UNIVERSAL TRANSFORMING GEOMETRIC NETWORK

Jin Li *
University of Chicago
Chicago, IL 60637, USA
jinli11@uchicago.edu

ABSTRACT

The recurrent geometric network (RGN), the first end-to-end differentiable neural architecture for protein structure prediction, is a competitive alternative to existing models. However, the RGN's use of recurrent neural networks (RNNs) as internal representations results in long training time and unstable gradients. And because of its sequential nature, it is less effective at learning global dependencies among amino acids than existing transformer architectures. We propose the Universal Transforming Geometric Network (UTGN), an end-to-end differentiable model that uses the encoder portion of the Universal Transformer architecture as an alternative for internal representations. Our experiments show that compared to RGN, UTGN achieve a 1.7 Å improvement on the free modeling portion and a 0.7 Å improvement on the template based modeling of the CASP12 competition.

1 INTRODUCTION

Proteins are chains of chemical units, called amino acids, that fold to form three dimensional structures. The ability to predict a protein's structure from its amino acid sequence remain to be the most elusive yet rewarding challenges in computational biology. The structure determines the protein's function, which can be used to help understand life threatening diseases and accelerate drug discover (Kuntz (1992)). However, experimental methods for solving a protein's structure is both time consuming and costly, and they only account for a small percentage of known protein sequences.

Earlier computational methods for predicting protein structure includes molecular dynamics (MD), which use physic based equations to simulate the trajectory of a protein's folding process into a stable final 3D conformation (Marx & Hutter (2010)). However, this method is computationally expensive and ineffective for larger proteins. Other approaches include using co-evolutional information to predict the residue-residue contact map, which can be used to guide structure prediction methods. With the help of deep learning architectures like convolutional neural networks, contact prediction remains to be the prevailing methods in structure prediction (Wang et al. (2016)). However, because these method does not provide an explicit mapping from sequence to structure, they lack the ability to capture intrinsic information between the sequence and structure.

RGNs solve that issue, as it is an end-to-end differentiable model that jointly optimizes the relationships between protein sequences and structure. However, because it uses RNNs as internal representations, training can be both difficult and time consuming (AlQuraishi (2019a)).

In this paper, we propose a modification to the RGN architecture. Inspired by the recent successes of the transformer models in the NLP community, we replace the LSTMs in the RGN model with the encoder portion of the Universal Transformer (UT) as the internal representation (Dehghani et al. (2019)). By doing so, the model is faster to train and it is contextually informed by all subsequent symbols. As a result, it is better at learning global dependencies among the amino acids than RNNs.

The UTGN operates by first taking a sequence of vector representation of the amino acids and applying the universal transformer architecture to iteratively refine a sequence of internal representations. Next, it uses the internal states to construct three torsional angles for each position, which is used to construct the 3D Cartesian structure (Figure 1).

*Worked performed at ShanghaiTech, School of Information Science and Technology.

Our experiments show that UTGN achieve an improvement of 1.7Å in RMSD and 0.013 in TM-Score for the free modeling portion of CASP12. In addition, the UTGN achieved an improvement of 0.7Å in RMSD and 0.008 in TM-Score for the template based modeling portion.

2 Model Description

2.1 Input Representation

We represent each amino acid in the protein sequence of size L as a 20 dimensional one-hot encoding. Next, we derive the Multiple Sequence Alignment (MSA) from JackHMMer and use it to calculate the $L \times 20$ Position-Specific Scoring Matrix (PSSM) (Potter et al. (2018)). Then, we normalize the PSSM values to between 0 and 1 and concatenate it with the one-hot encoding. After feeding this into a fully connected layer of dimension d, we add positional encodings as in Vaswani et al. (2017) as follows

$$PE_{(j,2i)} = \sin(\frac{j}{10000^{\frac{2i}{d}}}) \tag{1}$$

$$PE_{(j,2i+1)} = \cos(\frac{j}{10000^{\frac{2i}{d}}}) \tag{2}$$

where j is the position of the vector in the protein sequence and i is the index of that vector.

