Outlines the ideas behind the Recurrent Geometric Network (RGN)

For machine learning protein prediction models, most of them take a protein target, search for homolog and return multiple sequence alignment (MSA). The MSAs come with statistics of pairwise correlation between residues a long family of protein, and then feed the information to neuron networks to get some geometrical features[1]. The RGN was first introduced in 2018 with the basic idea of sequence to structure model where one takes a sequence of amino acids and PSSMs as an input and builds up 3D structure as an output. Like conventional RNN, the computational unit keeps feeding back up on itself. In RNN a sequence of amino acids is fed into computational unit of RGN and it proceeds that sequence information from the input as well as from the prior state. The computation was built based on the idea of gauging submechanisms that controls information flow in and out of the internal state (between amino acids in a protein sequence). The prediction is made at every time step, not just a single prediction at the end. The topology of this network is bi-directional, going from the N-terminus to the C-terminus, across the protein in both directions. Each unit returns torsion angles (φ,Ψ,ω) which go to the geometric unit. The last stage computes variation between the experimental structure and the prediction using distance-based root mean square deviation(dRMSD)[2].

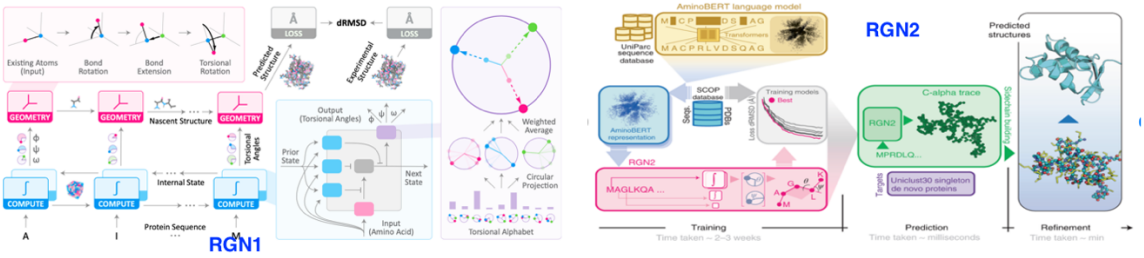
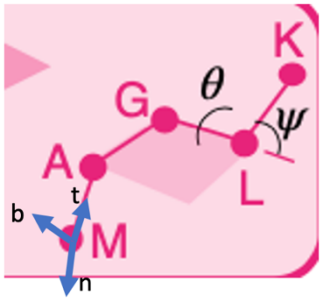


Figure 1 Overview structure of RGN1[2] *and RGN2* [3]

The idea of predicting protein structure from a single sequence has been developed as RGN2. Instead of utilizing evolution information, RGN2 models proteins using methods which are inspired by natural language processing, which can extract linguistic understanding from sentences. The models contain powerful neuron networks that can make sense of input with the mask (or the missing word) and then they can recover and fill in the missing words. In the end, the models acquire reasoning capabilities over sentences. In practice, the protein language model is fed in an amino acid sequence, and it can be treated like a sequence of words or tokens with in similar way. Some certain amino acids are masked and then fed into a transformer called AminoBERTH architecture. The task is to recover the identities of those amino acids and capture some internal representations. In the case of proteins, it is expected to capture structural understanding and higher-order representations. Individual masking amino acids can help the model capture local information, while masking contiguous segments can stimulate the model to learn some of those dependencies and capture global features. Making a use of a transformer turns out to be important for the language model and it is also relevant for protein prediction because the transformers can capture long-range dependencies of residues. Generally, residues that are far apart along the one-dimensional chain come close together in 3D space; therefore, the networks that are able to extract long-range dependencies are needed. AminoBerth is trained with around 260 million protein sequences; the training time can be estimated as 2-3 weeks. Frenet-Serret construction was introduced as a fast way to generate the backbone structure of the protein. The C-alpha trace can be considered effectively as a one-dimensional discrete curve in differential geometry, for a given curve in 3D, a reference frame for each point is assigned. In the case of proteins, each point refers to each C-alpha atom of each amino acid where the referent is a triplet of orthogonal vectors, the first vector is tangent (t) along the curve, and the other two are binormal (b) and normal vector (n), as shown in equation 1. They lie on a perpendicular plane to the tangent vector. Once the referent frame has moved along the curve, the geometry of the protein can be characterized by studying the kinetics of moving frame. There are two main parameters in this geometric studying, the first one is a torsion angle θ, which can be indicated by four consecutive atoms, the other is a curvature angle, which can be formed by three consecutive atoms. The motion of those frames of a long protein is fully governed

