Variational Autoencoders for Protein Structure Prediction

Fardina Fathmiul Alam
Dept of Computer Science
George Mason University
falam5@gmu.edu

Amarda Shehu*
Dept of Computer Science
George Mason University
amarda@gmu.edu

ABSTRACT

The universe of protein structures contains many dark regions beyond the reach of experimental techniques. Yet, knowledge of the tertiary structure(s) that a protein employs to interact with partners in the cell is critical to understanding its biological function(s) and dysfunction(s). Great progress has been made *in silico* by methods that generate structures as part of an optimization. Recently, generative models based on neural networks are being debuted for generating protein structures. There is typically limited to showing that some generated structures are credible. In this paper, we go beyond this objective. We design variational autoencoders and evaluate whether they can replace existing, established methods. We evaluate various architectures via rigorous metrics in comparison with the popular Rosetta framework. The presented results are promising and show that once seeded with sufficient, physically-realistic structures, variational autoencoders are efficient models for generating realistic tertiary structures.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Applied computing** → **Molecular structural biology**; **Bioinformatics**.

KEYWORDS

protein structure prediction; tertiary structure; generative modeling; variational autoencoders.

ACM Reference Format:
Fardina Fathmiul Alam and Amarda Shehu. 2020. Variational Autoencoders for Protein Structure Prediction. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '20), September 21–24, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3388440.3412471

1 INTRODUCTION

The protein universe contains many dark regions. Analysis of 546, 000 Swiss-Prot proteins estimates that 44–54% of these proteins in eukaryotes and viruses are beyond the reach of experimental techniques or homology modeling [25]; that is, not only do we not know what three-dimensional/tertiary structures these proteins employ to interact with other molecular partners in the cell, but such determination is challenging for wet and dry laboratories.

Knowledge of the biologically-active/native tertiary structure(s) of a protein molecule is critical towards understanding its array of biological functions as well as possible dysfunction in the living cell [19]. Shining a light on the dark proteome is a great motivation for computational approaches to tackle the problem of template-free protein structure prediction (PSP) necessitated by the dark proteome. In this problem, the only direct information about a target protein is its amino-acid sequence.

Great progress has been made in template-free PSP. A detailed review is beyond the scope of this paper, but we note a key characteristic shared by popular methods and software platforms, such as Rosetta [15], Quark [32], and others [24, 33]. These methods operate under the umbrella of stochastic optimization. They generate tertiary structures of a given amino-acid sequence as part of a process that seeks local optima of a given objective function. The latter is also referred to as an energy/scoring function, as it sums up the interactions among atoms in a given tertiary structure to provide an energy or score with each tertiary structure. The end result is that a distribution of low-scoring (near-optimal) tertiary structures are generated for a given amino-acid sequence. The most successful methods so far have relied on generating structures by assembling them from shorter structural fragments compiled *a priori* in fragment libraries extracted from known native structures in the Protein Data Bank (PDB) [4].

Fundamental challenges remain in optimization-based methods on how to balance between exploration and exploitation; exploration refers to exploring more of the structure space so as not to miss novel structures, and exploitation refers to improving structures so as to reach local optima of the scoring function. Methods driven by the optimization of an energy/scoring function do not learn not to generate structures that are unfit according to the scoring function. Related efforts have been made via adaptive search/active learning [29], but it remains challenging how to connect the structure generation mechanism with structure evaluation [19].

A machine learning (ML) framework would provide a natural setting, and ML models are increasingly showing promise. In Section 1.1 we summarize growing efforts in generative models. In particular, buoyed by advances in neural network (NN) research, NN-based models are increasingly being debuted for generating protein structures. However, their evaluation is typically limited to showing that some generated structures are protein-like.

In this paper, we go beyond this objective and evaluate the potential of generative models for PSP. Specifically, we evaluate whether variational autoencoders (VAEs) can replace existing, established

*Corresponding Author

BCB '20, September 21–24, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7964-9/20/09…$15.00
https://doi.org/10.1145/3388440.3412471

methods after being seeded with sufficient physically-realistic protein structures. We evaluate various VAE models via rigorous metrics in comparison with the popular Rosetta framework. The presented results are promising and show that once seeded with sufficient, physically-realistic structures, VAEs are efficient models for generating realistic tertiary structures.

1.1 Related Work

The employment of deep NNs has now a decade-long history in protein modeling. The earliest efforts saw the potential in deep NNs to learn from existing, experimentally-available protein sequences and structures deposited in biological databases to evaluate the quality of given tertiary structures. The bulk of the work, which presents a steadily-advancing thrust of research to this day, leverages representations of protein structures, such as pairwise distance maps or contact maps. The former records the pairwise distances between central atoms in each amino acid in a given tertiary structure. The latter adds a threshold-based filter and turns the distances into a contact/1 (distance is no higher than the threshold) or no contact/0.

Early work in [14] employed bidirectional recursive NNs (BRNNs). RaptorX-Contact introduced residual convolutional NNs (CNNs) [31], and DNNCON2 leveraged a two-stage CNN [1]. DeepContact introduced a 9-layer residual CNN with 32 filters [18]. DeepCOV reduced the amount of evolutionary information needed in the features [12]. PconsC4 further limited input features and improved prediction time [21]. SPOT-Contact extended RaptorX-Contact by adding a 2D-RNN stage downstream of the CNN [10]. TripletRes trained four CNNs end-to-end [17].

The performance of these models varies [30], but it is important to note that these methods do not, in current form, present alternatives to established platforms, such as Rosetta [15], Quark [32], and others, for the problem of generating credible tertiary structures given the amino-acid sequence. An important contribution in this regard was made by the recent AlphaFold method [28]. A deep NN in AlphaFold is used to predict a pairwise distance matrix for a given amino-acid sequence. Predicted distances are encapsulated in a penalty scoring function to bias a gradient descent-based optimization algorithm assembling short structural fragments into tertiary structures [28]. The fragments are enriched with novel ones generated by a generative NN.

While it remains more challenging to directly generate a distribution of tertiary structures via generative models, notable efforts have been made [11, 22, 23, 27]. Work in [11] does not provide an end-to-end NN-based framework for PSP, but composes a neural energy function with a Langevin dynamics-based simulator and is able to predict more than one type of structure for a given protein sequence. A body of recent works [22, 23, 27] represent the state-of-the-art on generative NN-based frameworks for problems related to PSP. However, these methods, based on the generative adversarial network (GAN) framework, generate structural fragments of an *a-priori* determined length (32, 64, or 128) that do not in themselves constitute tertiary structures of a given amino-acid sequence. That is, the model has to be retrained with the desired length, or one has to resort to methods that assemble fragments to stitch together tertiary structures for a desired amino-acid sequence. This body of work has also been content so far with showing that generated fragments seem protein-like, i.e., structurally-credible.

As laid out in Section 1, we are driven by the PSP problem in this work, and our objective is to generate a distribution of tertiary structures given an amino-acid sequence. We investigate VAEs of various architectures and evaluate them via several metrics. More importantly, our evaluation is driven by the question of whether VAEs can be as effective as Rosetta and Quark, which take a very different approach to PSP.

