Graph convolutional network for protein sequence sampling and prediction using structural data

Gabriel A. Orellana
Protera Biosciences
Av. Santa Maria 2810
Providencia, Santiago, Chile
gorellana@proterabio.com

Javier Caceres-Delpiano
Protera Biosciences
Av. Santa Maria 2810
Providencia, Santiago, Chile
jcaceres@proterabio.com

Roberto Ibañez
Protera Biosciences
176 Avenue Charles de Gaulle
92522 Neuilly Sur Seine Cedex, Paris, France
ribanez@proterabio.com

Leonardo Alvarez*
Protera Biosciences
176 Avenue Charles de Gaulle
92522 Neuilly Sur Seine Cedex, Paris, France
leonardo@proterabio.com

Abstract

The increasing integration between protein engineering and machine learning has led to many interesting results. A problem still to solve is to evaluate the likelihood that a sequence will fold into a target structure. This problem can be also viewed as sequence prediction from a known structure.

In the current work, we propose improvements in the recent architecture of Geometric Vector Perceptrons [1] in order to optimize the sampling of sequences from a known backbone structure. The proposed model differs from the original in that there is: (i) no updating in the vectorial embedding, only in the scalar one, (ii) only one layer of decoding. The first aspect improves the accuracy of the model and reduces the use of memory, the second allows for training of the model with several tasks without incurring data leakage.

We treat the trained classifier as an Energy-Based Model and sample sequences by sampling amino acids in a non-autoreggresive manner in the empty positions of the sequence using energy-guided criteria and followed by Monte Carlo optimization. We improve the median identity of samples from 40.2% to 44.7%.

An additional question worth investigating is whether sampled and original sequences fold into similar structures independent of their identity. We chose proteins in our test set whose sampled sequences show low identity (under 30%) but for which our model predicted favorable energies. We used trRosetta server [2] and observed that the predicted structures for sampled sequences highly resemble the predicted structures for original sequences, with an average TM score of 0.848.

1 Introduction

Proteins are one of the most interesting and highly studied macromolecules due to their diverse functional features. Protein sequences strictly determine the spontaneous folding that characterize their 3D structures. Different groups have shown applications of Deep Learning (DL) methods to major topics in Biochemistry. An example of this is the protein folding problem, making it possible to predict structure of proteins just from amino acid sequences [2, 3].

*Corresponding author

Preprint. Under review.

