# Generating Physically-Realistic Tertiary Protein Structures with Deep Latent Variable Models Learning Over Experimentally-available Structures

Fardina Fathmiul Alam
*Dept of Computer Science*
*George Mason University*
Fairfax, VA 22030
falam5@gmu.edu

Amarda Shehu
*Dept of Computer Science*
*George Mason University*
Fairfax, VA 22030
amarda@gmu.edu

***Abstract*—Sophisticated deep neural networks have significantly advanced our ability to predict a native structure of a protein amino-acid sequence. However, going beyond a single-structure view remains challenging. While rapid advances are being made, fundamental questions on the ability of generative deep modeling to learn to generate physically-realistic tertiary structures remain. This paper makes two key contributions. It first extends deep convolutional variable autoencoder networks to be able to learn from experimentally-available tertiary structures of proteins of variable lengths. The presented models learn over distance matrix representations of tertiary structures. A systematic and detailed analysis demonstrates that the design of the training data is of primary importance to the ability of the proposed models to learn key characteristics of tertiary structures. The second contribution this paper makes is a careful analysis along several metrics that measure the physical realism of generated tertiary structures. The presented results are promising and show that once seeded with sufficient, physically-realistic structures, variational autoencoders are efficient models for generating physically-realistic tertiary structures.**

***Index Terms*—protein modeling; tertiary structure; generative model; variational autoencoder; spatial pyramidal pooling.**

## I. Introduction

Knowledge of the 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 [1]. The pluralism is intentional. We now know proteins are intrinsically dynamic, and many harness their ability to access significantly different structures at equilibrium to regular their interactions with molecules in the cell [2].

The key role that protein structure plays in protein function continues to motivate computational research. After several decades of organized efforts through the CASP competition, rapid breakthroughs ensued. A seemingly separate direction, that of computing contacts between pairs of amino acids, gained momentum by leveraging seminal developments in deep learning. Work by Cheng and others showed that deep neural networks can learn directly from sequences and structures of known proteins and accurately predict contacts of a novel amino-acid sequence [3]. In 2020, Xu's ResNet [4] showed a non-incremental advance had been made in our ability to compute contacts de novo. DeepMind then refined ResNet, adding to the network attention layers resulting in AlphaFold2 [5], which recently demonstrated that deep learning can predict the native structure of a protein sequence with extremely high accuracy.

RestNet and AlphaFold2 represent a seminal development that is sure to support many structure-function studies in molecular biology. However, going beyond a single-structure view remains central to our ability to fully model protein molecules. Decades of computational research on Molecular Dynamics-, Monte Carlo-, and other optimization-based frameworks show that we can see more of the structure space of a protein molecule relevant for function, but we can rarely do it from knowledge of the amino-acid sequence alone [1]. This remaining challenge and rapid momentum in deep learning is now renewing interest in deep generative models as alternative frameworks.

While rapid advances are being made in deep generative learning of tertiary protein structures, fundamental questions remain [6]. Most related work, much like the naming of AlphaFold2, confounds protein folding, structure prediction, and design. Deep learning researchers are eager to tackle another challenging scientific domain, but in that rush, the scientific objective is missed or not clearly formulated. A review of related work, detailed in Section I-A, informs that the main objective in most studies is to show that tertiary structures generated by a neural network look like they belong to proteins. Mostly, rigorous evaluation is lacking. Recent work by Rahman and Shehu introduces key metrics to evaluate whether a generated structure is protein-like [7].

Most work leverages the generative adversarial network (GAN) architecture and builds over image-based convolution, representing a tertiary protein structure as a contact map or distance matrix, which are both 2D-based representations of a 3D structure. However, even the GAN presented in [7] is limited to hand-curated datasets of matrices of a fixed size and is unable to learn from experimentally-available structures of

This work is supported in part by NSF Grant No. 1907805, 1900061, and 1763233.

proteins of varying length.

In this paper, we present a deep latent variable model, a convolutional variational autoencoder network (VAE) that learns over distance matrix representations of tertiary structures. The VAE architecture allows us to readily accommodate varying-size distance matrices through a technique known as spatial pyramidal pooling (SPP), permitting our network to learn directly from experimentally-available tertiary structures of varying-length proteins in the Protein Data Bank (PDB) [8]. This is our first key contribution. However, our study demonstrates that, when accommodating varying-size objects, it is important to ensure sufficient representation in the training dataset so that key characteristics are learned by the network. A systematic and detailed analysis informs on the construction of the training dataset. Finally, borrowing and adapting the structure-derived metrics proposed in [7] to handle varying-size structures, we demonstrate that the proposed SPP-VAE is able to generate distance matrices corresponding to physically-realistic tertiary structures. This is an important step towards more powerful networks that can then be enhanced to generate sequence-specific tertiary structures.

### A. Related Work

Notable efforts have been made in deep generative modeling of protein structures [7], [9]–[12]. Work in [11], [12] introduces a convolutional GAN architecture learning over fixed-size distance matrices corresponding to structural fragments of an *a-priori* determined length ($32$, $64$, or $128$). Learning remains a challenge. Some methods specialize the loss function to focus the network to learn the symmetry of contact maps [13] or the sparse long-range contacts [14]. Work in [7] shows that often the generated structures are not physically-realistic through key metrics that assess local and distal patterns. Rahman and colleagues also show that training GANs is not trivial, and care has to be taken to obtain convergence. The work proposes a Wasserstein GAN which improves the quality of generated distance matrices upon previous work.

