FOLD2SEQ: A JOINT SEQUENCE(1D)-FOLD(3D) EMBEDDING-BASED GENERATIVE MODEL FOR PROTEIN DESIGN

Anonymous authors
Paper under double-blind review

ABSTRACT

Designing novel protein sequences consistent with a desired 3D structure or fold, often referred to as the inverse protein folding problem, is a central, but non-trivial, task in protein engineering. It has a wide range of applications in energy, biomedicine, and materials science. However, challenges exist due to the complex sequence-fold relationship and difficulties associated with modeling 3D folds. To overcome these challenges, we propose **Fold2Seq**, a novel transformer-based generative framework for designing protein sequences conditioned on a specific fold. Our model learns a fold embedding from the density of the secondary structural elements in 3D voxels, and then models the complex sequence-structure relationship by learning a joint sequence-fold embedding. Experiments demonstrate improved performance of **Fold2Seq** in terms of speed and reliability for sequence design, compared to existing benchmarks including the state-of-the-art RosettaDesign and other deep neural net-based approaches.

1 INTRODUCTION

Computational protein design is the conceptual inverse of the protein structure prediction problem, and aims to infer an amino acid sequence that will fold into a given 3D structure. Designing protein sequences that will fold into a desired structure has a broad range of applications, from therapeutics to materials (Kraemer-Pecore et al., 2001). Despite significant advancements in methodologies as well as in computing power, inverse protein design still remains challenging, primarily due to the vast size of the sequence space - and the difficulty of learning a function that maps from the 3D structure space to the sequence space. Earlier works rely mostly on energy minimization-based approaches (Koga et al., 2012; Rocklin et al., 2017; Huang et al., 2011), which follow a scoring function (force field, statistical potentials, or machine learning model) and sample both sequence and conformational space. Such methods often suffer from drawbacks such as low accuracy of energy functions or force-fields (Khan & Vihinen, 2010) and low efficiency in sequence and conformational search (Koga et al., 2012).

Recently, as the data on both protein sequences (hundreds of millions) and structures (a few hundreds of thousands) is quickly accumulating, data-driven approaches for inverse protein design are rapidly emerging (Greener et al., 2018; Karimi et al., 2020; Ingraham et al., 2019). Generally, data-driven protein design, attempts to model the probability distribution over sequences conditioned on the structures: $P(\boldsymbol{x}|\boldsymbol{y})$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are protein sequences and structures, respectively. Two key challenges remain: (1) defining a good representation ($\boldsymbol{y}$) of the protein structure and (2) modelling the sequence generation process conditioned on $\boldsymbol{y}$. Current protein design methods use protein backbone information from a single protein structure (fixed backbone) or from a set of backbone structures consistent with a single fold (flexible backbone). In earlier studies, the protein structures are represented as a 1D string (Greener et al., 2018), a 1D vector (Karimi et al., 2020), a 2D image (Strokach et al., 2020), or a graph (Ingraham et al., 2019). The sequence generation methods used in the protein design studies can be classified as non-autoregressive (Karimi et al., 2020; Greener et al., 2018; Strokach et al., 2020) and autoregeressive (Ingraham et al., 2019; Madani et al., 2020). In non-autoregerssive methods, $\boldsymbol{y}$ is usually concatenated with a Gaussian random noise $\boldsymbol{z}$ (which is the latent vector of the sequence) to be the input to a sequence generator

$P(\boldsymbol{x}|\boldsymbol{y}) = f_g(\boldsymbol{y}, \boldsymbol{z})$, while in autoregressive methods, $P(\boldsymbol{x}|\boldsymbol{y})$ is decomposed through the chain rule: $P(\boldsymbol{x}|\boldsymbol{y}) = \prod_{i=1}^{n} P(x_i|x_1, x_2, ..., x_{i-1}, \boldsymbol{y})$, where $\boldsymbol{x} = (x_1, x_2, ..., x_n)$.

In this paper, we focus on designing sequences conditioned on a protein fold. A protein fold is defined as the arrangement (or topology) of the secondary structure elements of the protein relative to each other (Hou et al., 2003). A secondary structural element can be defined as the three dimensional form of local segments of a protein sequence. Protein folds are therefore necessarily based on sets of structural elements that distinguish domains. As protein structure is inherently hierarchical, the complete native structure can have multiple folds and a fold can be present in many protein structures. A single structure (fixed backbone) or an ensemble of structures (flexible backbone) can be used as representatives of a fold. The ensemble representation is often a better choice, as it captures the protein dynamics.

Despite the recent progress in using machine learning models for protein design, significant gaps still remain in addressing both aforementioned challenges (fold representation and conditional sequence generation). **First**, the current fold representation methods are either hand-designed, or constrained and do not capture the complete original fold space (See Sec. 2.2 for details), resulting in low generalization capacity or efficiency. **Second**, the sequence encoding and the fold encoding are learned separately in previous methods, which makes two latent domains heterogeneous. Such heterogeneity across two domains actually increases the difficulty of learning the complex sequence–fold relationship.

To fill the aforementioned gaps, the **main contributions** of this work are as follows:

- We propose a novel fold representation, through first representing the 3D structure by the voxels of the density of secondary structures elements (SSEs), and then learning the fold representation through a transformer-based structure encoder. Compared to previous fold representations, this representation has several advantages: first, it stores 3D information of SSEs, so that spatial information is not lost. Second, it does not need any pre-defined rules or structures, so that it is not biased. Third, the representation can be automatically extracted from a given protein without any manual work. Lastly, the density model also loosens the rigidity of structures so that the dynamics of the protein is better considered.
- We employ a novel joint sequence-fold embedding learning framework into the transformer-based auto-encoder model. By learning a joint latent space between sequences and folds, our model, **Fold2Seq**, mitigates the heterogeneity between two different domains and is able to better capture the sequence-fold relationship, as reflected in the results.
- Experiments demonstrate that our model has superior performance on perplexity, native sequence recovery rate and native structure recovery accuracy, when compared to competing methods including the state-of-the-art RosettaDesign and other neural net models. Ablation study shows that this superior performance is directly attributed to our algorithmic innovations.