Figure 2 Backbone Geometry



by a rotation matrix, as shown in equation 2, so the rotation matrix is required in order to understand the kinetics of moving frame. Atomic positions can be reconstructed starting with a triplet vector that specifies the referent frame n, b , t and r which indicates the position of each atom along the backbone, and the algorithm is following a frame and position at residue (i) where R is a rotation matrix, as shown in equation 3. A frame of the following position at residue (i+1) and the position r were computed from matrix multiplication, and with this algorithm, the 3-D structure of proteins can be predicted.

$$t_i = \frac{r_{i+1} - r_i}{|r_{i+1} - r_i|} \quad b_i = \frac{t_{i-1} \times t_i}{|t_{i-1} \times t_i|} \quad n_i = b_i \times t_i \qquad \ldots\ldots\ldots\ldots (1)$$

$$\begin{pmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta & -\sin\psi \\ -\sin\theta & \cos\theta & 0 \\ \sin\psi\cos\theta & \sin\psi\sin\theta & \cos\psi \end{pmatrix} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$\begin{pmatrix} n_{i+1} \\ b_{i+1} \\ t_{i+1} \\ r_{i+1} \end{pmatrix} = \begin{pmatrix} & & & 0 \\ & \mathcal{R}_{i+1,i} & & 0 \\ & & & 0 \\ 0 & 0 & 3.8 & 1 \end{pmatrix} \begin{pmatrix} n_i \\ b_i \\ t_i \\ r_i \end{pmatrix} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Outline differences and similarities with approach taken by Alphafold2.

Instead of returning intermediate features such as a distogram, RNG has a deep learning pipeline where it can output a consistent 3D structure. The structure module of Alphafold2 is similar to RGN1(End to End prediction) in a way that it is returning as a consistent 3D structure in Euclidean space. The geometry modules in both RGN2 and Alpha fold 2 have similarities that introduced by AlphaFold2, where they reason over reference frames[4]. The transformer architecture was introduced by AlphaFold2, they use it in both reasoning MSAs and over structures[4]. It turns out that other available neuron architectures, such as the convolution network or the recurrent network are incapable of capturing long-range dependencies[5]. RGN takes advantage of the attention mechanism from AlphaFold2. The mechanism works based on tokens put their attention on any other token along the sequence, and learning interaction between non-neighbouring tokens, so the model extract important pairwise correlations[6].

On the other hand, the main difference is that AlphaFold2 requires raw MSAs as input, while RGN2 does not require homolog or any evolution information, so takes a single sequence as input. Another difference in the structure model is that Alphafold2 starts with Ramachandran construction while RGN performs Frenet-Serret frames.

Outline the claimed advantages of the RGN2 method (second article) with respect to Al- phaFold2.

There is a substantial drop in model accuracy, suggesting that the AlphaFold2 requires good MSAs, as shown in figure 3[4]. The AlphaFold2 does not appear to have a good prediction for protein that lack of evolutionary information or proteins that cannot generate MSAs. Figure 4 shows that RGN2 outperforms the AlphaFold2 47% of the cases on 77 orphan proteins[3]. So, RGN2 is very useful when there are no known sequence homologs, classes of designed proteins, and proteins with shallow MSAs.

As a single-sequence protein structure prediction, RGN2 can be able to avoid expensive MSAs; therefore, it performs much faster, with a prediction time in the order of milliseconds whereas the Alphafold2 prediction takes longer, as in table 1[3].

Figure 3 Effect of MSA depth.

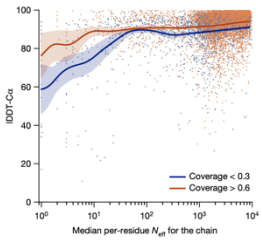


Figure 4 Predicting performance

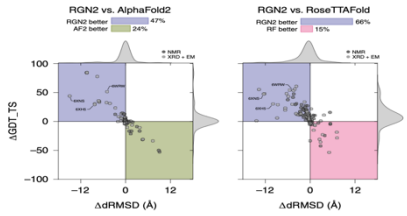


Table 1 Comparison of prediction times

(L) bins (no. residues)	targets	length (no. residues)	time per structure (s)		prediction time (s)	prediction time (s)	an RGN2 prediction time (ms)	Mean RGN2 prediction + refinement time (s)
			Distogram	3D structure				
$0 < L \leq 100$	184	37.5	1,768	1,004	831.5	412.6	2.7	132.99
$100 < L \leq 200$	93	148.7	2,791	1,927	851.6	408.3	2.2	139.89
$200 < L \leq 300$	28	258.3	2,877	1,752	828.4	492.7	3.1	154.34
$300 < L \leq 400$	17	333.4	3,647	2,140	825.6	501.6	5.7	179.52
$400 > L$	8	460.5	4,012	3,011	841.6	498.6	5.9	192.53

