Graph Representation Learning for Protein Conformation Sampling

Taseef Rahman[1], Yuanqi Du[1], and Amarda Shehu[1]

Department of Computer Science, George Mason University,
Fairfax VA 22030, USA
{trahman2,ydu6,amarda}@gmu.edu

Abstract. Significant research on deep neural networks, culminating in AlphaFold2, convincingly shows that deep learning can predict the native structure of a given protein sequence with high accuracy. In contrast, work on deep learning frameworks that can account for the structural plasticity of protein molecules remains in its infancy. Many researchers are now investigating deep generative models to explore the structure space of a protein. Current models largely use 2D convolution, leveraging representations of protein structures as contact maps or distance matrices. The goal is exclusively to generate protein-like, sequence-agnostic tertiary structures, but no rigorous metrics are utilized to convincingly make this case. This paper makes several contributions. It builds on momentum in graph representation learning and formalizes a protein tertiary structure as a contact graph. It demonstrates that graph representation learning outperforms models based on image convolution. This work also equips graph-based deep latent variable models with the ability to learn from experimentally-available tertiary structures of proteins of varying lengths. The resulting models are shown to outperform state-of-the-art ones on rigorous metrics that quantify both local and distal patterns in physically-realistic protein structures. We hope this work will spur further research in deep generative models for obtaining a broader view of the structure space of a protein molecule.

Keywords: graph representation learning · protein structure plasticity · conformation sampling

1 Introduction

Deep neural networks can learn directly from sequences and structures of proteins and accurately predict contacts of a novel amino-acid sequence. ResNet [21] was a precursor to AlphaFold2 [11], which recently showed that deep learning can predict tertiary structure from a sequence with high accuracy. This seminal development will support many structure-centric studies. Decades of research also show that a single-structure view ignores the inherent structural plasticity that proteins harness to regulate molecular interactions [2]. The Protein Data Bank (PDB) [1] is rich in proteins arrested in diverse biologically-active three-dimensional (3D)/tertiary structures [14]. Obtaining a broader view of the

structure space accessed by a protein is essential to advance our understanding of molecular mechanisms and support the development of therapeutics. Literature on methods for exploring the protein structure space is rich [15]. The majority operate under the umbrella of optimization and enhance the sampling capability of Monte Carlo- or Molecular Dynamics-based methods.

Momentum in deep generative models has recently spilled to protein structure modeling. A detailed review is beyond the scope of this paper, but we highlight here several lines of observation. Most related work confounds protein design, folding, and structure modeling. The objective is often not clearly stated. Largely, the goal is to show that generated tertiary structures *look like* structures of proteins, but rigorous evaluation is lacking. Most work does not connect with the long body of work and domain-specific insight that should inform and guide a quantitative evaluation of the realism of generated structures. Finally, most work leverages the generative adversarial network (GAN) architecture and builds over image-based convolution, representing a tertiary structure as a contact map or distance matrix, which are both 2D-based representations of a 3D structure. A more comprehensive survey of the landscape of deep generative models for protein structure modeling can be found in [10]. Several questions remain unanswered: (1) does 2D-based convolution suffice? (2) Which generative architecture is more powerful? (3) Are generated tertiary structures realistic?

In this paper, we provide answers to some of these questions and establish a clear, quantitative evaluation of the capability of deep generative models. First, we clarify our modest objective: Given known protein tertiary structures, learn to generate physically-realistic *conformations* of type-less amino-acid chains. Note how we utilize the concept of *a conformation*, which indicates a specific choice of representing a tertiary structure. A contact map is a conformation; a distance matrix is a conformation, resulting from a different representation choice. In this paper, our choice of conformation is a *contact graph*, which allows us to capture and leverage local and distal constraints inherent in a protein tertiary structure.

Guided by this objective, in this paper we carry out a detailed comparison that considers different architectures, such as GANs, Recurrent Neural Networks (RNNs), and Variational Autoencoders (VAEs). We evaluate the quality of generated conformations along several metrics that capture both local and distal patterns in (physically realistic) protein tertiary structures. Our most important contribution, from a methodological point of view, is our leveraging of graph representation learning, inspired by its promise for small molecule generation; *we present for the first time a graph-based model for protein conformation sampling.* We thus provide an additional dimension to the comparative evaluation, 2D convolution versus graph convolution models, and show through rigorous metrics that graph representation learning is more powerful to further advance protein conformation sampling. Before proceeding with methodological details, we first provide a concise review of the related work.

