From Interatomic Distances to Protein Tertiary Structures with a Deep Convolutional Neural Network

Yuanqi Du
Dept of Computer Science
George Mason University
ydu6@gmu.edu

Anowarul Kabir
Dept of Computer Science
George Mason University
akabir4@gmu.edu

Liang Zhao
Dept of Information Sciences and Technology
George Mason University
lzhao9@gmu.edu

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

ABSTRACT

Elucidating biologically-active protein structures remains a daunting task both in the wet and dry laboratory, and many proteins lack structural characterization. This lack of knowledge continues to motivate the development of computational methods for protein structure prediction. Methods are diverse in their approaches, and recent efforts have debuted deep learning-based methods for various sub-problems within the larger problem of protein structure prediction. In this paper, we focus on such a sub-problem, the reconstruction of three-dimensional structures consistent with given inter-atomic distances. Inspired by a recent architecture put forward in the larger context of generative frameworks, we design and evaluate a deep convolutional network model on experimentally- and computationally-obtained tertiary structures. Comparison with convex and stochastic optimization-based methods shows that the deep model is faster and similarly or more accurate, opening up several venues of further research to advance the larger problem of protein structure prediction.

CCS CONCEPTS

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

KEYWORDS

protein modeling; tertiary structure; coordinate reconstruction; deep learning.

ACM Reference Format:
Yuanqi Du, Anowarul Kabir, Liang Zhao, and Amarda Shehu. 2020. From Interatomic Distances to Protein Tertiary Structures with a Deep Convolutional Neural Network. 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, 8 pages. https://doi.org/10.1145/3388440.3414699

*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/10.1145/3388440.3414699

1 INTRODUCTION

Knowledge of the three-dimensional (tertiary) structure of a protein is central to understanding its biological activity in the cell [6]. Proteins exquisitely flex their chain(s) of amino acids to assume specific structures via which they "stick to" molecular partners [5]. Elucidating biologically-active structures of proteins, however, remains a daunting task both in the wet and dry laboratory [18]. In particular, according to current estimates, 44–54% of the proteome in eukaryotes and viruses lacks any structural characterization [23].

This current state continues to motivate computational approaches to protein structure prediction (PSP). The Critical Assessment of protein Structure Prediction (CASP) competition [10] pitches such approaches against one another to track progress and remaining challenges. Until recently, the dominating approaches in the most challenging category of *de-novo* PSP (where no structural template is available for a given amino-acid sequence) relied on a strategy known as molecular fragment replacement[16]. In this strategy, novel tertiary structures for a given amino-acid sequence are put together by stitching short fragments extracted from known, biologically-active structures deposited for proteins in the Protein Data Bank (PDB) [4]. Popular software suites, such as Rosetta [15], Quark [31], and others wrap such a strategy in a stochastic optimization framework that seeks structures which mimimize a given/designed empirical energy function; the latter sums atomic interactions in a given structure to associate with it a score. The score is a proxy of how relevant a computed tertiary structure is.

Over the past three years, the popularity and excitement associated with deep learning has found its way into the protein modeling community [3, 9, 11–13, 17, 19, 20, 24–27, 29]. For instance, work proposed by Ingraham et al. [12] employs a neural network (NN) to parameterize a sequence-conditioned scoring function of protein tertiary structure. Other works aim to learn energy functions and/or samplers in a variety of applications [3, 9, 11, 13, 17, 25, 27, 29].

Most notably, in the recent AlphaFold method [26], a deep Neural Network (NN) is first 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 stitching fragments into tertiary structures [26]. In [19], a generative adversarial network (GAN) is put forward to

replace the optimization-based fragment replacement strategy and directly predict a distribution of pairwise distance matrices for a given, fixed-length chain of amino acids. In [19], a so-called recovery network is additionally proposed to directly predict a tertiary structure (Cartesian coordinates of atoms) from a given matrix of pairwise distances among given atoms.

This recent body of work represents the enthusiasm of researchers keen to see the power of deep learning on a hallmark problem, such as PSP [28]. Interesting results have been obtained. For instance, the GAN in [19] generates protein-like structures, but it is not directly applicable to address PSP, as it does not take into account the amino-acid sequence. The GAN is also not applicable to the problem of protein design, which is stated as the motivation for designing a generative framework by the authors [19, 20], as the problem of protein design concerns finding an amino-acid sequence most compatible with a given tertiary structure.

Our synthesis of these initial efforts on debuting deep learning-based frameworks in PSP and more broadly, in molecular structure modeling research, suggests that these efforts are still very much in their nascency. In particular, generative deep learning-based approaches have yet to be demonstrated for PSP. However, our more immediate interest in this paper is the recovery network proposed in[20] to predict a tertiary structure from a given matrix of pairwise distances. This problem is a well-known one in protein modeling. The predominant approach is to treat the given distances as restraints/constraints, integrate each of them in a penalty term, and add these terms to a score/objective function [8, 14]. The same optimization-based approach used generally in PSP can then be used to obtain structures that reach low values of this scoring function where the distances are met as best as possible. One such protocol can also be put together in the Rosetta software suite, as demonstrated in Section 2 and utilized in section 3 to evaluate the performance of restraint scoring-based protocols.