The protein quality check program can provide a functionality to automatically download a protein from PDB, and it can be able to find polypeptide regions in the structure file that are potentially low quality. The main criteria of this analysis include resolution of the X-ray data, refinement (the R-factor & R-free), B-factors (temperature factors), occupancy of atoms, and model geometry (Ramachandran plot). The implementation consists of **class** Quality:, **def __init__(self, structure, path):** as a constructure that takes input as structure and path.
def poly_detail(self):this function connects to polypeptide builder ;it shows polypeptide chains and their length[7].

```
Poly-peptide detials

Sequence: 1, Length: 36
SNDIYFNFQRFNETNLILQRDASVSSSGQLRLTNLN
Sequence: 2, Length: 196
NGEPRVGSLGRAFYSAPIQIWDNTTGTVASFATSFTFNIQVPNNAGPADGLAFALVPVGSQPKDKGGFLGLFDGSNSNFHTVAVEFDTLYNKDWDPTERHIGIDVNSIRSIKTTRWD
FVNGENAEVLITYDSSTNLLVASLVYPSQKTSFIVSDTVDLKSVLPEWVSVGFSATTGINKGNVETNDVLSWSFASKLS
Sequence: 3, Length: 233
SNDIYFNFQRFNETNLILQRDASVSSSGQLRLTNLNGNGEPRVGSLGRAFYSAPIQIWDNTTGTVASFATSFTFNIQVPNNAGPADGLAFALVPVGSQPKDKGGFLGLFDGSNSNFH
TVAVEFDTLYNKDWDPTERHIGIDVNSIRSIKTTRWDFVNGENAEVLITYDSSTNLLVASLVYPSQKTSFIVSDTVDLKSVLPEWVSVGFSATTGINKGNVETNDVLSWSFASKLS
Sequence: 4, Length: 36
SNDIYFNFQRFNETNLILQRDASVSSSGQLRLTNLN
Sequence: 5, Length: 196
NGEPRVGSLGRAFYSAPIQIWDNTTGTVASFATSFTFNIQVPNNAGPADGLAFALVPVGSQPKDKGGFLGLFDGSNSNFHTVAVEFDTLYNKDWDPTERHIGIDVNSIRSIKTTRWD
FVNGENAEVLITYDSSTNLLVASLVYPSQKTSFIVSDTVDLKSVLPEWVSVGFSATTGINKGNVETNDVLSWSFASKLS
Sequence: 6, Length: 35
SNDIYFNFQRFNETNLILQRDASVSSSGQLRLTNL
Sequence: 7, Length: 196
NGEPRVGSLGRAFYSAPIQIWDNTTGTVASFATSFTFNIQVPNNAGPADGLAFALVPVGSQPKDKGGFLGLFDGSNSNFHTVAVEFDTLYNKDWDPTERHIGIDVNSIRSIKTTRWD
FVNGENAEVLITYDSSTNLLVASLVYPSQKTSFIVSDTVDLKSVLPEWVSVGFSATTGINKGNVETNDVLSWSFASKLS
```

def get_resolution(self): this function gets a resolution of structure from structure.header["resolution"]
def get_r_factor(self): Because, there is no r factor in structer.header keys, so I have to convert mmcif in to dictionary build_dict = MMCIF2Dict(path)[8].
def get_data(self): when this function is called, it shows overall data including, Polychain_number, residue number, residue name, atoms, B-factor, occupancy, torsion angel, and chain id.
def cal_bad_b_factor_region(self): and **def cal_b_factor(self):** with criteria that B-factor is greater than 80.
def ocupency(self): it shows occupancy if any atom shows occupancy < 1.
def rachamandran_outliers(self): and **def rachamandran_outliers_region(self):** to get information of amino residues in Rachamandrans disallow region.
def low_quality_region(self): this function returns low quality region containing chain_id, poly peptide number, residue id, residue name and remark (what makes it potentially low quality) combining from bad b-factor, lower occupancy and disallow Rachamandran.

Chain_ID	Polychain_number	res_id	Residue	Remark
<Chain id=A>	2	(, 38,)	ASN	Bad B-factor region
<Chain id=A>	2	(, 113,)	ASN	Bad B-factor region
<Chain id=A>	2	(, 114,)	SER	Bad B-factor region
<Chain id=C>	5	(, 38,)	ASN	Bad B-factor region
<Chain id=C>	5	(, 114,)	SER	Bad B-factor region
<Chain id=A>	1	(, 20,)	ARG	Disallow Rachmandran
<Chain id=A>	2	(, 87,)	GLY	Disallow Rachmandran
<Chain id=A>	2	(, 103,)	GLY	Disallow Rachmandran
<Chain id=A>	2	(, 107,)	GLY	Disallow Rachmandran

```
Protien structure resolution:  2.8

R-free:  ['0.2300000']              Total Quality Score:  (19, 'Medium')
```

def give_score(self): this function returns a total quality score and classify quality of structure (High:5, MediumL:3 Low:1) from high =5, medium=3 low=1 so the best score = 25 worst score = 5

