Generative models for graph-based protein design

John Ingraham, Vikas K. Garg, Regina Barzilay, Tommi Jaakkola
Computer Science and Artificial Intelligence Lab, MIT
{ingraham, vgarg, regina, tommi}@csail.mit.edu

Abstract

Engineered proteins offer the potential to solve many problems in biomedicine, energy, and materials science, but creating designs that succeed is difficult in practice. A significant aspect of this challenge is the complex coupling between protein sequence and 3D structure, with the task of finding a viable design often referred to as the inverse protein folding problem. In this work, we introduce conditional language models for protein sequences that directly condition on a graph specification of the target structure. Our approach efficiently captures the complex dependencies in proteins by focusing on those that are long-range in sequence but local in 3D space. Our framework improves in both speed and reliability over conventional and neural network-based approaches, and takes a step toward rapid and targeted biomolecular design with the aid of deep generative models.

1 Introduction

A central goal for computational protein design is to automate the invention of protein molecules with defined structural and functional properties. This field has seen tremendous progress in the past two decades [1], including the design of novel 3D folds [2], enzymes [3], and complexes [4]. However, the current practice often requires multiple rounds of trial-and-error, with first designs frequently failing [5, 6]. Several of the challenges stem from the bottom-up nature of contemporary approaches that rely on both the accuracy of energy functions to describe protein physics as well as on the efficiency of sampling algorithms to explore the protein sequence and structure space.

Here, we explore an alternative, top-down framework for protein design that directly learns a conditional generative model for protein sequences given a specification of the target structure, which is represented as a graph over the residues (amino acids). Specifically, we augment the autoregressive self-attention of recent sequence models [7] with graph-based descriptions of the 3D structure. By composing multiple layers of structured self-attention, our model can effectively capture higher-order, interaction-based dependencies between sequence and structure, in contrast to previous parameteric approaches [8, 9] that are limited to only the first-order effects.

The graph-structured conditioning of a sequence model affords several benefits, including favorable computational efficiency, inductive bias, and representational flexibility. We accomplish the first two by leveraging a well-evidenced finding in protein science, namely that long-range dependencies in sequence are generally short-range in 3D space [10–12]. By making the graph and self-attention similarly sparse and localized in 3D space, we achieve computational scaling that is linear in sequence length. Additionally, graph structured inputs offer representational flexibility, as they accommodate both coarse, ‘flexible backbone’ (connectivity and topology) as well as fine-grained (precise atom locations) descriptions of structure.

We demonstrate the merits of our approach via a detailed empirical study. Specifically, we evaluate our model at structural generalization to sequences of protein folds that were outside of the training set. For fixed-backbone sequence design, our model achieves considerably improved statistical
performance over a neural-network based model and also achieves higher accuracy and efficiency than Rosetta fixbb, a state-the-art program for protein design.

The rest of the paper is organized as follows. We first position our contributions with respect to the prior work in Section 1.1. We provide details on our methods, including structure representation, in Section 2. We introduce our Structured Transformer model in Section 2.2. The details of our experiments are laid out in Section 3, and the corresponding results that elucidate the merits of our approach are presented in Section 4.

1.1 Related Work

Generative models for protein sequence and structure A number of works have explored the use of generative models for protein engineering and design [13]. Recently [8, 9, 14] proposed neural network-based models for sequences given 3D structure, where the amino acids are modeled independently of one another. [15] introduced a generative model for protein sequences conditioned on a 1D, context-free grammar based specification of the fold topology. Multiple works [16, 17] have modeled the conditional distribution of single amino acids given surrounding structure and sequence context with convolutional neural networks. In contrast to these works, our model captures the joint distribution of the full protein sequence while grounding these dependencies in terms of long-range interactions arising from structure.

In parallel to the development of structure-based models, there has been considerable work on deep generative models for protein sequences in individual protein families [18–21]. While useful, these methods presume the availability of a large number of sequences from a particular family, which are unavailable in the case of designing novel proteins that diverge significantly from natural sequences.