Recovering Cartesian coordinates from pairwise distances is an interesting problem. Being able to do so is useful to complete end-to-end protocols that approach PSP as a problem, where the goal is to predict pairwise distances. Recovering the Cartesian coordinates consistent with pairwise distances is also of utility to structure determination in the wet laboratory, where it is often more expedient to obtain macroscopic measurements, such as NOE distances or mass spectrometry measurements [22]. It is worth noting that typically, in wet laboratories, ranges of distances rather than exact distances are obtained, and this type of information is better suited for restraint scoring-based approaches.

Inspired by work in [20], we explore deep neural networks to recover Cartesian coordinates from distances between pairs of atoms. The model used in [20] to complete the GAN-based framework and obtain tertiary structures from generated pairwise distance matrices is not detailed beyond the mention of it being a deep convolutional neural network (DCNN). DCNNs have been shown successful in many fields, such as computer vision, natural language processing, and others. Since they are capable of constructing linear and non-linear higher-level mapping of known and hidden low-level features, they are credible as a framework for drawing connections among atoms, driven by a suitable loss function.

In this paper, we explore the DCNN model space to find a model. We train the model on non-redundant protein structures and evaluate it rigorously on both experimentally- and computationally-obtained testing datasets. We demonstrate its performance in detail and evaluate it in a broader context by comparing it to other optimization-based approaches.

The rest of this paper is organized as follows. In Section 2 we describe the DCNN architecture, provide details of our model, its training and evaluation, and relate our implementation of alternative algorithms to which we compare DCNN. The evaluation is provided in Section 3. The paper concludes with a summary and list of future research directions in Section 4.

2 METHODS

The deep model trained in this paper takes as input pairwise atomic distances, which we detail first. We then describe the architecture of the model, the loss function we aim to minimize during training, other parameters of the trained model, and evaluation metrics.

2.1 Training, Validation, and Testing Datasets

In the interest of brevity, we assume that the reader is familiar with protein architecture. To obtain a training dataset for the deep model designed and evaluated in this paper, we first obtain experimentally-available, biologically-active tertiary structures of proteins. We extract a non-redundant image of the protein structures in the PBD via the PISCES server [30]. Specifically, we use the `cullpdb_pc20_res1.6_R0.25_d200702_chains3689.gz` pre-compiled dataset offered by the Dunbrack lab. This dataset removes proteins with more than 20% sequence similarity, with X-ray structures of resolution worse than than 1.6Å, or with NMR structures with resolution worse than an R factor of 0.25.

This dataset contains tertiary structures of proteins of various lengths (number of amino acids) at various resolution (varying number of atoms with specified Cartesian coordinates per amino acid). First, as in [20], we normalize all structures to have backbone-level resolution; that is, we discard all atoms but the backbone N, CA, C, and O atoms of each amino acid in a given tertiary structure.

Each of these structures (at backbone-level resolution) is then converted into a pairwise distance matrix. That is, the Euclidean distance between every pair of backbone atoms is computed and stored in a matrix. If a protein molecule has n amino acids, this means that we end up with a $4n \times 4n$ distance matrix.

Since the length of each protein varies, and we need a fixed-size distance matrix to feed into our model, we sample from a distance matrix. Sampling is done by a length of l residues with stride k amino acids in both row- and column-wise from a given distance matrix, where $k < l$. Thus, the sampled input distance matrix has size $4l \times 4l$. In our study, we take $l = 64$ and $k = 16$. Note that the sampling is a way of obtaining input matrices from proteins longer than 64 amino acids; structures of proteins shorter than 64 amino acids are discarded.

Finally, prior to feeding the obtained $4l \times 4l$ distance matrices to the deep model, we normalize the entries of a distance matrix to be in the range $[0, 1]$. Figure 1 shows such a normalized 256×256 distance matrix. The result of the process described above

is a dataset of 36,777 matrices corresponding to experimentally-available tertiary structures. We carry out a 60 : 20 : 20 split of this dataset to obtain a training, validation, and testing dataset, respectively.

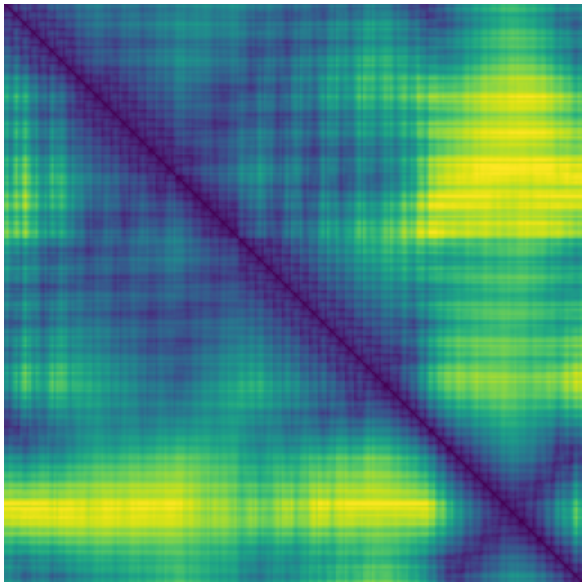

Figure 1: A distance matrix is shown as a heatmap for the purpose of illustration. A yellow-to-blue color scheme shows entries from 1 to 0 (corresponding to the normalized Euclidean distances among the backbone atoms).