2.2 Universal Transformer

We use the multi-layer encoder portion of the Universal Transformer (UT) for internal representation. This neural architecture operates by recurring over representations of each of the positions of the input sequences. Unlike recurrent neural networks, which recur over positions in the sequence, UT recurs over revisions of the vector representations of each position (Dehghani et al. (2019)). In each time step, the representations are revised by passing through N layers, where each layer consists of a self-attention mechanism to exchange information across all positions in the sequence in the previous representation, followed by a transition function.

More specifically, given an input sequence of length L, we initialize a matrix $\boldsymbol{H}^0 \in \mathbb{R}^{L \times d}$. Each new representation $\boldsymbol{H}^t \in \mathbb{R}^{L \times d}$ at time step t is determined by first applying the multi-head dot-product self-attention (Vaswani et al. (2017)) mechanism. We compute the scaled dot-product attention using queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$, and values $\boldsymbol{V}$ as follows

$$\text{ATTENTION}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{SOFTMAX}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V} \tag{3}$$

For each head i, we map state $\boldsymbol{H}^t$ to queries, keys, and values using learned matrices $\boldsymbol{W}_i^Q \in \mathbb{R}^{d \times \frac{d}{k}}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d \times \frac{d}{k}}$, and $\boldsymbol{W}_i^V \in \mathbb{R}^{d \times \frac{d}{k}}$, where k is the number of heads (Vaswani et al. (2017)). Next, we apply the scaled-dot product attention to each head, concatenate them, and multiply the result by a learned matrix $\boldsymbol{W}^O \in \mathbb{R}^{d \times d}$.

$$\text{MULTIHEAD}(\boldsymbol{H}^t) = \text{CONCAT}(\text{head}_1, ..., \text{head}_\text{k})\boldsymbol{W}^O \tag{4}$$

$$\text{head}_i = \text{ATTENTION}(\boldsymbol{H}^t\boldsymbol{W}_i^Q, \boldsymbol{H}^t\boldsymbol{W}_i^K, \boldsymbol{H}^t\boldsymbol{W}_i^V) \tag{5}$$

We pass through the first layer of the encoder as follows

$$\boldsymbol{H}_1^t = \text{LAYER}_1(\boldsymbol{H}^{t-1}) \tag{6}$$

where

$$\text{LAYER}_p(\boldsymbol{H}) = \text{LAYERNORM}(\boldsymbol{A}^t + \text{TRANSITION}(\boldsymbol{A}^t)) \tag{7}$$

$$\boldsymbol{A}^t = \text{LAYERNORM}(\boldsymbol{H} + \text{MULTIHEAD}(\boldsymbol{H})) \tag{8}$$

where LAYERNORM is defined in Ba et al. (2016) and TRANSITION is either a one dimensional separable convolution (Chollet (2016)) or a fully-connected layer. In addition, each layer has different weights and p indicates the layer number. Between the multi-head attention and transition function, we incorporate both residual connections and dropout (Srivastava et al. (2014)).

For an encoder with N layers, we have

$$\boldsymbol{H}^t = \boldsymbol{H}^t_N = \text{LAYER}_N(\boldsymbol{H}^t_{N-1}) \tag{9}$$

In contrast to the original UT model, we do not add positional encodings on each time step; rather, the positional encoding is only added at the initial starting phase (see Figure 2 for a complete model).

2.3 Dynamic Halting

Because we wish to expend more computing resources on amino acids with ambiguous relevancy, we use the Adaptive Computation Time (ACT) to dynamically halt changes in certain representations (Graves (2016)). For each step and each symbol, if the scalar halting probability predicted by the model exceeds a threshold, the state representation is simply copied to the next time step. Recurrence continues until all representations are halted or the maximum number of steps are met.