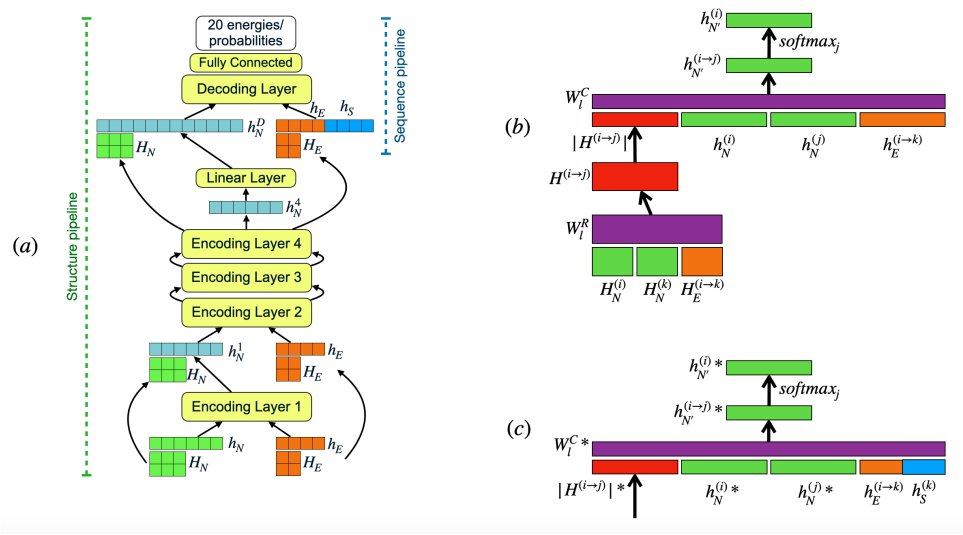

Figure 1: **(a)** Full network diagram integrating Encoder and Decoder layers from feature inputs to probability/energy **(b)** General architecture of an encoder layer, and **(c)** General architecture of decoder layer. Node features on green, edge features on orange, vector convolution output on red, sequence embeddings on blue, trainable weights on purple. Parameters in decoder layer with a * have a dimension of double the size as in encoder layers. In (a), length of pipelines for structure and sequence information can be seen on the sides of the full network diagram.

A relatively new approach for modeling proteins involves the use of graphs. Proteins can be seen as points in the space connected by different kinds of interactions. This information can be represented in a graph that consists of two sets, a set of nodes and a set of edges composed by pairs of nodes. [4] proposed an architecture based on the concept of Graph Convolutions, leading to new architectures named Graph Convolutional Networks (GCN).

The goal of the current work is to develop a GCN model and adjacent algorithms that address the problem of amino acid sequence prediction and sequence sampling given the structure of a backbone. This work is inspired by existing work [5, 1], but prioritizing the optimization of sampled sequences.

Similar works in the literature [5, 1] have trained their models in the task of Autoregressive Single Amino acid Prediction (ARSAP), consisting of predicting the current amino acid on a protein sequence, given the real protein structure and the real sequence prior to the current amino acid. ARSAP and identity between original and sampled sequences are the commonly used metrics. Previously used sample procedure [5] consists of predicting amino acids in an autoregressive (AR) manner (similar to ARSAP but with cumulative error). We propose to consider the logits in our model as a negative energy in the same fashion as it is used in Energy Based Models (EBM) [6], and then iteratively sample the amino acid with the lowest energy until all the sequence is filled.

Finally, we use trRosetta [2] to predict the structure of some of our sampled sequences and show that even in cases when identities are low, if energy of sequences is low, the structure of the sampled sequence predicted by trRosetta is highly similar to the predicted structure of the original sequence.

The contributions of this work are four: *(i)* we propose a new GCN layer architecture taking as starting point the one proposed by [5] along with the improvements made by [1], increasing the performance in ARSAP and sequence sampling; *(ii)* using the advantages given by the proposed architecture, we train our model with different auxiliary tasks, improving its performance in the two metrics evaluated; *(iii)* by exploiting some characteristics of our architecture we implement an energy-guided sampling procedure that outperformed the AR procedure in the sequence identity metric; *(iv)* we show that our model is able to generate sequences that fold into structures highly similar to the target structures even in cases when these sequences are not similar.

Table 1: Results in Autoregressive Single Amino acid Prediction (ARSAP) and Sequence Sampling with metaparameter k=30. [1]: Model proposed in [5]; [2]: Model proposed in [1]

	ARSAP		Sampling	
	Perplexity	Accuracy (%)	Identity short seqs (%)	Identity total (%)
AttGraph[1]	6.93	39.4	28.3	36.4
GVP[2]	5.29	46.6	32.1	40.2
AR Ours	5.43	46.6	35.5	42.4

2 Methods

Graph features. Following several works in modeling proteins as graphs [5, 7, 8, 9] we considered the use of node features, accounting for the features of single amino acids, and edge features, accounting for interactions between them. [5], based on the importance of close interactions in proteins, proposes that each node keeps the connections to its k-nearest neighbors in atomic distance. We performed tests to evaluate the optimal value of this meta-parameter and found it was $k = 35$.

Based on the recent success in the use of 3D vectors as complementary features for the this problem, presented in [1], we include two extra sets of features that corresponds to the 3D vector representation of *(i)* the normalized distances between all the atoms in the backbone for each amino acid, included as node features; and *(ii)* the normalized distances between the $C\alpha$ of each amino acid and the $C\alpha$ of the k nearest neighbors, included as edge features.

