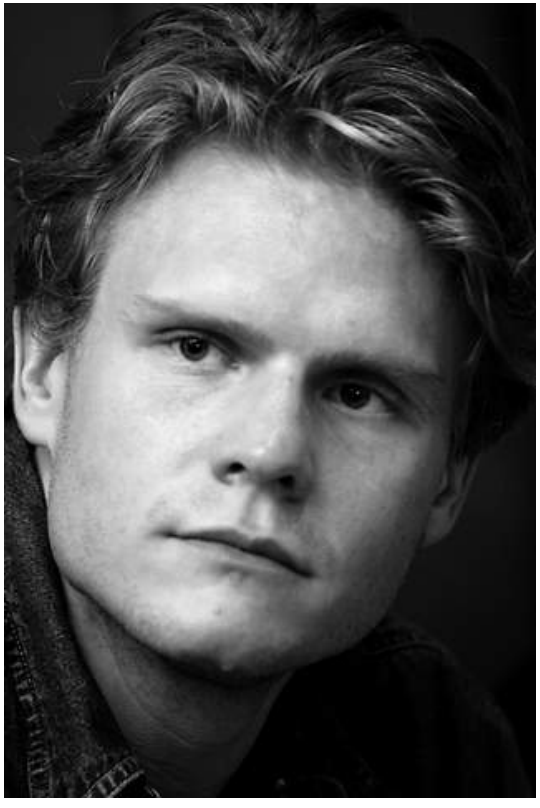
Highly accurate protein structure prediction with AlphaFold

AISC Healthcare Discussion Group

Willy Rempel

Saturday, July 24, 2021

Introduction



Willy Rempel

- HBSc Computer Science
- BSc Mathematics
- Research Associate, AISC
- seeking opportunities in the field

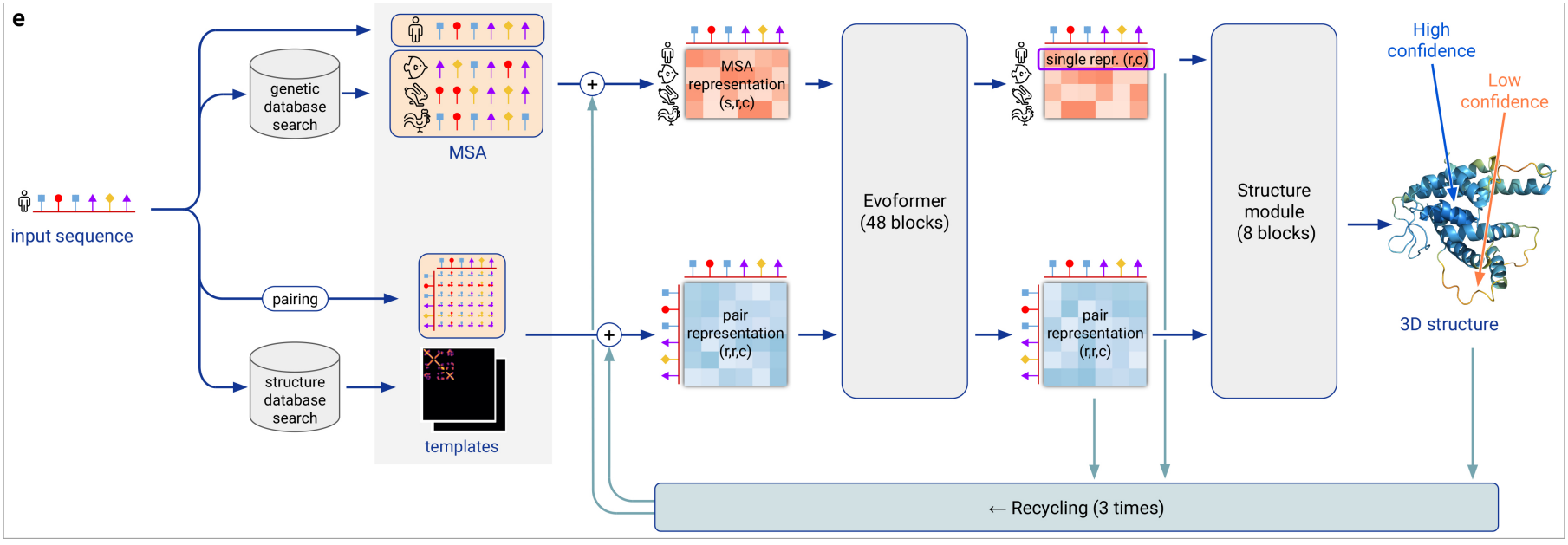
Introduction

Although all of the ideas in the model are doubtlessly clever, the main secret behind AlphaFold 2's success is the superb deep learning engineering. A close look at the model reveals an architecture with a large amount of small details that seem fundamental for the performance of the network. As we admire the end product, we should not turn a blind eye to the enormous budget, and the large team of full-time, handsomely paid engineers that made it possible. [3]