Several groups have obtained promising results using (unconditional) protein language models [22–25] to learn sequence representations that transfer well to supervised tasks. While serving different purposes, we emphasize that one advantage of conditional generative modeling is to facilitate adaptation to specific (and potentially novel) parts of structure space. Language models trained on hundreds of millions of evolutionary sequences are nevertheless ‘semantically’ bottlenecked by the much smaller number of evolutionary 3D folds (thousands) that these sequences represent. We propose evaluating protein language models with structure-based splitting of sequence data, and begin to see how unconditional language models may struggle to assign high likelihoods to sequences from out-of-training folds.

In a complementary line of research, several deep and differentiable parameterizations of protein structure [26–29] have been recently proposed that could be used to generate, optimize, or validate 3D structures for input to sequence design.

Protein design and interaction graphs For classical approaches to computational protein design, which are based on joint modeling of structure and sequence, we refer the reader to a review of both methods and accomplishments in [1]. Many of the major ‘firsts’ in protein design are due to Rosetta [30, 31], a leading framework for protein design. More recently, there have been successes with non-parametric approaches to protein design [32] that are based on finding substructural homologies between the target and diverse templates in large protein database. We will focus on comparisons to Rosetta, since it is based on a shared parametric energy function that captures the sequence-structure relationship.

Self-Attention Our model extends the Transformer [33] to capture sparse, pairwise relational information between sequence elements. The dense variation of this problem was explored in [34] and [35]. As noted in those works, incorporating general pairwise information incurs \(O(N^2)\) memory (and computational) cost for sequences of length \(N\), which can be highly limiting for training on GPUs. We circumvent this cost by instead restricting the self-attention to the sparsity of the input graph. Given this graph-structured self-attention, our model may also be reasonably cast in the framework of message-passing or graph neural networks [36, 37]. Our approach is similar to Graph Attention Networks [38], but augmented with edge features and an autoregressive decoder.
For a rigid-body design problem, the structure for conditioning is a fixed set of backbone coordinates $X = \{x_i \in \mathbb{R}^3 : 1 \leq i \leq N\}$, where $N$ is the number of positions. We desire a graph representation of the coordinates $G(X)$ that has two properties:

- **Invariance.** The features are invariant to rotations and translations.
- **Locally informative.** The edge features incident to $v_i$ due to its neighbors $N(i,k)$, i.e., $\{e_{ij}\}_{j \in N(i,k)}$, contain sufficient information to reconstruct all adjacent coordinates $\{x_j\}_{j \in N(i,k)}$ up to rigid-body motion.

While invariance is motivated by standard symmetry considerations, the second property is motivated by limitations of current graph neural networks [36]. In these networks, updates to node features $v_i$ depend only on the edge and node features adjacent to $v_i$. However, typically, these features are

---

1 Here we consider a single representative coordinate per position when deriving edge features but may revisit multiple atom types per position for features such as backbone angles or hydrogen bonds.
insufficient to reconstruct the relative neighborhood positions \( \{x_j\}_{j \in \mathcal{N}(i,k)} \), so individual updates cannot fully depend on the ‘local environment’. For example, pairwise distances \( D_{ij} \) and \( D_{il} \) are insufficient to determine if \( x_j \) and \( x_l \) are on the same or opposite sides of \( x_i \).

**Relative spatial encodings** We develop invariant and locally informative features by first augmenting the points \( x_i \) with ‘orientations’ \( O_i \) that define a local coordinate system at each point (Figure 2). We define these in terms of the backbone geometry as

\[
O_i = [b_i \quad n_i \quad b_i \times n_i],
\]

where \( b_i \) is the negative bisector of angle between the rays \( (x_{i-1} - x_j) \) and \( (x_{i+1} - x_j) \), and \( n_i \) is a unit vector normal to that plane. Formally, we have

\[
u_i = \frac{x_i - x_{i-1}}{|x_i - x_{i-1}|}, \quad b_i = \frac{u_i - u_{i+1}}{|u_i - u_{i+1}|}, \quad n_i = \frac{u_i \times u_{i+1}}{|u_i \times u_{i+1}|}.
\]