The four kinds of features (vectorial and scalar node features plus vectorial and scalar edge features) and the amino acid sequence are then encoded using a standard trainable embedding and introduced into the model, which is trained to output one value per each possible amino acid (20 in total) for each position in the sequence following a classification paradigm.

Architecture. Inspired by the graph network application to this problem shown in [5] and the improvements made in [1], we propose a GCN with four layers of only structural information (encoding layers) and a single layer of both structural and sequence information (decoding layer). This design decision differs from the architectures previously cited, where the number of encoding and decoding layers were the same. We proposed this change expecting: *(i)* more dependence on structural information. We expect the performance to decrease on the ARSAP task, but increase or maintain in the sequence sampling task, since the latter relies heavily on structural information while the former uses both structure and sequence information; *(ii)* more flexibility in the training of our model, since we have just one decoding layer, there is no problem of data leakage with masks different that AR mask; and *(iii)* more speed in sampling because, for each step in the process the inputs of the decoding layer are the same and, therefore, just that layer needs to be executed.

Graph layer. The functioning of layers used in our architecture is described in pseudocode in appendix A and displayed in figure 1,

Loss functions. Under the regular probabilistic setting, the model is trained to approximate the probabilistic distributions of the target. Considering the use of the softmax function, equation 1 describes the loss function L_1, with q_a as the logits for class a. In case of the EBM, the loss function attempts to minimize the energy of the target class while increasing the energies of the rest. Here we use a function L_2 from the family of Contrastive Free Energy losses [10] (equation 2).

$$L_1 = -\log(\frac{\exp^{q_y}}{\sum_a \exp^{q_a}}) = -q_y + \log(\sum_a^A \exp^{q_a}) \tag{1}$$

$$L_2 = G_y(W, X) + \frac{1}{\beta}\log(\sum_a^A \exp^{-\beta G_a(W,X)}) \tag{2}$$

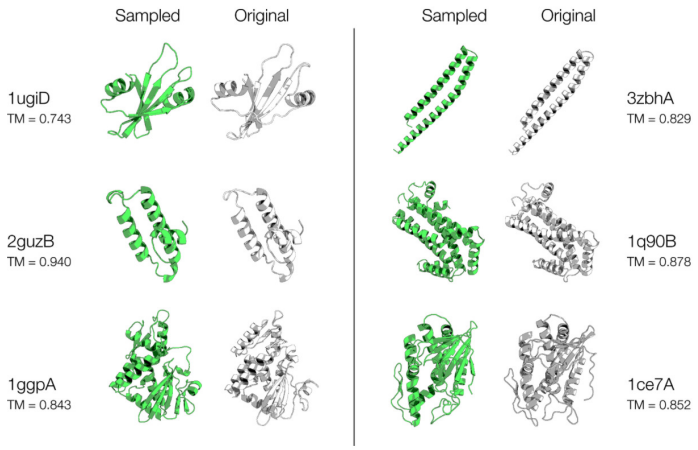

Figure 2: Alignment of structures generated by trRosetta from original (white) and sampled (green) sequences. PDB codes and chains are shown for each structure, as well as the calculated TM-score between original and sampled structures.

As it can be seen, equations 1 and 2 are very similar, differing in the sign on the energy factors. Based on this similarity we employ the logits trained in the standard probabilistic setting as negative energy in an EBM setting.

Tasks. We describe the main task used in previous works [5, 1] as well as the 3 auxiliary tasks employed in this work in appendix B.

Evaluation of energy for Sequence-Structure tuple (ESST). We define a formula for calculating the energy of a sequence and a structure using our model (see appendix C).