1.1 Related Work

Current deep generative methods represent a tertiary structure as a contact map or distance matrix, encoding the spatial proximity of pairs of amino acids (often

collapsing an amino acid to its central carbon(CA) atom). Two CA's not further than 8Å in Euclidean space are considered in contact, and this is denoted by a 1 in the corresponding $[i, j]$ entry in the contact map/matrix, which is indexed by the position of CAs along the protein chain (from N- to C-terminus). A distance matrix records the actual Euclidean distances between CA pairs.

As the review in [10] shows, early works employed dihedral angle-based representations, but generated conformations contained many steric clashes, as such representations are under-constrained. State-of-the-art (SOTA) work uses contact maps or distance matrices and leverage 2D convolution (C)GANs popular in computer vision. Some methods specialize the loss function to focus the network to learn the symmetry of contact maps [9] or the sparse contacts/distances between amino acids far apart in the chain [5]. Recent work in [18] shows that, while GANs have become predominant frameworks, the quality of the contact or distance matrices they produce varies. Work in [18] proposes a Wasserstein GAN (WGAN) model that improves the quality of generated contact maps, and we use it as a baseline model here. We recall that WGAN uses 2D convolution, effectively treating a contact map or distance matrix as an image. Work in [18] also debuts metrics that quantify the local and distal patterns in physically-realistic tertiary structures, which we extend here to evaluate generated conformations of chains of varying lengths.

2 Methods

A conformation here is a contact graph, where amino acids become vertices, and contacts between pairs of CAs representing amino acids become edges. Vertices are labeled with the position of the corresponding CA atom in the chain. Using such graph representations allows us to adapt and evaluate GraphRNN and GraphVAE. As a baseline, to show the power of the graph representation, we utilize WGAN, which uses 2D convolution over contact maps. We summarize next GraphRNN and provide more details into GraphVAE, which we extend to learn from tertiary structures of varying-length protein chains.

GraphRNN: Consider an undirected graph $G = (A, E)$, where A represents the adjacency matrix of the graph, and E represents the edge attributes/features attached to each edge [22]. The goal of GraphRNN is to learn a distribution $p(G)$ over a set of observed graphs. GraphRNN defines a mapping f_S from graphs to sequences $f_S(G, \pi) = (S_1^\pi, ..., S_n^\pi)$, where π is a vertex ordering, and S_i^π denotes an adjacency vector representing edges between vertex v_i and previous vertices already in the graph. GraphRNN recasts $p(G)$ as the marginal distribution of the joint distribution $p(G, S^\pi)$: $p(G) = \sum_{S^\pi} p(S^\pi)\mathbb{1}[f_G(S^\pi) = G]$. Here, $p(S^\pi)$ is the distribution that needs to be learned. By approaching graph generation as a sequential process, GraphRNN formulates $p(S^\pi)$ as a product of conditional distributions over the elements; that is, $p(S^\pi) = \prod_{i=0\ n} p(S_i^\pi | S_0^\pi, ..., S_{i-1}^\pi)$. Further details can be found in [22]. GraphRNN has been applied to learn grid networks, community (network) structures, and other real networks. It has also been employed in the generation of small molecules of a few dozen atoms. This paper evaluates it for the first time on the generation of protein structures.

GraphVAE: First proposed in [20], GraphVAE is able to generate probabilistic fully-connected graphs, where the output tensor consists of graphs of a fixed size. As illustrated in Fig. 1, GraphVAE, like traditional VAEs, consists of two separate networks, an encoder and a decoder that are trained simultaneously [6–8]. Let $G = (A, E, F)$ be a graph specified with its adjacency matrix A, edge features E, and node features F. The encoder maps a graph G to a latent space z by calculating the variational posterior $q(z|G)$. The decoder, a densely-connected layer, reconstructs the input graph in the form of $\tilde{G} = (\tilde{A}, \tilde{E}, \tilde{F})$ from the latent space z by a generative distribution $p(G|z)$. It subsequently employs a *reparameterization* trick that allows backpropagation of loss. Similar to VAEs [13], GraphVAE's loss also consists of Kullback-Liebler divergence (KL) loss and reconstruction loss. These respectively measure the divergence between $q(z|G)$ and $p(z)$ and the accuracy of the reconstructed graph $\tilde{G}$ as compared to the original graph G. One important aspect in which our GraphVAE differs from that of [20] is that we have circumvented the need for an explicit likelihood loss by making the encoder node-invariant through employing graph convolutions and node aggregation, inspired by the work in [4]. This also implies that the encoder can choose any one of the possible node orderings for a particular task, whereas the GraphVAE in [20] utilizes a max pooling matching algorithm [3].