Finally, we derive the spatial edge features \( e_{ij}^{(s)} \) from the rigid body transformation that relates reference frame \( (x_i, O_i) \) to reference frame \( (x_j, O_j) \). While this transformation has 6 degrees of freedom, we decompose it into features for distance, direction, and orientation as

\[
e_{ij}^{(s)} = \left( r\left(\|x_j - x_i\|\right), \quad O_i^T \frac{x_j - x_i}{\|x_j - x_i\|}, \quad q(O_j^T O_j) \right).
\]

Here the first vector is a distance encoding \( r(\cdot) \) lifted into a radial basis\(^2\), the second vector is a direction encoding that corresponds to the relative direction of \( x_j \) in the reference frame of \( (x_i, O_i) \), and the third vector is an orientation encoding \( q(\cdot) \) of the quaternion representation of the spatial rotation matrix \( O_j^T O_j \). Quaternions represent 3D rotations as four-element vectors that can be efficiently and reasonably compared by inner products [39].\(^3\)

**Relative positional encodings** Taking a cue from the original Transformer model, we obtain positional embeddings \( e_{ij}^{(p)} \) that encode the role of local structure around node \( i \). Specifically, we need to model the positioning of each neighbor \( j \) relative to the node under consideration \( i \). Therefore, we obtain the position embedding as a sinusoidal function of the gap \( i - j \). We retain the sign of

\(^2\)We used 16 Gaussian RBFs isotropically spaced from 0 to 20 Angstroms.

\(^3\)We represent quaternions in terms of their vector of real coefficients.
the distance $i - j$ because protein backbones are asymmetric. These relative distances contrast the absolute positional encodings of the original Transformer, and instead matches the relative encodings in [34].

**Node and edge features** Finally, we obtain an aggregate edge encoding vector $e_{ij}$ by concatenating the structural encodings $e_{ij}^{(s)}$ and the positional encodings $e_{ij}^{(p)}$ and then linearly transforming them to have the same dimension as the model. We only include edges in the $k$-nearest neighbors graph of $X$, with $k = 30$ for all experiments. This $k$ is generous, as typical definitions of residue-residue contacts in proteins will result in $< 20$ contacts per residue. For node features, we compute the three dihedral angles of the protein backbone $(\phi_i, \psi_i, \omega_i)$ and embed these on the 3-torus as $\{\sin, \cos\} \times (\phi_i, \psi_i, \omega_i)$.

**Flexible backbone features** We also consider ‘flexible backbone’ descriptions of 3D structure based on topological binary edge features and coarse backbone geometry. We combine the relative positional encodings with two binary edge features: **contacts** that indicate when the distance between $C_\alpha$ residues at $i$ and $j$ are less than 8 Angstroms and **hydrogen bonds** which are directed and defined by the electrostatic model of DSSP [41]. For coarse node features, we compute virtual dihedral angles and bond angles between backbone $C_\alpha$ residues, interpret them as spherical coordinates, and represent them as points on the unit sphere.

### 2.2 Structured Transformer

**Autoregressive decomposition** We decompose the joint distribution of the sequence given structure $p(s|x)$ autoregressively as

$$p(s|x) = \prod_i p(s_i|x, s_{<i}),$$

where the conditional probability $p(s_i|x, s_{<i})$ of amino acid $s_i$ at position $i$ is conditioned on both the input structure $x$ and the preceding amino acids $s_{<i} = \{s_1, \ldots, s_{i-1}\}$. These conditionals are parameterized in terms of two sub-networks: an **encoder** that computes refined node embeddings from structure-based node features $V(x)$ and edge features $E(x)$ and a **decoder** that autoregressively predicts letter $s_i$ given the preceding sequence and structural embeddings from the encoder (Figure 1).

**Encoder** Our encoder module is designed as follows. A transformation $W_h: \mathbb{R}^{d_e} \rightarrow \mathbb{R}^d$ produces initial embeddings $h_i = W_h(v_i)$ from the node features $v_i$ pertaining to position $i \in [N] \triangleq \{1, 2, \ldots, N\}$.

