Conference on Machine Learning*, pages 10358–10368. PMLR, 2021b.

Ming-Ru Wu, Lior Nissim, Doron Stupp, Erez Pery, Adina Binder-Nissim, Karen Weisinger, Casper Enghuus, Sebastian R Palacios, Melissa Humphrey, Zhizhuo Zhang, et al. A high-throughput screening and computation platform for identifying synthetic promoters with enhanced cell-state specificity (specs). *Nature communications*, 10(1):1–10, 2019.

Jianyi Yang, Ivan Anishchenko, Hahnbeom Park, Zhenling Peng, Sergey Ovchinnikov, and David Baker. Improved protein structure prediction using predicted interresidue orientations. *Proceedings of the National Academy of Sciences of the United States of America*, 117(3):1496–1503, 2020.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.