Two related representations of tertiary protein structures have been popular for deep generative modeling research, distance matrices and contact maps. Both encode the spatial proximity of pairs of amino acids; most works collapse each amino acid to its central carbon – CA – atom. A matrix or map is indexed by the position of CAs along the protein chain of amino acids (ordered from the N- to the C-terminus). In a distance matrix, one records the actual Euclidean distances between all the CA pairs. A contact map is a binary version of a distance matrix. Typically, a proximity threshold of $8$Å is utilized; two CAs not further than $8$Å in Euclidean space are in *in contact*, and so their corresponding entry in the contact map is set to $1$; otherwise, the entry is set to $0$.

To the best of our knowledge, the VAE architecture has remained under-utilized for generating tertiary structures. In earlier work we showed its early promise when trained over computationally-generated tertiary structures of a given protein; in fact, we showed that after sufficient training data, a VAE could then replace Rosetta; its generated structures were virtually indistinguishable from those generated by Rosetta [15]. However, this work was focused on the single-structure setting, built one model per sequence, and though able to directly learn over Cartesian coordinates and so generate tertiary structures, it was also not able to accommodate experimentally-available structures of different lengths.

As laid out in Section I, our focus in this paper is to advance research in deep generative models for going beyond the single-structure view and instead accessing the structure space of a given amino-acid sequence. Like related work, though all in GANs, we leverage the distance matrix representation, which allows us to utilize image-based convolution. Unlike related work, we utilize a VAE architecture, which has the capacity to directly expose the underlying latent code. Unlike related work, our convolutional VAE learns directly over experimentally-available tertiary structures of varying lengths. We now relate further methodological details in Section II.

## II. Methods

### A. Tertiary Structure Representation and Training Datasets

The information present in a given tertiary protein structure is distilled into its distance matrix. The dataset that is used to train our SPP-CVAE consists of distance matrices corresponding to experimentally-available tertiary structures of proteins found in the Protein Data Bank [8]. In a key distinction from work in [7], [11], [12], we do not utilize the Namrata dataset of $115,850$ tertiary structures extracted from the PDB; we note that in this thread of works, a targeted chain length is determined, and then fragments of that length are extracted from the dataset of $115,850$. As work in [7] notes, this dataset is possibly highly-redundant. Therefore, in this work, we utilize instead a representative view of the PDB obtained via the PISCES server [16]. We leverage the pre-compiled datasets; compromising between good-quality tertiary structures (high resolution) and sufficient size of the training dataset, we choose the "cullpdb_pc20_res1.6_R0.25_d2021_07_02_chains2953" set of $2953$ tertiary structures. Their lengths vary between $20$ and $830$ amino acids. If we utilize this dataset as is, we find that the SPP-CVAE is unable to learn key characteristics of tertiary structures (local and distal patterns present in the backbone, short-range, and long-range contacts). So, instead, we pursue five settings, where we 'mix' with varying percentages distance matrices of different sizes, resulting in five models that we compare along various metrics.

### B. SPP-CVAE

While we assume some familiarity with the VAE architecture, we summarize here its main characteristics for the sake of completeness. A VAE builds over the AE architecture of an encoder and decoder network, each containing one or more layers of neurons/units. The encoder maps the input layer $x$ to its output layer $y$. The decoder performs the reverse mapping from the same layer $y$ to the output layer $z$. When

dealing with real values, training an AE entails minimizing the reconstruction error $||x - z||^2$.

A VAE is similar to an AE in that it consists of an encoder (inference model) and a decoder (generative model) network, which are connected but independently-parameterized. The training process in a VAE is regularized to ensure that the latent space has the properties necessary to generate data via the decoder [17]. The input distribution is assumed to be Gaussian so that the encoder can be trained to return the mean and covariance matrix. This assumption allows the latent space regularization to be expressed naturally.

Specifically, the encoder assumes that $x \sim N(\mu_x, \sigma_x)$ exists and attempts to learn the input distribution's parameters $\mu_x$ and $\sigma_x$. Similarly, the latent representation $z$ is assumed to be Gaussian; i.e., $z \sim N(\mu_{x,x})$. The loss function is $L = |x - y|^2 + KL(N(\mu_{x,x}) - N(0, I))$. The reconstruction loss, which ensures that the decoder's output is similar to the input, is the first term. The second term measures the Kullback-Leibler (KL) divergence between two probability distributions. The loss function reflects the fact that to prevent a VAE from behaving like an AE, both the covariance matrix and the mean of the distribution returned by the encoder must be regularized. This is accomplished by constraining the distribution to resemble a standard normal distribution as closely as possible (centered and reduced). As the loss function indicates, the covariance matrix is required to be close to zero, and the mean is also forced to be close to zero.

SPP-CVAE builds over the VAE architecture. The network utilizes convolution layers in the encoder and decoder to learn patterns in the 2D distance matrices corresponding to known tertiary structures. A schematic is shown in Fig. 1. Recent work in computer vision [18], proposes to extend CVAEs to learn from varying-size images via the SPP mechanism, which we also leverage here, as described next.