Each layer of the encoder implements a multi-head self-attention component, where head $\ell \in [L]$ can attend to a separate subspace of the embeddings via learned query, key and value transformations [7]. The queries are derived from the current embedding at node $i$ while the keys and values from the relational information $r_{ij} = (h_j, e_{ij})$ at adjacent nodes $j \in N(i, k)$. Specifically, $W_q^{(\ell)}$ maps $h_i$ to query embeddings $q_i^{(\ell)}$, $W_k^{(\ell)}$ maps pairs $r_{ij}$ to key embeddings $k_{ij}^{(\ell)}$ for $j \in N(i, k)$, and $W_v^{(\ell)}$ maps the same pairs $r_{ij}$ to value embeddings $v_{ij}^{(\ell)}$ for each $i \in [N], \ell \in [L]$. Decoupling the mappings for keys and values allows each to depend on different subspaces of the representation.

We compute the attention $a_{ij}^{(\ell)}$ between query $q_i^{(\ell)}$ and key $k_{ij}^{(\ell)}$ as a function of their scaled inner product:

$$a_{ij}^{(\ell)} = \frac{\exp(m_{ij}^{(\ell)})}{\sum_{j' \in N(i,k)} \exp(m_{ij'})}, \quad \text{where} \quad m_{ij}^{(\ell)} = \frac{q_i^{(\ell)\top} k_{ij}^{(\ell)}}{\sqrt{d}}.$$

The results of each attention head $l$ are collected as the weighted sum

$$h_i^{(\ell)} = \sum_{j \in N(i,k)} a_{ij}^{(\ell)} v_{ij}^{(\ell)},$$

We anticipate that alternative orderings for decoding the sequence may be favorable but leave this to future work.
Table 1: **Null perplexities** for common statistical models of proteins.

<table>
<thead>
<tr>
<th>Null model</th>
<th>Perplexity</th>
<th>Conditioned on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>20.00</td>
<td>-</td>
</tr>
<tr>
<td>Natural frequencies</td>
<td>17.83</td>
<td>Random position in a natural protein</td>
</tr>
<tr>
<td>Pfam HMM profiles</td>
<td>11.64</td>
<td>Specific position in a specific protein family</td>
</tr>
</tbody>
</table>

and then concatenated and transformed to give the update

\[
\Delta h_i = W_o \text{Concat}(h_i^{(1)}, \ldots, h_i^{(L)}).
\]

We update the embeddings with this residual, and alternate between these self-attention layers and position-wise feedforward layers as in the original Transformer [7]. We stack multiple layers atop each other, and thereby obtain continually refined embeddings as we traverse the layers bottom up. The encoder yields the embeddings produced by the topmost layer as its output.

**Decoder** Our decoder module has the same structure as the encoder but with augmented relational information \( r_{ij} \) that allows access to the preceding sequence elements \( s_{<i} \) in a *causally consistent* manner. In contrast to the encoder, where the keys and values are based on the relational information \( r_{ij} = (h_j, e_{ij}) \), the decoder can additionally access sequence elements \( s_j \) as

\[
r_{ij}^{(\text{dec})} = \begin{cases} 
(h_j^{(\text{dec})}, e_{ij}, g(s_j)) & i > j \\
(h_j^{(\text{enc})}, e_{ij}, 0) & i \leq j 
\end{cases}
\]

Here \( h_j^{(\text{dec})} \) is the embedding of node \( j \) in the current layer of the decoder, \( h_j^{(\text{enc})} \) is the embedding of node \( j \) in the final layer of the encoder, and \( g(s_j) \) is a sequence embedding of amino acid \( s_j \) at node \( j \). This concatenation and masking structure ensures that sequence information only flows to position \( i \) from positions \( j < i \), but still allows position \( i \) to attend to subsequent structural information unlike the standard Transformer decoder.

We now demonstrate the merits of our approach via a detailed empirical analysis. We begin with the experimental set up including our architecture, and description of the data used in our experiments.

### 3 Training

**Architecture** In all experiments, we used three layers of self-attention and position-wise feedforward modules for the encoder and decoder with a hidden dimension of 128.