The model is trained on the training dataset over iterations, and its performance over the training and validation dataset is monitored to ensure no over- or under-fitting issues emerge, as we show in Section 3. The model's performance on the testing dataset is related in Section 3, as well. We now describe the DCNN model summarized in Section 1 in greater detail.

2.2 From Pairwise Distances to Tertiary Structures via a DCNN

The architecture of the DCNN model we train over the training dataset is related in Figure 2. The model effectively maps (input) pairwise distances matrices of size $4l \times 4l$ into (output) coordinate matrices of size $4l \times 3$.

In the interest of brevity, we assume that the reader has some basic familiarity with neural networks and convolutional neural networks (CNNs), in particular. Our DCNN is built upon a simple architectural block as a 2d convolutional layer (conv2d) followed by a batch normalization layer and a ReLU activation layer. We have such 3 blocks and a final prediction layer. The kernel size in the first three convolutional layers is set to 4. In the final layer, the kernel size is set to 2. As Figure 2 shows, by going through four conv2d layers with batch normalization and ReLU activation, we gradually go from inputs (pairwise distance matrices) of size $4l \times 4l$ to outputs (Cartesian coordinates) of size $4l \times 3$, where l is the number of amino acids in a protein chain, and 4 refers to the number of backbone atoms in an amino acid; recall that in this paper we restrict l to 64.

During training, we use an Adam optimizer with learning rate of 0.0005, $\beta_1 = 0.5$, and $\beta_2 = 0.999$. We utilize a stochastic gradient descent and train the model over 5 epochs (1 epoch is over the whole training set once). It is worth noting that we do not know how this architecture relates to the one utilized in [20], as no details can be found in the paper. We also note that the hyper-parameters related above are the result of brief experimentation on our part on finding a suitable and effective model in the model space.

2.2.1 Loss Function. The loss function minimized during training over the training dataset consists of two terms. The first term is a scaled autoencoder reconstruction loss which penalizes mistakes in the reconstructed pairwise distance matrix. Specifically, the loss on a particular input pairwise distance matrix $m_{\text{input,i}}$ compares it to its reconstructed version $m_{\text{reconstructed,i}}$ (where i varies over the instances in the training dataset) as in:

$$L_1(m_{\text{input,i}}) = \frac{(m_{\text{reconstructed,i}} - m_{\text{input,i}})^2}{m^2_{\text{input,i}}}$$

.

The second term of the loss function now compares the input coordinates to the reconstructed coordinates. Let us refer to the input coordinates corresponding to an input pairwise distance matrix $m_{\text{input,i}}$ as $x(m_{\text{input,i}})$ and to the reconstructed coordinates corresponding to the reconstructed pairwise distance matrix $m_{\text{reconstructed,i}}$ as $x(m_{\text{reconstructed,i}})$. The reason for adding this term that compares coordinates is so as to recover the correct placement of the input coordinates over the space of rigid-body motions (translations and rotations) in three-dimensional (SE3) space. Specifically, this second term is defined as:

$$L_2(m_{\text{input,i}}) = ||R(x(m_{\text{input,i}})) + t - x(m_{\text{reconstructed,i}})||_2$$

where R and t denote 3D rotation and translation, respectively.

Note that this term effectively calculates what is also known as root-mean-squared-deviation (RMSD); RMSD is a popular dissimilarity metric to compare two structures of the same molecule. The above formula effectively removes the rigid-body motions by determining the 3D rotation and translation that minimize the Euclidean distance among the input and reconstructed coordinates.

We note that L_1 is implemented as in [20], whereas L_2 is different. While the work in [20] describes a similar reason for a second term of the loss function, it compares reconstructed coordinates to the input distance matrix, perhaps by mistake. As in [20], we combine these two terms in a single loss function as in: $L = L_1 + \lambda \cdot L_2$. The λ parameter weighs the contribution of the second term and is set to $1e-2$, as in [20].

2.3 Optimization-based Protocols to Reconstruct Tertiary Structures

In order to have a broad view of the contribution of our DCNN model, we implement two additional protocols to recover backbone Cartesian coordinates from pairwise distances of backbone atoms (for protein fragments of length 64 amino acids). Specifically, we leverage the ADMM algorithm, and the optimization of a distance restraint-based scoring function in the pyRosetta software.