In Section 2, we relate details regarding our experimental setup, the architectures we design, the training process, the training data, and the evaluation metrics and protocols. The results are related in Section 3. Section 4 concludes the paper with a summary and an exposition of further directions of research.

2 METHODS

2.1 Experimental Setup

Let us delay details regarding the VAE framework and the various VAE models that we investigate in this paper for their utility in PSP in the interest of laying out the experimental setup. Let us broadly assume that we have a generative model "NewGenModel" that we are debuting for PSP. The model needs to learn from a given distribution of tertiary structures. In this paper, due to our experience with the publicly-available Rosetta platform, we generate N tertiary structures with the Rosetta AbInitio protocol. The protocol takes as input the amino-acid sequence for a given protein, as well as two libraries, one of structural fragments of 3 amino acids in length, and another of structural fragments of 9 amino acids, which are generated *a priori* for the given sequence via the Rosetta server. So, we train our "NewGenModel" over the N structures and then use it to generate N new structures.

The experimental setup investigates three basic questions:

(1) Did "NewGenModel" learn the input distribution? This is a basic property for a well-trained generative model. So, our first objective is to demonstrate that "NewGenModel" has learned the latent space and is generating tertiary structures that "resemble" the tertiary structures in the training dataset. We carry this out via various metrics that allow comparing two distributions.

(2) What is the quality of the tertiary structures generated by "NewGenModel"? This question needs exploring beyond the visualization of some generated structures and showing that they "look like" structures of proteins. While there can be many ways of evaluating the quality of a tertiary structure, often dependent on the application of interest, since our interest here is in the utility of generative models for PSP, we use the distance of a tertiary structure from the native structure as a proxy of its quality. We realize that this is a coarse proxy, but it is nonetheless informative and allows obtaining an aggregate view of the quality of structures generated by a model.

(3) Does "NewGenModel" have a sufficient-size training dataset to learn the latent space and to generate tertiary structures of good quality? This question effectively necessitates re-addressing the above questions as a function of increasing N. In the evaluation related in Section 3, we address this question at $N = 10,000$ and $N = 20,000$ tertiary structures.

(4) Even if the answers to (1) -(3) are satisfactory, a relevant question is what would be the motivation to adopt a generative model, which requires the additional burden of training over structures generated with an existing and established method. The answer to this question can be best addressed via a cost-benefit analysis. Methods such as Rosetta and others that assemble fragments to generate structures and seek local minima of an increasingly sophisticated energy function are expensive. They can take many minutes to generate one all-atom structure, and the time increases at least quadratically with the number of amino acids (mainly due to the cost of the energy function). After considering the generation of the training dataset as a form of pre-processing cost, a new generative model would only be useful if the cost to then generate a tertiary structure with it is lower than the cost to generate a tertiary structure with an energy function-minimizing method, while the quality of structures does not suffer. Therefore, an important evaluation is to compare the cost and the quality of N additional structures generated by "NewGenModel" with the cost and quality of N additional structures generated by Rosetta (or, more broadly, by an energy minimization-based method).

These four questions guide and structure our evaluation in Section 3. We now provide more details on the metrics used to compare two distributions, the overall VAE framework, the various models we evaluate, the training process and parameters, and the datasets employed for evaluation.

2.2 Metrics to Compare Two Distributions

As related above, it is important to assess whether the model has learned the input distribution by comparing the training to the generated/testing distribution. Since tertiary structures are multi-dimensional objects and one needs to additionally consider the issue of representation, we employ the root-mean-square-deviation (RMSD) as a proxy variable summarizing a tertiary structure.

RMSD is a common metric used in PSP to compare the distance of a tertiary structure to the native structure for a protein target. The metric is a version of the Euclidean distance averaged over the number of atoms considered after the two structures are superimposed optimally over each-other to remove differences due to rigid-body motions in SE3 (whole-body translation and rotation in three dimensions [20]. We utilize the RMSD distributions of two sets of structures, beyond the training and the generated dataset, as detailed in Section 3, as a means of comparing two sets of interest.

We make use of several metrics to compare two distributions, such as the Maximum Mean Discrepancy (MMD), the Bhattacharya distance (BD), and the Earthmover Distance (EMD), which we briefly summarize below.

Maximum Mean Discrepancy (MMD). The MMD metric allows measuring the distance between two distributions $p(x)$ and $q(y)$ and is based on kernel embedding of the distributions. Briefly, MMD is defined as the squared distance between the embedding in a reproducing kernel Hilbert space (RKHS): $\text{MMD}(p,q) = |\mu_x - \mu_y|^2_{\mathcal{H}}$ [9]. MMD has been recently used in training generative adversarial models [7, 16] to measure the distance of generated samples to some reference target set. Here, we follow work in [5] to use MMD for model selection, so we can distinguish between different VAE models. Specifically, rather than train a model using the MMD distance to a reference distribution (as opposed to KL divergence, for instance), we use MMD to evaluate the relative performance of various VAE models and find models that generate samples significantly closer to the reference/training distribution.

Bhattacharya Distance (BD). BD [13] measures the distance between two distributions $p(x)$ and $q(x)$ defined over the same domain X. It is defined as $DB(p,q) = -\ln(BC(p,q)$. The Bhattcharaya coefficient $BC(p,q)=\sum_{x\in X}\sqrt{p(x)q(x)}$. BC varies from 0 to 1. BD varies from 0 to ∞.

Earth Mover's Distance (EMD). EMD [26] is also known as the Wasserstein metric. EMD measures the distance between two probability distributions over a domain. If the distributions are interpreted as two different ways of piling up a certain amount of dirt over the domain, EMD returns the minimum cost of turning one pile into the other. The cost is assumed to be the amount of dirt moved times the distance by which it is moved. EMD can be computed by solving an instance of the transportation problem, using any algorithm for minimum cost flow problem, such as the network simplex algorithm [26].

2.3 Summary of the VAE Architecture

We have recently published work showing the ability of autoencoders (AEs) to learn low-dimensional representations of protein tertiary structures [2, 3]. In summary, an AE is composed of an encoder and a decoder. Each contain one or more layers of neurons/units. The encoder maps the input layer x to its output layer y. The decoder mirrors the encoder, mapping the same layer y to its output layer z. The learned code/representation consists of the layer y. When dealing with real values (such as Cartesian coordinates of atoms), training an AE involves optimizing the reconstruction error $||x - z||^2$.

A VAE is similar to an AE in that it consists of an encoder and decoder, but the training process in a VAE is regularized to avoid overfitting and so ensure that the latent space has the properties needed to enable generating data via the decoder [8]. In order to introduce the regularization of the latent space, the encoding-decoding process is modified. The input is encoded as a distribution over the latent space. A point from the latent space is sampled from that distribution. The sampled point is decoded, and the reconstruction error is computed. The reconstruction error is back-propagated as usual.