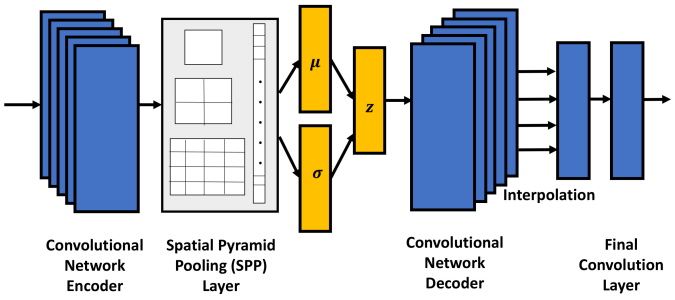


Fig. 1. The schematic shows the main components of the proposed CVAE-SPP network. The network contains Convolutional layers in both the encoder and decoder, so that it can learn patterns over distance matrices. An SPP layer allows the encoder to handle distance matrices of varying size.

SPP works by performing pooling operations of different sizes to extract features from data and process it to be the same size. For each SPP layer, a resolution parameter $n$ must be set; larger values correspond to a higher resolution, where smaller features can be identified by the pooling operation. Usually, there will be multiple values assigned in a single pooling layer. The sub-layers with smaller values of $n$ will extract larger features in the data, and the larger values of $n$ will extract smaller features. This gives the layers its pyramidal structure and allows the model to learn from features of different scales. In this way, using SPP makes the model agnostic of input size.

In summary, a encoder of SPP-CVAE is a stack of Convolutional layers with activation function LeakyReLU and Batch Normalization layers to help with the model optimization. Each Convolutional layer is able to extract useful information about the model so that the final latent representation will be information-rich and allow the decoder to generate useful information. At the end of the encoder, a SPP layer performs the steps required to get a fixed-length output. In SPP-CVAE, the SPP layer is comprised of 3 sub-layers of $n = 1, 2, 4$.

The first part of the decoder is a stack of Convolutional transpose layers (along with activation function LeakyReLU and Batch Normalization layers), which increases the input data point from the smaller dimension all the way to the larger one. Each Convolutional transpose layer performs decoding by doubling the size of the height and width.

The SPP-CVAE architecture produces tensors that will be of a fixed size but that are likely to be a lot smaller and possibly of a different size than the target output. This is solved using interpolation. Essentially what this does is align the corners of data points and fills in the gaps by fitting different functions. We use the "nearest" interpolation mode. This allows reshaping the data while still allowing backpropagation to take place. Before this step is finished, a final Convolutional layer is used as a final processing step to allow for better data reconstruction/generation.

### C. Evaluation Metrics

Building over work in [7], we employ and extend domain-specific metrics to evaluate generated distance matrices of varying lengths. Let us first summarize the three main metrics that summarize a distance matrix with one value that measures the presence of a backbone, short-range contacts, and long-range contacts, respectively.

*a) Evaluating the Presence of Backbone:* We first evaluate whether a backbone is present in a distance matrix, as this is a key characteristic of tertiary protein structures. More specifically, we measure how many of the consecutive CAs (entries $[i, i+1]$ in a distance matrix) are at a distance between $3.6$ and $4.0$Å from each other. Ideally, this distance should be closer to $3.83$Å, but we allow for some variability, as we observe small fluctuations even in experimentally-available tertiary structures. If $k-1$ consecutive CAs meet this criterion in a distance matrix of size $k$, then that distance matrix is considered to have a backbone. To allow for a metric that is independent of matrix size, we utilize '% Backbone' for the evaluation in Section III – the percentage of distance matrices in a set of interest (training or generated) that have a backbone.

*b) Evaluation of the Presence of Short-range Contacts:* The second metric summarizes a distance matrix with the number of short-range contacts present in it; these are measured as the number of $[i, j]$ entries, where $1 < |j - i| \leq 4$ (the lower bound excludes the backbone) and the distance between corresponding CAs is no higher than $8$Å. We refer to this

metric as SR-Nr, for short-range (contact) number. To extend this metric to generated distance matrices of varying sizes, we normalize this number by the number of CAs. We refer to this metric as SR-Score.

*c) Evaluation of the Presence of Long-range Contacts:* As work in [7] shows, learning the off-backbone contacts that characterize tertiary protein structures is challenging, and even more so for long-range contacts, where $|j - i| > 4$. So, we measure the number of long-range contacts as an additional metric and refer to it as LR-Nr for long-range (contact) number. To allow for varying-size distance matrices, we normalize this number by the number of CAs. We refer to this metric as LR-Score.

### D. Evaluation Metrics to Compare Two Distributions

Each of the above metrics provides us with one value to summarize a distance matrix. Doing so over a set of matrices, one can then obtain a distribution that characterizes the set. For instance, one can measure the number of short-range contacts, SR-NR, over the distance matrices in the training dataset. Similarly, one can obtain a distribution of the same value over the distance matrices generated by a trained SPP-CVAE model. Obtaining two distributions that characterize the training and generated dataset, respectively, permits answering the question of physical realism, using the training dataset as an example of what physical realism is.

Distance functions can be utilized to compare two distributions. Here, we utilize the Earthmover Distance (EMD) [19], also known as the Wasserstein distance. EMD treats the two given distributions as two different ways of piling up a certain amount of dirt over the domain and calculates the minimum cost to turn one pile into the other. The cost is the amount of dirt moved times the distance by which it is moved. EMD can be computed using any algorithm for the minimum cost flow problem, such as the network simplex algorithm [19].