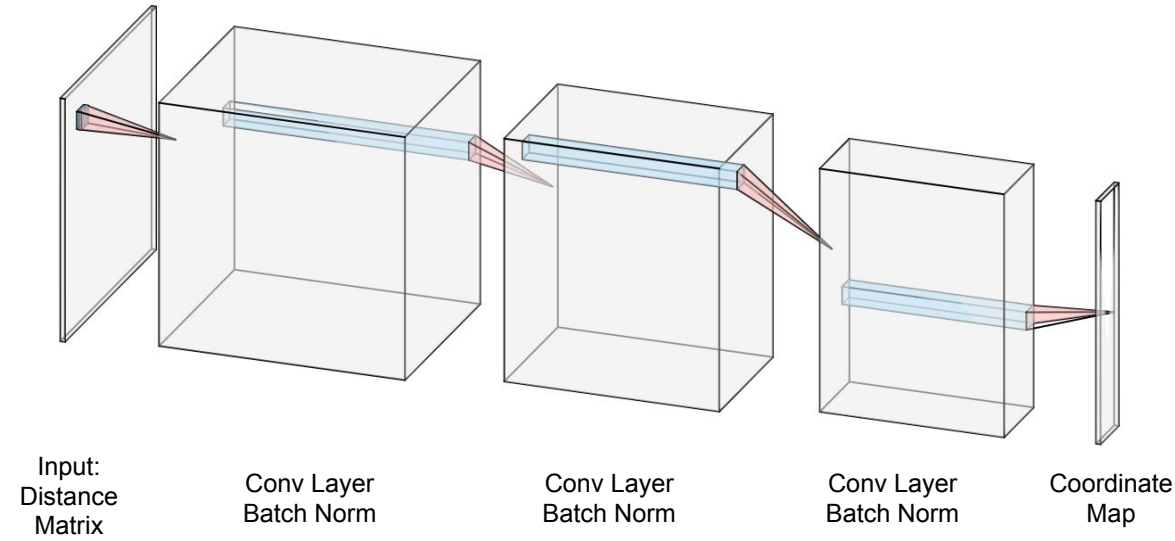

Figure 2: The DCNN model's architecture is related here. The model contains four conv2d layers with batch normalization and ReLU activation and gradually goes from inputs (pairwise distance matrices) of size $4l \times 4l$ to outputs (Cartesian coordinates) of size $4l \times 3$.

ADMM stands for "Alternating Direction Method of Multipliers" [7]. The algorithm combines dual decomposition and the technique of multipliers and is based on a formulation of the tertiary structure reconstruction problem as a convex optimization one [1]. For further details, we direct the reader to work in [7]. Our experience with the ADMM algorithm is that it fails to complete when applied to 256×256 pairwise distance matrices. So, for the purpose of our comparative analysis, we restrict ADMM to submatrices of size 64×64 which specify only CA-CA pairwise distances.

The reconstruction of tertiary structures from pairwise distances can also be approached as a stochastic optimization problem, if the distances are considered to be restraints. Each distance can be used to define a scoring term $(d - d_0)^2$, which penalizes structures where the distance d between a pair of atoms is different from the desired value d_0. These quadratic terms can be summed over all the given/desired distances and then added to an energy-based scoring function, such as the suite of scoring functions in the pyRosetta platform, to obtain a pseudo-scoring function for optimization.

Per instructional materials from the DeMaio laboratory on how to reconstruct tertiary structures in pyRosetta, we implement a simple protocol that samples backbone dihedral angles with values in Ramachandran maps and carries out the minimization of a pseudo-scoring function (with distance-based penalty terms as described above). This process is not guaranteed to reach a reasonable local minimum, and may be repeated several times. In fact, our experience suggests that this process is highly inefficient, needing to be run $10 - 20$ times in order to reach a reasonable tertiary structure consistent with many (not always all) of the restraints.

2.4 Implementation Details

All implementation is carried out in Python. Training our DCNN on the training dataset described above takes around 3 hours. Our implementation of the ADMM algorithm leverages the open source code in github available at ADMM_Link. The instructional materials by DeMaio are available at http://faculty.washington.edu/dimaio/files/HW_2_2020.pdf. Computational demands of the trained DCNN model to reconstruct the tertiary structure (of size 256×3) of a given testing distance matrix instance (of size 256×256) are around 0.06 seconds. As described above, ADMM cannot handle the 256×256 input instances, so we restrict instead to 64×64 distance matrices that specify only CA-CA pairwise distances. On such a matrix, ADMM takes around 1.5 minutes to complete and report CA coordinates. In contrast, each run of Rosetta (of which, as described above, an average of 10-20 are needed to obtain a good-quality structure) can take around 5 minutes; so the entire process to go from one 256×256 distance matrix to the corresponding tertiary structure can take $1 - 2$ hours.

3 RESULTS

3.1 Evaluation Performance Metrics

We carry out three sets of analyses. First, we evaluate the generality of our proposed DCNN model by plotting the training/validation loss curve and recording the average loss and RMSD over the corresponding dataset. Note that the RMSD is part of the loss function, but we hone in on it due to its importance in the protein modeling community and greater familiarity with interpreting it. We additionally show loss and RMSD over the testing dataset to relate the performance of the model. Second, we evaluate the performance of the model on 15 datasets of computationally-obtained tertiary structures (for 15 protein molecules). Specifically, we utilize a benchmark list of proteins [2, 21, 32, 33] for which tertiary structures have been generated via the Rosetta AbInitio protocol [15]. We refer the reader to previous work on details of how the protocol is used to generate tertiary structures [2, 32]. Finally, we compare

the model to existing, optimization-based approaches to recovering tertiary structures from pairwise distances which we implement in the Rosetta framework.

3.2 Performance of the DCNN Model on the Training, Validation, and Testing Datasets

Figure 3(a) shows the decrease of the loss function over training iterations over the training dataset (see red curve). As the model is trained, its performance in terms of loss over the validation dataset is also tracked (see blue curve). Figure 3(a) clearly shows that there is no over- or under-fitting, as the loss over the validation dataset tracks closely that over the training dataset. (Note we train the model for 5 epochs, where each epoch is around 20,000 iterations)

Figure 3(b) hones in and shows the distribution of RMSD values (the second part of the loss function) over the training and validation datasets, respectively. We recall that RMSD here measures the distance between a recovered tertiary structure and the original one utilized to construct a pairwise distance matrix fed as input to the model. So, the distribution of the RMSD values over the training dataset is comparing reconstructed/recovered structures from pairwise distance matrices in the training dataset to the original tertiary structures utilized to extract these distance matrices. The distribution of the RMSD values over the validation dataset is comparing recovered to original tertiary structures in the validation dataset withheld from training. As can be clearly seen in Figure 3(b), the distributions of the RSMD values over the training and the validation dataset track very closely to each-other.