Sampling procedure. We believe that the AR sampling algorithm used in literature [5, 1] does not take advantage of two things: *(i)* As we described previously, normal sampling is inefficient because it needs to run the entire forward of the protein at every step. In order to deal with this we modified the architecture to have just one decoding layer and modified the sampling procedure to run the encoding part of the forward just once. *(ii)* AR sampling is forced to sample the next position in the sequence instead of other positions that may have less energy (more certainty). Based on this, we develop a non-AR Energy-Based sample procedure (see appendix D)

3 Results

Following the setting of [5] and also adopted [1], i.e. ARSAP task with metaparameter $k = 30$, we evaluate the performance of the three models in table 1. Regarding the performance of our model with the proposed additions, table S2 displays the results of our model trained in the main task and the addition of the auxiliary tasks described in the previous section, with a parameter $k = 35$. The model trained with all the 4 tasks proposed achieves a perplexity of 5.34 and an accuracy of 47.2%. On the other hand, table S3 shows the results on the Sampling Sequence task, comparing the model trained with the different tasks and using AR and non-AR sampling procedures. The non-AR procedure using the model trained with 4 tasks achieves an average identity on the whole test set of 44.3% and just on short sequences of 35.1%. For calculating metrics in tables S2 and S3, we removed proteins with His-tags from training, validation and test data sets (around 20%). The effect of this modification can be seen in table S1.

4 Discussion

Based on our interpretation of the logits as the negative of the energy, a natural point of interest is whether the average energy of sampled sequences is correlated with performance. We found a high correlation between these two variables with a Pearson's r of -0.8778 (figure S1).

In the works directly related to the solution of the problem of sequence sampling from a structure [1, 5], authors focus on metrics of similarity between original sequence and the sampled sequences generated by the models. This is natural since the models are trained using the original sequences as the target, hence a similarity metric between them and the sampled sequences can be seen as a metric of how successful the training was. An important question unaddressed in these works is whether sequences generated by the models could exhibit low levels of identity with the original sequence and still generate 3D foldings with high similarity with the input structure. There is evidence of some structural similarity between foldings of sequence with identities of 20% [11]. If these cases occur, it would imply that the model is able to learn how to produce a similar structure using different patterns of amino acids in the sequence. A way to predict the performance of the model on an input structure is to consider how low an energy it assigns to each pair of input structure and sampled sequence. As a way to obtain an anecdotal insight of this phenomena, we select a subset of proteins in the test set where their sampled sequences have the lowest energy and identities lower than 30% and use trRosetta [2] to predict their structure as well the structure of the original sequence (see figure 2 and tables S4 and S5), obtaining an average TM score of 0.848.

References

[1] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael JL Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. *arXiv preprint arXiv:2009.01411*, 2020.

[2] 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*, 117(3):1496–1503, 2020. Publisher: National Acad Sciences.

[3] 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*, pages 1–11, 2021.

[4] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[5] John Ingraham, Vikas Kamur Garg, Regina Barzilay, and Tommi S. Jaakkola. Generative models for graph-based protein design. 2021. Publisher: Neural Information Processing Systems Foundation, Inc.

[6] Yann LeCun, Sumit Chopra, Raia Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[7] John Ingraham, Adam Riesselman, Chris Sander, and Debora Marks. Learning protein structure with a differentiable simulator. In *International Conference on Learning Representations*, 2018.

[8] Alex M. Fout. *Protein interface prediction using graph convolutional networks*. PhD Thesis, Colorado State University, 2017.

[9] Rafael Zamora-Resendiz and Silvia Crivelli. Structural learning of proteins using graph convolutional neural networks. *bioRxiv*, page 610444, 2019. Publisher: Cold Spring Harbor Laboratory.

[10] Yann LeCun and Fu Jie Huang. Loss Functions for Discriminative Training of Energy-Based Models. In *AIStats*, volume 6, page 34. Citeseer, 2005.

[11] Burkhard Rost. Twilight zone of protein sequence alignments. *Protein engineering*, 12(2):85–94, 1999. Publisher: Oxford University Press.

[12] Andriy Kryshtafovych, Torsten Schwede, Maya Topf, Krzysztof Fidelis, and John Moult. Critical assessment of methods of protein structure prediction (casp)—round xiii. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1011–1020, 2019.

Appendices

A Graph convolutional layer

The functioning of layers used in our architecture is described on pseudocode in algorithm A.1 and displayed in figure 1.b and 1.c, where h_N are the scalar node features and h_E are scalar edge features, both $\in \mathbb{R}^S$, while H_N and H_E are the vectorial node and edge features respectively, both $\in \mathbb{R}^{V\times 3}$. being S the hidden dimension of scalar features and V the hidden dimension of vectorial features.