### E. Trained and Evaluated SPP-CVAE Models

In recent related work in computer vision, it was demonstrated that SPP was effective to accommodate varying-size images, but the training dataset consisted of two different sizes represented at similar ratios in the training dataset. Our PISCES-derived dataset has great variability in the resulting sizes of distance matrices. Indeed, utilizing it as is to train a SPP-CVAE model results in very poor generated distance matrices (results not shown). In this paper, we show the importance of having sufficient representation of varying-size distance matrices by exploring three main settings: one where all distance matrices are of size $64 \times 64$ (that is, corresponding to fragments of $64$ amino acids extracted from the PISCES dataset); another where the training dataset contains two types of matrices, of size $64 \times 64$ and of size $72 \times 72$ at varying percentages; finally, a setting where the training dataset includes a third type of matrices of size $90 \times 90$.

In all, we train five different models, to which we refer as: SPP-CVAE$_{64}$, SPP-CVAE$_{64(70\%)+72(30\%)}$, SPP-CVAE$_{64(50\%)+72(50\%)}$, SPP-CVAE$_{64(40\%)+72(30\%)+90(30\%)}$, and SPP-CVAE$_{64(34\%)+72(33\%)+90(33\%)}$. Each of these models is trained separately until convergence. Each trained model is then used to generate/sample $2,954$ distance matrices (as many as in the training dataset) which are analyzed in detail to comparably evaluate model performance. For the purpose of evaluation, particularly when we compare the generated to the training datasets, the percentage of distance matrices of a given size is kept as in the training dataset; that is, for instance, $70\%$ of the distance matrices sampled by the trained SPP-CVAE$_{64(70\%)+72(30\%)}$ are of size $64 \times 64$, and the rest are of size $72 \times 72$.

### F. Implementation Details

We use Pytorch Lightning [20] to implement, train, and evaluate the various SPP-CVAE models. PyTorch Lightning is an open-source python library that provides a high-level interface for the PyTorch deep learning framework. Each of the investigated models is trained for a total of $90$ epochs (until convergence is reached). For the SPP-CVAE$_{64}$ model, we use a batch size of $32$. For the other four models, we use a batch size of $1$ to handle variable-length input data, as well as to avoid mixed sizes within a batch. A learning rate of $0.0003$ is used to prevent premature convergence. Training times for the models vary from $120.889$ to $161.127$ seconds. All models converge at around $40$ epochs, as shown in Fig. 2.

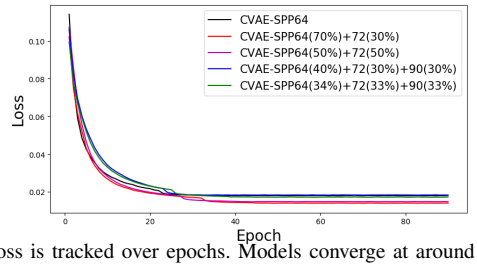


Fig. 2. Loss is tracked over epochs. Models converge at around epoch 40.

## III. Results

Here we evaluate the five models describe in Section II on their ability to learn directly from distance matrices of experimentally-available tertiary structures and generate physically-realistic distance matrices. We organize the evaluation around the three metrics presented in Section II that capture the presence of a backbone, short-range contacts, and long-range contacts.

### A. Evaluating the Presence of Backbone

Table I summarizes the performance of the various models, which are listed in Column 1. Column 2 shows the '% Backbone' over generated distance matrices. For instance, Table I shows that 99% of distance matrices generated by CPP-CVAE$_{64}$ contain their backbone. The value out of parentheses shows the '% Backbone' over all distance matrices generated by a model. The values in parentheses show this value over subsets of distance matrices, grouped by size. For instance, Table I shows that 96% of the distance matrices generated by SPP-CVAE$_{64(70\%)+72(30\%)}$ have their backbone. When

separating the generated distance matrices by size ($64 \times 64$ versus $72 \times 72$), the corresponding values become $97\%$ and $92.5\%$. Overall, Table I shows that at least $93\%$ of the distance matrices generated by any model have their backbone. Slight reductions are observed as the training datasets become more varied, but overall all models learn the backbone well.

### *B. Evaluating Distribution of Short-range Contacts*

Column 3 in Table I shows the EMD between the distribution of SR-Nr (number of short-range contacts in a distance matrix) measured over the generated dataset and the distribution of SR-NR measured over the training dataset. Low SR-Nr values provide evidence that the generated dataset are similar to the training dataset. Column 3 shows that the lowest value is reached by the CVAE-SPP$_{64}$, which serves as a baseline of performance. As the training datasets become varied in the size of distance matrices, EMD values increase (though they deviate only slightly from the baseline); interestingly, however, when the representation of the varied distance matrices are equal, as in SPP-CVAE$_{64(50\%)+72(50\%)}$, the EMD value is the second-lowest and very close to that of the baseline model. Proportional representation lowers the EMD value, as the performance of SPP-CVAE$_{64(34\%)+70(33\%)=90(33\%)}$ indicates, but in such cases, larger datasets are surely to improve the performance of the model even further. When normalizing by the size of a distance matrix, as in Column 5 in Table I, the EMD values appear closer among the models but largely follow the same trend.