Finally, the trained model is evaluated over the testing dataset. Figure 3(c) zooms in and shows the distribution of RMSD values over the testing dataset, superimposing it over the RMSD distribution over the training dataset. The distributions are very similar, showing in greater detail that the trained model performs just as well over the testing dataset.

The summary performance is related in Table 1, where the average loss, the average RMSD, the minimum RMSD, the maximum RMSD, and the standard deviation over RMSDs are related for the training, validation, and testing datasets. Table 1 shows that the model achieves comparable performance in terms of each of these summary metrics over each of the datasets.

3.3 Comparative Performance of the DCNN Model on Computed Datasets

We now test the model on 15 proteins of different lengths and folds that are benchmark targets used in algorithm development for PSP [21, 33]. The proteins vary in fold and length (from 64 to 146 amino acids). We abuse notation in the interest of space by identifying 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.

We note that though these proteins have different lengths, the same sampling procedure described in Section 2 is utilized to obtain testing datasets each of consisting of 5,000 fixed-size input distance matrices on which we can compare the performance of DCNN to other algorithms. As described in Section 2, the only feasible

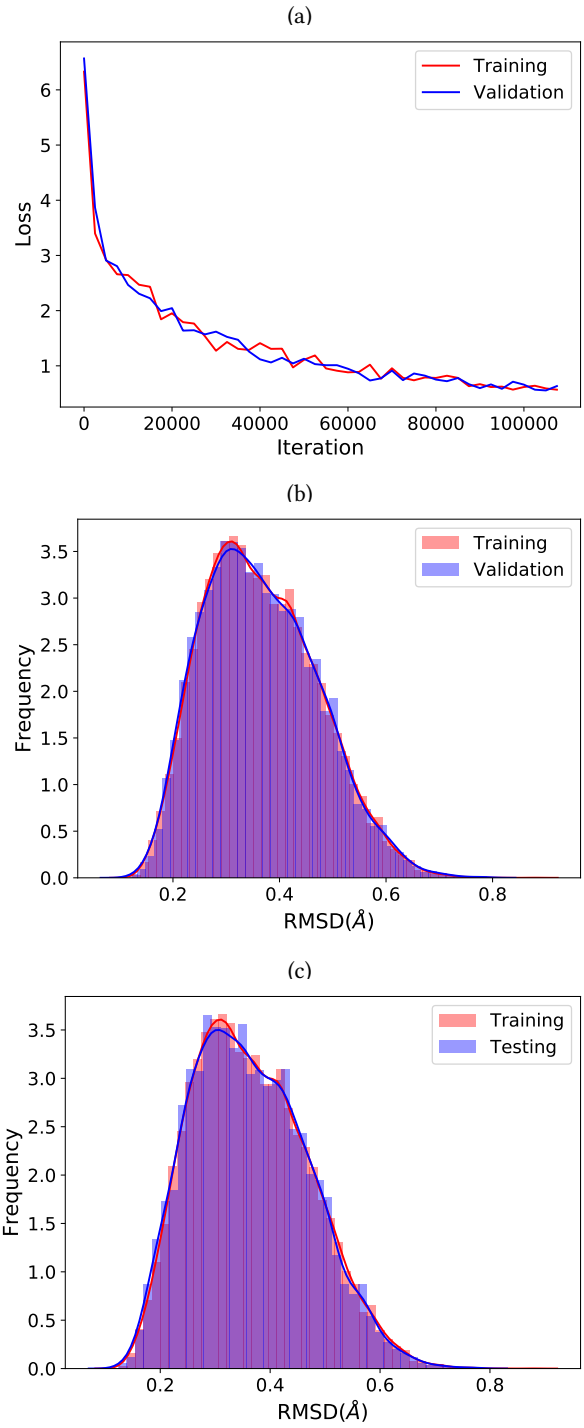

Figure 3: (a) The average loss over the training and validation dataset is shown per epoch of training. (b) The distribution of RMSDs between reconstructed and original tertiary structures in training and validation datasets are shown superimposed over each-other. (c) The distribution of RMSDs between reconstructed and original tertiary structures in training and testing datasets are shown superimposed over each-other.

Table 1: The performance of the model over the training, validation, and testing datasets is summarized here by showing the average loss, average RMSD, minimum RMD, maximum RMSD, and RMSD standard deviation over each of the datasets.

Dataset	Avg Loss	Avg RMSD (Å)	Min RMSD (Å)	Max RMSD (Å)	RMSD Std Dev (Å)
Training	0.519	0.365	0.125	0.880	0.106
Validation	0.521	0.365	0.118	0.789	0.106
Testing	0.519	0.363	0.123	0.778	0.105

algorithm among the ADMM and the Rosetta ones we describe is ADMM.