2 Related Work

Data-driven Protein Design A significant surge of protein design studies that deeply exploit the data through modern artificial intelligence algorithms has been witnessed in the last two years. Greener et al. (2018) used the conditional variational autoencoder for generating protein sequences conditioned on a given fold. Karimi et al. (2020) developed a guided conditional Wasserstein Generative Adversarial Networks (gcWGAN) also for inverse fold design. Madani et al. (2020) trained an extreme large (1.2B parameters) language model conditioned on taxonomic and keyword tags such as molecular functions for generating protein sequences. Ingraham et al. (2019) developed a graph-based transformer for generating protein sequences conditioned on a fixed backbone. Lastly, Strokach et al. (2020) formulated the inverse protein design as a constraint satisfaction problem (CSP) and applied the graph neural networks for generating protein sequences conditioned on the residue-residue distance map that is a static representation of the structure.

Protein Fold Representation For an extensive overview of molecular representations, including those of proteins, please see David et al. (2020). Murzin et al. (1995) and Orengo et al. (1997) manually classified protein structures in a hierarchical manner based on their structural similarity.

These classifications can be regarded as one-hot encoding of the fold representations. Taylor (2002) represents a protein fold using a "periodic table". This representation was later used for inverse fold design (Greener et al., 2018). However, it only considers three pre-defined shapes for a set of structures, which significantly limits the generalizablity in the fold space. Hou et al. (2003) chose hundreds of representative proteins and calculated the similarity scores among them. This similarity matrix was then used for kernel Principle Component Analysis (kPCA). A similar idea was used in Karimi et al. (2020) for inverse protein design. This representation needs a pre-defined set of structures along with a similarity metric, and could lead to biased or constrained representations of the fold space. Finally, Koga et al. (2012) summarized three rules that describe the junctions between adjacent secondary structure elements for a specific fold. Again, these rules are hand designed for a subset of structures, which makes the representation restricted to a small part of the fold space and offers limited generalizability during conditional sequence generation.

Joint Embedding Learning Joint embedding learning across multiple different data modalities was first proposed by Ngiam et al. (2011) on audio and video signals. Since then, such approaches have been then widely used in cross modal retrieval or captioning tasks (Arandjelovic & Zisserman, 2018; Gu et al., 2018; Peng & Qi, 2019; Chen et al., 2018; Wang et al., 2013; Dognin et al., 2019). In few/zero-shot learning, joint feature-label embedding was used for predicting the label of instances belonging to unseen classes (Zhang & Saligrama, 2016; Socher et al., 2013). Several papers have demonstrated that learning joint embedding is useful for the single modal classification tasks (Ngiam et al., 2011; Wang et al., 2018; Toutanova et al., 2015). Moreover, Chen et al. (2018) used joint embedding learning for text to shape generation. Lastly, Joint sequence-label embedding is also applied for molecular prediction/generation (Cao & Shen, 2020; Das et al., 2018).

3 METHODS

3.1 BACKGROUND AND NOTATION

A protein is formed by a linear chain of amino acids (residues) that defines its 1D sequence. Chemical nature, as well as physical and chemical interactions with neighboring residues drive the folding of a sequence into different secondary structure elements or SSEs (helix, beta-sheet, loop, etc., see Fig 1(a)), that eventually forms a complete native 3D structure. A **protein fold** captures the topology and the composition of secondary structure elements, thus serving as an intermediate between the 1D sequence and the full 3D structure.

3.2 STRUCTURE REPRESENTATION THROUGH 3D VOXELS OF THE DENSITY OF SSES

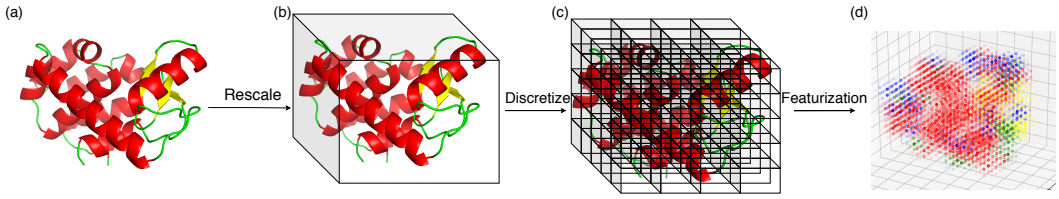

Figure 1: (a) The structure of PDB 107L. The secondary structures are colored as: helices in red, beta sheets in yellow and loops in green. (b) The structure is rescaled to fit the $40\text{Å} \times 40\text{Å} \times 40\text{Å}$ cubic box. (c) The box is discretized into voxels. (d) Features of each voxel are obtained from the structure content of the voxel (bend/turn in blue).

In this subsection, we describe how we represent the 3D structure to explicitly capture the fold information. We denote the position(3D coordinate) of each residue by its alpha carbon. For a given protein with length N, we first translate the structure to match its center of mass with the origin of the coordinate system. We then rotate the protein around the origin to let the first residue be on the negative side of z-axis. We denote the resulting residue coordinates as $\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_N$. We assign the secondary structure label to each residue based on their SSE assignment (Kabsch & Sander, 1983) in Protein Data Bank (Berman et al., 2000). We consider 4 types of secondary structure labels: helix, beta strand, loop and bend/turn. In order to consider the distribution of different secondary structure

labels in the 3D space, we discretize the 3D space into voxels, as shown in Fig 1. A technical challenge here is that the sizes of different proteins vary drastically. As we are only considering the arrangement of SSEs, not their exact coordinates, we here rescale the original structure, so that it fits into a fixed-size cubic box. Based on the distribution of sizes of single-chain proteins in the CATH database (Sillitoe et al., 2019), we choose a $40\text{Å} \times 40\text{Å} \times 40\text{Å}$ box with each voxel of size $2\text{Å} \times 2\text{Å} \times 2\text{Å}$. We denote the scaling ratio as $\boldsymbol{r} \in \mathcal{R}^3$. For voxel i, we denote the coordinates of its center as $\boldsymbol{v}_i$. We assume that the contribution of residues j to voxel i follows the Gaussian form:

$$\boldsymbol{y}_{ij} = \exp(-\frac{||\boldsymbol{c}_j \odot \boldsymbol{r} - \boldsymbol{v}_i||_2^2}{\sigma^2}) \cdot \boldsymbol{t}_j, \tag{1}$$

where $\boldsymbol{t}_j \in \{0,1\}^4$ is the one-hot encoding of the secondary structure label of amino acid j. The standard deviation is chosen to be 2Å. We sum up all residues together to obtain the final features of the voxel i: $\boldsymbol{y}_i = \sum_{j=1}^{N} \boldsymbol{y}_{ij}$. The structure representation $\boldsymbol{y}$ is the vector of $\boldsymbol{y}_i$'s over all voxels. In the next subsection, we will describe how we encode $\boldsymbol{y}$ into the latent space and how we learn a joint sequence-fold embedding in order for generating sequences consistent with a desired 3D structure.