The top panel in Fig. 3 shows the EMD values comparing the distribution of SR-NR (in red) and SR-Score (in blue) over the generated and training datasets are tracked over training epochs for CVAE-SPP$_{64}$. To clarify, after every $10$ epochs of training, the model is arrested and used to obtain a generating dataset, which is then compared to the training dataset (we recall that the loss function converges at epoch $40$). The top panel in Fig. 3 shows improvements in the physical realism of generated distance matrices (over models arrested at varying number of epochs) in terms of short-range contact metrics, and convergence is observed over the EMD values as the loss function converges, as well; The bottom panel adds the other four models and shows EMD values over SR-Score distributions over epochs. Overall, all models reduce the EMD values; that is, the generated distance matrices resemble the training ones more closely (in terms of short-range contacts) over epochs. In agreement with the summary results shown in Table I, the lowest EMD values are obtained by the baseline model, followed closely by the models where the varying-size matrices have similar representation in the training dataset.

### *C. Evaluating Distribution of Long-range Contacts*

Fig. 4 replicates the above analysis for long-range contacts. The steep reduction in the EMD values is obtained by the baseline model over epochs, and convergence is not demonstrated (though each model's loss function converges before or at epoch $45$). The bottom panel, which adds the other four models but shows EMD values over LR-Score distributions, confirms

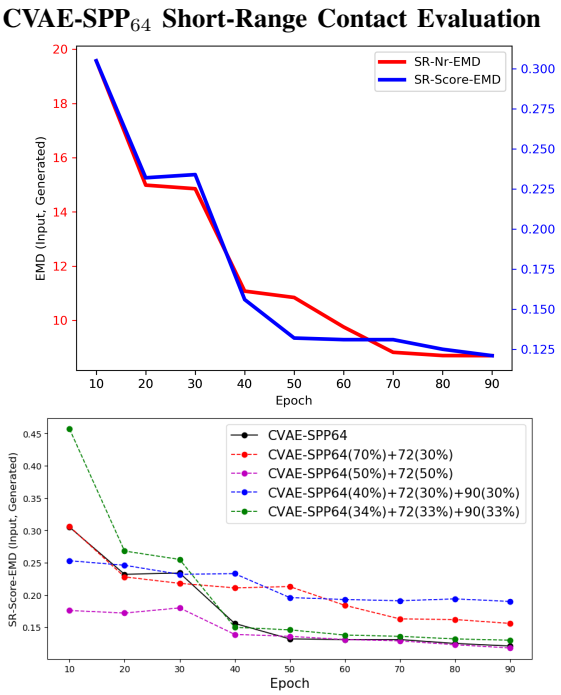


Fig. 3. Top panel: EMD values comparing the distribution of SR-NR (in red) and SR-Score (in blue) over the generated and training datasets are tracked over training epochs for CVAE-SPP$_{64}$. Since SR-Nr and SR-Score have different ranges, two y axes are utilized. Bottom panel adds the other four models but shows only EMD over SR-Score distributions.

that all models reduce the EMD values; generated distance matrices resemble the training ones more closely (in terms of long-range contacts) over increasing epochs. Interestingly, as Column 6 in Table I shows, normalizing for length, as in LR-Score, makes all models competitive and actually suggests that higher diversity (with proportional representation) improves realism on long-range contacts.

### *D. Visualization of Distributions and Distance Matrices*

Fig. 5 shows the distribution of SR-NR, SR-Score, LR-NR, and LR-Score over the dataset generated by each model (left to right panels). The distribution over the generated dataset (in blue) is superimposed over the corresponding one over the training dataset (in orange). Several observations can be made. Fig. 5 shows that the distributions over the generated and training datasets are closer for the baseline model, with slightly more discrepancies as the training dataset varies in the size of distance matrices. These visual observations confirm the quantitative analysis related above.

Finally, we relate some distance matrices by drawing them as heatmaps in Fig. 6. We select five at random over the training dataset and over each dataset generated by each of the models. The values are normalized, and a red-to-black color scheme is used to indicate higher-to-lower distances; that is, the contacts (backbone, short-range, and long-range) visually emerge as black lines. Fig. 6 relates the richness of patterns, further confirming the above quantitative analysis that generated distance matrices are highly realistic and resemble those of experimentally-available tertiary structures.

TABLE I

COLUMN 1 LISTS THE MODELS UNDER COMPARISON. COLUMN 2 SHOWS THE '% BACKBONE' OVER DISTANCE MATRICES GENERATED BY A MODEL. THE VALUES IN PARENTHESES SHOW THE '% BACKBONE' OVER DISTANCE MATRICES OF THE SAME SIZE. COLUMN 3 SHOWS THE EMD DISTANCE BETWEEN THE DISTRIBUTION OF SR-NR VALUES OVER THE GENERATED AND TRAINING DATASETS, RESPECTIVELY. WE RECALL THAT SR-NR MEASURES THE NUMBER OF SHORT-RANGE CONTACTS IN A DISTANCE MATRIX. COLUMN 4 DOES SO FOR LR-NR, WHICH MEASURES THE NUMBER OF LONG-RANGE CONTACTS. COLUMNS 5 AND 6 NORMALIZE SR-NR AND LR-NR, RESPECTIVELY, BY THE SIZE OF A DISTANCE MATRIX.

