Sequence-guided protein structure determination using graph convolutional and recurrent networks

Po-Nan Li  
Dept. of Electrical Eng.  
Stanford University  
Stanford, CA, USA  
liponan@stanford.edu

Saulo H. P. de Oliveira  
Division of Biosciences  
SLAC National Accelerator Laboratory  
Menlo Park, CA, USA  
oliveira@slac.stanford.edu

Soichi Wakatsuki  
Dept. of Structural Biology  
Stanford University  
Stanford, CA, USA  
soichi.wakatsuki@stanford.edu

Henry van den Bedem  
Atomwise  
San Francisco, CA, USA  
vbedem@atomwise.com

Abstract—Single particle, cryogenic electron microscopy (cryo-EM) experiments now routinely produce high-resolution data for large proteins and their complexes. Building an atomic model into a cryo-EM density map is challenging, particularly when no structure for the target protein is known a priori. Existing protocols for this type of task often rely on significant human intervention and can take hours to many days to produce an output. Here, we present a fully automated, template-free model building approach that is based entirely on neural networks. We use a graph convolutional network (GCN) to generate an embedding from a set of rotamer-based amino acid identities and candidate 3-dimensional CO locations. Starting from this embedding, we use a bidirectional long short-term memory (LSTM) module to order and label the candidate identities and atomic locations consistent with the input protein sequence to obtain a structural model. Our approach paves the way for determining protein structures from cryo-EM densities at a fraction of the time of existing approaches and without the need for human intervention.

Index Terms—Machine learning, Computational biology, Electron microscopy, Recurrent neural networks, Neural networks

I. INTRODUCTION

Recent advancements in cryogenic electron microscopy (cryo-EM), including better detector technologies and algorithmic developments, enabled high-resolution and atomic scale imaging of proteins and large biological complexes [1]. Determining protein structures is of fundamental importance to our understanding of protein function, disease mechanisms, and structure-based development of pharmaceuticals.

To construct an atomic model for a protein, typically tens of thousands of single-particle images are collected, sorted and aligned to reconstruct a 3-D density map volume. Next, atomic coordinates are built into the density map. The latter routine is known as the map-to-model process, which typically requires a considerable amount of human intervention and inspection, notwithstanding the availability of automated tools to aid the process [2]–[4].

Despite significant progress in machine learning techniques in 2-D or 3-D object detection [6]–[9] and protein folding [10], deep learning approaches to modeling atomic coordinates into cryo-EM densities remain relatively unexplored. Multiple research groups have proposed convolutional neural networks (CNNs) for detecting amino acid residues in a cryo-EM density map, but either did not address the final map-to-model step [11]–[14], or use a conventional optimization algorithm to construct the final model (see Related Work). Conventional search algorithms have high time- and space-complexity, constituting a bottleneck for large protein complexes, and are unable to exploit rich structural information encoded in genetic information [10].

Here, we address these shortcomings by presenting an approach for protein structure determination from cryo-EM densities based entirely on neural networks. First, we use a 3D CNN with residual blocks [17] we called RotamerNet to locate and predict amino acid and rotameric identities in the 3-D density map volume. Next, we apply a graph convolutional network (GCN) [18] to create a graph embedding using the nodes with 3D structural information generated by RotamerNet. Inspired by the UniRep approach [19], we then apply a bidirectional long short-term memory (LSTM) module to select and impose an ordering consistent with the protein sequence on the candidate amino acids, effectively generating a refined version of the graph with directed edges connecting amino acids (Fig. 1). We trained our LSTM on sequence data (UniRef50) alone, taking advantage of structural information encoded in the vast amount of genetic information [10]. In this paper we focus on the protein structure generation part with a GCN and an LSTM, which together we termed the Structure Generator. Our main contributions are:

- The first, to our knowledge, entirely neural network based approach to generate a protein structure from a set of candidate 3D rotameric identities and positions.
- Exploitation of genetic information learned from UniRef50 sequences to help generate a 3-D structure from cryo-EM data using a GCN embedding and a bidirectional LSTM.