Algorithm A.1: Graph convolutional layer

Input: h_N, h_E, H_N, H_E
$h'_N = GCU(h_N, h_E, H_N, H_E)$
$h_N = norm_1(h_N + Dropout(h'_N))$
$h_N = norm_2(dense(h_N))$
return h_N

With GCU being the Graph convolutional unit, $norm_1$ and $norm_2$ are standard normalization layers with trainable terms for gain and bias; and $dense$ is a standard Position-wise Feedforward layer used as a dense layer. As it can be seen, in contrast with [1], our model does not update both scalar and vectorial node features in each layer, but just the scalar node features. This also allows us to keep the standard dense layers used in [5]. The GCU is shown in pseudocode on algorithm A.2, where K is the hyperparameter for the number of nearest neighbors nodes considered in the elaboration of the graph, while $W_R \in \mathbb{R}^{S\times(3V)}$, $W_C \in \mathbb{R}^{(4S)\times S'}$ and $W_{out} \in \mathbb{R}^{S'\times S}$ are weight matrices with S' as an intermediate hidden layer dimension.

Algorithm A.2: Graph convolutional unit

Input: h_N, h_E, H_N, H_E
$H^{(i\to k)} = W_R \star concat(H_N^{(i)}, H_E^{(i\to k)}, H_N^{(k)})$
${h'_N}^{(i\to k)} = concat(h_N^{(i)}, h_E^{(i\to k)}, h_N^{(k)}) \star W_C$
${h'_N}^{(i)} = \sum_k^K {h'_N}^{(i\to k)} \times \frac{\exp({h'_N}^{(i\to k)})}{\sum_{k'}^K \exp({h'_N}^{(i\to k')})}$
${h'_N}^{(i)} = W_{out} \star {h'_N}^{(i)} + b$
return h'_N

For the decode layer, the sequence information $h_{seq}^{(k)}$ is concatenated to $h_E^{(i\to k)}$ and then introduced into the regular data flow (see figure 1.b). The hidden dimension of scalar features is then duplicated in order to fit the size of the data input using a standard linear layer.

B Tasks

The regular task associated with this problem and already described in [5] is the autoregressive single amino acid prediction, that can be stated as:

$$P(seq_i | seq_{[1,\ldots,i-1]} \cup struc) \tag{B.1}$$

Where seq_i is the sequence in the i position. In notation of Energy-Based Model, we can describe it as:

$$softmax_a(-E(W, i, a, struc, seq_{[1,\ldots,i-1]})) \tag{B.2}$$

In our study we employ three additional tasks:

1. Knowing the following amino acids in the sequence, but no the previous ones (inverse autoregressive prediction): $P(seq_i|seq_{[i+1,\dots,N]} \cup struc)$ or $softmax_a(-E(W, i, a, struc, seq_{[i+1,\dots,N]}))$
2. Knowing all the amino acids in the sequence except the current one: $P(seq_i|seq - \{seq_i\} \cup struc)$ or $softmax_a(-E(W, i, a, struc, seq - \{seq_i\}))$
3. Knowing none of the amino acids in the sequence: $P(seq_i|struc)$ or $softmax_a(-E(W, i, a, struc, \emptyset))$

It is worth mentioning that the three additional tasks can be employed in our model due to the decision of having just one decoding layer (see figure 1.b), otherwise the training would suffer from data leakage.

C ESST

We consider the tuple of a structure Str and a sequence Seq. We run the tuple through the model in evaluation mode in the same configuration as in the auxiliary task 2. We define function for belonging to sequence seq in position i for an amino acid a as

$$A(seq, i, a) = \begin{cases} 1, & \text{if } seq_i = a \\ 0, & \text{if } seq_i \neq a \end{cases} \tag{C.1}$$

Then we define:

$$ESST(W, struc, seq) = \sum_{i=1}^{N} \sum_{a=1}^{20} E(W, a, i, struc, seq - \{seq_i\}) \times A(seq, i, a) \tag{C.2}$$

Being the summation of energies only from amino acids in sequence seq given its immediate surrounding in sequence and structure.