VAEs assume that the input distribution is Gaussian so that the encoder can be trained to return the mean and the covariance matrix. This assumption allows expressing very naturally the latent space regularization. The distribution returned by the encoder is also enforced to be close to Gaussian. To achieve this, loss function is modified to contain in addition to the reconstruction error a regularization term. The latter uses the Kullback-Leibler (KL) divergence between the returned distribution and a Gaussian; the KL divergence between two Gaussian distributions has a closed form expressed in terms of the means and the covariance matrices of the distribution. Specifically, the encoder assumes that $x \sim N(\mu_x, \sigma_x)$ and tries to learn the parameters μ_x and σ_x of the input distribution. The latent representation z is also assumed to be Gaussian; that is,

$z \sim N(\mu_{x,x})$. The loss function $L = |x-y|^2 + KL(N(\mu_{x,x}) - N(0, I))$. This loss function reflects the fact that to avoid having a VAE behave like an AE, one has to regularize both the covariance matrix and the mean of the distribution returned by the encoder. This is achieved by enforcing the distribution to be close to a standard normal distribution (centered and reduced). So, as the loss function shows, the covariance matrix is forced to be close to the identity, and the mean to be close to 0.

2.4 Designed VAE Models

What one controls in the architecture of a VAE is the number of layers, neurons per layer, and activation functions. With guidance from our previous work on AEs for learning low-dimensional embedding of tertiary structures [2], we investigate seven different models, as listed in Table 1. The notation summarizes a layer with the number of units/dimensions in it. For instance, in VAE2, there are m input units in the encoder that are then connected to the hidden layer of 2 units, which contains the latent encoding. The decoder is a mirror image of the encoder, overlapping with the encoder in the latent layer and then feeding the information to the output layer of 2 units in the VAE2 model.

VAE1	$m \rightarrow 250 \rightarrow 125 \rightarrow 2$
VAE2	$m \rightarrow 2$
VAE3	$m \rightarrow 10$
VAE4	$m \rightarrow 20$
VAE5	$m \rightarrow 30$
VAE6	$m \rightarrow 30 \rightarrow 20 \rightarrow 10 \rightarrow 2$
VAE7	$m \rightarrow m$

Table 1: VAE models listed here show the dimensionality of the layers. The decoder is a mirror image of the encoder.

The architectures are shown in Table 1 reflect some of the lessons learned in our prior, detailed evaluation of AEs on protein tertiary structures [3]. One of the lessons is that deep architectures (with many layers in the encoder) result in hundreds of thousands or more parameters that necessitate very large datasets to avoid overfitting. The first model, VAE1, is an example of a deep model whose architecture we included in our analysis of AEs in prior work. The other models are much shallower. The last model carries no dimensionality reduction but serves as a baseline that allows evaluating whether the assumption that the input and learned distributions are normal is a valid one when dealing with protein tertiary structures.

Finally, based on our prior work on AEs, we do not expand here on trying out various activation functions. We use a rectified linear unit in all layers of the encoder and decoder except for the last layer of the decoder, where we use a linear activation function.

2.5 Training and Training Datasets

We experiment with 18 proteins of different lengths and folds that are benchmark targets used in algorithm development for PSP [24, 34]. These proteins are listed in Table 2. In an abuse of notation but in the interest of space, we identify each protein not by its name but by the four-letter id of the entry where a representative native structure for it is stored in the PDB. This is referred to as PDB ID in Column 1 in Table 2. The fifth letter shown in parentheses refers to the chain selected for a multi-chain protein molecule. Column 2 in Table 2 shows the fold of the native structure, and Column 3 shows the length (in terms of the number of amino acids), confirming that the targets are diverse in fold and length.

Table 2: Training datasets (* denotes a predominant β fold with a short helix). The chain extracted from a multi-chain PDB entry (in Column 1) to be used as the native structure is shown in parentheses. The fold of the known native structure is shown in Column 2. The length of the protein sequence (number of amino acids) is shown in Column 3.

PDB ID	Fold	Length
1. 1bq9	β	53
2. 1dtd(B)	$\alpha + \beta$	61
3. 1isu(A)	$coil$	62
4. 1hz6(A)	$\alpha + \beta$	64
5. 1c8c(A)	β^*	64
6. 2ci2	$\alpha + \beta$	65
7. 1sap	β	66
8. 1wap(A)	β	68
9. 1fwp	$\alpha + \beta$	69
10. 1ail	α	70
11. 1dtj(A)	$\alpha + \beta$	74
12. 1aoy	α	78
13. 1cc5	α	83
14. 1tig	$\alpha + \beta$	88
15. 2ezk	α	93
16. 1hhp	β^*	99
17. 2h5n(D)	α	123
18. 1aly	β	146

While the tertiary structures obtained from the Rosetta AbInitio protocol are all-atom, we only retain the coordinates of the central carbon atom (also known as the CA atom) in each amino acid. The input layer of m units in all the VAE models implemented here reflects the fact that the protein targets are of different lengths. Once the tertiary structures of a given protein target are reduced to their CA atoms, they are then superimposed/aligned over/to the first structure obtained in order to remove differences due to rigid-body motions (whole-body rotation and translation). The resulting coordinates for a structure are then passed to the input neurons. In this manner, the decoder in our VAE models samples Cartesian coordinates for the CA atoms of a given protein target.

2.6 Implementation Details

We use Keras [6] to implement, train and evaluate the various VAE models. Keras is an open-source neural-network library written in Python. Each of the investigated VAEs is trained for a total of 100 epochs with a batch size of 256. A learning rate of 0.0005 is employed to prevent premature convergence to local optima. Training times vary from 566.479 to 1136.575 seconds depending on the size of the training dataset.

3 RESULTS

3.1 Comparing the Learned to the Input Distribution across All VAE Models

We evaluate whether the generated tertiary structures from the learned distribution follow that of the training structures. We do so for each of the 7 VAE models described in Section 2 in order to determine a few top models in this regard. As described in Section 2, we make use of three metrics, MMD, BD, and EMD. Since the distributions are multi-dimensional, we indirectly compare them via their respective distribution of RMSDs from the corresponding native structure of a protein target. We do so over 1/3 of the benchmark set (6 targets) in order to narrow down our focus on a few top models. This evaluation is repeated at $N = 10,000$ and at $N = 20,000$ to possibly observe an impact by the size of the training dataset; a model is trained on N Rosetta-generated structures, then used to generate N more structures, and the evaluation compares the distribution of the training dataset (the Rosetta dataset) to the VAE-generated dataset. Tables 3-5 relate the comparisons.

MMD-based Comparison

ID	N	V1	V2	V3	V4	V5	V6	v7
1hz6(A)	10K	0.0233	0.0588	0.0206	0.0189	0.0185	0.0259	**0.0183**
	20K	0.0240	0.0610	0.0212	0.0193	0.0189	0.0268	**0.0187**
1c8c(A)	10K	0.0073	0.0479	0.0022	0.0008	**0.0005**	0.0104	**0.0005**
	20K	0.0073	0.0480	0.0023	0.0008	**0.0005**	0.0104	0.0006
1ail	10K	0.0132	0.0790	0.0181	0.0138	0.0131	0.0369	**0.0121**
	20K	0.0132	0.0790	0.0183	0.0140	0.0132	0.0370	**0.0122**
1dtj(A)	10K	0.0110	0.0338	0.0050	0.0029	**0.0026**	0.0104	0.0032
	20K	0.011	0.0347	0.0051	0.0029	**0.0027**	0.0105	0.0033
1aoy	10K	0.0130	0.0642	0.0038	0.0008	0.0005	0.0166	**0.0002**
	20K	0.0130	0.0639	0.0039	0.0009	**0.0004**	0.0166	**0.0004**
1aly	10K	0.0790	0.1381	0.0291	0.0063	0.0017	0.0859	**0.0001**
	20K	0.0790	0.1385	0.0291	0.0063	0.0018	0.0858	**0.0001**

Table 3: The training and generated distributions are compared via MMD. Lowest values are in bold font. PDB ID is abbreviated as PID. VAE1-7 are abbreviated as V1-7.