II. RELATED WORK

A. Map-to-model for cryo-EM maps

Over the last few years, cryo-EM has evolved as a major experimental technique for determining novel structures of large proteins and their complexes. Computational techniques to process and analyze the data, and build protein structures are challenged by this avalanche of data. For example, widely-used de novo cryo-EM structure determination tools, e.g.,...
phenix.map_to_model [3, 4] or rosettaCM [5] partially automate cryo-EM data interpretation and reconstruction, but typically take many hours to generate a preliminary model and can require significant manual intervention. The underlying algorithms are often decades old, and are difficult to adapt to faster (e.g. graph processing unit, GPU) architectures. It will be critical to modernize these approaches, and capitalize on recent advances of deep learning and GPUs to expedite this procedure.

Several deep-learning based approaches for protein structure determination from cryo-EM data have been proposed. Li and coworkers [11] introduced a CNN-based approach to annotate the secondary structure elements in a density map, an approach later also proposed by Subramaniya et al. and Mostosi et al. [13, 14]. The feasibility of an end-to-end map-to-model pipeline with deep learning has also been explored. Xu and colleagues trained a number of 3-D CNNs with simulated data to localize and identify amino acid residues in a density map and use a Monte-Carlo Tree Search (MCTS) algorithm to build the protein backbone [15]. Using an entirely different architecture, Si and coworkers divided the map-to-model procedure into several tasks addressed by a cascade of CNNs. However, their procedure also relied on a conventional Tabu-Search algorithm to produce the final protein model [16].

### B. Graph neural networks

Graph neural networks [18, 20] are natural representations for molecular structures with atoms as nodes and covalent bonds as edges. Duvenaud and coworkers pioneered this approach using a GCN to learn molecular fingerprints, which are important in drug design [21]. Numerous other applications of GCNs to predict or generate molecular properties can be found in the literature. For example, Li and colleagues demonstrated the utility of a generative GCN to construct 3-D molecules from SMILES strings, among other applications [22].

### C. Long short-term memory

Recurrent neural networks (RNNs) are widely used in natural language processing tasks. Their architecture is designed to process, classify, or predict properties of sequences as input and can output sequences with desired properties [23]. Among many RNN architectures, the LSTM model was proposed to address the gradient vanishing problem for long sequences and a number of variants have since been studied to further increase its capacity, such as multi-layer and bidirectional LSTMs. LSTMs are often used in conjunction with other neural network models. An image captioning system, for example, can be realized by using a 2-D CNN that extracts high-dimensional features from an image and an LSTM that outputs a sentence describing the input image [24].

### III. Method

#### A. Model

In this section we present the Structure Generator, a neural network model for protein model building consisting of a GCN and, subsequently, a bidirectional LSTM module. The input for the Structure Generator is a set of nodes labeled with 3-D coordinates and amino acid identity. To generate a set of candidate amino acids we have previously implemented RotamerNet (unpublished), a 3-D CNN based on the ResNet architecture [17] that can identify amino acids and their rotameric identities in an EM map. This set of candidate amino acids is not constrained by the sequence, and and their 3-D locations are located based entirely on their density profiles. The set can contain false positives (an amino acid rotamer is proposed at a location where there is none) or false negative (a correct amino acid rotamer was not identified). RotamerNet outputs an amino acid and rotamer identity together with proposed coordinates for its Cα atom. In the remainder, we will only consider the amino acid identity. A node \( v \) is a proposed amino acid identity with the Cα coordinates.

Next, we generate a Cα contact map for all predicted Cα coordinate locations. We connect any two proposed Cα’s with a distance less than a given threshold (4.0 Å) with an undirected edge. We represent the input with two matrices: an \( m \times 20 \) matrix of node features and an \( m \times m \) adjacency matrix that describes the connectivity between nodes.

We generate a high dimensional embedding for each node \( v \),

\[
H_{node} = \text{GCN}(A, F),
\]

where \( H = [h^{(1)}_{node}, \ldots, h^{(m)}_{node}]^{\top} \), \( A \) is the adjacency matrix with \( a_{i,j} = 1 \) for each neighbor pair \((i, j)\) or the node itself, i.e. \( i = j \), and \( F = [f^{(1)}_{node}, \ldots, f^{(m)}_{node}]^{\top} \) are input features. Features are generated with \( f^{(s)}_{node} = \text{NN}(s^{(v)}) \), where \( s \in \mathbb{R}^{20} \) is the normalized softmax score vector for a node \( v \) obtained from the RotamerNet and \( \text{NN}() \) denotes a single-layer neural network. We implemented the GCN module following [18] (Fig. 2a). Note that the GCN can be applied in \( T \) layers to propagate messages, thereby increasing the capacity of the network [21, 22]. As depicted in Fig. 2a, in each of the GCN layer, messages propagate through edges, sharing the embedding of a node with its neighbors. For example when \( T = 2 \), \( H_{node} = \text{GCN}^{(2)}(A, \text{GCN}^{(1)}(A, F)) \), and these two GCN layers can share the same set or have different sets of parameters.