3.3 FOLD2SEQ WITH JOINT SEQUENCE-FOLD EMBEDDING

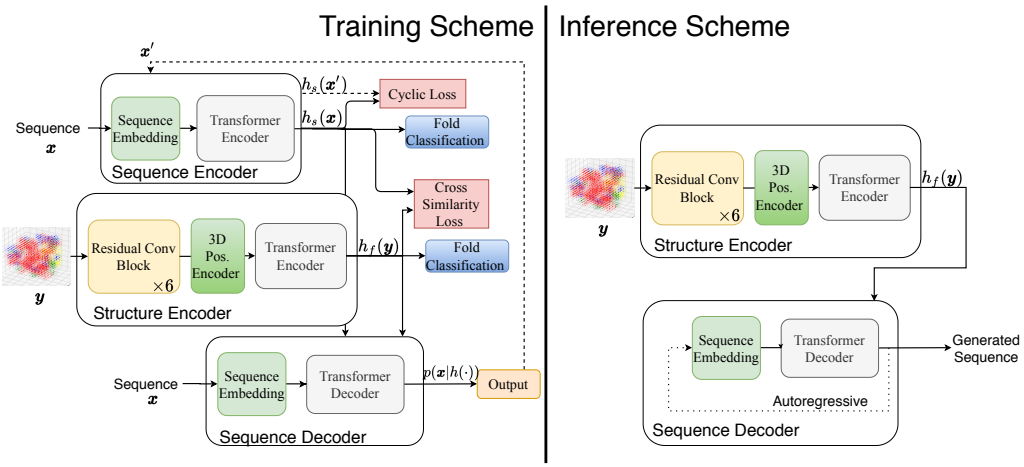

Figure 2: The architecture of the model during the training and inference stages. (**Training Scheme**): During training, the model includes three major components: (top) Sequence Encoder, (middle) Structure Encoder and (bottom) Sequence Decoder. The dashed arrows represent the process for getting cyclic loss. (**Inference Scheme**): During the inference, the model only needs the structure encoder and the sequence decoder for decoding sequences.

Model architectures In the training stage, our model consists of three major components: a sequence encoder: $h_s(\cdot)$, a structure encoder: $h_f(\cdot)$ and a sequence decoder: $p(\boldsymbol{x}|h(\cdot))$, as shown in Fig. 2 (Left). Both sequence encoder and decoder are implemented using the vanilla transformer model (Vaswani et al., 2017). The structure encoder contains 6 residual blocks followed by a 3D positional encoding. Each residual block has two 3D-convolutional layers ($3 \times 3 \times 3$) and batch normalization layers. The 3D positional encoding is a simple 3D extension of the sinusoidal encoding described in the vanilla transformer model, as shown in Appendix A. After the positional encoding, the 3D vector is flattened to be 1D as the input of a transformer encoder. The length of the transformer input is fixed to be $l_f = 5^3 = 125$. The output of the transformer encoder: $h_f(\boldsymbol{y})$ is the latent fold representation of $\boldsymbol{y}$.

We propose a simple fold-to-sequence reconstruction loss based on the auto-encoder model: $\mathbf{RE}_f = p(\boldsymbol{x}|h_f(\boldsymbol{y}))$. However, as mentioned earlier, training based on $\mathbf{RE}_f$ alone suffers due to the heterogeneity of $\boldsymbol{x}$ and $\boldsymbol{y}$. To overcome this challenge, we first encode the sequence $\boldsymbol{x}$ through the sequence encoder into the latent space as: $h_s(\boldsymbol{x})$, which could be done through a simple sequence-to-sequence reconstruction loss: $\mathbf{RE}_s = p(\boldsymbol{x}|h_s(\boldsymbol{x}))$. We then learn a joint latent space between $h_f(\boldsymbol{y})$ and $h_s(\boldsymbol{x})$ through a novel sequence-fold embedding learning framework.

Joint Embedding Learning Typically, learning a joint embedding across two domains needs two intra-domain losses and one cross-domain loss (Chen et al., 2018). An intra-domain loss forces two semantically similar samples from the same domain to be close to each other in the latent space, while a cross-domain loss forces two semantically similar samples in different domains to be closer. In our case, the meaning of 'semantically similar' is that the proteins should have the same fold(s). Therefore, we consider a supervised learning task for learning intra-domain similarity: fold classification. Specifically, the outputs of both encoders: $h_f(\boldsymbol{y}) \in \mathcal{R}^{l_f \times d}$ and $h_s(\boldsymbol{x}) \in \mathcal{R}^{l_s \times d}$ will be averaged along l_f and l_s dimensions, followed by a MLP+softmax layer to perform fold classification (shown as two blue blocks in Fig. 2), where l_s, l_f and d are the length of the sequence, the fold and the latent state, respectively. The two MLP layers' parameters are shared. The category labels follow the fold (topology) level of hierarchical protein structure classification in CATH4.2 dataset (Sillitoe et al., 2019) (see Section 3.4). As a result, we propose the following two intra-domain losses: $\mathbf{FC}_f$ and $\mathbf{FC}_s$, the cross entropy losses of fold classification from $h_f(\boldsymbol{y})$ and $h_s(\boldsymbol{x})$ respectively. The benefits of these two classification tasks are obvious: First, it will force the structure encoder to learn the fold representation. Second, as we perform the same supervised learning task on the latent vectors from two domains, it will not only learn the intra-domain similarity, but also cross-domain similarity. However, without explicit cross-domain learning, the two latent vectors $h_f(\boldsymbol{y})$, $h_s(\boldsymbol{x})$ could still have minimal alignment between them. We will describe the cross-domain loss in the rest of this subsection.