BD-based Comparison

ID	N	V1	V2	V3	V4	V5	V6	v7
1hz6(A)	10K	0.0524	0.4134	0.0339	**0.0061**	0.0066	0.0922	0.0131
	20K	0.0431	0.3960	0.0438	0.0147	**0.0099**	0.100	0.0112
1c8c(A)	10K	0.0415	0.2339	0.0357	0.0066	**0.0002**	0.0744	0.0011
	20K	0.0646	0.2801	0.0605	0.0154	0.0054	0.1083	**0.0037**
1ail	10K	0.0143	0.1110	0.0170	0.0124	**0.0045**	0.0555	0.0076
	20K	0.1393	0.1131	0.0140	0.0099	**0.0030**	0.0566	0.0040
1dtj(A)	10K	0.0401	0.1411	0.0407	0.0094	**0.0037**	0.1090	0.0090
	20K	0.2416	0.1319	0.0420	0.0093	**0.0034**	0.1000	0.0041
1aoy	10K	0.0263	0.2588	0.0238	0.0002	**0.0013**	0.0844	0.0078
	20K	0.1815	**0.1075**	0.1925	0.3347	0.3382	0.1561	0.3378
1aly	10K	0.0803	0.2980	0.0585	0.0241	**0.0004**	0.1072	0.0037
	20K	0.2367	0.3847	0.2507	0.0744	**0.0018**	0.1497	0.0131

Table 4: The training and generated distributions are compared via BD. Lowest values are in bold font. PDB ID is abbreviated as PID. VAE1-7 are abbreviated as V1-7.

EMD-based Comparison

ID	N	V1	V2	V3	V4	V5	V6	v7
1hz6(A)	20K	0.4240	1.1763	0.2668	0.1535	**0.1240**	0.4161	0.1533
	20K	0.4248	1.1796	0.2696	0.1533	**0.1231**	0.4224	0.1521
1c8c(A)	10K	0.4957	0.5936	0.4281	0.2037	**0.0792**	0.5871	0.1201
	20K	0.49005	0.5934	0.4225	0.2019	0.1098	0.5821	**0.0800**
1ail	10K	1.1030	1.2332	0.4320	0.1961	**0.1909**	0.8063	0.2542
	20K	1.0985	1.2331	0.4306	0.1953	**0.1915**	0.8073	0.2546
1dtj(A)	10K	0.7014	1.4431	0.4689	0.1867	**0.1236**	0.6692	0.1701
	20K	0.6727	1.4403	0.4646	0.1857	**0.1243**	0.6687	0.1690
1aoy	10K	0.6769	1.5131	0.5493	0.2138	0.1082	0.9331	**0.0900**
	20K	3.8124	2.8833	3.3388	3.6763	**2.7802**	2.9811	3.8005
1aly	10K	3.0111	3.3354	2.2717	1.250	0.7155	3.180	**0.130**
	20K	2.9924	3.3333	2.2649	1.2454	0.7135	3.177	**0.1308**

Table 5: The training and generated distributions are compared via EMD. Lowest values are in bold font. PDB ID is abbreviated as PID. VAE1-7 are abbreviated as V1-7.

The above tables indicate that the lowest distances between the training and the generated datasets are achieved by VAE5 and VAE7. The results suggest that VAE5 and VAE7 are the top two models that mimick the input distribution best (in terms of RMSD from a given native structure) in the generated distribution. It is worth noting that the performance of VAE7, where there is no dimensionality reduction, suggests that the normal assumption is reasonable.

3.2 Comparing the Learned to the Input Distribution for top VAE models over All Targets

The rest of the evaluation focuses on VAE5 and VAE7. Figure 1 expands the above analysis to all 18 proteins, at the two settings of $N = 10,000$ or $N = 20,000$ tertiary structures in the training dataset. Specifically, for each target, we show the ratio of a distance metric achieved by VAE7 over the corresponding metric achieved by VAE5. Cases, where VAE5 achieves a lower distance metric, are easily identifiable as ratios above 1.

The left panel in Figure 1, which relates the comparison at $N = 10,000$, shows that VAE5 achieves no higher MMD values than VAE7 on 8/18 of the targets, no higher BD values on 14/18 of the targets, and no higher EMD values on 11/18 of the targets. The right panel in Figure 1, which relates the comparison at $N = 20,000$, shows that VAE5 achieves no higher MMD values than VAE7 on 9/18 of the targets, no higher BD values on 13/18 of the targets, and no higher EMD values on 11/18 of the targets.

Altogether, the above analysis suggests that VAE5 is a better model and warrants conducting further detailed analysis of its performance. It is worth noting that VAE5 is also more appealing than VAE7, as it forces the model to learn a 30-dimensional latent space, which is then used to reconstruct tertiary structures at their original dimensions. We make use of the lower dimensionality to

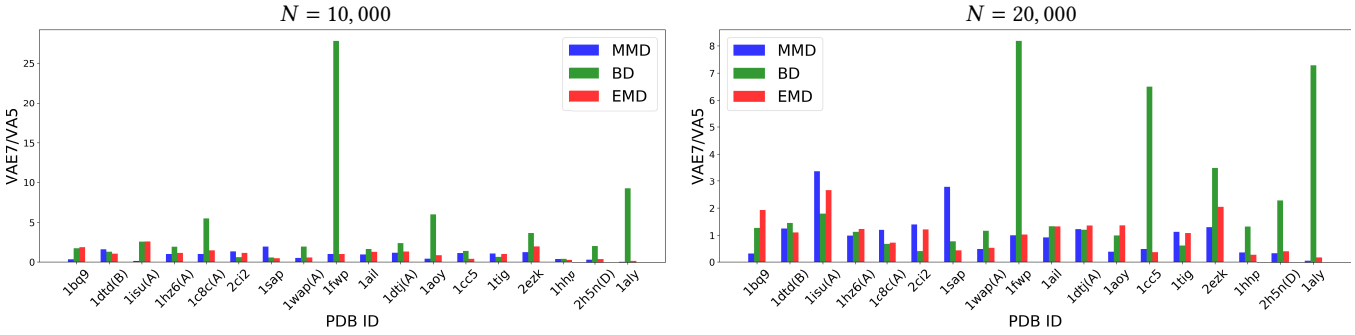


Figure 1: For each of the 18 targets, the plots show the ratio of the distance (according to the MMD, BD, and EMD metrics, respectively) between the training distribution and the distribution generated by VAE7 over the distance (according to the MMD, BD, and EMD metrics) between the training distribution and the distribution generated by VAE5.

visualize the latent space in some of the visualization-based analysis later in this section.