The Structure Generator then uses a bidirectional LSTM module as a decoder for the refined protein chain generation.
We use zero vectors for initial hidden and cell states. At each time step \( t \), an embedding of an amino acid in the sequence \( h^{(t)}_{\text{seq}} = \text{NN}(v^{(t)}_{\text{emb}}) \), where \( v^{(t)}_{\text{emb}} \in \mathbb{R}^{20} \) is a one-hot encoding for an amino acid at position \( t \) in the sequence, is fed into the LSTM cell. The cell output at each time step, \( h^{(t)}_P \), can be viewed as the current graph representation for \( P \), the protein structure to be built. A score \( z^{(v)}_{\text{node}} \in \mathbb{R} \) of a candidate node to be selected as the next node to be added to \( P \) is determined by \( z^{(v)}_{\text{node}} = \text{NN}(h^{(v)}_{\text{add}} + h^{(v)}_P) \), where \( h^{(v)}_{\text{add}} = \text{NN}(h^{(v)}_P) \). At each time step \( t \), the node with the highest softmax score \( p^{(v)}_{\text{node}} = \exp(z^{(v)}_{\text{node}}) / \sum_v \exp(z^{(v)}_{\text{node}}) \) is selected and added to \( P \). The selection process continues until the end of the sequence \( t = N \), where \( N \) is the length of the sequence, is reached, at which point the cross entropy loss is computed for the entire sequence in the training stage, or bitwise accuracy for the inference stage. We implemented the decoder with a bidirectional LSTM, in which the outputs from one LSTM fed with a forward sequence and another fed with a backward sequence are concatenated to obtain \( h^{(t)}_P \) for each time step \( t \). We found that a bidirectional LSTM consistently outperformed a uni-directional LSTM. We also found that using the ensemble of inference results with a forward (from N-terminus) and backward (from C-terminus) sequence further improves the accuracy. Fig. 2(b) illustrates the generation process with the sequence as the input at each time step (top) and the best corresponding node prediction as output (bottom). Importantly, the sequence information is used both in the training and inference stages to guide the protein modeling.

\[
\text{Loss} = - \sum_{t=1}^{n} \log p_j
\]

where \( n \) is the length of the ground truth sequence and \( m \) is the number of nodes in the raw graph, is calculated and back-propagated through the entire network, i.e. the LSTM and then the GCN. In the inference stage, the average accuracy

\[
AA = \frac{1}{K} \sum_{t=1}^{N_{\text{prot}}} \frac{1}{N_{\text{prot}}} \sum_{j=1}^{N_{\text{prot}}} 1(j_t = j_t),
\]

i.e., the fraction of amino acids whose identity was predicted correctly, is used to evaluate the performance of the Structure Generator on a set of \( K \) protein structures, where \( N_{\text{prot}} \) denotes the sequence length of a protein, \( j_t \) and \( j_t \) are the ground truth and predicted note index for the \( t \)-th step in the LSTM, respectively.

We trained the GCN with sequence embedding dimension 32, node embedding dimension 128 and LSTM hidden state dimension 2 \( \times \) 256 (256 for each direction). During training, we added (500\( - n \)) dummy (false positive) nodes with random edges in each iteration to complicate the training data.