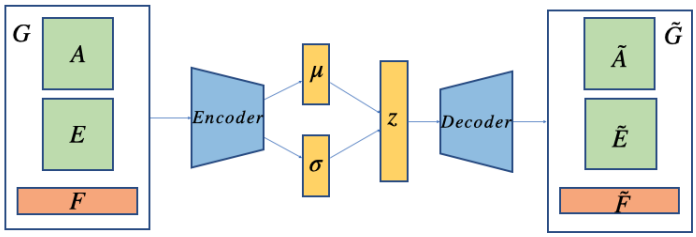

Fig. 1. Schematic shows the main components of GraphVAE.

vGraphVAE: We extend GraphVAE so that it can handle variable-size input contact graphs. We refer to the resulting model as *vGraphVAE*. We first find the longest chain in the input dataset. This becomes our target length (the target number of vertices in contact graphs). Each contact graph of a tertiary structure whose chain is shorter than the maximum length is padded with dummy vertices; the dummy vertices participate in no edges. During the interpretation of the output contact graph, we look for the first position where the aforementioned dummy vertex occurs, and we conduct the subsequent calculation up to that particular position.

Datasets: Our input dataset of tertiary structures is as in [16, 18]; 115, 850 tertiary structures are extracted from the PDB from entries listed in [16]. This set contains proteins of different lengths, which both GraphRNN and vGraphVAE can handle, but WGAN cannot. We create three different views/training datasets of fixed-length chains to evaluate WGAN. We refer to these as FL=16, FL=64, and FL=128 to indicate respective chain lengths of 16, 64, and 128 amino

acids. As in [18], non-overlapping protein fragments of a given length l are extracted from chain 'A' for each protein structure starting at the first residue. The corresponding contact map is calculated and added to the training dataset for WGAN. This process is followed to obtain $115,850$ maps in the FL=16 dataset and $98,966$ maps in the FL=64 and FL=128 datasets. Each model, WGAN, GraphRNN, and GraphVAE are separately trained on each of these datasets and then evaluated in each setting. Finally, GraphRNN and vGraphVAE are compared to one another when the restriction of fixed-length is lifted. In that case, we consider the whole dataset of $115,850$ structures, from which we extract contact graphs. In each case, a $0.8 : 0.1 : 0.1$ split is followed for training, validation, and testing.

Metrics to Evaluate a Conformation: Inspired by the work in [18], we use domain-specific metrics to evaluate a generated conformation. First, we evaluate the presence of a backbone in a contact graph, which should be evident as an edge between two nodes corresponding to consecutive CAs. We sum up the number of such edges and refer to this metric as BackboneScore, thus summarizing each contact graph/conformation with a BackboneScore. Ideally, for a model trained over an $FL = l$ dataset, the average BackboneScore value over generated conformations should be $l - 1$. Smaller values indicate missing portions of the backbone. The order of amino acids in a given protein sequence tells us where the backbone is. So, we do not really need to learn it. However, it is a fundamental task and intrinsic characteristic in every structure that a powerful model ought to learn easily. Learning off-backbone, distal patterns is more challenging. We employ the concept of short- versus long-range contacts, inspired by the work in [18]. For short-range contacts, we count the number of (i, j) edges (which connect vertices labeled i and j) in a contact graph, where $1 < |j - i| \leq 4$ (the lower bound excludes the backbone). For long-range contacts, we restrict $|j - i| > 4$. Work in [18] considers contact maps of a fixed size. Here we evaluate models that can learn from a training dataset of varying-size contact graphs. So, we propose modifications of the above metrics to evaluate a contact graph by we normalizing them (dividing them by the number of vertices in a graph).