3.3 Evaluating the Quality of VAE5-generated Structures

We evaluate in greater detail the quality of the tertiary structures generated by VAE5 by comparing summary metrics of the RMSD distributions corresponding to training and learned/generated tertiary structures. We recall that we use RMSD as a proxy of the relevance of a tertiary structure, as it measures the distance from a known native structure. Again, we conduct this comparison at the two settings $N = 10,000$ structures and $N = 20,000$ structures.

In Figures 2(a)-(c), we show scatterplots that juxtapose a summary metric of the RMSD distribution of the generated distribution versus the same summary metric on the training distribution for each of the targets at $N = 10,000$. The top left panel in Figure 2 does so for the minimum RMSD and utilizes the size of disks to show the minimum RMSD in the generated distribution. Disks below the identity line correspond to targets where the generated distribution achieves a lower minimum RMSD. The Pearson correlation coefficient (PCC) between the minimum RMSDs over all targets is shown to be higher than 0.9, which confirms that the generated distributions are of as high quality as the training dataset in this regard.

Figure 2 reveals similar information on the mean and median RMSDs, respectively. An additional interesting observation related by these plots is that the generated distributions have the lower mean and median RMSDs than the training distributions. Similar observations are drawn from the bottom panel that relates the setting of $N = 20,000$, which again informs on the high quality of the generated datasets, as well as on $N \in \{10,000, 20,000\}$ being sufficient for VAE5 to learn the latent space.

Figure 3 provides some more detail by showing the distribution of RMSDs of the training versus the generated datasets for three selected proteins. These are representative of the full benchmark set, as they show a case (top panel) where the generated distribution covers well and improves overall (in terms of minimum, mean, and median) RMSD upon the training distribution, a case (middle panel) where the generated distribution has the same mean and median RMSD but slightly higher minimum RMSD than the training distribution, and a case (bottom panel) where the generated distribution is overall shifted to the right of the training distribution.

The comparison of the input versus the generated distribution can also be conducted visually, by utilizing the 30 latent coordinates via which VAE5 encodes tertiary structures. We utilize t-SNE to additionally project the 30-dimensional embedding of the input and latent space onto 2 dimensions. Figure 4 overlays the two-dimensional embedding and shows overall similarity between them.

3.4 Cost-Benefit Evaluation of VAE5 versus Rosetta

Altogether, the above analyses above suggest that VAE and, in particular, VAE5, is a good generative model that performs as well as Rosetta, an established method for template-PSP. However, the caveat, as related in Section 2 is that generative models need to seeded with relevant structures. So, we now provide a cost-benefit analysis detailed in Section 2 to warrant any employment of VAE5 as a generative model. Specifically, we consider now the following setting. After training VAE5 with N Rosetta-generated tertiary structures, we determine whether one should continue using Rosetta for an additional batch of N tertiary structures or whether one should switch to using VAE5 instead.

We compare the VAE5-generated dataset of N structures with N new, additional structures generated by Rosetta; note that the latter is not the training dataset, but new structures. We relate this comparison in two ways. First, we compare the corresponding RMSD distributions (RMSDs to the native structure) via distance metrics. We limit to MMD and EMD in the interest of space. Then, we relate the minimum, mean, and median of the corresponding RMSD distributions. We do so in two regimes, at $N = 10,000$ and $N = 20,000$. Note that at a particular N, the VAE has been trained with N Rosetta-generated structures, has been used to generate N more structures which are here compared to N new structures (not in the training dataset) generated with Rosetta.

Table 6 relates the MMD and EMD comparison.

Table 6 shows that the MMD values between the two settings of $N = 10,000$ and $N = 20,000$ are similar; no higher MMD values obtained on 10/18 of the targets when N is $20,000$ compared to when $N = 10,000$. EMD values are also similar; no higher EMD

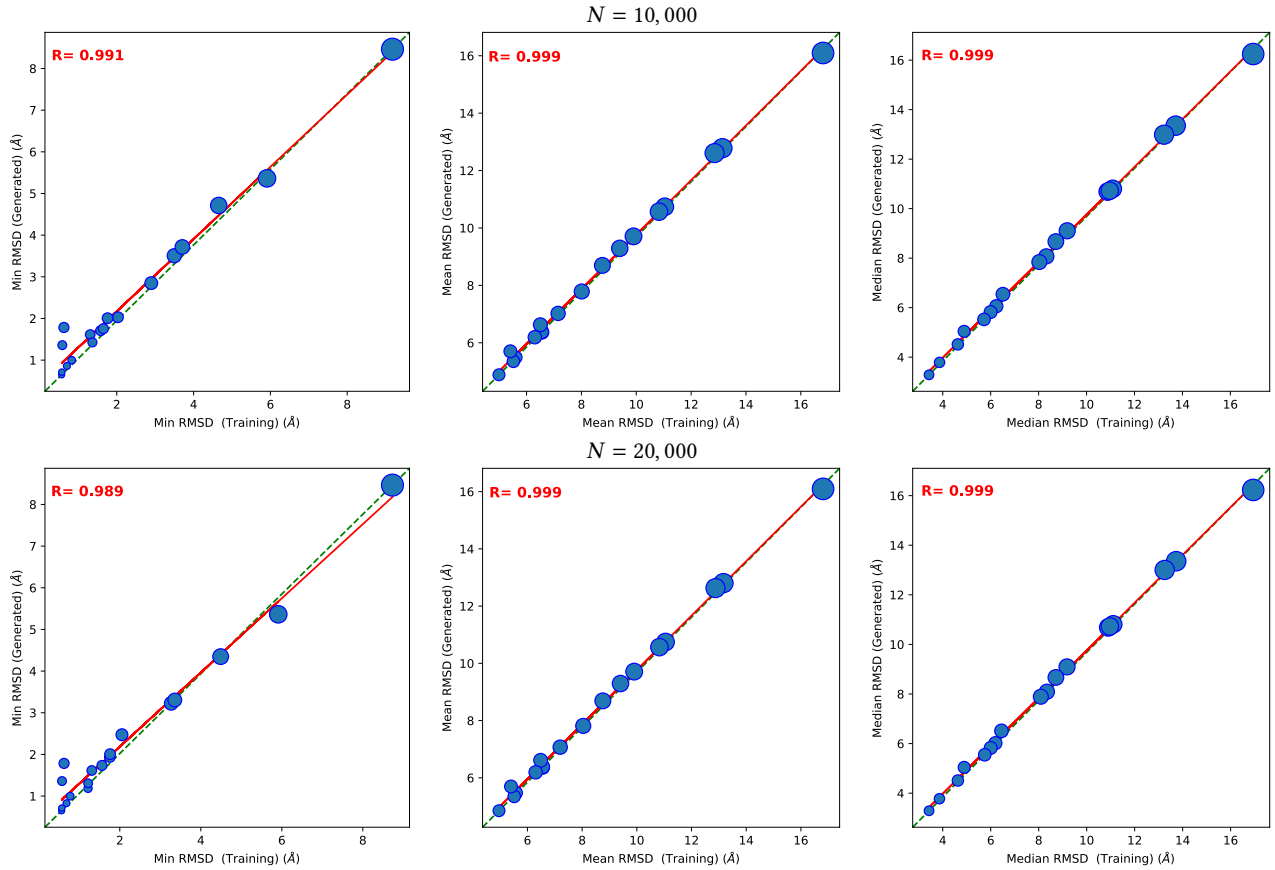


Figure 2: Comparison of summary metrics of the RMSD distribution of the generated distribution versus the training distribution for each of the targets at $N = 10,000$. The size of a disk relates the minimum RMSD in the generated dataset. The identity line is shown in green. The regression line is in red. The Pearson correlation is also shown.