2.4 Structure Construction

As in AlQuraishi (2019a), we use the final states for each position to construct the three torsional angles φ, ψ, ω. These angles represent the geometry of the protein spanned by the backbone atoms N, C^α, C'. Though bond lengths and angles also vary, their variation is limited enough that we can assume them to be fixed. We will also ignore the side chains of the protein, as our focus is on the backbone atoms. The resulting angles at each position is then translated into the 3D coordinates for the backbone.

More specifically, at position j, the corresponding angle triplet $\phi_j = (\psi_j, \varphi_j, \omega_j)$ is calculated as follows

$$\phi_j = arg(p_j exp(i\Phi)) \tag{10}$$

$$p_j = \text{SOFTMAX}(\boldsymbol{W}_\phi \boldsymbol{h}_j + \boldsymbol{b}_\phi) \tag{11}$$

where $\boldsymbol{h}_j$ is the j^{th} row of matrix $\boldsymbol{H}^T$, $\boldsymbol{W}_\phi$, $\boldsymbol{b}_\phi$, Φ are learned weights, and arg is the complex valued argument function. In addition, Φ defines an alphabet of size m whose letters correspond to triplets of torsional angles over the 3-torus. Next, recurrent geometric units convert the sequence of torsional angles $(\phi_1, ..., \phi_L)$ into 3D Cartesian coordinates as follows

$$\tilde{c_k} = r_{k \bmod 3} \begin{pmatrix} \cos(\theta_{k \bmod 3}) \\ \cos(\phi_{\lfloor \frac{k}{3} \rfloor, k \bmod 3}) sin(\theta_{k \bmod 3}) \\ \sin(\phi_{\lfloor \frac{k}{3} \rfloor, k \bmod 3}) sin(\theta_{k \bmod 3}) \end{pmatrix} \tag{12}$$

$$m_k = c_{k-1} - c_{k-2}$$
$$n_k = m_{k-1} \times \widehat{m}_k$$
$$M_k = (\widehat{m}_k, \widehat{n}_k \times \widehat{m}_k, \widehat{n}_k)$$
$$c_k = M_k \tilde{c_k} + c_{k-1}$$

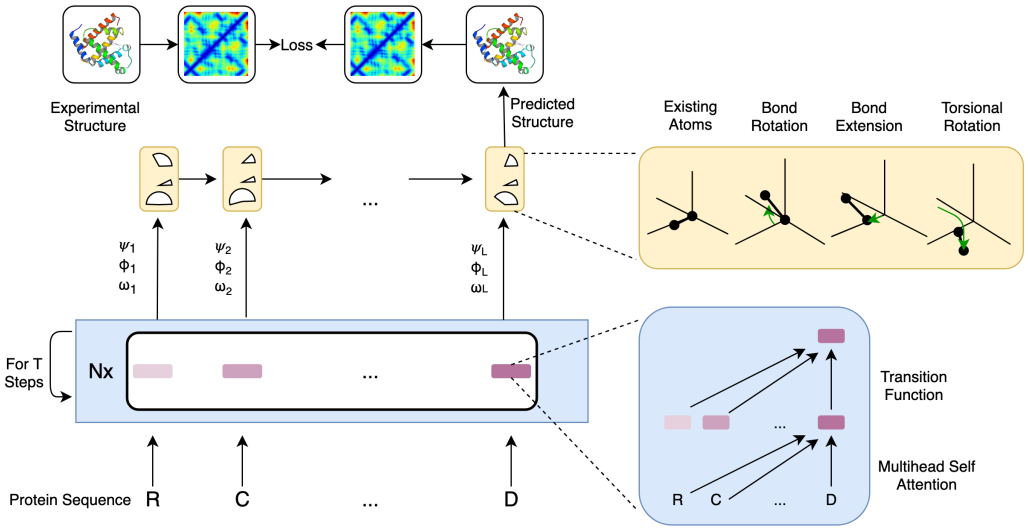

Figure 1: Vector representation of the amino acids are fed into the encoder of the UT. During each time step, the encoder applies a multi-head self attention mechanism and then a transition function to incorporate information from all previous states. In addition, the ACT mechanism decides which representation remain static. During each time step, the UT iteratively refines the internal representations of the inputs until a maximum time step is reached or until ACT mechanism halts every representation. Next, the UTGN constructs 3 torsional angle from each internal representation, which is used to create the 3D Cartesian coordinates for the protein structure. Then the loss function dRMSD is calculated by creating the distance map for the experimental structure and the predicted structure and calculating the distances between those two distance map. Back-propagation is used to optimize the weights of the architecture.