In the transformer decoder, each element in the **non**-self attention matrix is calculated by the cosine similarity between the latent vectors from the encoder and the decoder, respectively. Inspired by this observation, we consider to maximize the cosine similarity (shown as the 'Cosine Similarity' pink block in Fig 2) between $h_f(\boldsymbol{y}) \in \mathcal{R}^{l_f \times d}$ and $h_s(\boldsymbol{x}) \in \mathcal{R}^{l_s \times d}$ as the cross-domain loss. We first calculate the matrix-product between $h_f(\boldsymbol{y})$ and $h_s(\boldsymbol{x})$ as $\boldsymbol{Q} = h_f(\boldsymbol{y}) \cdot h_s(\boldsymbol{x})^T, \boldsymbol{Q} \in \mathcal{R}^{l_f \times l_s}$. The ith row in $\boldsymbol{Q}$ represents the similarity between ith position in the fold and every position of the sequence. We would like to find the best-matching sequence piece with each position in the fold. To achieve this, the similarity matrix $\boldsymbol{Q}$ first goes through a row-wise average pooling with kernel size k, followed by the row-wise $\max$ operation:

$$\boldsymbol{q} = \max_{row}(\text{AvgPool}^k_{row}(\boldsymbol{Q})), \boldsymbol{q} \in \mathcal{R}^{l_f \times 1} \tag{2}$$

where row means the operation is row-wised. We choose $k = 3$, which means the scores of every 3 continuous positions in the sequence will be averaged. We finally average over all positions in the fold to get the final similarity score: $\mathbf{CS} = \text{mean}(\boldsymbol{q})$.

Besides the cosine similarity loss, inspired from the earlier CycleGAN work (Zhu et al., 2017), we add a cyclic loss (shown as the 'Cyclic Loss' red block in Fig 2.) to be another term of our cross-domain loss. Specifically, we take the argmax of the output of fold-to-sequence model: $\boldsymbol{x}' = \arg\max p(\boldsymbol{x}|h_f(\boldsymbol{y}))$, and send it back to the sequence encoder for generating the cyclic-seq latent state: $h_s(\boldsymbol{x}')$ (shown as the dashed line in Fig 2). This cyclic-seq latent state will compare with the native seq latent state $h_s(\boldsymbol{x})$ through the square of the L2 distance:

$$\mathbf{CY} = ||h_s(\boldsymbol{x}') - h_s(\boldsymbol{x})||_2^2 \tag{3}$$

To summarize, the complete loss objective is the following:

$$L = \lambda_1 \mathbf{RE}_f + \lambda_2 \mathbf{RE}_s + \lambda_3 \mathbf{FC}_f + \lambda_4 \mathbf{FC}_s + \lambda_5(\mathbf{CY} - \mathbf{CS}) \tag{4}$$

where λ_1 through λ_5 are the hyperparameters for controlling the importance of these losses.

Training and Decoding Strategy During experiments we found that, if the sequence encoder and the structure encoder were trained together, the structure encoder had little parameter improvement while the sequence encoder dominated the training. To overcome this issue, we consider a two-stage training strategy. In the first stage, we train the sequence-to-sequence model regularized by the sequence intra-domain loss: $L_1 = \lambda_2 \mathbf{RE}_s + \lambda_4 \mathbf{FC}_s$. In the second stage, we train the fold-to-sequence model regularized by the fold intra-domain loss and the cross-domain loss while keeping the sequence encoder frozen: $L_2 = \lambda_1 \mathbf{RE}_f + \lambda_3 \mathbf{FC}_f + \lambda_5(\mathbf{CY} - \mathbf{CS})$.

We implement our model in Pytorch (Paszke et al., 2019). Each transformer block has 4 layers and $d = 256$ latent dimensions. The learning rate schedule follows the original transformer paper (Vaswani et al., 2017). We use the exponential decay (Blundell et al., 2015) for $\lambda_5 = 1/2^{\#\text{epoch}-e}$

in the loss function, while λ_1 through λ_4 and e are tuned based on the validation set, resulting $\lambda_1 = 1.0, \lambda_2 = 1.0, \lambda_3 = 0.02, \lambda_4 = 1.0, e = 3$. We train our model on 2 Tesla K80 GPUs, with batch size 128. In every training stage we train up to 200 epochs with an early stopping strategy based on the loss on the validation set. We will release the code for reproducing our experiments on acceptance.

During inference, we only need the Fold2seq model for decoding sequences (Fig 2 (Right)). Top-k sampling strategy (Fan et al., 2018) is used for sequence generation, where k is set to be 5 (tuned based on the validation set).

3.4 Datasets

We used the protein structure data from CATH 4.2 (Sillitoe et al., 2019) filtered by 100% sequence identity. We remove proteins that (1) are multi-chain or non-continuous in sequence; (2) contain other than 20 natural amino acids; (3) have length longer than 200. We randomly split the dataset based on the fold-level classification of protein structures into 95%, 2.5%, 2.5% as dataset (a), (b) and (c), which means that the three datasets have non-overlapping folds. We further randomly split the dataset (a) into 95%, 2.5% and 2.5% as dataset (a1), (a2) and (a3). Datasets (a1) (a2) (a3) have overlapping folds. We use dataset (a1) as the training set, (b)+(a2) as the validation set, (a3) as the In-distribution (ID) test set and (c) as the Out-of-distribution (OD) test set. The folds of the ID test set overlaps with the training set, whereas the folds of OD test set do not overlap with the training set. The statistics of these datasets are presented in Appendix B.