values are obtained on 11/18 of the targets when $N = 20,000$ than when $N = 10,000$. This suggests that this range of number of structures is sufficient. In the interest of space, the rest of our analysis focuses on $N = 20,000$ structures.

We now compare summary metrics of the RMSD distributions. Figure 5 shows scatter plots that juxtapose a summary metric of the RMSD distribution of the VAE5-generated distribution versus the same summary metric on the Rosetta-generated distribution for each of the targets at $N = 20,000K$. The left panel shows the minimum RMSD, the middle panel shows the mean RMSD, and the right panel shows the median RMSD. The size of disks shows the minimum RMSDs in the Rosetta-generated distributions. Disks below the identity line correspond to targets where the VAE5-generated distribution achieves a lower minimum RMSD. The Pearson correlation coefficient (PCC) is also shown. The plots and the PCC show that the minimum RMSDs are comparable, and there is strong agreement on the median RMSDs. The means are generally higher on the VAE5-generated dataset, pointing to the existence of outlier structures.

3.4.1 Structure Visualization. The three proteins we have selected above to show more details into the distribution of tertiary structures are used once again for the purpose of visualizing some structures. In Figure 6 we draw the best (lowest RMSD from the native structure) tertiary structure generated by VAE5 over the best tertiary structure generated by Rosetta; as in the experiment above, we use Rosetta to sample 20,000 more structures, and the one with the lowest RMSD among them from the native structure is recorded and drawn below. Figure 6 shows that the VAE5- and Rosetta-generated structures are very similar, with RMSDs ranging from 0.5Åto 1.7Å; even the higher RMSD of 1.7Åis distributed mainly in the flexible loops connecting the secondary structure elements.

3.4.2 VAE or Rosetta? The above analysis shows that using VAE5 comes at no cost to data quality. Indeed, all the comparisons show that the tertiary structures generated by VAE5 are of as high structural quality and near-nativeness as the ones generated by a top platform, such as Rosetta.

The question remains on whether there is any benefit? We now look at computational cost to answer this question. The time to generate 20,000 tertiary structures with VAE5 varies from 314.041 to 500.802 seconds depending on protein length. It is an average of

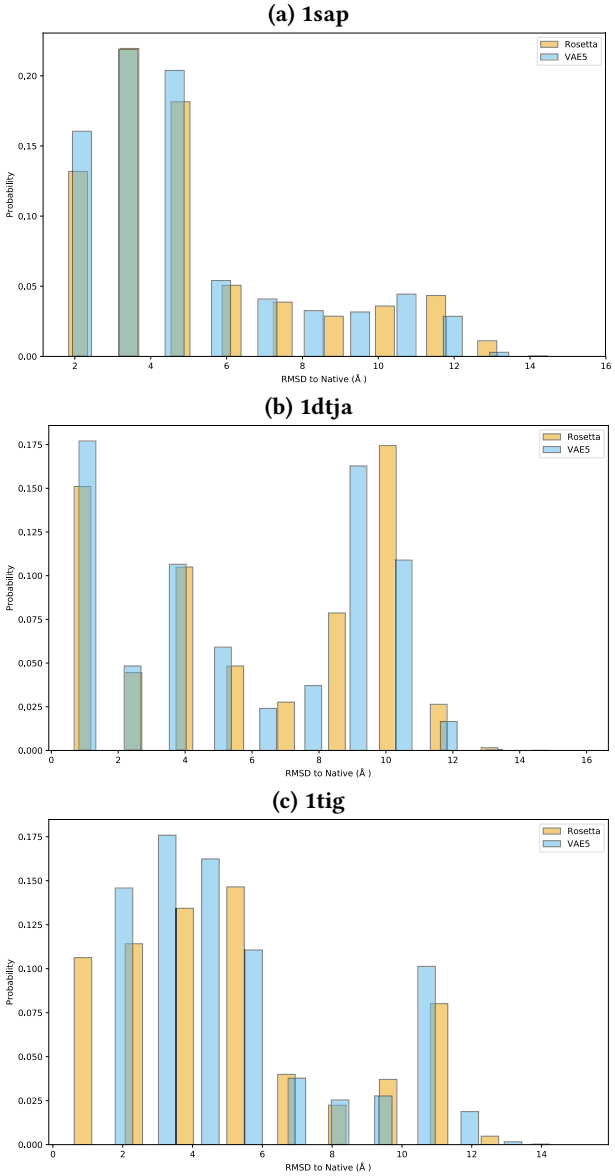


Figure 3: RMSD distributions of the training dataset (in orange) and the VAE-generated dataset (in blue) for three selected proteins at $N = 20,000$.

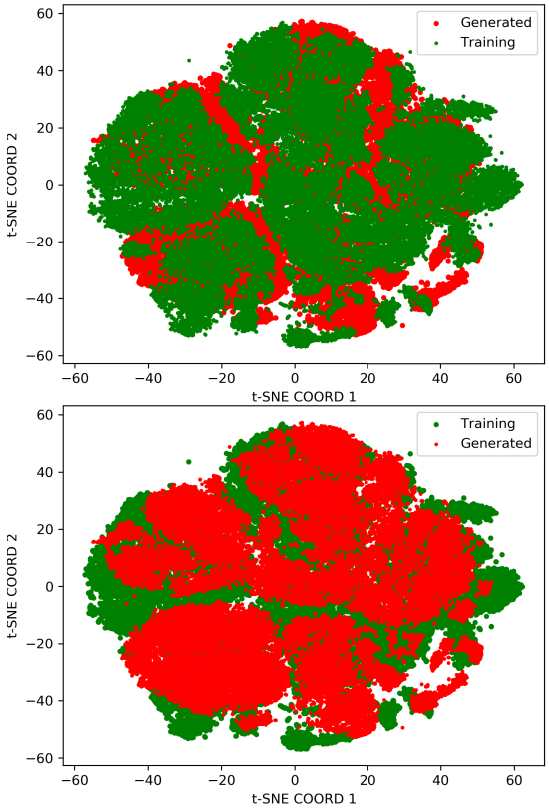


Figure 4: t-SNE is applied to the 30-dimensional learned embedding of VAE of the input structures and the generated structures. The two-dimensional projections are drawn on top of each-other to visualize the structure space for a selected protein with native structure under PDB id 1dtj(A).