where r_k is the length of the bond connecting atoms $k-1$ and k, θ is the bond angle formed by atoms $k-2$, $k-1$, and k, $\phi_{\lfloor \frac{k}{3} \rfloor, k \bmod 3}$ is the predicted torsional angle formed by atoms $k-2$ and $k-1$, $\widehat{m}$ is the unit-normalized version of m, $\times$ is the cross product, and c_k is the position of the newly predicted atom k. The sequence $(c_1, ...c_{3L})$ form the final Cartesian coordinates of the protein backbone chain structure.

For training, the weights are optimized through the dRMSD loss function between the predicted and expected coordinates. This computes the pairwise distances between each atom in either the predicted or expected structure, and then finds the distance between those distances. More specifically,

$$\begin{aligned} \tilde{d_{j,k}} &= \|c_j - c_k\|_2 \\ d_{j,k} &= \tilde{d_{j,k}}^{(exp)} - \tilde{d_{j,k}}^{(pred)} \\ \text{dRMSD} &= \frac{\|\boldsymbol{D}\|_2}{L(L-1)} \end{aligned} \tag{13}$$

where $d_{j,k}$ are elements of matrix $\boldsymbol{D}$. We chose this loss function because it is differentiable and captures both local and global aspects of the protein structure.

3 Experiments and Analysis

3.1 Training Data and Batching

We evaluate our models with the CASP12 ProteinNet dataset with a thinning of 90%, which consists of around 50,000 structures (AlQuraishi (2019b)). The train and validation set contains all sequences and structures that exist prior to the CASP12 competition. The test set is the targets of CASP12, which consists of both the template-based modeling (TBM), intended to assess the prediction of

targets with structural homologs in the Protein Data Bank, and the free modeling (FM), intended to test a model's ability to predict novel structures (Moult et al. (1995)). In the train and validation set, entries with missing residues were annotated and are not included in the calculation of dRMSD. Sequences with similar lengths are batched together with a batch size of 32.

3.2 Model Parameters

The dimension of the feed forward layer that connected the input to the UT encoder was 256. We use 8 heads and 6 layers for the UT encoder architecture. The ACT threshold is 0.5 and the maximum number of ACT recurrence was 10. If a feed forward layer was used for the transition function (UTGN-FF), the feed forward dimension was 128. If a separable convolution is used instead (UTGN-SepConv), the kernel size is set to 3 and the stride was set to 1. In addition, we set the alphabet size to 60 for the angularization layer. The UTGN architecture amounts to about 2 million trainable parameters. For point of comparison, we train the RGN model with a size of 240, which is also around 2 million trainable parameters.

3.3 Optimizer

We used the ADAM optimizer with $\beta_1 = 0.95$, $\beta_2 = 0.99$, and learning rate of 0.001 (Kingma & Ba (2014)). In addition, the loss function for optimization was length normalized (dRMSD / protein length).

3.4 Regularization

We apply a dropout probability of 0.10 in the UT encoder architecture (Srivastava et al. (2014)). In addition, gradients are clipped using norm re-scaling with a threshold of 5.0 (Pascanu et al. (2012)). Furthermore, we perform early stopping when the validation loss failed to change noticeably in 10 epochs.