| Model | % Backbone | SR-Nr-EMD | LR-Nr-EMD | SR-Score-EMD | LR-Score-EMD |
|---|---|---|---|---|---|
| CVAE-SPP$_{64}$ | 99.0 | 8.694 | 17.098 | 0.121 | 0.301 |
| CVAE-SPP$_{64(70\%)+72(30\%)}$ | 96 ( 97, 92.5) | 10.031 | 18.738 | 0.156 | 0.335 |
| CVAE-SPP$_{64(50\%)+72(50\%)}$ | 97.9 (99, 96.91) | 8.758 | 16.947 | 0.118 | 0.221 |
| CVAE-SPP$_{64(40\%)+72(30\%)+90(30\%)}$ | 94.3 (97.5, 94, 93.4) | 12.370 | 22.421 | 0.190 | 0.321 |
| CVAE-SPP$_{64(34\%)+72(33\%)+90(33\%)}$ | 95.29 (96.4, 94.3, 95.15) | 11.182 | 20.148 | 0.130 | 0.209 |

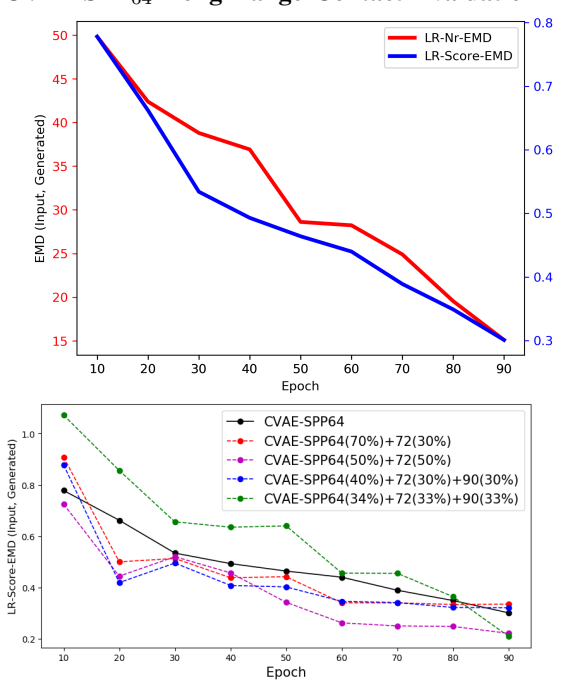


Fig. 4. Top panel: EMD values comparing the distribution of LR-NR (in red) and LR-Score (in blue) over the generated and training datasets are tracked over training epochs for CVAE-SPP$_{64}$. Since LR-Nr and LR-Score have different ranges, two y-axes are utilized. The bottom panel adds the other four models but shows only EMD over LR-Score distributions.

## IV. CONCLUSION

Growing momentum in deep generative modeling presents an opportunity and an alternative framework to structure space sampling and going beyond the single-structure view. The majority of existing deep generative models leverage the GAN architecture and are trained over fixed-size contact maps or distance matrices (that distill the essential information in a tertiary protein structure). Fundamental questions on whether deep generative models can learn to generate physically-realistic tertiary structures (whether in contact map or distance matrix representation) need to be addressed.

In this paper, we advance deep generative modeling for sampling the structure space of a sequence-agnostic protein chain in several directions. First, we debut a convolutional VAE architecture. We equip it with an SPP layer so that the resulting model can learn directly from experimentally-available tertiary structures (and so accommodate distance matrices of varying size in the training dataset). We demonstrate that the training dataset needs to be carefully constructed and have an adequate representation of distance matrices of varying-size. A rigorous analysis along metrics that interrogate a generated distance matrix for the presence of backbone, short-range, and long-range contacts as in experimentally-available structures show that all presented models generate distance matrices that correspond to physically-realistic tertiary structures.

Our evaluation shows that the presented SPP-CVAE network results in several models that generate tertiary-structures of the same physical realism as the training dataset, insofar as the metrics that measure the presence of backbone, short-range contacts, and long-range contacts provide such information. Interestingly, adding distance matrices of varying size often helps the model; for instance, better results are obtained on the metrics that interrogate the presence of short-range contacts; when normalized for chain length, the models with the highest diversity of distance matrices in the training dataset perform better even on long-range contacts. We caution that the presented study constitutes a well-controlled experiment and that adding more diversity, while it may further improve the physical realism, may also require a larger training dataset for adequate representation of proteins of different length.

Several directions of future research remain. We need to investigate conditioning a generated dataset on a given amino-acid sequence. Scoring of generated structures will also be useful. End-to-end frameworks that include dihedral angles in order to directly generate Cartesian coordinates of tertiary structures will also be helpful for further and more detailed analysis of generated structures. Graph-based representations of tertiary structures may also be appealing and open the way for graph-based GANs and VAEs, as well as graph neural network architectures, as preliminary work in [21] shows for prediction of a single structure.


## ACKNOWLEDGMENT

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.