	High : 5	Medium: 3	Low:1
Resolution	< 2.2	2.2 to 2.8	> 2.8
R-free	< 0.22	0.22 to 0.4	> 0.4
B-factor	0%	< 20%	> 20%
Occupancy	0	show occupancy add 3	
Disallow Racha	< 10%	10 to 15%	> 15%
Total score	> 21	17 to 21	< 17

Resolution of x-ray: high resolution guarantees better structure (a lower number). The quality of data can be varying a lot depends on what kind of crystals, high resolution is at atomic level (2.4Å or lower) but I define it in this program at less than 2.2 Å, a medium resolution is good enough to see the chain; they recommend 2.5 to 3.5 Å [9] but I take 2.2 to 2.8 Å and low resolution is greater than 2.8 Å. **R-free** shows how well the model explains the experimental data; below 0.22 is high quality[10]. **B-factor** is a variance in atoms positions, for any residue that has a b-factor great than 80, it will be counted as having a bad b-factor region[11]. In this implement, if the **occupancy** less than one, a score of three is assigned. **Disallow Ramanchadrans region** (phi from 0 to 180 and psi from 0 to -180) is expected to be lower than 10% for high quality, between 10 and 15% for medium, and then low quality the value that is greater than 15% [12]. The total score is the sum of all quality parameters, the best score is at 25, above 21 can be graded as high quality, 17 to 21 for medium quality, and the score below 17 for low quality. The last part of this program is to rank the quality of structures, for example there are three protein structures with three differences in total score ("1FAT", 19), ("3LZT", 23), ("2WXW", 15). It ranks from high to low.

```
3LZT
└── 1FAT
    └── 2WXW
```

Reference

[1] R. Rao *et al.*, “MSA Transformer,” *bioRxiv*, p. 2021.02.12.430858, 2021, [Online]. Available: https://www.biorxiv.org/content/10.1101/2021.02.12.430858v3%0Ahttps://www.biorxiv.org/content/10.1101/2021.02.12.430858v3.abstract

[2] M. AlQuraishi, “End-to-End Differentiable Learning of Protein Structure,” *Cell Syst.*, vol. 8, no. 4, pp. 292-301.e3, 2019, doi: 10.1016/j.cels.2019.03.006.

[3] R. Chowdhury *et al.*, “Single-sequence protein structure prediction using a language model and deep learning,” *Nat. Biotechnol.*, vol. 40, no. 11, pp. 1617–1623, 2022, doi: 10.1038/s41587-022-01432-w.

[4] J. Jumper *et al.*, “Highly accurate protein structure prediction with AlphaFold,” *Nature*, vol. 596, no. 7873, pp. 583–589, 2021, doi: 10.1038/s41586-021-03819-2.

[5] A. M. Schaefer, S. Udluft, and H.-G. Zimmermann, “Learning long-term dependencies with recurrent neural networks,” *Neurocomputing*, vol. 71, no. 13, pp. 2481–2488, 2008, doi: https://doi.org/10.1016/j.neucom.2007.12.036.

[6] C. Corvalan, “AlphaFold 2: Attention Mechanism for Predicting 3D Protein Structures.” https://piip.co.kr/en/blog/AlphaFold2_Architecture_Improvements (accessed Jan. 21, 2023).

[7] T. Hamelryck, “Bio.PDB.Polypeptide,” 2002, [Online]. Available: https://biopython.org/docs/1.76/api/Bio.PDB.Polypeptide.html

[8] Thomas Hamelryck, “Bio.PDB.MMCIF2Dict module,” 2002. https://biopython.org/docs/1.76/api/Bio.PDB.MMCIF2Dict.html (accessed Jan. 20, 2023).

[9] “Quality Criteria for Structures: Experimental Parameters.” https://proteinstructures.com/experimental/structure-quality/ (accessed Jan. 20, 2023).

[10] D. Vitkup, D. Ringe, M. Karplus, and G. A. Petsko, “Why protein R-factors are so large: A self-consistent analysis,” *Proteins Struct. Funct. Genet.*, vol. 46, no. 4, pp. 345–354, 2002, doi: 10.1002/prot.10035.

[11] “PDB format and content.” https://proteinstructures.com/structure/protein-databank/ (accessed Jan. 23, 2023).

[12] “Ramachandran plot.” https://www.ebi.ac.uk/thornton-srv/software/PROCHECK/manual/examples/plot_01.html (accessed Jan. 21, 2023).