Since ADMM takes about 1.5 minutes on an instance, we restrict its evaluation while dedicating to it the same fixed budget as to DCNN. We consider the 3 hours of training needed by DCNN, as well as its total of 5 minutes on a testing dataset of 5,000 instances to determine the total budget of 10,805 seconds for ADMM. Considering its average of 90 seconds of running time on an instance, this fixed budget translates to around 120 instances, which we sample from a testing dataset in order to evaluate the performance of ADMM. We recall that while DCNN is evaluated on backbone tertiary structure reconstruction, ADMM is evaluated only on CA-only tertiary structure reconstruction. Since the running time of the Rosetta protocol is prohibitive, we do not include Rosetta in this systematic evaluation.

Table 2 summarizes the performance of DCNN on each of the 15 datasets by showing the average loss, average RMSD and standard deviation, as well as minimum and maximum RMSD. The performance of ADMM is listed next on each dataset by reporting each algorithm's average RMSD (and standard deviation) minimum, and maximum RMSD on a dataset.

Table 2 shows that both DCNN and ADMM perform well. Since ADMM has fewer constraints to consider (only CA-CA distances) and fewer cartesian coordinates to recover, it achieves lower average, minimum, and maximum RMSDs over DCNN. The average RMSDs obtained by DCNN, however, are all below 1Å(with maximum RMSDs not exceeding 1.3Å), which relates that the reconstructed structures are highly similar to the original structures; while higher than the values obtained by ADMM, the standard deviations obtained by DCNN are lower. Comparing the performance of DCNN on these datasets with its performance over experimentally-obtained data, one can observe that DCNN yields slightly higher loss over the computationally-generated datasets.

The distribution of RMSDs between the DCNN-reconstructed and original tertiary structures on some of the datasets are shown in Figure 4. In the interest of space, we do not show the related histograms for ADMM. Taken altogether, the comparative evaluation suggests that DCNN is a highly efficient and accurate algorithm.

3.4 Comparison with Rosetta

Since the computational demands of the Rosetta protocol are prohibitive for a systematic evaluation, we restrict relating its performance here to a few instances. We select the 1c8c(A) dataset, in which we select 15 instances/structures as the ground truth. We extract the distance matrix (of the backbone atoms) from each instance, and provide the distances as restraints to the Rosetta protocol described in Section 2. The protocol is run 10 times on each input to obtain 10 tertiary structures, of which we report the lowest-energy and the lowest-RMSD (to the input) ones. We compare these structures in terms of RMSD to the input structures and juxtapose these RMSDs to those obtained by DCNN on each of the instances, as well. The results are related in Table 3.

Table 3 shows that the tertiary structures reconstructed with the Rosetta protocol overall have RMSDs above 1Å; the lowest-energy structures are often the lowest-RMSD ones, but this is not always the case for all 15 instances. Though our comparison is mostly for the purpose of illustration, due to the prohibitive cost of running the protocol on all 5,000 instances of each of the 15 datasets on which we evaluate DCNN, Column 4 in Table 3 shows that the tertiary structures reconstructed with DCNN have sub-angstrom RMSDs from the ground truth/input tertiary structures. Anecdotally, we also report that we have not been able to find instances where the Rosetta protocol outperforms the DCNN model in terms of RMSD to the ground truth.

4 CONCLUSION

In this paper we have designed a DCNN model to recover Cartesian coordinates of tertiary protein structures from knowledge of the pairwise Euclidean distances between backbone atoms. A detailed analysis demonstrates the accuracy of the DCNN model in retrieving the ground truth. The comparison with a popular optimization-based approach shows that the DCNN model is similarly accurate while superior in terms of computational efficiency.

There are several directions we would like to investigate in future work. One direction concerns reducing the demand on the input representations, from distances between backbone atoms to distances between main-chain CA atoms only, and even distances between fragments of consecutive amino acids or a partial, incomplete list of the atoms. The latter will open up new avenues of applicability of the model to structure determination from experimental, macroscopic measurements. Another direction concerns expanding the model to consider not distances but contacts, reducing the presence or absence of distances below a threshold to 1s and 0s, respectively.

Finally, we will integrate these models in generative NN-based frameworks that will test the utility and possible advantages of such frameworks for structure modeling and prediction in comparison of established software. In this context, it will be important to understand the sensitivity of the model to missing or noisy input.

5 ACKNOWLEDGMENTS

The work is supported in part by the National Science Foundation Grant No. 1907805 and a Jeffress Trust Awards Program in Interdisciplinary Research Award. This material is additionally based upon work by AS supported by (while serving at) the National

Table 2: The performance of the model over the Rosetta-generated datasets is summarized here by showing the average loss, average RMSD (and standard deviation), minimum RMD, and maximum RMSD over each of the datasets. The performance of ADMM on a sampled subset of each Rosetta-generated dataset is summarized by showing the average RMSD (and standard deviation), minimum RMD, and maximum RMSD.