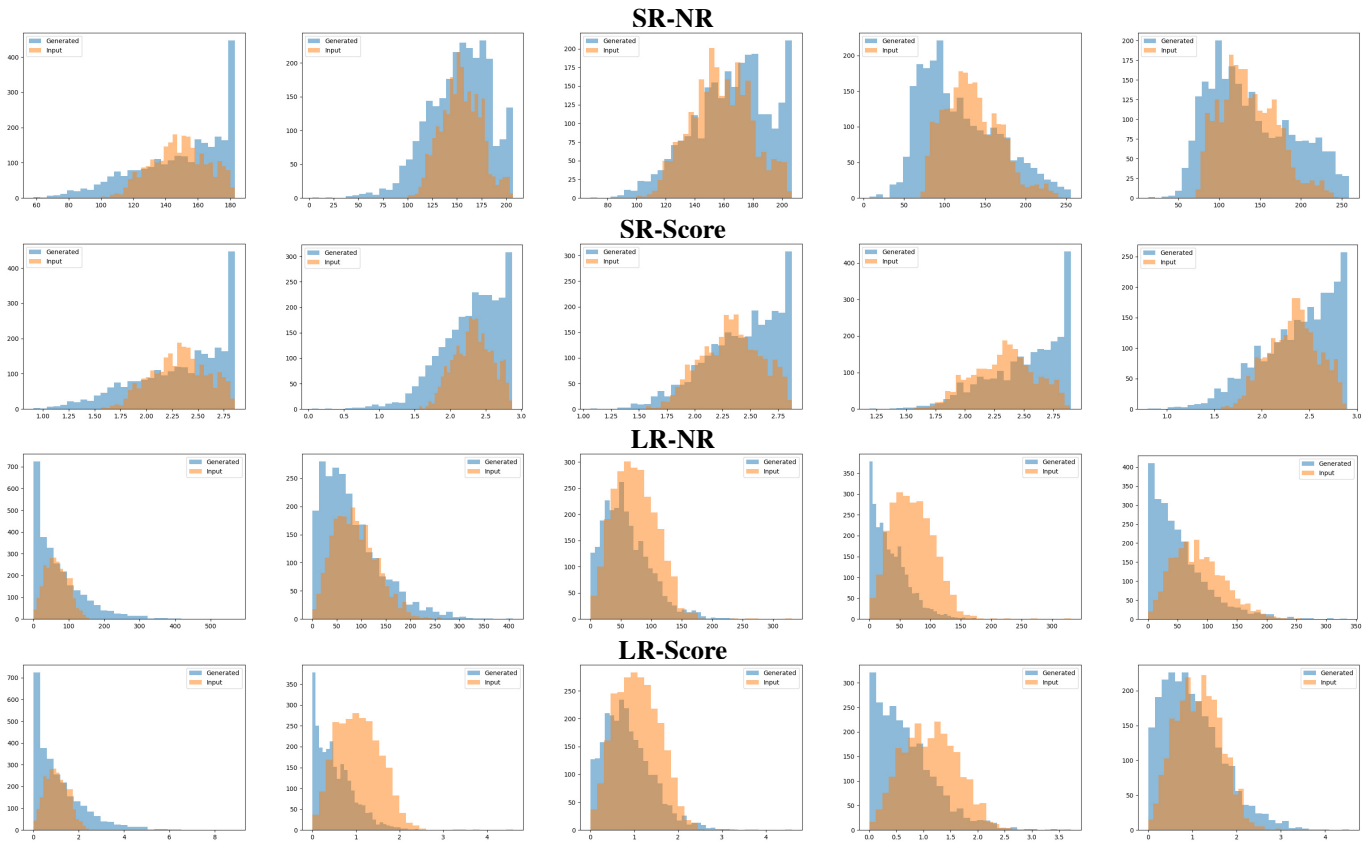


Fig. 5. Left to right: SPP-CVAE$_{64}$, SPP-CVAE$_{64(70\%)+72(30\%)}$, SPP-CVAE$_{64(50\%)+72(50\%)}$, SPP-CVAE$_{64(40\%)+72(30\%)=90(30\%)}$, and SPP-CVAE$_{64(34\%)+70(33\%)=90(33\%)}$. Distributions of SR-NR, SR-Score, LR-NR, and LR-Score over the generated dataset (blue) are superimposed over the training dataset (orange).

## REFERENCES


[1] T. Maximova, R. Moffatt, B. Ma, R. Nussinov, and A. Shehu, "Principles and overview of sampling methods for modeling macromolecular structure and dynamics," *PLoS Comp. Biol.*, vol. 12, no. 4, p. e1004619, 2016.

[2] D. D. Boehr, R. Nussinov, and P. E. Wright, "The role of dynamic conformational ensembles in biomolecular recognition," *Nature Chem Biol*, vol. 5, no. 11, pp. 789–96, 2009.

[3] J. Hou, T. Wu, R. Cao, and J. Cheng, "Protein tertiary structure modeling driven by deep learning and contact distance prediction in CASP13," *Proteins*, vol. 87, no. 12, p. 1165–1178, 2019.

[4] J. Xu, M. McPartlon, and J. Lin, "Improved protein structure prediction by deep learning irrespective of co-evolution information," *Nature Mach Intel*, vol. 3, pp. 601–609, 2020.

[5] J. Jumper, R. Evans *et al.*, "Highly accurate protein structure prediction with alphafold," *Nature*, 2021.

[6] P. Hoseini, L. Zhao, and A. Shehu, "Generative deep learning for macromolecular structure and dynamics," *Curr. Opinion Struct. Biol.*, vol. 67, pp. 170–177, 2020.

[7] T. Rahman, Y. Du, L. Zhao, and A. Shehu, "Generative adversarial learning of protein tertiary structures," *Molecules*, vol. 26, no. 5, p. 1209, 2021.

[8] H. M. Berman, K. Henrick, and H. Nakamura, "Announcing the worldwide Protein Data Bank," *Nature Structural Biology*, vol. 10, no. 12, pp. 980–980, 2003.

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

[10] S. Sabban and M. Markovsky, "RamaNet: Computational de novo protein design using a long short-term memory generative adversarial neural network," *BioRxiv*, p. 671552, 2019.

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

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

[13] H. Hang, M. Wang, Z. Yu, X. Zhao, and A. Li, "GANcon: Protein contact map prediction with deep generative adversarial network," *IEEE Access*, vol. 8, pp. 80 899–80 907, 2020.