3.5 Analysis

We evaluate our model using two metrics: root mean squared deviation (RMSD) and Template Modelling (TM) Score (Zhang & Skolnick (2004)). RMSD is calculated by

$$\mathrm{RMSD}(\boldsymbol{v}, \boldsymbol{w}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|v_i - w_i\|^2} \tag{14}$$

where $\boldsymbol{v}$ and $\boldsymbol{w}$ are two sets of n points. This metric has the advantage that it does not require two structures to be globally aligned, and is able to detect regions of high agreement even if the global structure is not aligned. However, RMSD is very sensitive to protein length, leading to higher RMSD for longer proteins. The TM Score is calculated by

$$\mathrm{TMScore}(\boldsymbol{v}, \boldsymbol{w}) = \sum_i \frac{1}{1 + (\frac{D_i}{D_0})^2} \tag{15}$$

where $D_i = \|\boldsymbol{v} - \boldsymbol{w}\|$. TM scores are length normalized, and take values between 0 and 1, with higher values indicating better alignment. A TM score < 0.17 corresponds to random alignment whereas a TM score of > 0.5 correspond to the same protein fold (Xu & Zhang (2010)).

After evaluating our model, we found that the UTGN with separable convolution as the transition function performed better than the RGN model in the free modeling category by 9 percent in the RMSD metric and by 7 percent in the TM score metric (Table 2). For the template based modeling, the improvement was 4 percent for RMSD and 4 percent for TM score (Table 1).

From training both the RGN and UTGN model, we note that RGNs tend to suffer from heavily from exploding gradients, whereas the UTGN model never had that issue. In addition, the RGN model takes around 6 times longer for each epoch, and the UTGN model converged to its result about 2

times faster. Furthermore, we found that UTGNs have more stable initializations, whereas different initializations in RGNs can produce very different evaluation results.

Because RGNs and UTGNs must learn very deep neural networks from scratch and do not include any biophysical priors into the model, training a state-of-the-art model would require months of training and 10 times more parameters. Though we only train for a few days and with only 2 million parameters, we show that UTGNs have the potential to outperform RGNs.

Model	**dRMSD** (Å)	**TM score**
RGN	17.8	0.200
UTGN-FF	17.6	0.198
UTGN-SepConv	**17.1**	**0.208**

Table 1: The average dRMSD (lower is better) and TM score (higher is better) achieved by RGN and UTGN models in the TBM category for CASP12.

Model	**dRMSD** (Å)	**TM score**
RGN	19.8	0.181
UTGN-FF	19.4	0.174
UTGN-SepConv	**18.1**	**0.194**

Table 2: The average dRMSD (lower is better) and TM score (higher is better) achieved by RGN and UTGN models in the FM category for CASP12.

4 DISCUSSION

Before RGN was introduced, the protein prediction competition was dominated by complex models that fuse together multiple pipelines (Yang et al. (2014)). They tend to incorporate biological priors, like co-evolutionary information and secondary structure, that significantly improved their model performance. But the RGN model show to be a very competitive option without biological priors, outperforming the CASP11 model in the free modeling category (AlQuraishi (2019a)). Just like end to end differentiable models were able to replace complex pipelines in image recognition, we expect end-to-end differentiable architectures like UTGN to eventually replace the complex pipeline in protein structure prediction.

The biggest bottlenecks for training RGNs is time. The recurrent neural network portion is unstable, leading to gradient explosions. In addition, different initializations produce very different model performance, requiring researchers to try many different initializations. Furthermore, a fully refined model may take months to train (AlQuraishi (2019a)). As a result, it may take a significant amount of time to search for optimal parameters. UTGNs, however, are able to solve many of these problems, as replacing RNNs with transformers lead to a more stable and easily parallelizable model.

UTGNs are also better at learning global dependencies than RGNs. In RNNs, as the length of the path between two amino acids increase, information flow decreases. In contrast, the UT effectively has a global receptive field, as each new representation is contextually informed by all previous representations.