**Optimization** We trained models using the learning rate schedule and initialization of the original Transformer paper [7], a dropout rate of 10% [42], a label smoothing rate of 10%, and early stopping based on validation perplexity. The unconditional language models did not include dropout or label smoothing.

**Dataset** To evaluate the ability of our models to generalize across different protein folds, we collected a dataset based on the CATH hierarchical classification of protein structure [40]. For all domains in the CATH 4.2 40% non-redundant set of proteins, we obtained full chains up to length 500 and then randomly assigned their CATH topology classifications (CAT codes) to train, validation and test sets at a targeted 80/10/10 split. Since each chain can contain multiple CAT codes, we first removed any redundant entries from train and then from validation. Finally, we removed any chains from the test set that had CAT overlap with train and removed chains from the validation set with CAT overlap to train or test. This resulted in a dataset of 18024 chains in the training set, 608 chains in the validation set, and 1120 chains in the test set. There is zero CAT overlap between these sets.

### 4 Results

A challenge in evaluating computational protein design methods is the degeneracy of the relationship between protein structure and sequence. Many protein sequences may reasonably design the same
Table 2: **Per-residue perplexities for protein language modeling** (lower is better). The protein chains have been cluster-split by CATH topology, such that test includes only unseen 3D folds. While a structure-conditioned language model can generalize in this structure-split setting, unconditional language models struggle.

<table>
<thead>
<tr>
<th>Test set</th>
<th>Short</th>
<th>Single chain</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure-conditioned models</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Structured Transformer (ours)</td>
<td>8.54</td>
<td>9.03</td>
<td>6.85</td>
</tr>
<tr>
<td>SPIN2 [8]</td>
<td>12.11</td>
<td>12.61</td>
<td>-</td>
</tr>
<tr>
<td><strong>Language models</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LSTM ( (h = 128) )</td>
<td>16.06</td>
<td>16.38</td>
<td>17.13</td>
</tr>
<tr>
<td>LSTM ( (h = 256) )</td>
<td>16.08</td>
<td>16.37</td>
<td>17.12</td>
</tr>
<tr>
<td>LSTM ( (h = 512) )</td>
<td>15.98</td>
<td>16.38</td>
<td>17.13</td>
</tr>
<tr>
<td><strong>Test set size</strong></td>
<td>94</td>
<td>103</td>
<td>1120</td>
</tr>
</tbody>
</table>

3D structure [43], meaning that sequence similarity need not necessarily be high. At the same time, single mutations may cause a protein to break or misfold, meaning that high sequence similarity isn’t sufficient for a correct design. To deal with this, we will focus on three kinds of evaluation: (i) likelihood-based, where we test the ability of the generative model to give high likelihood to held out sequences, (ii) native sequence recovery, where we evaluate generated sequences vs the native sequences of templates, and (iii) experimental comparison, where we compare the likelihoods of the model to high-throughput data from a de novo protein design experiment.

We find that our model is able to attain considerably improved statistical performance in its likelihoods while simultaneously providing more accurate and efficient sequence recovery.

4.1 **Statistical comparison to likelihood-based models**

**Protein perplexities** What kind of perplexities might be useful? To provide context, we first present perplexities for some simple models of protein sequences in Table 1. The amino acid alphabet and its natural frequencies upper-bound perplexity at 20 and \( \sim 17.8 \), respectively. Random protein sequences under these null models are unlikely to be functional without further selection [44]. First order profiles of protein sequences such as those from the Pfam database [45], however, are widely used for protein engineering. We found the average perplexity per letter of profiles in Pfam 32 (ignoring alignment uncertainty) to be \( \sim 11.6 \). This suggests that even models with high perplexities of this order have the potential to be useful models for the space of functional protein sequences.

The importance of structure We found that there was a significant gap between unconditional language models of protein sequences and models conditioned on structure. Remarkably, for a range of structure-independent language models, the typical test perplexities are \( \sim 16-17 \) (Table 2), which were barely better than null letter frequencies (Table 1). We emphasize that the RNNs were not broken and could still learn the training set in these capacity ranges. All structure-based models had (unsurprisingly) considerably lower perplexities. In particular, our Structured Transformer model attained a perplexity of \( \sim 7 \) on the full test set. It would seem that protein language models trained on one subset of 3D folds (in our cluster-splitting procedure) generalize poorly to predict the sequences

Table 3: **Ablation of graph features and model components**. Test perplexities (lower is better).