[14] W. Ding and H. Gong, "Predicting the real-valued inter-residue distances for proteins," *Advanced Science*, vol. 7, no. 19, p. 2001314, 2020.

[15] A. F. Fardina and A. Shehu, "Variational autoencoders for protein structure prediction," in *ACM Conf on Bioinf and Comp Biol*, New York, NY, USA, 2020.

[16] G. Wang and R. Dunbrack, "PISCES: a protein sequence culling server," *Bioinformatics*, vol. 19, no. 12, pp. 1589–1591, 2003.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[18] A. Ashiquzzaman, H. Lee, K. Kim, H.-Y. Kim, J. Park, and J. Kim, "Compact spatial pyramid pooling deep convolutional neural network based hand gestures decoder," *Applied Sciences*, vol. 10, no. 21, p. 7898, 2020.

[19] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *International journal of computer vision*, vol. 40, no. 2, pp. 99–121, 2000.

[20] e. a. Falcon, WA, "Pytorch lightning," *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, vol. 3, 2019.

[21] T. Xia and W. Ku, "Geometric graph representation learning on protein structure prediction," in *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 2021, pp. 1873–1883.

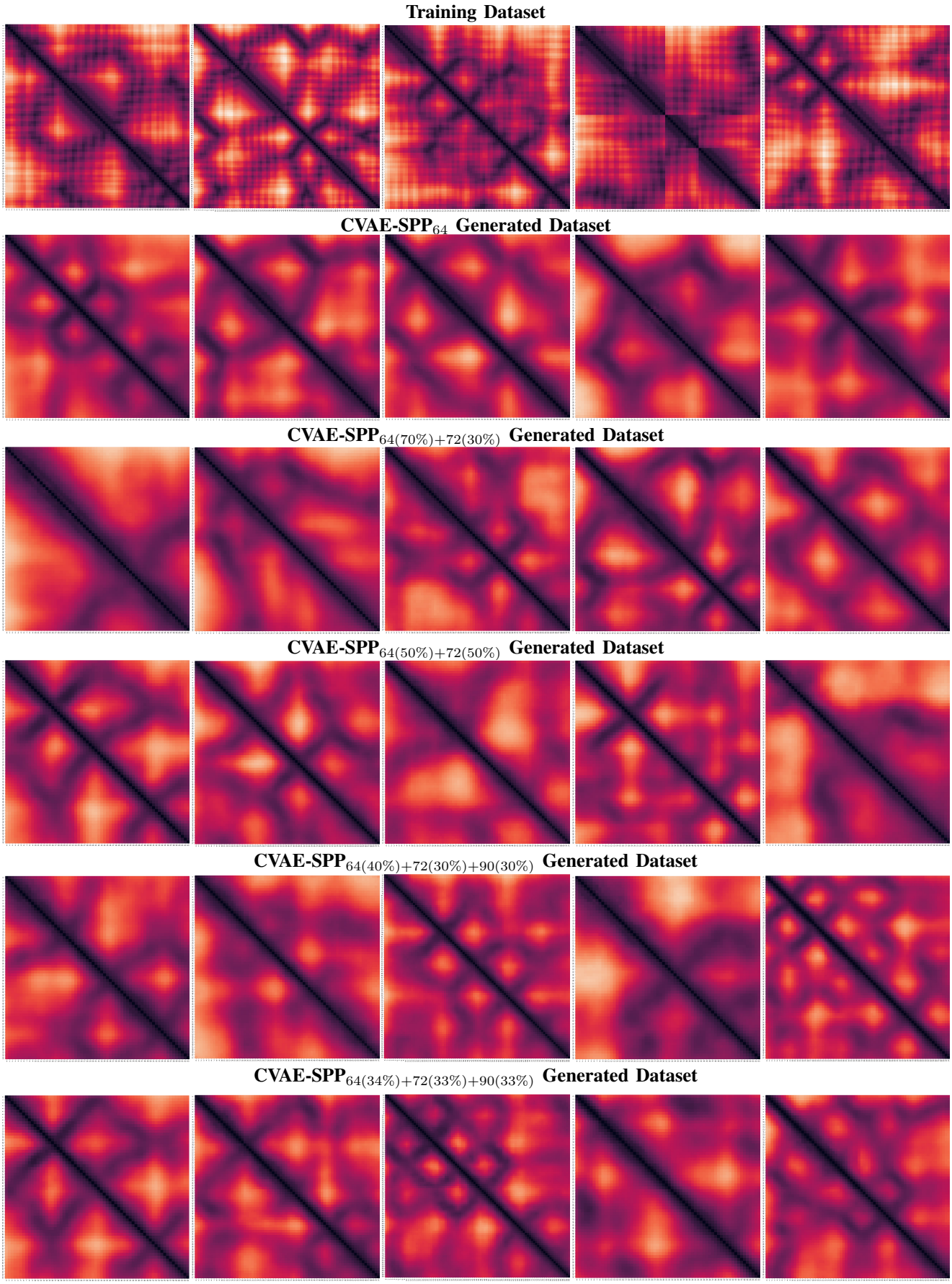


Fig. 6. We show here a few distance matrices sampled at random over the training dataset (top panel) and the datasets generated by each of the other 4 models (other four rows). Distance matrices are drawn as heatmaps. Distance values are normalized, and darker colors indicate lower values.