Some possible extensions for UTGN include using pre-trained embedding representations of amino acids like UniRep (Alley et al. (2019)). This can replace the need to calculate PSSMs for each new sequence, which further reduces the prediction time. In addition, instead of using static positional encodings, we could train relative position representations along with the transformer (Shaw et al. (2018)). Or we could incorporate more information in the input sequence, like secondary structure predictions.

5 CONCLUSION

This paper introduces UTGN, an end-to-end protein structure prediction architecture that uses a universal transformer as an internal representation. As opposed to the existing RGN model, UTGN is better at learning relationships of long range dependencies in the amino acids. In addition, the UTGN perform slightly better, converge much quicker, and is more stable to train. This progress shows that end-to-end differentiable protein prediction architectures can become competitive models in the protein folding problem.

The code for UTGN is available at: `https://github.com/JinLi711/3DProteinPrediction`

ACKNOWLEDGMENTS

We are grateful for Jie Zheng and Suwen Zhao for providing insightful guidance and commentary. We are also thankful for ShanghaiTech, School of Information Science and Technology for providing access to its computer cluster.

REFERENCES

Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M. Church. Unified rational protein engineering with sequence-only deep representation learning. 2019.

Mohammed AlQuraishi. End-to-end differentiable learning of protein structure. *Cell systems*, 8 4: 292–301.e3, 2019a.

Mohammed AlQuraishi. Proteinnet: a standardized data set for machine learning of protein structure. In *BMC Bioinformatics*, 2019b.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

François Chollet. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2016.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *ArXiv*, abs/1807.03819, 2019.

Alex Graves. Adaptive computation time for recurrent neural networks. *ArXiv*, abs/1603.08983, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Irwin D. Kuntz. Structure-based strategies for drug design and discovery. *Science*, 257 5073:1078–82, 1992.

Dominik Marx and Jurg Hutter. *Ab initio molecular dynamics: basic theory and advanced methods*. Cambridge University Press, 2010.

John Moult, Jesper Tejlgaard Pedersen, Richard S. Judson, and Krzysztof Fidelis. A large-scale experiment to assess protein structure prediction methods. *Proteins*, 23 3:ii–v, 1995.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *ArXiv*, abs/1211.5063, 2012.

Simon C. Potter, Aurelien Luciani, Sean R. Eddy, Youngmi Park, Rodrigo Lopez, and Robert D. Finn. Hmmer web server: 2018 update. In *Nucleic Acids Research*, 2018.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *NAACL-HLT*, 2018.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15: 1929–1958, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Sheng Wang, Siqi Sun, Zhen Li, Renyu Zhang, and Jinbo Xu. Accurate de novo prediction of protein contact map by ultra-deep learning model. *bioRxiv*, pp. 073239, 2016.

Jinrui Xu and Yang Zhang. How significant is a protein structure similarity with tm-score = 0.5? *Bioinformatics*, 26 7:889–95, 2010.

Jianyi Yang, Renxiang Yan, Ambrish Roy, Dong Lai Xu, Jonathan Poisson, and Yang Arthur Zhang. The i-tasser suite: protein structure and function prediction. *Nature Methods*, 12:7–8, 2014.

Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins*, 57 4:702–10, 2004.