<table>
<thead>
<tr>
<th>Node features</th>
<th>Edge features</th>
<th>Aggregation</th>
<th>Short</th>
<th>Single chain</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rigid backbone</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dihedrals</td>
<td>Distances, Orientations</td>
<td>Attention</td>
<td>8.54</td>
<td>9.03</td>
<td>6.85</td>
</tr>
<tr>
<td>Dihedrals</td>
<td>Distances, Orientations</td>
<td>PairMLP</td>
<td>8.33</td>
<td>8.86</td>
<td>6.55</td>
</tr>
<tr>
<td>C( \alpha ) angles</td>
<td>Distances, Orientations</td>
<td>Attention</td>
<td>9.16</td>
<td>9.37</td>
<td>7.83</td>
</tr>
<tr>
<td>Dihedrals</td>
<td>Distances</td>
<td>Attention</td>
<td>9.11</td>
<td>9.63</td>
<td>7.87</td>
</tr>
<tr>
<td><strong>Flexible backbone</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C( \alpha ) angles</td>
<td>Contacts, Hydrogen bonds</td>
<td>Attention</td>
<td>11.71</td>
<td>11.81</td>
<td>11.51</td>
</tr>
</tbody>
</table>
of unseen folds, which is important to consider when training protein language models for protein engineering and design.

**Improvement over deep profile-based methods** We also compared to a recent method SPIN2 that predicts, using deep neural networks, protein sequence profiles given protein structures [8]. Since SPIN2 is computationally intensive (minutes per protein for small proteins) and was trained on complete proteins rather than chains, we evaluated it on two subsets of the full test set: a ‘Small’ subset of the test set containing chains up to length 100 and a ‘Single chain’ subset containing only those models where the single chain accounted for the entire protein record in the Protein Data Bank. Both subsets discarded any chains with structural gaps (chain break). We found that our Structured Transformer model significantly improved upon the perplexities of SPIN2 (Table 2).

**Graph representations and attention mechanisms** The graph-based formulation of protein design can accommodate very different formulations of the problem depending on how structure is represented by a graph. We tested different approaches for representing the protein including both more ‘rigid’ design with precise geometric details, and ‘flexible’ topological design based on spatial contacts and hydrogen bonding (Table 3). For the best perplexities, we found that using local orientation information was indeed important above simple distance measures. At the same time, even the topological features were sufficient to obtain better perplexities than SPIN2 (Table 2), which uses precise atomic details.

In addition to varying the graph features, we also experimented with an alternative aggregation function from message passing neural networks [36]. We found that a simple aggregation function $\Delta h_i = \sum_j \text{MLP}(h_j, h_i, e_{ij})$ led to the best performance of all models, where $\text{MLP}(\cdot)$ is a two layer perceptron that preserves the hidden dimension of the model. We speculate that this is due to potential overfitting by the attention mechanism. Although all following experiments still use multi-head self-attention and full rigid backbone features, this suggests room future improvements.

### 4.2 Benchmarking protein redesign

**Decoding strategies** Generating protein sequence designs requires a sampling scheme for drawing high likelihood sequences from the model. While beam-search or top-$k$ sampling [46] are commonly used heuristics for decoding, we found that simple biased sampling from the temperature adjusted distributions $p(T)(s|x) = \prod_i \frac{p(s_i|x, s_{<i})^{1/T}}{\sum_{a} p(a|x, s_{<i})^{1/T}}$ was sufficient for obtaining sequences with higher likelihoods than native. We used a temperature of $T = 0.1$ selected from sequence recovery on validation. For conditional redesign of a subset of positions in a protein, we speculate that the likelihood calculation is sufficiently fast such that MCMC-based approaches such as Gibbs sampling may be feasible.