This, and many other tricks, are described in exhaustive detail in the Supplementary Information. A reduced subset has been analysed in a brief ablation study, but ultimately, how important are each of the minor details is anybody's guess. [3]

(above blog post is recommended reading)

Model Overview [1]



Initial Input: mmCIF or FASTA files

PDBx/mmCIF

Flexible, extensible, and verbose format with rich metadata, well suited for archival purposes (mmcif.wwpdb.org)

redundant annotations

inefficient representation

repetitive information

```
loop_
_atom_site.group_PDB
_atom_site.id
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_alt_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_entity_id
_atom_site.label_seq_id
_atom_site.pdbx_PDB_ins_code
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.Cartn_x_esd
_atom_site.Cartn_y_esd
_atom_site.Cartn_z_esd
_atom_site.occupancy_esd
_atom_site.B_iso_or_equiv_esd
_atom_site.pdbx_formal_charge
_atom_site.auth_seq_id
_atom_site.auth_comp_id
_atom_site.auth_asym_id
_atom_site.auth_atom_id
_atom_site.pdbx_PDB_model_num
ATOM   1    N  N   . TRP A 1 5   ? 8.519   -0.751  10.738  1.00 13.37 ? ? ? ? ? 5   TRP A N   1
ATOM   2    C  CA  . TRP A 1 5   ? 7.743   -1.668  11.585  1.00 13.42 ? ? ? ? ? 5   TRP A CA  1
ATOM   3    C  C   . TRP A 1 5   ? 6.786   -2.502  10.667  1.00 13.47 ? ? ? ? ? 5   TRP A C   1
ATOM   4    O  O   . TRP A 1 5   ? 6.422   -2.085  9.607   1.00 13.57 ? ? ? ? ? 5   TRP A O   1
ATOM   5    C  CB  . TRP A 1 5   ? 6.997   -0.917  12.645  1.00 13.34 ? ? ? ? ? 5   TRP A CB  1
ATOM   6    C  CG  . TRP A 1 5   ? 5.784   -0.209  12.221  1.00 13.40 ? ? ? ? ? 5   TRP A CG  1
ATOM   7    C  CD1 . TRP A 1 5   ? 5.681   1.084   11.797  1.00 13.29 ? ? ? ? ? 5   TRP A CD1 1
ATOM   8    C  CD2 . TRP A 1 5   ? 4.417   -0.667  12.221  1.00 13.34 ? ? ? ? ? 5   TRP A CD2 1
ATOM   9    N  NE1 . TRP A 1 5   ? 4.388   1.418   11.515  1.00 13.30 ? ? ? ? ? 5   TRP A NE1 1
ATOM   10   C  CE2 . TRP A 1 5   ? 3.588   0.375   11.797  1.00 13.35 ? ? ? ? ? 5   TRP A CE2 1
ATOM   11   C  CE3 . TRP A 1 5   ? 3.837   -1.877  12.645  1.00 13.39 ? ? ? ? ? 5   TRP A CE3 1
ATOM   12   C  CZ2 . TRP A 1 5   ? 2.216   0.208   11.656  1.00 13.39 ? ? ? ? ? 5   TRP A CZ2 1
ATOM   13   C  CZ3 . TRP A 1 5   ? 2.465   -2.043  12.504  1.00 13.33 ? ? ? ? ? 5   TRP A CZ3 1
ATOM   14   C  CH2 . TRP A 1 5   ? 1.654   -1.001  12.009  1.00 13.34 ? ? ? ? ? 5   TRP A CH2 1
```