### IV. Results

We first examined the effect of the GCN on structure determination. We found that the number of GCN layers can dramatically improve the average accuracy on the validation set. For example, using two rather than one GCN layer, i.e. \( T = 1 \) to \( T = 2 \), yields an 20\% improvement (Fig. 3(a)). Encouraged by this improvement, we further trained the Structure Generator with \( T = \{3, 4\} \). Fig. 3(a) shows the error rate \((1 - AA)\) curves on the validation data for different number of GCN layers \( T \). While increasing to \( T = 3 \) again gives another 0.3\% increase in average accuracy, \( T = 4 \) adds only 0.04\% (Table I). This observation suggest that \( T = 2 \),
TABLE I
AVERAGE ACCURACY ON VARIOUS VALIDATION DATASETS WITH DIFFERENT GCN LAYER AND INFERENCE SETTINGS.

| Dataset        | Inference | T = 0     | T = 1     | T = 2     | T = 3     | T = 4     |
|----------------|-----------|-----------|-----------|-----------|-----------|-----------|
| UniRef50       | Forward   | 0.0347    | 0.7952    | 0.9949    | 0.9981    | 0.9985    |
|                | Ensemble  | 0.0347    | 0.8383    | 0.9954    | 0.9986    | 0.9987    |
| ProteinNet     | Forward   | 0.8214    | 0.9899    | 0.9957    | 0.9960    |           |
|                | Ensemble  |           | 0.8577    | 0.9908    | 0.9960    | 0.9963    |
| RotamerNet     | Forward   | 0.6162    | 0.7443    | 0.6912    | 0.6853    |           |
|                | Ensemble  |           | 0.6409    | 0.7538    | 0.7060    | 0.6965    |

which can be interpreted as learning 5-mer spatial motifs in the graph (see discussion in IV-B) is sufficient for the model to capture the implicit structural information in the graph and the sequence. In the remainder, unless stated otherwise we fixed $T = 2$ for inference in all experiments. Fig. 3(b) shows the error counts on the 100 protein structure in the validation set as a function of sequence length, suggesting that the error counts increased very mildly with the length of sequence.

![Fig. 3](image-url)

Fig. 3. Validation results. $N = 100$. (a) Error rate, defined as $1 - \text{average accuracy}$, vs. training iterations with GCNs with different layers. (b) Error counts, i.e., the number of incorrectly assigned amino acid positions in a protein structure for the $T = 2$ model.

To study the efficacy of the GCN, we also tested a GCN with $T = 0$, i.e., the node features $F$ are added directly to the LSTM outputs. This model dramatically reduced the average accuracy to 0.0347, which is approximately the probability of randomly selecting the correct node out of candidate nodes that have the amino acid identity matching the sequence input. This result indicates that a GCN embedding of RotamerNet’s output is required for the LSTM to predict an ordered graph consistent with the sequence and the GCN.

A. Performance of the Structure Generator on the ProteinNet dataset

Next, we evaluated the Structure Generator on the ProteinNet data set, a standardized machine learning sequence-structure dataset with standardized splits for the protein structure prediction and design community [27]. The CASP12 ProteinNet validation set used here has 224 structures with sequence lengths ranging from 20 to 689, with median 163 and mean 204.4. We selected the same parameters as those for the UniRef50 dataset to generate simulated feature vectors and generated Cα contact maps based on the backbone atom coordinates from ProteinNet. We note that a small number of Cα coordinates are absent in ProteinNet owing to lack of experimental data. Compared to the UniRef50 validation set, the ProteinNet validation set is therefore more challenging, as edges in the input graph can be missing. Validation results on ProteinNet are given in the second row of Table I.

Nonetheless, well over 50% of the structures in the data set are correctly predicted without any errors (Fig. 4(a)). Remarkably, the Structure Generator can correctly predict amino acids for which the Cα records are missing. For example, atomic coordinates for the first three and last two amino acids of prosurvival protein A1 (PDB ID 2vog) are missing, but the Structure Generator can still completely reconstruct the protein model (Fig. 4(b)). Several amino acids, for example glutamine (Q), occur multiple times in the sequence. As a result, the last two rows in the Structure Generator output have repeating patterns (Fig. 4(c)), which did not prevent the Structure Generator from predicting the correct nodes for each of the positions corresponding to glutamine.

![Fig. 4](image-url)

Fig. 4. Test on the ProteinNet CASP12 validation set. (A) Error counts, i.e., number of wrong amino acid in a protein structure, and the histogram, with the $T = 2$ model. (b) Contact map of prosurvival protein A1 (PDB ID 2vog). Green pixels are contacts and purple pixels indicate where Cα coordinates are unknown and thus the contacts are missing. (c) The output of the Structure Generator on 2vog. Red means higher probability whereas blue means less likely.