---

5We thank one of our reviewers for this suggestion.
Table 5: Structure-conditioned likelihoods correlate with mutation effects in de novo-designed miniproteins. Shown are Pearson correlation coefficients ($R$, higher is better) between the log-likelihoods of mutated sequences and high-throughput mutation effect data from a systematic design of miniproteins [6]. Each design (column) includes 775 experimentally tested mutant protein sequences.

<table>
<thead>
<tr>
<th>Design</th>
<th>$\beta\beta\alpha\beta_{37}$</th>
<th>$\beta\beta\alpha\beta_{1498}$</th>
<th>$\beta\beta\alpha\beta_{1702}$</th>
<th>$\beta\beta\alpha\beta_{1716}$</th>
<th>$\alpha\beta\alpha_{779}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rigid backbone</td>
<td>0.47</td>
<td>0.45</td>
<td>0.12</td>
<td>0.47</td>
<td>0.57</td>
</tr>
<tr>
<td>Flexible backbone</td>
<td>0.50</td>
<td>0.44</td>
<td>0.17</td>
<td>0.40</td>
<td>0.56</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design</th>
<th>$\alpha\beta\alpha_{223}$</th>
<th>$\alpha\beta\beta_{726}$</th>
<th>$\alpha\beta\alpha_{872}$</th>
<th>$\alpha\alpha_{134}$</th>
<th>$\alpha\alpha_{138}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rigid backbone</td>
<td>0.36</td>
<td>0.11</td>
<td>0.21</td>
<td>0.24</td>
<td>0.33</td>
</tr>
<tr>
<td>Flexible backbone</td>
<td>0.33</td>
<td>0.21</td>
<td>0.23</td>
<td>0.36</td>
<td>0.41</td>
</tr>
</tbody>
</table>

Comparison to Rosetta  To evaluate the performance of our model at generating realistic protein sequences, we performed two experiments that compare with Rosetta [30], a state-of-the-art framework for computational protein design. We found that our model was more accurate and significantly faster than Rosetta (Table 4). In the first, we used the latest version of Rosetta (3.10) to design sequences for our ‘Single chain’ test set with the fixbb fixed-backbone design protocol and default parameters (Table 4, a). In the second, we also compared to a prior benchmark from members of the Rosetta community [47, 48] across 40 diverse proteins. For this set of proteins, we re-split our dataset to form new training and validation sets with no CAT overlap to the 40 templates for design. Although this reduced the size of the training set from ∼18,000 to ∼10,000 chains, we found our model to be both more accurate than and several orders of magnitude faster than Rosetta (Table 4, b).

4.3 Unsupervised anomaly detection for experimental protein design

While synthesis and testing of designed sequences is the gold standard of evaluating protein design methods we can measure what our structure-conditioned language model ‘knows’ about protein function by comparing the likelihoods it assigns to functional and non-functional mutant proteins from recent high-throughput design experiments as a kind of unsupervised anomaly detection [18]. We compare to a recent high-throughput design and mutagenesis experiment in which several de novo designed mini-proteins were subject to systematic mutagenesis to all possible point mutations [6]. We find that the log-likelihoods of our model non-trivially reflect mutational preferences of designed proteins (Table 5). Importantly, we see that the performance is not dependent on precise 3D geometric features (e.g. distances and orientations) but can also be realized with coarse information (e.g. contacts, hydrogen bonds, and coarse backbone angles).

5 Conclusion

We presented a new deep generative model to ‘design’ protein sequences given a graph specification of their structure. Our model augments the traditional sequence-level self-attention of Transformers [7] with relational 3D structural encodings and is able to leverage the spatial locality of dependencies in molecular structures for efficient computation. When evaluated on unseen folds, the model achieves significantly improved perplexities over recent neural network-based generative models and more accurate and efficient sequence generation than the state-of-art program Rosetta.

Our framework suggests the possibility of being able to efficiently design and engineer protein sequences with structurally-guided deep generative models, and underscores the central role of modeling sparse long-range dependencies in biological sequences.

Acknowledgments

We thank members of the MIT MLPDS consortium, the MIT NLP group, and the reviewers for helpful feedback. This work was supported by the Machine Learning for Pharmaceutical Discovery and Synthesis (MLPDS) consortium.
References