Initial Input: mmCIF or FASTA files

```
;LCBO - Prolactin precursor - Bovine
; a sample sequence in FASTA format
MDSKGSSQKGSRLLLLLVVSNLLLCQGVVSTPVCPNGPGNCQVSLRDLFDRAVMVSHYIHDLSS
EMFNEFDKRYAQGKGFITMALNSCHTSSLPTPEDKEQAQQTHHEVLMSLILGLLRSWNDPLYHL
VTEVRGMKGAPDAILSRAIEIEEENKRLLEGMEMIFGQVIPGAKETEPYPVWSGLPSLQTKDED
ARYSAFYNLLHCLRRDSSKIDTYLKLLNCRIIYNNNC*

>MCHU - Calmodulin - Human, rabbit, bovine, rat, and chicken
MADQLTEEQIAEFKEAFSLFDKDGDGTITTKELGTVMRSLGQNPTEAELQDMINEVDADGNGTID
FPEFLTMMARKMKDTDSEEEIREAFRVFDKDGNGYISAAELRHVMTNLGEKLTDEEVDEMIREA
DIDGDGQVNYEEFVQMMTAK*

>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

Parsing [1]

- only certain metadata (more from mmCIF)
- change MSE residues into MET

Genetic Search [1]

For MSAs

- JackHMMER
  - MGnify: MSA depth 5,000
  - UniRef90: MSA depth 10,000
- HHBlits
  - Uniclust30 + BFD: MSA depth unlimited
- MSAs duplicated and stacked

flags: JackHMMER: -N 1 -E 0.0001 –incE 0.0001 –F1 0.0005 –F2 0.00005 –F3 0.0000005. HHBlits: -n 3 -e 0.001 -realign$_{\text{max}}$ 100000 -maxfilt 100000 -min$_{\text{prefilterhits}}$ 1000 -maxseq 1000000.

Template Search [1]

- UniRef90 MSA from prior search used for PDB70 search using HHSearch.
- Filter out:
  - released after the input sequence
  - or identical to the input sequence
  - too small
- At inference use top 4 templates

Training Data [1]

- 75:25 self-distillation : known structure (PDB)
- stochastic filters (next)

Filtering [1]

- stochastic filters:
  - Input mmCIFs are restricted to have resolution less than 9 Å. This is not a very restrictive filter and only removes around 0.2% of structures.
  - Longer protein chains are selected with higher probability.
  - Also favour protein chains from smaller clusters. They use 40% sequence identity clusters of the Protein Data Bank clustered with MMSeqs2.
  - Sequences are filtered out when any single amino acid accounts for more than 80% of the input primary sequence. This filter removes about 0.8% of sequences.

MSA block deletion [1]

- MSAs grouped by tool, sorted by block output? (e-value?)
  - similar sequences are likely to be adjacent
  - block deletion tends to remove similarities (ie. whole branch phylogeny)

MSA clustering [1]

- Similarity clusters used to randomly select subset of MSA sequences
  - to reduce computational cost from attention modules, reduce $N_s eq$