Metrics to Compare Distributions: The above metrics provide us with various ways to summarize a contact graph/conformation. They allow comparing the training to the generated dataset by comparing distributions of specific metrics. We make use of the Bhattacharya distance (BD) and the Earthmover Distance (EMD). BD [12] measures the distance between two distributions $p(x)$ and $q(x)$ defined over the same domain X. It is defined as $BD(p, q) = -\ln(BC(p, q))$. 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 ∞. EMD [19] is also known as the Wasserstein distance. 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 [19].

Implementation Details: All models are implemented using Pytorch [17]. Experiments are run on an NVIDIA Tesla V100 GPU, where an epoch of GraphRNN, vGraphVAE, and WGAN takes 19.51, 425.77, and 184.0 seconds, respectively.

3 Results

3.1 Experimental Setup

Part I of our experiments compare the performance of models trained on experimental protein tertiary structures of fixed-length chains. Trained WGAN, GraphRNN, and GraphVAE are compared on the quality of the dataset generated from each, using domain-agnostic and domain-specific metrics. Part II of our experiments compares the performance of models trained on experimental tertiary structures of varying-length chains. In each setting, we carry out three major analyses. We evaluate whether a trained model has learned to generate a backbone, short-range contacts, and long-range contacts. We make use of BD and EMD to compare distributions over the generated versus training dataset, as well as some visualizations of selected distributions. We arrest models at 10, 20, 30, and 50 epochs during the training process, so we can obtain a dynamic view. Inspection of loss over epochs shows that all models converge fast, within a few epochs (data not shown).

3.2 Evaluation of Models on Fixed-Length Chains

As described in Section 2, we design three experiments, constructing three separate training datasets; contacts graphs of 16, 64, and 128 vertices; for WGAN, these correspond to the number of rows and columns in contact maps. We refer to these datasets as FL = 16, FL = 64, and FL = 128, respectively. Each model is trained separately on each dataset and then utilized to generate contact graphs (of the corresponding size as in the respective training dataset), and the generated contact graphs are evaluated.

We first evaluate the presence of a backbone. Table 1 reports the average BackboneScore value over all instances generated by a model. Ideally, we expect average values to be nearly identical to FL − 1. Table 1 allows making several observations. The worst-performing model is GraphRNN. The average BackboneScore values it reports deviate significantly from the ideal ones. In comparison, WGAN and GraphVAE perform much better. However, WGAN's performance increases when the fragment length increases. The model has trouble on the shorter chains (see FL = 16). While better performing on the longer chains, FL=64 and FL=128, its performance varies significantly over training epochs; that is, this model is not stable with respect to learning the backbone. In contrast, GraphVAE is both the best performing, reporting values nearly identical to what is expected and stable over the training epochs.

The distribution of the number of short-range contacts over generated conformations is now compared to the corresponding distribution over the training data via BD and EMD. In the interest of space, we show here only the BD values,

Table 1. For each training dataset, the table reports the average of the generated distribution of BackboneScores from models arrested at 10, 20, 30, and 50 epochs.

FL	WGAN				GraphRNN				GraphVAE			
	10	**20**	**30**	**50**	**10**	**20**	**30**	**50**	**10**	**20**	**30**	**50**
128	126.98	125.26	114.23	71.37	6.17	6.09	6.17	6.29	126.86	126.90	126.91	126.89
64	63.0	61.45	62.04	55.87	5.39	5.56	5.14	5.43	62.93	62.95	62.96	62.93
16	1.82	0.84	0.31	0.04	2.60	2.83	2.81	3.81	14.99	14.99	14.99	14.99

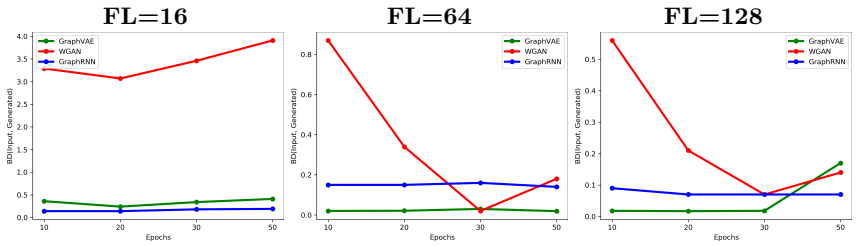

Fig. 2. The distribution of the number of short-range contacts in the generated dataset is compared to that in the training dataset via BD. The progression of BD's as a function of the number of training epochs for a specific model is tracked to show its impact on the quality of the generated dataset. This comparison is conducted separately for the models trained on the FL=16, FL=64, and FL=128 datasets.