PDB ID	DCNN				ADMM		
	Avg Loss	RMSD (Å)			RMSD (Å)		
		Avg (Std Dev)	Min	Max	Avg (Std Dev)	Min	Max
1. 1hz6(A)	1.874	0.888 (0.058)	0.654	1.176	0.350 (0.111)	0.157	0.680
2. 1c8c(A)	1.932	0.904 (0.051)	0.684	1.136	0.401 (0.087)	0.122	0.604
3. 2ci2	1.722	0.835 (0.070)	0.579	1.156	0.400 (0.105)	0.120	0.632
4. 1sap	1.907	0.893 (0.051)	0.702	1.129	0.324 (0.114)	0.108	0.577
5. 1wap(A)	1.951	0.896 (0.081)	0.614	1.250	0.411 (0.081)	0.223	0.609
6. 1fwp	1.557	0.769 (0.091)	0.444	1.214	0.392 (0.094)	0.120	0.622
7. 1ail	1.571	0.789 (0.069)	0.577	1.089	0.399 (0.093)	0.167	0.610
8. 1dtj(A)	1.905	0.890 (0.066)	0.658	1.253	0.448 (0.105)	0.266	0.723
9. 1aoy	1.703	0.831 (0.063)	0.599	1.151	0.344 (0.098)	0.126	0.604
10. 1cc5	1.629	0.811 (0.068)	0.564	1.154	0.384 (0.113)	0.145	0.730
11. 1tig	1.788	0.870 (0.089)	0.601	1.188	0.374 (0.117)	0.134	0.643
12. 2ezk	1.519	0.767 (0.075)	0.507	1.052	0.399 (0.092)	0.172	0.622
13. 1hhp	1.679	0.818 (0.082)	0.554	1.166	0.369 (0.122)	0.075	0.718
14. 2h5n(D)	1.446	0.730 (0.095)	0.401	1.158	0.380 (0.085)	0.188	0.594
15. 1aly	1.554	0.774 (0.092)	0.357	1.145	0.345 (0.102)	0.113	0.640

Table 3: The performance of the Rosetta protocol is related on 10 instances selected from the 1c8c(A) dataset. The lowest-energy and lowest-RMSD structures obtained by 10 runs of the protocol on each instance (pairwise distances) are related in terms of RMSD to the ground truth/input instance. These RMSDs are juxtaposed to the RMSD obtained by DCNN on the same input instances.

Instance	RMSD (Å)		
	Rosetta		DCNN
	Lowest-energy	Lowest-RMSD	
1.	1.638	1.509	0.879
2.	1.525	1.318	0.893
3.	1.435	0.984	0.905
4.	1.256	1.256	0.913
5.	1.375	1.375	0.913
6.	1.364	1.215	0.865
7.	1.440	1.239	0.968
8.	1.578	1.578	0.909
9.	1.062	1.062	0.975
10.	1.723	1.658	0.910
11.	1.218	1.218	0.903
12.	1.107	1.107	0.882
13.	1.280	1.280	0.909
14.	1.459	1.409	0.897
15.	1.530	1.530	0.889

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] 2004. *Convex optimization.* Cambridge University Press.

[2] N. Akhter and A. Shehu. 2018. From Extraction of Local Structures of Protein Energy Landscapes to Improved Decoy Selection in Template-free Protein Structure Prediction. *Molecules* 23, 1 (2018), 216.

[3] D. Belanger, B. Yang, and A. McCallum. 2017. End-to-end learning for structured prediction energy networks. *Intl Conf Mach Learn (ICML)*, 429–439.

[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] D. D. Boehr, R. Nussinov, and P. E. Wright. 2009. The role of dynamic conformational ensembles in biomolecular recognition. *Nature Chem Biol* 5, 11 (2009), 789–96.

[6] D. D. Boehr and P. E. Wright. 2008. How do proteins interact? *Science* 320, 5882 (2008), 1429–1430.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Research in Machine Learning* 3, 1 (2011), 1–122.

[8] G. Chelvanayagam, L. Knecht, T. Jenny, S. A. Benner, and G. H. Gonnet. 1998. A combinatorial distance-constraint approach to predicting protein tertiary models from known secondary structure. *Folding & Design* 3, 3 (1998), 149–160.

[9] Z. Dai, A. Almahairi, P. Bachman, E. Hovy, and A. Courville. 2017. Calibrating energy-based generative adversarial networks. In *Intl Conf on Learning Representations (ICLR).* 1–17.

[10] Arne Elofsson, Keehyoung Joo, Chen Keasar, Jooyoung Lee, Ali HA Maghrabi, Balachandran Manavalan, Liam J McGuffin, David Ménendez Hurtado, Claudio Mirabello, Robert Pilstål, et al. 2018. Methods for estimation of model accuracy in CASP12. *Proteins: Struct, Funct, and Bioinf* 86 (2018), 361–373.

[11] Y. Feng, D. Wang, and Q. Liu. 2017. Learning to draw samples with amortized stein variational gradient descent. In *Intl Conf on Uncertainty in Artificial Intelligence.* 1–10.

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

[13] Taesup Kim and Yoshua Bengio. 2016. Deep Directed Generative Models with Energy-Based Probability Estimation. In *arXiv.* arXiv:1606.03439 http://arxiv.org/abs/1606.03439

[14] A. Kloczkowski, R. L. Jernigan, Z. Wu, G. Song, L. Yang, A. Kolinski, and P. Pokarowski. 2009. Distance Matrix-Based Approach to Protein Structure Prediction. *J Struct Funct Genomics* 10, 1 (2009), 67–81.

[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] J. Lee, P. Freddolino, and Y. Zhang. 2017. Ab initio protein structure prediction. In *From Protein Structure to Function with Bioinformatics* (2 ed.), D. J. Rigden (Ed.). Springer London, Chapter 1, 3–35.