In order to quantitatively measure their difficulty levels, we calculate the averaged maximum sequence similarity (**amsi**) between a given test set $\boldsymbol{T}$ and the training set, defined as: $\text{amsi}_{\boldsymbol{T}} = \frac{1}{|D_{\boldsymbol{T}}|} \sum_{i \in D_{\boldsymbol{T}}} \max_{j \in D_{\text{train}}} (\text{SIM}(\boldsymbol{x}_i, \boldsymbol{x}_j))$, where D_{train} and $D_{\boldsymbol{T}}$ are the training set and test set $\boldsymbol{T}$, respectively; $\text{SIM}(\boldsymbol{x}_i, \boldsymbol{x}_j)$[1] is the sequence similarity between sequence $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. As a result, we found we have $\text{amsi}_{\text{ID}} = 36.3\%$ and $\text{amsi}_{\text{OD}} = 16.3\%$. This shows that the sequence similarity between the ID test set and the training set is more than twice that between the OD test set and the training set. These values clearly demonstrate that the OD test set represents a much more difficult generalization task compared to the ID test set.

4 Results

Ideally, the most appropriate criteria for evaluating inverse design methods is the extent that the structure of the generated sequence matches the desired fold/structure. However, as protein structure prediction is very time-consuming, and similar sequences usually indicate similar folds/structures, many earlier methods report performance in the sequence domain. We hereby perform comprehensive evaluations of Fold2Seq against two data-driven methods: cVAE (Greener et al., 2018) and gcWGAN (Karimi et al., 2020) as well as the state-of-the-art principle-driven method, RosettaDesign[2] (Huang et al., 2011). For this comparison, we leave out methods that do not focus on inverse folding problem with a flexible backbone constraint (e.g. (Ingraham et al., 2019), (Madani et al., 2020), (Strokach et al., 2020)). Three evaluation metrics are used: 1) Perplexity: The ability to assign high likelihood to the native (ground-truth) sequences; 2) Native sequence recovery rate: The ability to recover the native sequences; 3) Native structural recovery accuracy: The ability to recover the native structures.

Perplexity and Sequence Recovery Comparison. We first compare Fold2Seq against gcWGAN and cVAE in terms of per-residue Perplexity. Perplexity is a metric widely used in the language modeling and is not applicable to RosettaDesign. For reference, we also show the per-residue perplexity under the uniform distribution and the frequencies through all natural sequences in UniRef50 (Suzek et al., 2015). Performance on two test sets is summarized in Table 1a, showing that Fold2Seq has the smallest per-residue perplexity for both ID and OD test sets. We also note that the performance

[1] Sequence similarity is measured through the Needleman Wunsch algorithm (Needleman & Wunsch, 1970) with Blossum62 scoring matrix.

[2] RosettaDesign is a principle-based method. It uses MCMC sampling and energy calculation to search for best sequences. The input to RosettaDesign consists of the backbone of the native structure and the SSE of each residue.

Table 1: Performance of different methods assessed in the sequence domain.

(a) Per-residue Perplexity.

Model	ID Test	OD Test
Uniform	20.0	20.0
Natural	18.0	18.0
cVAE	14.8	16.3
gcWGAN	13.5	15.2
Fold2Seq	**8.1**	**11.9**

(b) Avg. sequence recovery rate $\pm$ std. dev. (%).

Model	ID Test	OD Test
Random across two folds	12.8 ± 7.94	12.8 ± 7.94
cVAE	17.7 ± 7.34	15.3 ± 5.34
gcWGAN	17.5 ± 6.35	14.1 ± 3.45
RosettaDesign	20.3 ± 5.13	20.2 ± 2.98
Fold2Seq	$\mathbf{27.1} \pm \mathbf{6.31}$	$\mathbf{24.1} \pm \mathbf{2.64}$
Random within same fold	39.1 ± 9.35	39.1 ± 9.35

of all methods on the OD test set is worse than that on the ID test set, which matches our expectation of their difficulty levels.

Next, we compare the ability of different methods for recovering the native sequences given a desired fold. We calculate the averaged sequence similarity between the native sequence and the generated sequence on two test sets. Also, for comparison, we calculate the expected similarity between two random sequences in our whole dataset belonging to two different folds and belonging to the same folds. The results are summarized in Table 1b. In general, Fold2Seq performs significantly better, when compared to existing methods. Specifically, compared to RosettaDesign (second-best), Fold2Seq improves the recovery rate by 33.5% and by 19.3% on the ID and OD test set, respectively.

Structural Recovery Comparison with RosettaDesign. Although the comparison in sequence domain reflects the reliability of generated sequences to some extent, our final goal is to ensure that the structure of a generated sequence is similar to the native structure. Specifically, we used the iTasser Suite (Yang et al., 2015), one of the state-of-the-art protein structure prediction software, to predict the structure of the designed sequences. As iTasser usually takes at least one day for predicting the structure of a single protein, we only compare Fold2Seq to RosettaDesign in terms of the structure recovery. The similarity between two 3D protein structures is measured by the TM-score $\in [0, 1]$ (Zhang & Skolnick, 2004). The higher the score, more similar the two structures are.

For the ith protein in the test set, we compare $\text{TM}(\boldsymbol{s}_i^{\textbf{Fold2Seq}}, \boldsymbol{s}_i^{\textbf{Native}})$ against $\text{TM}(\boldsymbol{s}_i^{\textbf{Rosetta}}, \boldsymbol{s}_i^{\textbf{Native}})$, where $\boldsymbol{s}_i^{\textbf{Fold2Seq}}$, $\boldsymbol{s}_i^{\textbf{Rosetta}}$ and $\boldsymbol{s}_i^{\textbf{Native}}$ are the 3D structures of the generated sequence from Fold2Seq, the generated sequence from RosettaDesign, and the native sequence, respectively.

We first compare the distributions of $\text{TM}(\boldsymbol{s}_i^{\textbf{Fold2Seq}}, \boldsymbol{s}_i^{\textbf{Native}})$ and $\text{TM}(\boldsymbol{s}_i^{\textbf{Rosetta}}, \boldsymbol{s}_i^{\textbf{Native}})$ on two test sets (ID and OD). The results are shown in Fig 3(a). For the ID test set, Fold2Seq shows significant improvement against RosettaDesign, while for the OD test set, Fold2Seq shows slightly better performance. We then define $\Delta\text{TM}_i = \text{TM}(\boldsymbol{s}_i^{\textbf{Fold2Seq}}, \boldsymbol{s}_i^{\textbf{Native}}) - \text{TM}(\boldsymbol{s}_i^{\textbf{Rosetta}}, \boldsymbol{s}_i^{\textbf{Native}})$ and perform one-sided one-sampled t-test over ΔTM, with null hypothesis as "$\Delta\text{TM} \leq 0.0$" on two test sets, revealing $\textbf{P-value}_{\text{ID}} = 2.14\text{E}-60$, $\textbf{P-value}_{\text{OD}} = 0.019$, which demonstrate that Fold2Seq can overall generate more reliable structures compared to RosettaDesign. The two distributions over ΔTM are shown in Fig 4 in Appendix C. We also randomly pick some designed structures with $\Delta\text{TM} > 0.0$ and $\Delta\text{TM} < 0.0$, and visualize them on Fig 5 and Fig 6 in Appendix D, respectively.