D Energy-Based Sampling

The initial step in the Energy-Based Sampling procedure is to start from an empty sequence and sample single amino acids based on their energy in a non-AR manner. We call this procedure **Energy-Guided Sampling** (EGS) and it is explained in algorithm D.1.

Algorithm D.1: Energy-Guided Sampling

Input: $model, struct, lenghtSeq$
$seq \leftarrow zeros(lengthSeq)$
$mask \leftarrow zeros(lengthSeq)$
for *i in lengthSeq* **do**
  $energies \leftarrow model(seq, struct)$
  $sortedEnergies \leftarrow sort(energies, axis = 2)$
  $diffEnergies \leftarrow sortedEnergies[:, 1] - sortedEnergies[:, 2]$
  $minSpotIndex \leftarrow argmin(diffEnergies + mask \times B)$
  $minAAIndexx \leftarrow argmin(energies[minSpotIndex, :])$
  $seq[minSpotIndexx] = minAAIndex$
  $mask[minSpotIndex] = 1$
end
return seq

Following EGS we expect that, according to the model, the sampled amino acids are optimal given the surroundings at the point when they are sampled. On the downside, once all the amino acids are sampled, it is not guaranteed that amino acids are optimal given their surroundings since they

Algorithm D.2: Energy-Guided Mutation Optimization

Input: $model, struct, seq$
$energies \leftarrow model(seq, struct)$
$minEnergies \leftarrow min(energies, axis = 2)$
$spots \leftarrow minEnergies \neq energies[seq]$
$spots \leftarrow shuffle(spots)$
while *not isempty(spots)* **do**
  $spotIndex \leftarrow pop(spots)$
  $newSeq[spotIndex] = argmin(energies[spotIndex])$
  if *esst(model,struct,seq)* $>$ *esst(model,struct,new*$_s eq$) **then**
    $seq \leftarrow newSeq$
    $energies \leftarrow model(seq, struct)$
    $minEnergies \leftarrow min(energies, axis = 2)$
    $spots \leftarrow minEnergies \neq energies[seq]$
    $spots \leftarrow shuffle(spots)$
  end
end
return seq

changed from the moment where the amino acids were sampled. Considering this we employ a step of **Energy-Guided Mutation Optimization** (see algorithm D.2).

After the two steps of Energy-Guided Sampling and Optimization, the optimality of the energy of sequence is still not guaranteed. We implement a final step of Random Mutation Optimization (RMO), when we generate a high number ($N = 1000$) of random single mutations and evaluate whether they improve the energy of the sequence employing ESST.