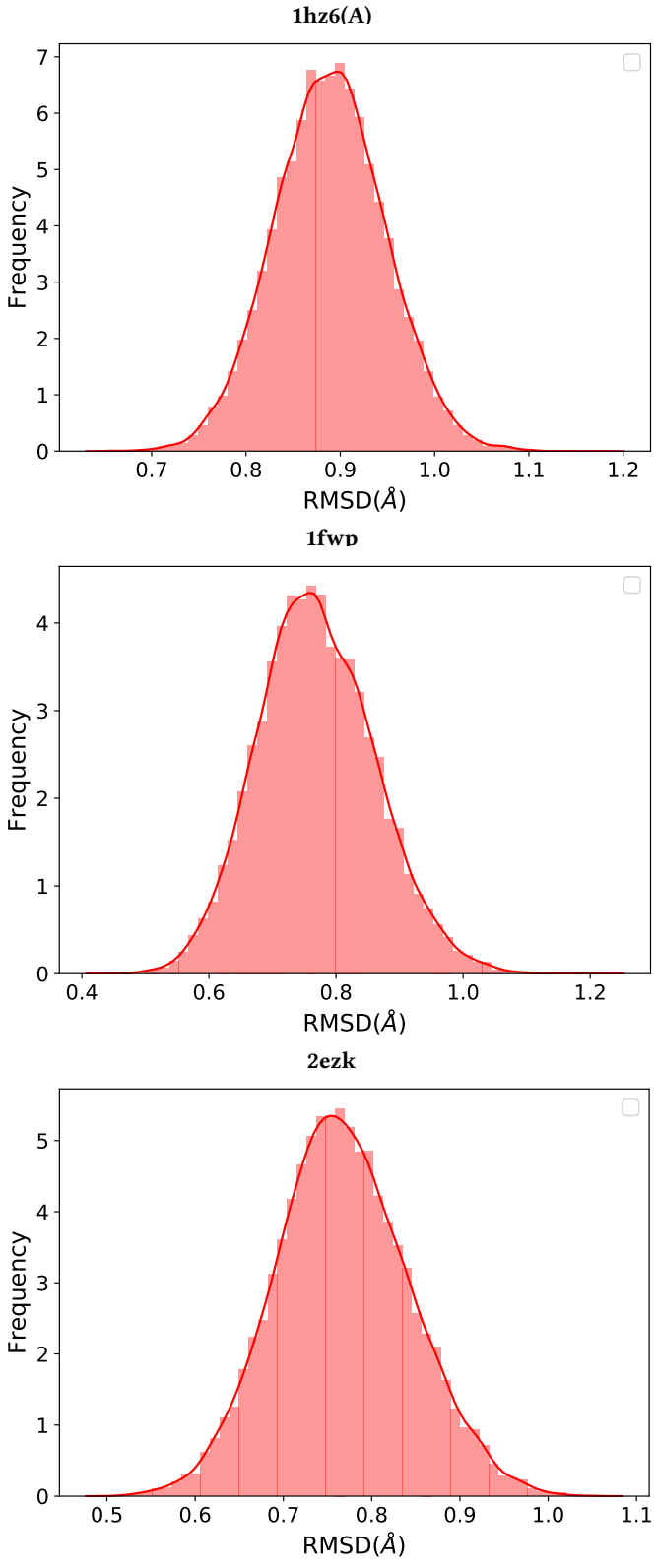

Figure 4: The histogram of RMSD values between DCNN-reconstructed and original tertiary structures in three representative Rosetta datasets are shown here.

[17] D. Levy, M. D. Hoffman, and J. Sohl-Dickstein. 2018. Generalizing Hamiltonian Monte Carlo with neural networks. In *Intl Conf on Learning Representations (ICLR)*. 1–15.

[18] 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.

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

[20] 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*.

[21] 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.

[22] Z. Orbá-Németh, R. Beveridge, D. M. Hollenstein, E. Rampler, T. Stranzl, O. Hudecz, J. Doblmann, P. Schlögelhofer, and K. Mechtler. 2018. Structural prediction of protein models using distance restraints derived from cross-linking mass spectrometry data. *Nature Protocols* 13, 3 (2018), 478–494.

[23] 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.

[24] 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.

[25] T. Salimans, D. Kingma, and M. Welling. 2015. Markov chain Monte Carlo and variational inference: Bridging the gap. In *Intl Conf on Machine Learning (ICML)*. 1218–1226.

[26] 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.

[27] J. Song, S. Zhao, and S. Ermon. 2017. A-nice-MC: Adversarial training for MCMC. *Neural Information Processing Systems* (2017), 5140–5150.

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

[29] D. Tran, R. Ranganath, and D. Blei. 2017. Hierarchical implicit models and likelihood-free variational inference. *Advances in Neural Information Processing Systems* (2017), 5523–5533.

[30] G. Wang and R. L. Dunbrack. 2003. PISCES: a protein sequence culling server. *Bioinformatics* 19, 12 (2003), 1589–1591.

[31] 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.

[32] A. Zaman and A. Shehu. 2019. Balancing multiple objectives in conformation sampling to control decoy diversity in template-free protein structure prediction. *BMC Bioinformatics* 20, 1 (2019), 211. https://doi.org/10.1186/s12859-019-2794-5

[33] 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.