The sequence generation efficiency of Fold2Seq against RosettaDesign is shown in Fig 3(b). Fold2Seq is almost 100 times faster than RosettaDesign when running on CPU, and is 5000 times faster when running on GPU, while RosettaDesign can only run on CPU.

Ablation Study. In order to rigorously delineate the contributions of each algorithmic innovations, we perform an ablation study as following:

- cVAE: We use cVAE (Greener et al., 2018) as the baseline with 1D string fold representation and MLP-based variational autoencoder (VAE).
- Trans_string_$\textbf{RE}_f$: We replace the MLP-based VAE in cVAE with transformer autoencoder model. The loss is $L = \textbf{RE}_f$.

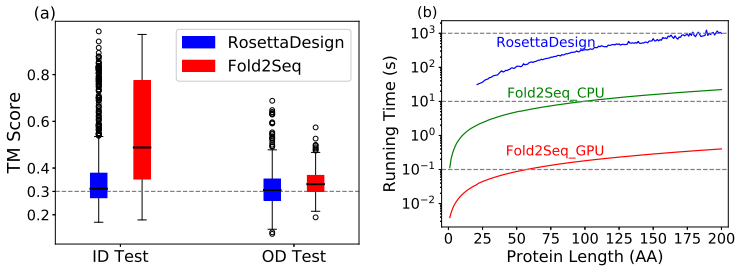

Figure 3: (a). The distributions of TM scores of RosettaDesign and Fold2Seq on two test sets. (b). The running time of Fold2Seq and RosettaDesign for generating one protein sequence. (CPU version: Intel Xeon E5-2680 v4 2.40GHz, GPU version: Nvidia Tesla K80.)

Table 2: Avg. sequence recovery rate±std. dev. (%) of different models in the ablation study.

Model	ID Test	OD Test
cVAE	17.7 ± 7.34	15.3 ± 5.34
Trans_string_$\mathbf{RE}_f$	18.5 ± 8.26	16.1 ± 3.23
Trans_voxel_$\mathbf{RE}_f$	22.3 ± 8.23	19.2 ± 3.46
+$\mathbf{RE}_s$+**CS**	22.4 ± 7.54	19.1 ± 2.24
+2**FC**	25.3 ± 6.24	22.2 ± 2.13
+**CY** (Fold2Seq)	$\mathbf{27.1 \pm 6.31}$	$\mathbf{24.1 \pm 2.64}$

- Trans_voxel_$\mathbf{RE}_f$: We replace the 1D string fold representation in "Trans_string_$\mathbf{RE}_f$" with 3D voxel representation. We also add the convolutional residual block and 3D positional encoding. The loss is also $L = \mathbf{RE}_f$.
- +$\mathbf{RE}_s$+**CS**: We add the sequence encoder, together with the reconstruction loss and the cosine similarity loss to the previous loss: $L = \lambda_1 \mathbf{RE}_f + \lambda_2 \mathbf{RE}_s - \lambda_5 \mathbf{CS}$.
- +2**FC**: We add the two **FC** losses. $L = \lambda_1 \mathbf{RE}_f + \lambda_2 \mathbf{RE}_s + \lambda_3 \mathbf{FC}_f + \lambda_4 \mathbf{FC}_s - \lambda_5 \mathbf{CS}$.
- +**CY** (Fold2Seq): We add the cyclic loss into the former model with the final loss $L = \lambda_1 \mathbf{RE}_f + \lambda_2 \mathbf{RE}_s + \lambda_3 \mathbf{FC}_f + \lambda_4 \mathbf{FC}_s + \lambda_5 (\mathbf{CY} - \mathbf{CS})$.

We use native sequence recovery rate as the assessment metric. The performance on two test sets is summarized in Table 2. Our **key** observations are: (i) By changing 'string' to 'voxel' and adding 2 **FC** losses, the performance has the largest improvement by 3-4%. (ii) By adding the cyclic loss (**CY**), the performance also improves around 2%. (iii) In contrast, by adding $\mathbf{RE}_s$ and **CS**, the improvement is trivial. It also shows that including the two **FC** losses as the intra-domain loss is crucial for the joint embedding learning. In conclusion, our novel design of the 3D voxel representation and the joint embedding learning framework, which includes intra-domain and cyclic losses, results in significant improvement of the performance.

5 Conclusion

In this paper, we design novel neural network model to learn a fold representation from the 3D voxels of density of secondary structure elements. In order to mitigate the heterogeneity between the sequence domain and the fold domain, we learn the joint sequence-fold representation through novel intra–domain and cross–domain losses. Our model, Fold2Seq outperforms existing data-driven methods and the state-of-the-art principle-driven method RosettaDesign, in terms of perplexity, sequence recovery rate, and structural recovery accuracy. We also show that our method is significantly faster compared to RosettaDesign. Ablation study shows that this superior performance can be directly attributed to our novel algorithmic innovations, including the fold representation, joint sequence-fold embedding, and various losses. Future work will focus on bridging the performance gap between ID and OD test sets, improving the fold representation by using ensemble of input structures, accounting for additional constraints (other than fold) for sequence design, and finally extending Fold2Seq to novel structure design.

REFERENCES

Relja Arandjelovic and Andrew Zisserman. Objects that sound. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 435–451, 2018.

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1): 235–242, 2000.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Yue Cao and Yang Shen. Tale: Transformer-based protein function annotation with joint sequence–label embedding. *bioRxiv*, 2020. doi: 10.1101/2020.09.27.315937. URL https://www.biorxiv.org/content/early/2020/09/28/2020.09.27.315937.

Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. In *Asian Conference on Computer Vision*, pp. 100–116. Springer, 2018.

Payel Das, Kahini Wadhawan, Oscar Chang, Tom Sercu, Cicero Dos Santos, Matthew Riemer, Vijil Chenthamarakshan, Inkit Padhi, and Aleksandra Mojsilovic. Pepcvae: Semi-supervised targeted design of antimicrobial peptide sequences. *arXiv preprint arXiv:1810.07743*, 2018.

Laurianne David, Amol Thakkar, Rocío Mercado, and Ola Engkvist. Molecular representations in ai-driven drug discovery: a review and practical guide. *Journal of Cheminformatics*, 12(1):1–22, 2020.

Pierre Dognin, Igor Melnyk, Youssef Mroueh, Jerret Ross, and Tom Sercu. Adversarial semantic alignment for improved image captions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10463–10471, 2019.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*, 2018.

Joe G Greener, Lewis Moffat, and David T Jones. Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific reports*, 8(1):1–12, 2018.

Jiuxiang Gu, Jianfei Cai, Shafiq R Joty, Li Niu, and Gang Wang. Look, imagine and match: Improving textual-visual cross-modal retrieval with generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7181–7189, 2018.

Jingtong Hou, Gregory E Sims, Chao Zhang, and Sung-Hou Kim. A global representation of the protein fold space. *Proceedings of the National Academy of Sciences*, 100(5):2386–2390, 2003.

Po-Ssu Huang, Yih-En Andrew Ban, Florian Richter, Ingemar Andre, Robert Vernon, William R Schief, and David Baker. Rosettaremodel: a generalized framework for flexible backbone protein design. *PloS one*, 6(8):e24109, 2011.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. In *Advances in Neural Information Processing Systems*, pp. 15820–15831, 2019.

Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers: Original Research on Biomolecules*, 22(12):2577–2637, 1983.

Mostafa Karimi, Shaowen Zhu, Yue Cao, and Yang Shen. De novo protein design for novel folds using guided conditional wasserstein generative adversarial networks. *Journal of Chemical Information and Modeling*, 2020.

Sofia Khan and Mauno Vihinen. Performance of protein stability predictors. *Human mutation*, 31 (6):675–684, 2010.

Nobuyasu Koga, Rie Tatsumi-Koga, Gaohua Liu, Rong Xiao, Thomas B Acton, Gaetano T Montelione, and David Baker. Principles for designing ideal protein structures. *Nature*, 491(7423): 222–227, 2012.

Christina M Kraemer-Pecore, Andrew M Wollacott, and John R Desjarlais. Computational protein design. *Current Opinion in Chemical Biology*, 5(6):690–695, 2001.

Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. *arXiv preprint arXiv:2004.03497*, 2020.

Alexey G Murzin, Steven E Brenner, Tim Hubbard, Cyrus Chothia, et al. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, 1995.

Saul B Needleman and Christian D Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *ICML*, 2011.

Christine A Orengo, Alex D Michie, Susan Jones, David T Jones, Mark B Swindells, and Janet M Thornton. Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pp. 8026–8037, 2019.

Yuxin Peng and Jinwei Qi. Cm-gans: Cross-modal generative adversarial networks for common representation learning. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(1):1–24, 2019.

Gabriel J Rocklin, Tamuka M Chidyausiku, Inna Goreshnik, Alex Ford, Scott Houliston, Alexander Lemak, Lauren Carter, Rashmi Ravichandran, Vikram K Mulligan, Aaron Chevalier, et al. Global analysis of protein folding using massively parallel design, synthesis, and testing. *Science*, 357 (6347):168–175, 2017.

Ian Sillitoe, Natalie Dawson, Tony E Lewis, Sayoni Das, Jonathan G Lees, Paul Ashford, Adeyelu Tolulope, Harry M Scholes, Ilya Senatorov, Andra Bujan, et al. Cath: expanding the horizons of structure-based functional annotations for genome sequences. *Nucleic acids research*, 47(D1): D280–D284, 2019.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pp. 935–943, 2013.

Alexey Strokach, David Becerra, Carles Corbi-Verge, Albert Perez-Riba, and Philip M Kim. Fast and flexible design of novel proteins using graph neural networks. *bioRxiv*, pp. 868935, 2020.

Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

William R Taylor. A 'periodic table'for protein structures. *Nature*, 416(6881):657–660, 2002.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1499–1509, 2015.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*, 2018.

Kaiye Wang, Ran He, Wei Wang, Liang Wang, and Tieniu Tan. Learning coupled feature spaces for cross-modal matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2088–2095, 2013.

Jianyi Yang, Renxiang Yan, Ambrish Roy, Dong Xu, Jonathan Poisson, and Yang Zhang. The i-tasser suite: protein structure and function prediction. *Nature methods*, 12(1):7–8, 2015.

Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

Ziming Zhang and Venkatesh Saligrama. Zero-shot learning via joint latent similarity embedding. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6034–6042, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

A 3D EXTENSION OF THE SINUSOIDAL ENCODING

We use a simple extension of the sinusoidal encoding described in the original transformer model (Vaswani et al., 2017) to encode each position in our Structure Encoder.

$$\begin{aligned} \text{PE}(x, y, z, 2i) &= \sin(x/10000^{2i/h}) + \sin(y/10000^{2i/h}) + \sin(z/10000^{2i/h}) \\ \text{PE}(x, y, z, 2i+1) &= \cos(x/10000^{2i/h}) + \cos(y/10000^{2i/h}) + \cos(z/10000^{2i/h}) \end{aligned} \tag{5}$$

B DATASET STATISTICS

The statistics of our various datasets are given below.

- Training set includes 45995 proteins belonging to total 1093 folds.
- Validation set includes 1590 proteins belonging to total 256 folds.
- In-distribution (ID) test set includes 1230 proteins belonging to total 230 folds.
- Out-of-distribution (OD) test set includes 204 proteins belonging to 28 folds.