APPENDICES

A UNIVERSAL TRANSFORMER ARCHITECTURE

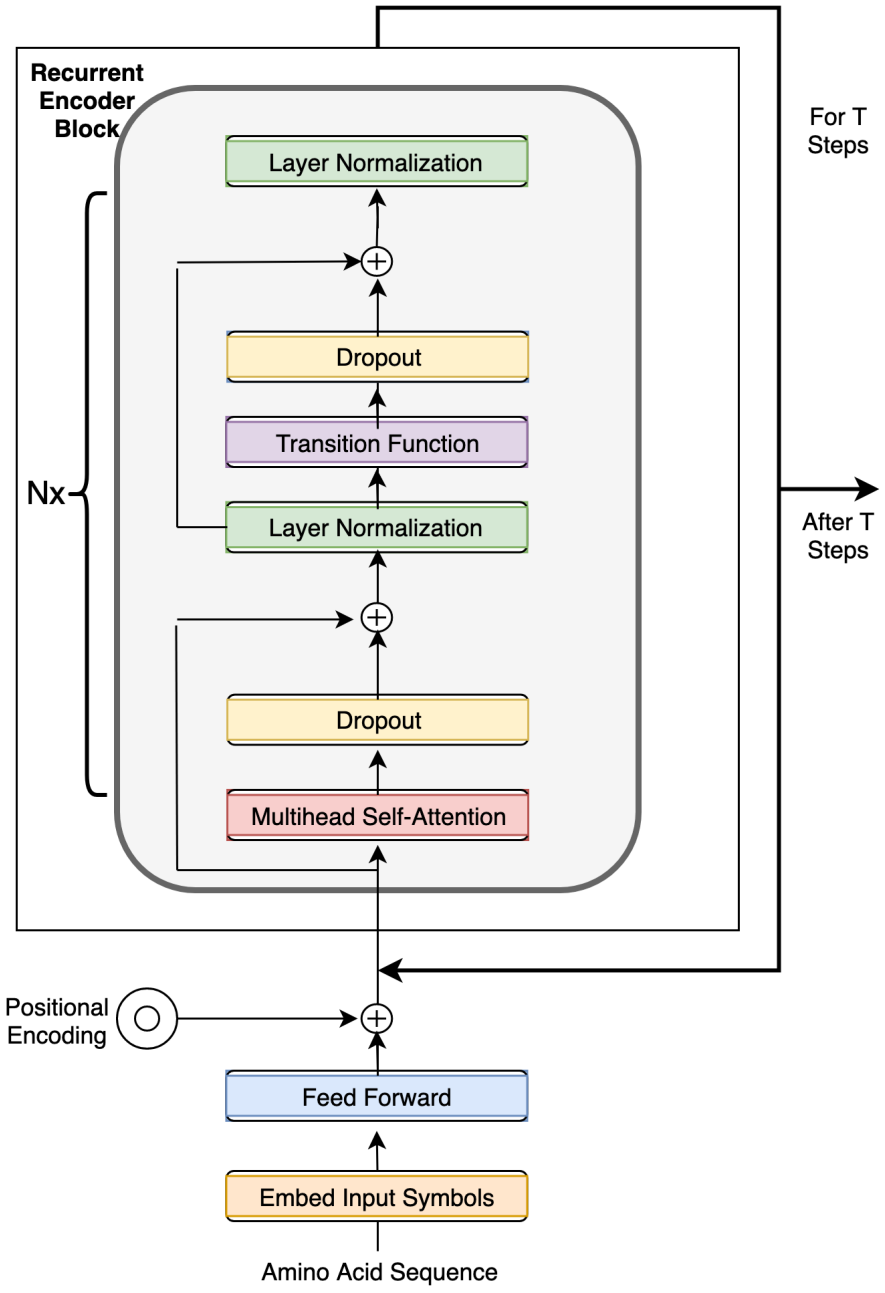

Figure 2: Complete architecture of the encoder portion of the universal transformer.