15.70205ms to 25.0401ms per tertiary structure. We note that this average time is to produce structures at the CA-atom detail. A direct time comparison with Rosetta is not possible; Rosetta (and methods, such as Quark, and others) does not generate CA coordinates first, but instead includes backbone atoms and a centroid pseudo-atom for each amino-acid side chain. However, our experience with Rosetta is that each of the stages in it, which gradually add more energetic penalties, takes more time on average than the average time for VAE5 to generate one CA-atom resolution structure.

4 CONCLUSION

In this study, we design and evaluate VAEs on whether they can be as useful as existing, established methods for template-free PSP. Our evaluation shows that once seeded with sufficient, physically-realistic structures, these generative models are effective at generating realistic and near-native tertiary structures.

The presented results are encouraging. However, we caution that generative models for protein structure modeling are in their nascent stage. In particular, such models are still a long way from supplanting established platforms that generate tertiary structures as a process of optimizing an energy/scoring function. The detailed evaluation in this paper suggests that generative models are useful but they require physically-realistic structures for training.

Further research needs to investigate extensions of deep generative models that allow utilizing experimentally-available structures. Issues such as different lengths and amino-acid sequences need to be addressed. Another direction can investigate complementary frameworks, such as generative adversarial networks, though these present more challenges in training then VAEs. More studies are also needed on how to further diversify generated structures to potentially explore a broader structure space, as well as how to

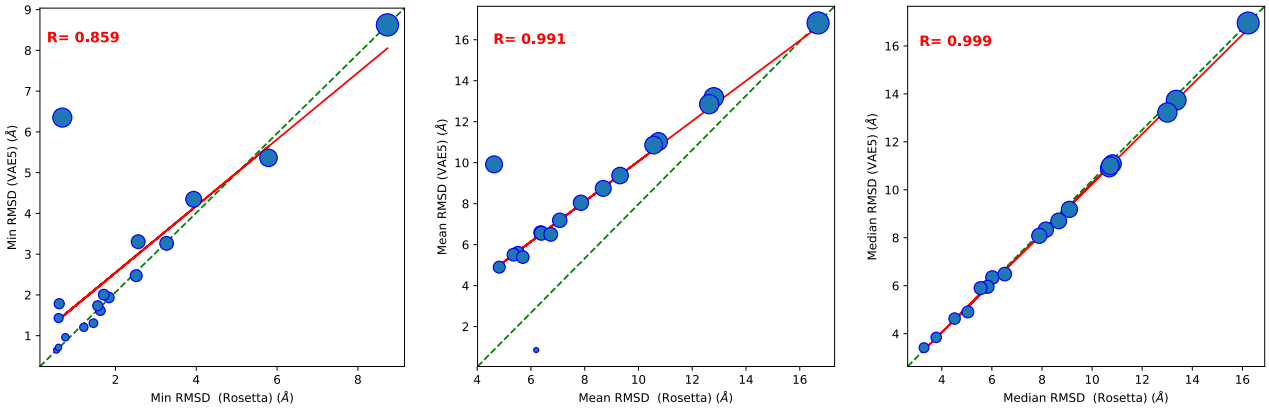
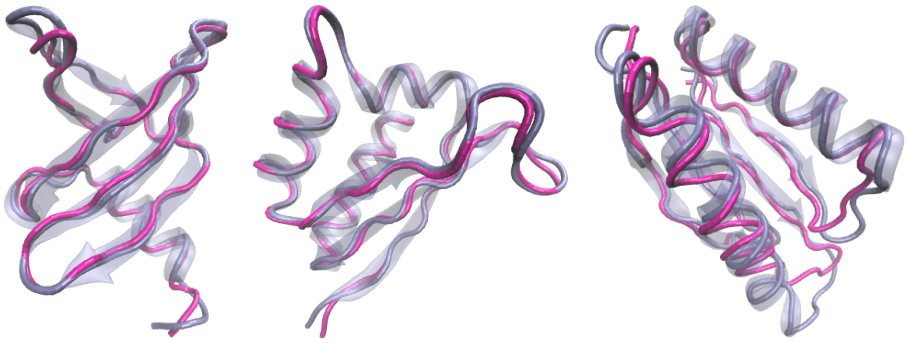


Figure 5: Comparison of summary metrics of the RMSD distribution of the VAE5-generated distribution versus the Rosetta-generated distribution for each of the targets at $N = 20,000$. The size of the disks relates the minimum RMSD in the Rosetta dataset. The identity line is shown in green. The regression line is in red. The Pearson correlation is also shown.



CA RMSD = 0.5Å CA RMSD = 0.55Å CA RMSD = 1.7Å

Figure 6: The lowest-RMSD Rosetta-generated structure is drawn in transparent blue. Its CA atoms are connected with a tube drawn in the same color in opaque. The lowest-RMSD VAE5-generated structure is superimposed on top and is drawn in opaque magenta, with its CA atoms connected with a tube. The structures are rendered with VMD /citevmd96. This is done for three selected proteins with PDB IDs 1sap (top panel), 1dtja (middle panel), and 1tig (bottom panel).

add more structural detail beyond the CA-atom representation employed here in an end-to-end framework that generates all-atom detail tertiary structures.

5 ACKNOWLEDGMENTS

This work is supported in part by NSF IIS Grants (No. 1763233 and 1763246). Computations were run in part on ARGO, a research computing cluster provided by the Office of Research Computing at George Mason University, VA (URL: http://orc.gmu.edu). This material is additionally based upon work supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] B. Adhikari, J. Hou, and J. Cheng. 2018. DNCON2: improved protein contact prediction using two-level deep convolutional neural networks. *Bioinformatics* 34 (2018), 1466–1472.

[2] F. F. Alam, T. Rahman, and A. Shehu. 2019. Learning Reduced Latent Representations of Protein Structure Data. In *Conf on Bioinf and Comput Biol (BCB) Workshops: Comput Struct Biol Workshop (CSBW)*. ACM, Niagara Falls, NY, 592–597.

[3] F. F. Alam, T. Rahman, and A. Shehu. 2020. Evaluating Autoencoder-based Featurization and Supervised Learning for Protein Decoy Selection. *Molecules* 25, 5 (2020), 1146.

[4] H. M. Berman, K. Henrick, and H. Nakamura. 2003. Announcing the worldwide Protein Data Bank. *Nature Structural Biology* 10, 12 (2003), 980–980.

[5] W. Bounliphone, E. Belilovsky, M. B. Blaschko, I. Antonoglou, and A. Gretton. 2016. A Test of Relative Similarity For Model Selection in Generative Models. In *Intl Conf Learn Representations (ICLR)*. 1–16.

[6] François Chollet et al. 2015. Keras. https://keras.io.

[7] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani. 2015. Training generative neural networks via maximum mean discrepancy optimization. In *Intl Conf Uncertainty in AI*. 1–10.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[9] A. Gretton, K. M. Borgwardt, M. J. Rasch, Scholkopf B., and A. Smola. 2012. A kernel two-sample test. *J Mach Learn Res* 13, 1 (2012), 723–773.