Supplementary information

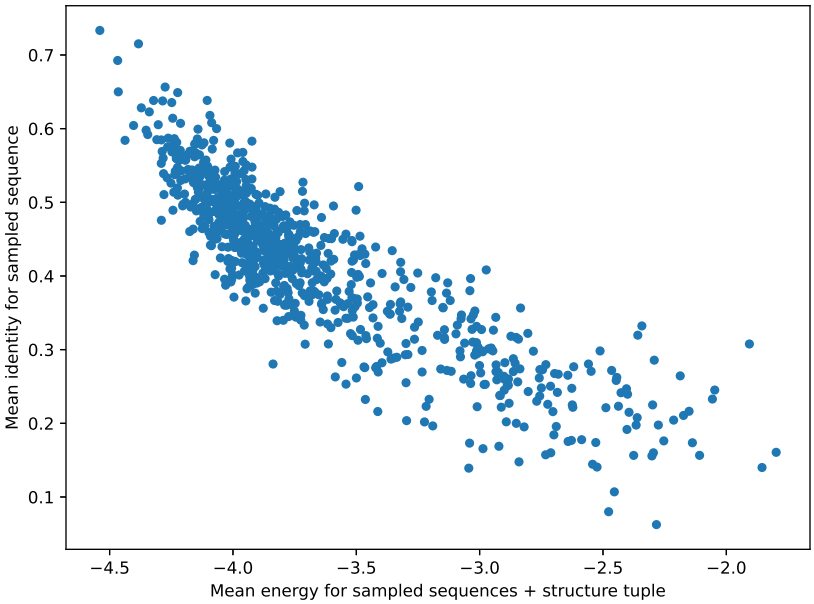

Figure S1: Mean protein identity in the test data set compared to their Energy of Sequence-Structure Tuple (ESST). Pearson's r of -0.8778.

Table S1: Results on both tasks comparing our final version of our model trained and tested in the CASP dataset [12] with and without proteins with histags. Model were trained with main (ARSAP) task only. Accuracy and identities are presented in percentage.

	ARSAP		Sampling	
	Perplexity	Accuracy	Ident. short seqs	Ident. total
Ours with His-tag proteins	5.40	47.0	34.0	41.7
Ours without His-tag proteins	5.52	46.4	33.4	41.2

Table S2: Results in Autoregressive Single Amino Acid Prediction. Our model was trained in the main task and an increasing number of auxiliary tasks. Meta parameter k=35. Databases for this measure were modified, removing all the protein with His-tags from training, validation and test subsets.

Training	Perplexity	Accuracy (%)
Main task only	5.52	46.4
Main + aux task 1	5.64	46.2
Main + aux tasks 1, 2	5.40	46.7
Main + aux tasks 1, 2, 3	5.44	46.6

Table S3: Results in Autoregressive (AR) or non-Autoregressive (nAR) sequence sampling tasks. Model was trained in the main task and an increasing number of auxiliary tasks. Meta parameter k=35. Databases for this measure were modified, removing all the protein with His-tags from training, validation and test subsets.

	Ident. short seq (%)		Ident. total (%)		Energy	
	Mean	Median	Mean	Median	Mean	Median
AR main task only	33.9	32.4	41.5	43.1	-3.61	-3.76
nAR main task only	33.4	31.7	41.2	42.5	-3.79	-3.97
nAR main + aux task 1	34.5	32.9	41.6	43.2	-3.83	-4.01
nAR main + aux tasks 1, 2	34.7	32.9	42.1	43.9	-3.56	-3.74
nAR main + aux tasks 1, 2, 3	34.4	32.5	42.3	44.1	-3.69	-3.86

Table S4: Statistics of sequences sampled from selected proteins used in analysis with trRosetta. Statistics correspond to the mean of the 10 sampled sequences.

Protein	Length	Energy	Identity (%)
1ugi.D	82	-3.635	29.27
2guz.B	65	-3.518	29.54
1ggp.A	234	-3.415	27.09
3zbh.A	90	-3.321	26.00
1q90.B	212	-3.226	28.21
1ce7.A	241	-3.224	28.96

Table S5: Metrics of individual sequences sampled from selected proteins used in analysis with trRosetta. Metrics corresponding to sequences in 3 stages of sequence sampling procedure (EGS, EGMO, RMO).

	1st step			2nd step			3rd step		
	Energy	Identity (%)	TM-Score	Energy	Identity (%)	TM-Score	Energy	Identity (%)	TM-Score
1ugi.D	-3.609	30.5	0.773	-3.748	26.8	0.607	-3.779	26.8	0.743
2guz.B	-3.490	29.3	0.908	-3.595	30.8	0.950	-3.617	30.8	0.940
1ggp.A	-3.429	26.9	0.888	-3.464	26.5	0.849	-3.478	26.1	0.843
3zbh.A	-3.342	25.6	0.819	-3.377	26.7	0.854	-3.384	27.8	0.829
1q90.B	-3.232	27.8	0.799	-3.250	28.8	0.800	-3.290	28.8	0.878
1ce7.A	-3.255	29.0	0.860	-3.297	28.6	0.851	-3.323	29.9	0.852
Average			0.841			0.819			0.848