plotted in Fig. 2 for each model (on each training dataset) over epochs 10, 20, 30, and 50. Fig. 2 shows that WGAN's generated distribution deviates significantly from the training distribution for the FL=16 setting and does not improve with further training. The model improves with further training on the longer chains (FL=64 and FL=128). While GraphVAE and GraphRNN are close in performance, GraphVAE outperforms GraphRNN for the longer chains. There is an increase after 30 epochs on FL=128, which suggests local instability. Altogether, the results suggest that GraphVAE is very effective at learning and reproducing the patterns of short-range contacts as in the training dataset. The EMD-based analysis confirms this (data not shown). We also show the actual distributions

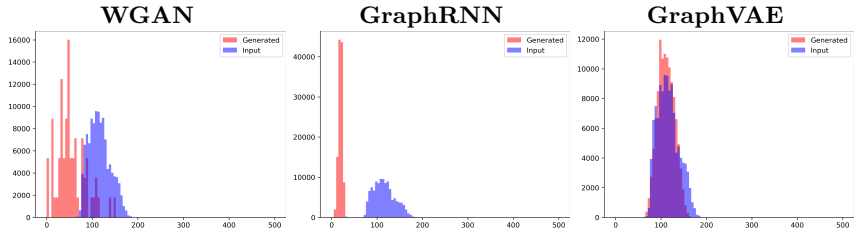

Fig. 3. The distribution of the number of short-range contacts corresponding to the generated dataset for each of the models is shown here. The visualization is limited to models trained on the FL=64 dataset and arrested at 50 training epochs.

of short-range contacts over the training and generated dataset. The visualiza-

tion in Fig. 3 is limited to models trained on the FL=64 dataset and arrested at 50 training epochs. Fig. 3 visually confirms the quantitative, rigorous analysis above, showing that the distribution of short-range contacts over the generated dataset is closer to the training distribution for GraphVAE, followed then by GraphRNN and by WGAN.

We repeat the above analysis but now for long-range contacts, again showing only BD values. Fig. 4 shows that WGAN is not the best-performing model at any of the three training regimes. On FL=16 and FL = 64, GraphVAE and GraphRNN performs similarly. On FL=128, the performance of GraphVAE improves steadily over training epochs. The experiments suggest that GraphRNN is the more stable, followed closely by GraphVAE. The EMD-based analysis and the visualization of distributions confirm these observations (data not shown).

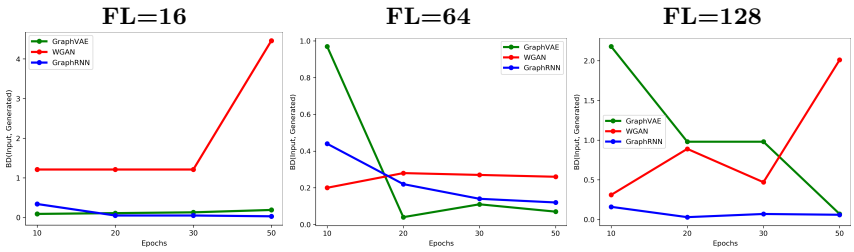

Fig. 4. The distribution of the number of long-range contacts in the generated dataset is compared to that in the training dataset via BD. The progression of BD's as a function of the number of training epochs for a specific model is tracked here to show its impact on the quality of the generated dataset. This comparison is conducted separately, for the models trained on the FL=16, FL=64, and FL=128 datasets.

3.3 Evaluation of Models on Variable-Length Chains

The rest of the experiments compare GraphRNN to vGraphVAE. We fist evaluate the presence of a backbone. Table 2 reports the average of the normalized BackboneScore over all contact graphs in the generated dataset (x 100%). A dynamic view is provided over training epochs. Table 2 clearly shows that GraphRNN fails to produce even 50% of the backbone on average, whereas vGraphVAE produces over 95% of the backbone on average.