B. Performance of the Structure Generator on RotamerNet data

Finally, to demonstrate the utility of our approach with an upstream machine learning approach, we tested the Struc-
ture Generator on output from RotamerNet. The RotamerNet validation data set consists of the amino acid type classification scores for simulated cryo-EM density maps from 45 protein structures of various lengths from 15 to 278 with various nominal resolution ranging from 1.4 to 1.8Å. The average accuracy of the RotamerNet validation set is 0.852, meaning that a non-trivial fraction of input features for the Structure Generator is noisy or incorrect. Fig. 5(a) shows the RotamerNet and the Structure Generator accuracy of the 45 proteins with various sequence lengths. Remarkably, while the performance of the Structure Generator is largely limited by the RotamerNet accuracy (data points beneath the dashed gray line in Fig. 5(a)), as indicated by the correlation, a number of proteins have higher Structure Generator accuracy than RotamerNet accuracy, suggesting that the Structure Generator can tolerate and recover errors from upstream machine learning approaches. To understand the characteristics of the Structure Generator, we plot the confusion matrix for the C-terminal calponin homology domain of alpha-parvin (PDB ID 2vzg). Among those amino acids whose identity and position are correctly predicted by RotamerNet and the Structure Generator (Fig. 5(b), blue dots on the diagonal), there are two red dots indicating that the prediction error from RotamerNet does not necessarily prevent the Structure Generator from making correct predictions. Again we point out that the Structure Generator has been trained only on UniRef50 sequences and simulated features, and has not been fine-tuned with the RotamerNet data. We anticipate that either doing so or training with the upstream model will further enhance the performance of the Structure Generator.

We observe in Table I that the $T = 2$ model performs best on the RotamerNet data. The Structure Generator relies on learning the correlation between the graph embedding and the motifs in the sequence. Increasing the number of GCN layers in principle allows the Structure Generator to recognize longer n-grams and spatial motifs of increased connectivity length. However, such correspondences will become increasingly noisy as lengths increase. Based on this observation, we therefore suggest that $T = 2$ is a practical choice.

V. Conclusion

Building an atomic model into a map is a time- and labor-intensive step in single particle cryo-EM structure determination, and mostly relies on traditional search algorithms that cannot exploit recent advancements in GPU computing and deep learning. To address these shortcomings, we have presented the Structure Generator, a full-neural network pipeline that can build a protein structural model from a set of unordered candidate amino acids generated by other machine learning models. Our experiments show that a GCN can effectively encode the output from the upstream model as a graph while a bidirectional LSTM can precisely decode and generate a directed amino acid chain, even when the input contains false or erroneous entries. Our experiments suggest that a two-layer GCN is sufficient for processing the raw graph while preventing over-fitting to the training data.

Fig. 5. Results on the RotamerNet data. (a) Structure Generator accuracy vs. RotamerNet accuracy. Each data point represents a protein structure. The color code indicates the length of the structure. (b) Amino acid position of a select structure, PDB ID 2vzg, predicted by the Structure Generator. Red pixels are where RotamerNet made incorrect prediction on the amino acid type. Sequence on the top is derived from RotamerNet output and on the right is the ground truth.

The Structure Generator exploits genetic information to guide the protein structure generation, and showed promising results on the RotamerNet data set without fine tuning. Training on the ProteinNet dataset and fine-tuning on the RotamerNet dataset will further enhance performance. While a practical machine learning model for cryo-EM map-to-model is still a work-in-progress, in part because of the lack of high-resolution experimental data for training, our proposed framework can complement the existing approaches and ultimately pave ways toward a fully trainable end-to-end machine learning map-to-model pipeline, making human intervention-free protein modelling in a fraction of a minute possible.

CURRENT AFFILIATION

This work was initiated when S.H.d.O. and H.v.d.B. were at SLAC National Accelerator Laboratory. S.H.d.O. is currently at Frontier Medicines, CA, USA. In addition to his position at Atomwise, H.v.d.B. is on the faculty of the Department of Bioengineering and Therapeutic Sciences, University of California, San Francisco, CA, USA.

REFERENCES

[1] Ewen Callaway, “Revolutionary cryo-EM is taking over structural biology,” Nature 578, 201 (2020).
[2] F. DiMaio and W. Chiu, “Chapter Ten – Tools for Model Building and Optimization into Near-Atomic Resolution Electron Cryo-Microscopy Density Maps” in Methods in Enzymology 579, 255–276, edited by R.A. Crowther (2016).