[10] J. Hanson, K. Paliwal, T. Litfin, Y. Yang, and Y. Zhou. 2018. Accurate prediction of protein contact maps by coupling residual two-dimensional bidirectional long short-term memory with convolutional neural networks. *Bioinformatics* 34 (2018), 4039–4045.

[11] J Ingraham, , A. Riesselman, C. Sander, and D. Marks. 2019. Learning protein structure with a differentiable simulator. In *Intl Conf on Learning Representations (ICLR)*.

[12] D. T. Jones and S. M. Kandathil. 2018. High precision in protein contact prediction using fully convolutional neural networks and minimal sequence features. *Bioinformatics* 34 (2018), 3308–3315.

PDB ID	$N = 10,000$		$N = 20,000$	
	MMD	EMD	MMD	EMD
1. 1bq9	**0.0003**	0.282	0.0007	**0.231**
2. 1dtd(B)	**0.0005**	**0.209**	0.0006	0.242
3. 1isu(A)	0.0026	0.317	**0.0021**	**0.285**
4. 1c8c(A)	**0.0006**	**0.074**	**0.0006**	0.127
5. 1hz6(A)	**0.0191**	0.133	0.0193	**0.110**
6. 2ci2	0.0015	0.187	**0.0011**	**0.167**
7. 1sap	0.0025	**0.156**	**0.0020**	0.158
8. 1wap(A)	**0.0002**	**0.208**	0.0003	0.214
9. 1fwp	0.0138	0.226	**0.0136**	**0.118**
10. 1ail	0.0133	0.202	**0.0132**	**0.186**
11. 1dtj(A)	0.0028	**0.130**	**0.0027**	0.172
12. 1aoy	0.0006	0.119	**0.0005**	**0.077**
13. 1cc5	**0.0005**	**0.254**	0.0007	0.293
14. 1tig	0.0713	0.334	**0.0710**	**0.313**
15. 2ezk	0.0026	**0.055**	**0.0021**	**0.055**
16. 1hhp	**0.0001**	0.415	0.0009	**0.383**
17. 2h5n(D)	**0.0002**	0.290	0.0008	**0.224**
18. 1aly	0.0017	**0.713**	**0.0015**	0.723

Table 6: The VAE5- and Rosetta-generated distributions are compared via MMD and EMD.

[13] Thomas Kailath. 1967. The divergence and Bhattacharyya distance measures in signal selection. *IEEE transactions on communication technology* 15, 1 (1967), 52–60.

[14] P. Kukic, P. Mirabello, G. Tradigo, I. Walsh, P. Veltri, and G. Pollastri. 2014. Toward an accurate prediction of inter-residue distances in proteins using 2d recursive neural networks. *BMC Bioinf* 15 (2014), 6.

[15] A. Leaver-Fay et al. 2011. ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. *Methods Enzymol* 487 (2011), 545–574.

[16] Y. Li, K. Swersky, and R. Zemel. 2015. Generative moment matching networks. In *Intl Conf Mach Learn (ICML)*. 1718–1727.

[17] Y. Li, C. Zhang, E. W. Bell, D.-J. Yu, and Y. Zhang. 2019. Ensembling multiple raw coevolutionary features with deep residual neural networks for contact-map prediction in CASP13. *Proteins: Struct, Funct, Bioinf* 87, 12 (2019), 1082–1091.

[18] Y. Liu, P. Palmedo, Ye Q., B. Berger, and J. Peng. 2018. Enhancing evolutionary couplings with deep convolutional neural networks. *Cell Syst* 6, e3 (2018), 65–74.

[19] T. Maximova, R. Moffatt, B. Ma, R. Nussinov, and A. Shehu. 2016. Principles and Overview of Sampling Methods for Modeling Macromolecular Structure and Dynamics. *PLoS Comp. Biol.* 12, 4 (2016), e1004619.

[20] A. D. McLachlan. 1972. A mathematical procedure for superimposing atomic coordinates of proteins. *Acta Cryst A* 26, 6 (1972), 656–657.

[21] M. Michel, D. M. Hurtado, and A. Elofsson. 2019. PconsC4: fast, accurate and hassle-free contact predictions. *Bioinformatics* 35 (2019), 2677–2679.

[22] A. Namrata and H. Po-Ssu. 2018. Generative modeling for protein structures. In *Advances in Neural Information Processing Systems*. 7494–7505.

[23] A. Namrata, E. Raphael, and H. Po-Ssu. 2019. Fully differentiable full-atom protein backbone generation. In *Intl Conf on Learning Representations (ICLR) Workshops: DeepGenStruct*.

[24] B. Olson and A. Shehu. 2013. Multi-Objective Stochastic Search for Sampling Local Minima in the Protein Energy Surface. In *ACM Conf on Bioinf and Comp Biol (BCB)*. Washington, D. C., 430–439.

[25] N. Perdigao, J. Heinrich, C. Stolte, K. S. Sabir, M. J. Buckley, B. Tabor, B. Signal, B. S. Gloss, C. J. Hammang, B. Rost, A. Schafferhans, and S. I. O'Donoghue. 2015. Unexpected features of the dark proteome. *Proc Natl Acad Sci USA* 112, 52 (2015), 15898–1590.

[26] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision* 40, 2 (2000), 99–121.

[27] S. Sabban and M. Markovsky. 2019. RamaNet: Computational De Novo Protein Design using a Long Short-Term Memory Generative Adversarial Neural Network. *BioRxiv* (2019), 671552.

[28] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, et al. 2019. Protein structure prediction using multiple deep neural networks in CASP13. *Proteins: Struct, Funct, Bioinf* 87, 12 (2019), 1141–1148.

[29] A. Shehu and B. Olson. 2010. Guiding the Search for Native-like Protein Conformations with an Ab-initio Tree-based Exploration. *Intl. J. Robot. Res.* 29, 8 (2010), 1106–1127.

[30] M. Torrisi, G. Pollastri, and Q. Le. 2020. Deep learning methods in protein structure prediction. *Comput and Struct Biotech J* S2001037019304441 (2020), 1–10.

[31] S. Wang, S. Sun, Z. Li, R. Zhang, and J. Xu. 2017. Accurate de novo prediction of protein contact map by ultra-deep learning model. *PLoS Comput Biol* 13 (2017), e1005324.

[32] D. Xu and Y. Zhang. 2012. Ab initio protein structure assembly using continuous structure fragments and optimized knowledge-based force field. *Proteins: Struct. Funct. Bioinf.* 80, 7 (2012), 1715–1735.

[33] A. Zaman, P. Parthasarathy, and A. Shehu. 2019. Using Sequence-Predicted Contacts to Guide Template-free Protein Structure Prediction. In *ACM Conf on Bioinf and Comp Biol (BCB)*. Niagara Falls, NY, 154–160.

[34] G. J. Zhang, G. Zhou, X, X. F. Yu, H. Hao, and L. Yu. 2017. Enhancing protein conformational space sampling using distance profile-guided differential evolution. *IEEE/ACM Trans Comput Biol and Bioinf* 14, 6 (2017), 1288–1301.