B CASP12 COMPARISON

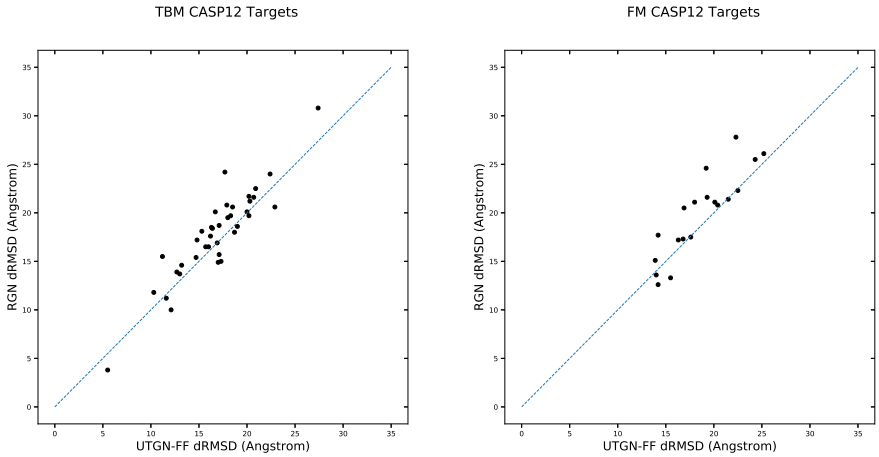

Figure 3: Scatter-plot comparing individual FM and TBM predictions of RGN and UTGN feed forward. Points above the blue line indicates better UTGN performance.

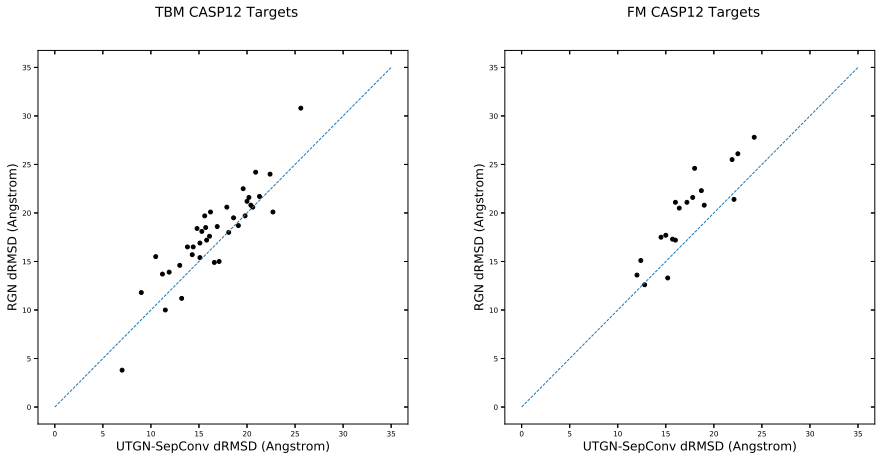

Figure 4: Scatter-plot comparing individual FM and TBM predictions of RGN and UTGN with separable convolution. Points above the blue line indicates better UTGN performance.

C CASP12 SAMPLES

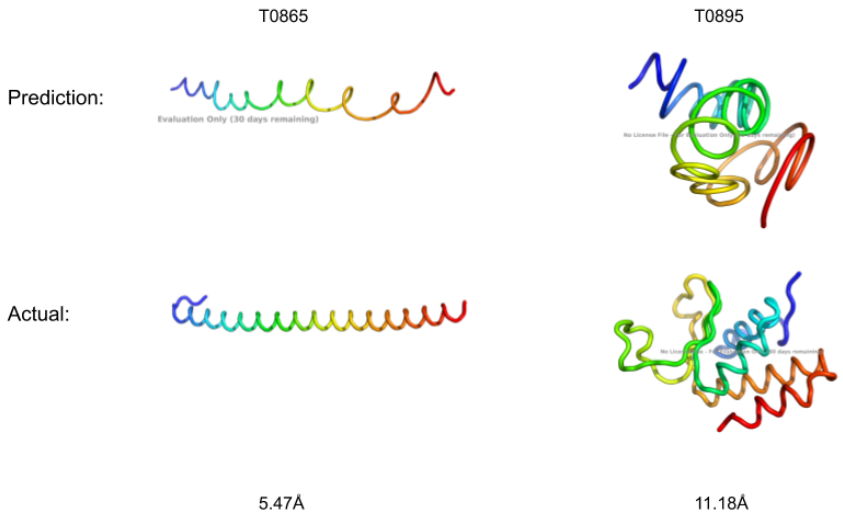

Figure 5: Comparison between predicted and actual structure of proteins T0865 (RMSD of 5.47Å) and T0895 (RMSD of 11.18Å) from CASP12 competition.