C DISTRIBUTIONS OVER ΔTM

For the ith protein, we define $\Delta\text{TM}_i = \text{TM}(\boldsymbol{s}_i^{\textbf{Fold2Seq}}, \boldsymbol{s}_i^{\textbf{Native}}) - \text{TM}(\boldsymbol{s}_i^{\textbf{Rosetta}}, \boldsymbol{s}_i^{\textbf{Native}})$. The distributions of ΔTM for the ID and OD tests are shown in Figure 4.

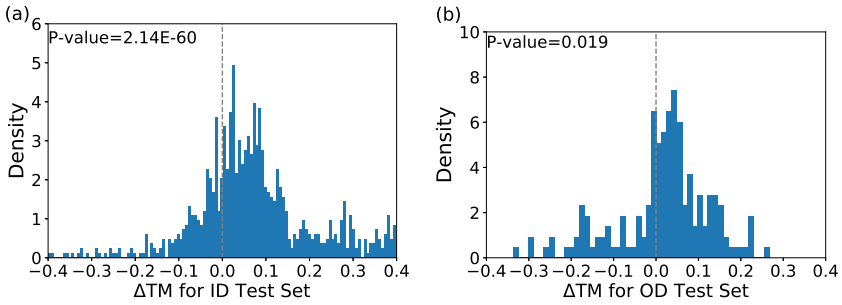

Figure 4: The distributions of ΔTM for two test sets.

D COMPARISON OF FOLDED STRUCTURES

In this section, we show some representative folded structures whose sequences are designed by RosettaDesign and Fold2Seq. The folded structures were predicted using iTasser, a state of the art program for protein structure prediction. Figure 5 shows some structures where Fold2Seq performs better than RosettaDesign and Figure 6 shows some structures where RosettaDesign is better.

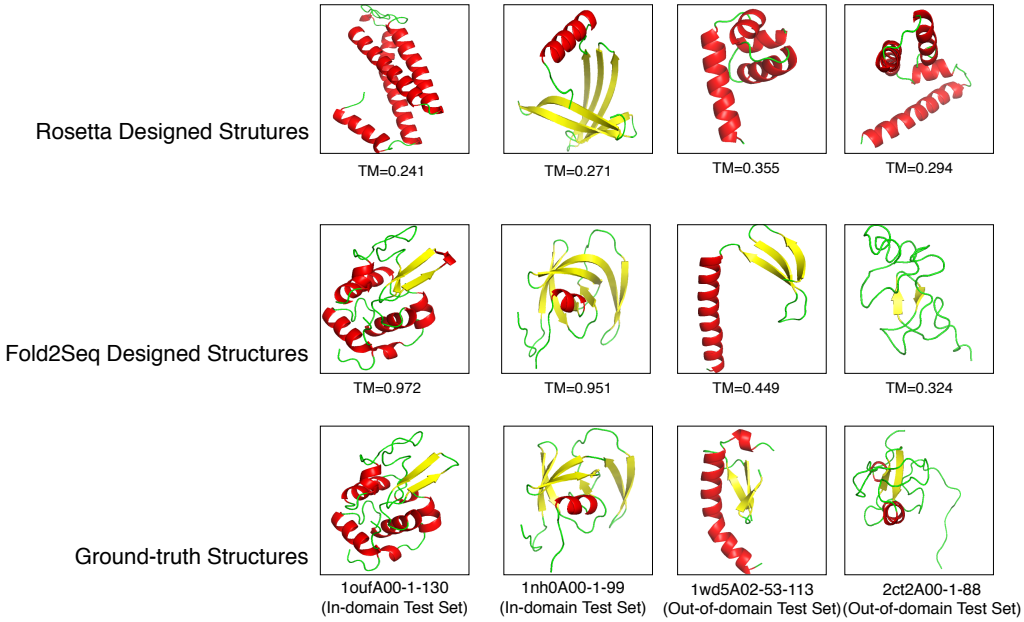

Figure 5: The native and designed structures with $\Delta\text{TM}_i > 0$. The IDs at the bottom are the CATH domain names of each structure.

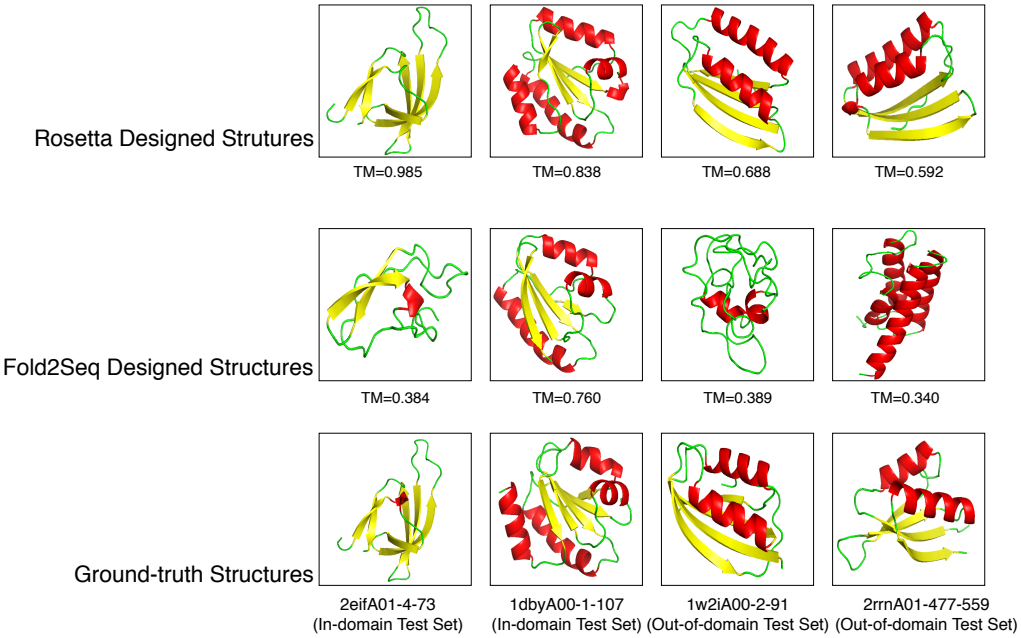

Figure 6: The native and designed structures with $\Delta\text{TM}_i < 0$. The IDs at the bottom are the CATH domain names of each structure.