Table 2. The table reports the average of the generated distribution of normalized BackboneScores (x 100%) from each model arrested at 10, 20, 30, and 50 epochs.

	Epochs			
Model	**10**	**20**	**30**	**50**
GraphRNN	42.67	40.03	36.87	39.77
vGraphVAE	98.36	95.04	97.56	92.68

The distribution of normalized short-range contacts values over the training and generated dataset is compared for each model via BD and EMD and reported

in the top panel of Fig. 5 over the training epochs. The bottom panel shows the actual distributions. Fig. 5 clearly shows that vGraphVAE achieves the best performance (lowest BD and EMD values) over all training epochs, as well.

The distribution of normalized long-range values over the training and generated dataset is compared for each model via BD and EMD and reported in the top panel of Fig. 6 over the training epochs. The bottom panel shows the actual distributions. The top panel of Fig. 6 shows that both models achieve low BD and EMD values. vGraphVAE achieves lower EMD values. The bottom panel shows a better overlap between the input and generated distributions, which EMD seems to capture better.

Finally, we visualize some contact graphs selected at random over those generated by GraphRNN and vGraphVAE. Fig. 7 draws them as contact maps, with bright yellow indicating edge/contact and dark blue indicating absence. The drawn contact maps are of high quality, with backbone, short-range, and long-range contacts, but those obtained by vGraphVAE are of higher quality. The visualization lends additional support to the conclusion that vGraphVAE is more effective than GraphRNN.

4 Conclusion

This paper shows that the contact graph formalization is a very useful modality that advances representation learning for protein tertiary structures. A detailed comparison that pitches convolution-based to graph-based models, considers different architectures, such as GANs, RNNs, and VAEs, and evaluates the quality of protein tertiary structures along several informative metrics, suggests that the GraphVAE architecture is a good step towards generative models for protein structure generation. The ability to learn directly from experimental structures of proteins of varying lengths deposited in databases, such as the PDB, further advances the state of the art.

While the focus of this paper has been on improving the quality of generated structures, our end goal is to advance deep generative models, so that they can give us a complete and detailed view of the structure space of a given protein molecule. Further directions of work will include making such models sequence-specific. In this paper, we utilize the highly informative contact graph representation of a tertiary structure. Many algorithms exist to reconstruct tertiary structures from contact maps, and a natural direction is to expand our work to end-to-end models that readily provide tertiary structures.

Acknowledgment

This work is supported in part by NSF Grant No. 1907805, 1900061, and 1763233. Computations were run on ARGO, a research computing cluster provided by the Office of Research Computing at George Mason University, VA. This material is additionally based upon work by AS supported while serving at the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

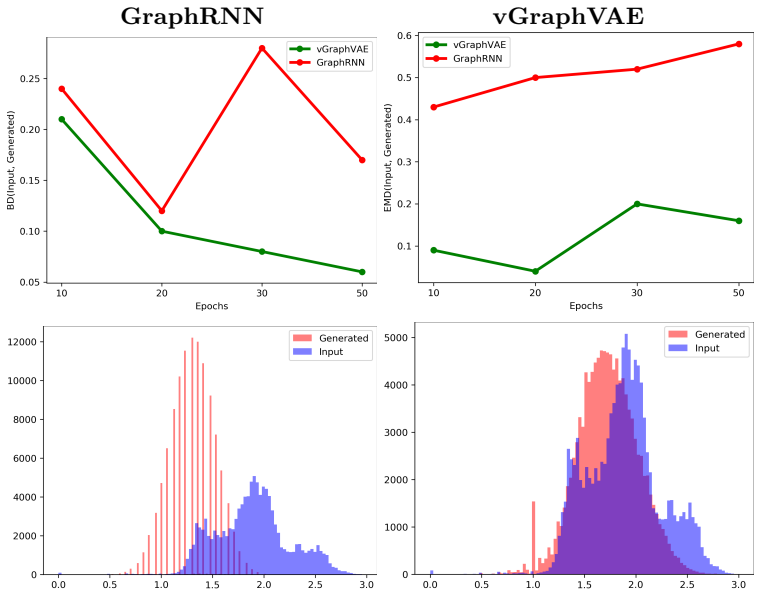

Fig. 5. Top panel: the distribution of normalized short-range contacts over the training and generated dataset are compared via (left) BD and (right) EMD. Bottom panel: the distributions are shown, superimposing the generated over the training distribution.

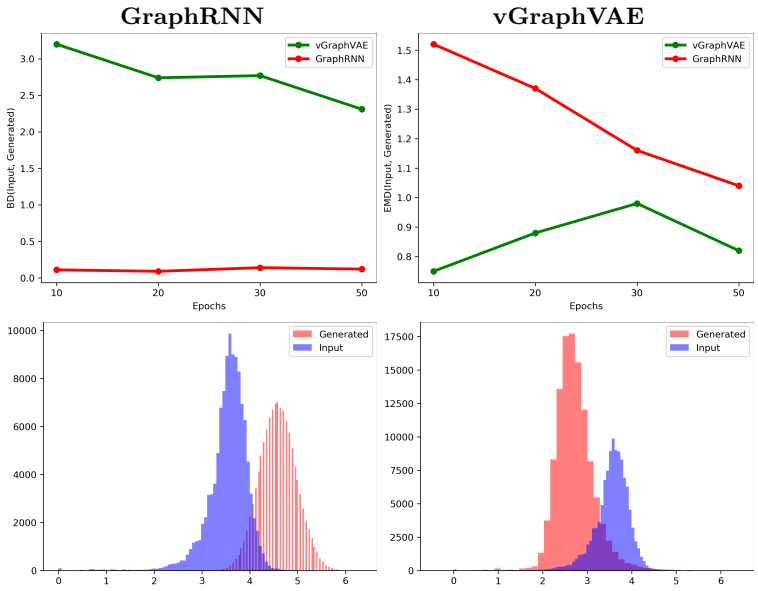

Fig. 6. Top panel: the distribution of normalized long-range contacts over the training and generated dataset are compared via (left) BD and (right) EMD. Bottom panel: the distributions are shown, superimposing the generated over the training distribution.

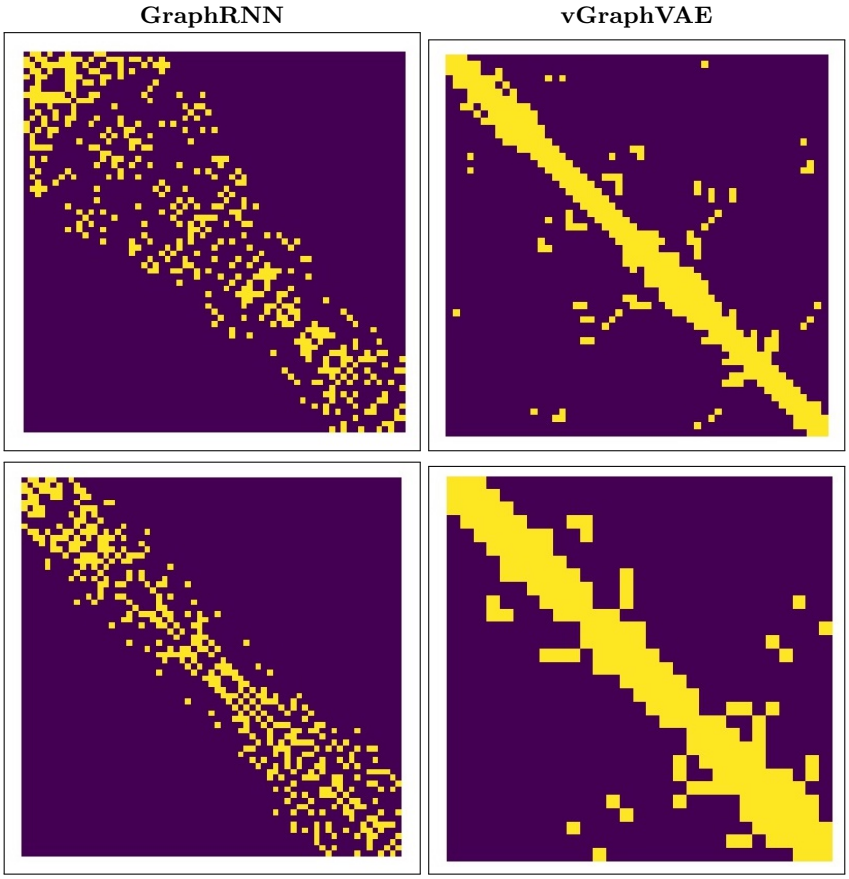

Fig. 7. Contact graphs are selected at random over generated data. Bright yellow indicates the presence of an edge/contact, and dark blue indicates the absence.

References

1. Berman, H.M., Henrick, K., Nakamura, H.: Announcing the worldwide Protein Data Bank. Nat. Struct. Biol. **10**(12), 980–980 (2003)
2. Boehr, D.D., Nussinov, R., Wright, P.E.: The role of dynamic conformational ensembles in biomolecular recognition. Nature Chem Biol **5**(11), 789–96 (2009)
3. Cho, M., Sun, J., Duchenne, O., Ponce, J.: Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2091–2098 (2014). https://doi.org/10.1109/CVPR.2014.268
4. De Cao, N., Kipf, T.: Molgan: An implicit generative model for small molecular graphs. arXiv preprint arXiv:1805.11973 (2018)
5. Ding, W., Gong, H.: Predicting the real-valued inter-residue distances for proteins. Advanced Science **7**(19), 2001314 (2020)

6. Du, Y., Guo, X., Shehu, A., Zhao, L.: Interpretable molecule generation via disentanglement learning. In: Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics. pp. 1–8 (2020)
7. Du, Y., Wang, S., Guo, X., Cao, H., Hu, S., Jiang, J., Varala, A., Angirekula, A., Zhao, L.: Graphgt: Machine learning datasets for graph generation and transformation. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2) (2021)
8. Guo, X., Du, Y., Zhao, L.: Property controllable variational autoencoder via invertible mutual dependence. In: International Conference on Learning Representations (2020)
9. Hang, H., Wang, M., Yu, Z., Zhao, X., Li, A.: GANcon: Protein contact map prediction with deep generative adversarial network. IEEE Access **8**, 80899–80907 (2020)
10. Hoseini, P., Zhao, L., Shehu, A.: Generative deep learning for macromolecular structure and dynamics. Curr. Opinion Struct. Biol. **67**, 170–177 (2020)
11. Jumper, J., Evans, R., et al.: Highly accurate protein structure prediction with alphafold. Nature (2021). https://doi.org/10.1038/s41586-021-03819-2
12. Kailath, T.: The divergence and bhattacharyya distance measures in signal selection. IEEE transactions on communication technology **15**(1), 52–60 (1967)
13. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
14. Maximova, T., Carr, D., Plaku, E., Shehu, A.: Sample-based models of protein structural transitions. In: ACM Conf Bioinf & Comp Biol (BCB). pp. 128–137. Seattle, WA (2016)
15. Maximova, T., Moffatt, R., Ma, B., Nussinov, R., Shehu, A.: Principles and overview of sampling methods for modeling macromolecular structure and dynamics. PLoS Comp. Biol. **12**(4), e1004619 (2016)
16. Namrata, A., Raphael, E., Po-Ssu, H.: Fully differentiable full-atom protein backbone generation. In: Intl Conf on Learning Representations (ICLR) Workshops: DeepGenStruct (2019)
17. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`
18. Rahman, T., Du, Y., Zhao, L., Shehu, A.: Generative adversarial learning of protein tertiary structures. Molecules **26**(5), 1209 (2021)
19. Rubner, Y., Tomasi, C., Guibas, L.J.: The earth mover's distance as a metric for image retrieval. International journal of computer vision **40**(2), 99–121 (2000)
20. Simonovsky, M., Komodakis, N.: Graphvae: Towards generation of small graphs using variational autoencoders. In: International conference on artificial neural networks. pp. 412–422. Springer (2018)
21. Xu, J., McPartlon, M., Lin, J.: Improved protein structure prediction by deep learning irrespective of co-evolution information. Nature Mach Intel **3**, 601–609 (2020)
22. You, J., Ying, R., Ren, X., Hamilton, W., Leskovec, J.: GraphRNN: Generating realistic graphs with deep auto-regressive models. In: International conference on machine learning. pp. 5708–5717. PMLR (2018)