- K-means, input sequence used as first cluster center
- masking
- hamming distance measure for remaining selections

Residue cropping [1]

During training:

1. unclamped & clamped - sampling start index from uniform distributions
2. Cropped with fixed size N_res

Featurization and model inputs [1]

- $\text{target}_{\text{feat}}$ This is a feature of size [Nres, 21] consisting of the "aatype" feature.
- $\text{residue}_{\text{index}}$ This is a feature of size [Nres] consisting of the "$\text{residue}_{\text{index}}$" feature.
- msa_{feat} This is a feature of size [Nclust, Nres, 49] constructed by concatenating "$\text{cluster}_{\text{msa}}$", "$\text{cluster}_{\text{hasdeletion}}$", "$\text{cluster}_{\text{deletionvalue}}$", "$\text{cluster}_{\text{deletionmean}}$", "$\text{cluster}_{\text{profile}}$". We draw Ncycle$\times$Nensemble random samples from this feature to provide each recycling/ensembling iteration of the network with a different sample (see subsubsection 1.11.2).
- $\text{extra}_{\text{msafeat}}$ This is a feature of size [$\text{Nextra}_{\text{seq}}$, Nres, 25] constructed by concatenating "$\text{extra}_{\text{msa}}$", "$\text{extra}_{\text{msahasdeletion}}$", "$\text{extra}_{\text{msadeletionvalue}}$". Together with "$\text{msa}_{\text{feat}}$' above we also draw Ncycle $\times$ Nensemble random samples from this feature (see subsubsection 1.11.2).

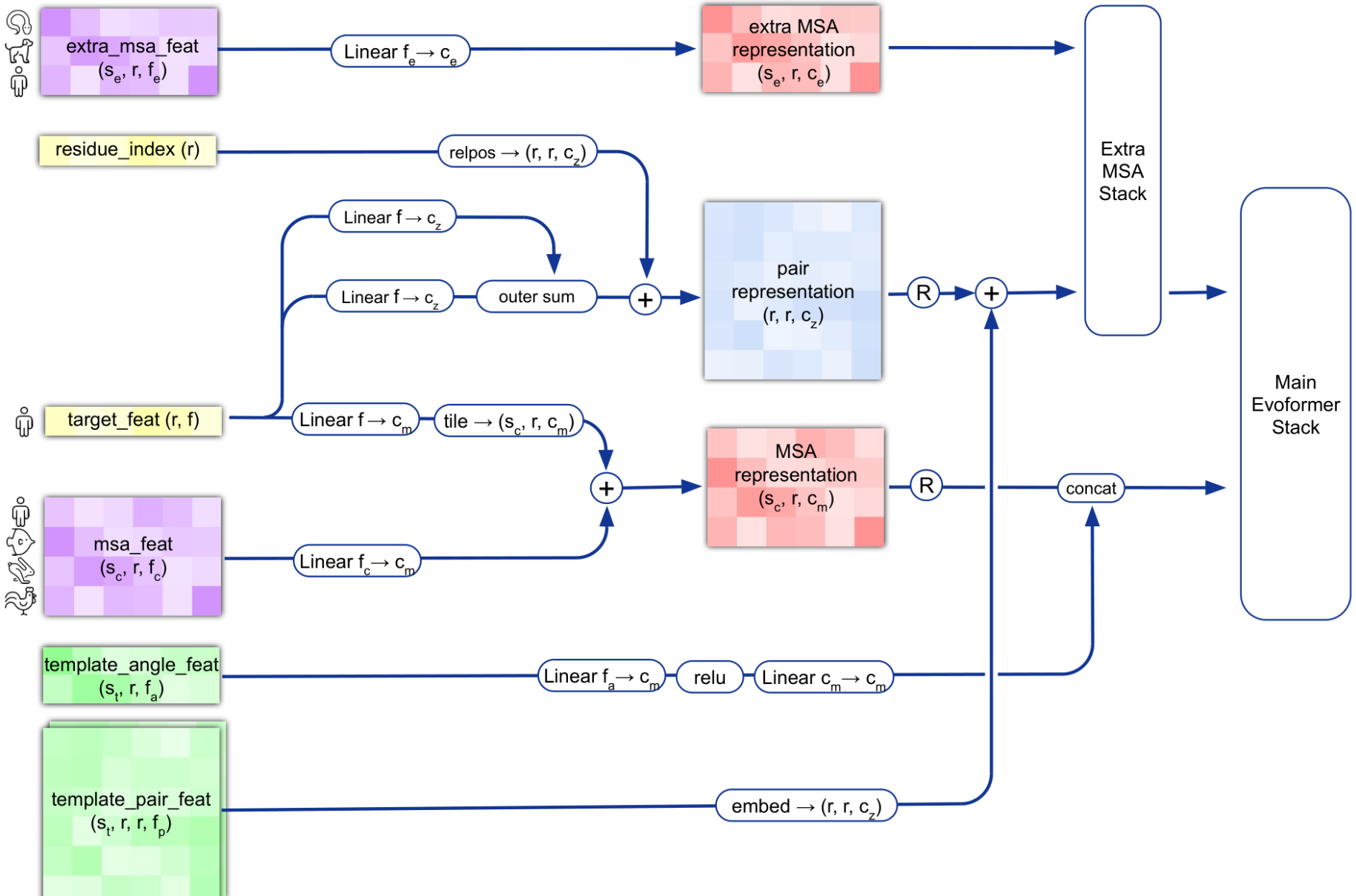
Featurization and model inputs [1]

- $\text{template}_{\text{pairfeat}}$ This is a feature of size [Ntempl, Nres, Nres, 88] and consists of concatenation of the pair residue features "$\text{template}_{\text{distogram}}$", "$\text{template}_{\text{unitvector}}$", and also several residue features, which are transformed into pair features. The "$\text{template}_{\text{aatype}}$" feature is included via tiling and stack- ing (this is done twice, in both residue directions). Also the mask features "$\text{template}_{\text{pseudobetamask}}$" and "$\text{template}_{\text{backboneframemask}}$" are included, where the feature fij = maski · maskj. - $\text{template}_{\text{anglefeat}}$ This is a feature of size [Ntempl, Nres, 51] constructed by concatenating the following features: "$\text{template}_{\text{aatype}}$", "$\text{template}_{\text{torsionangles}}$", "$\text{template}_{\text{alttorsionangles}}$", and "$\text{template}_{\text{torsionanglesmask}}$".

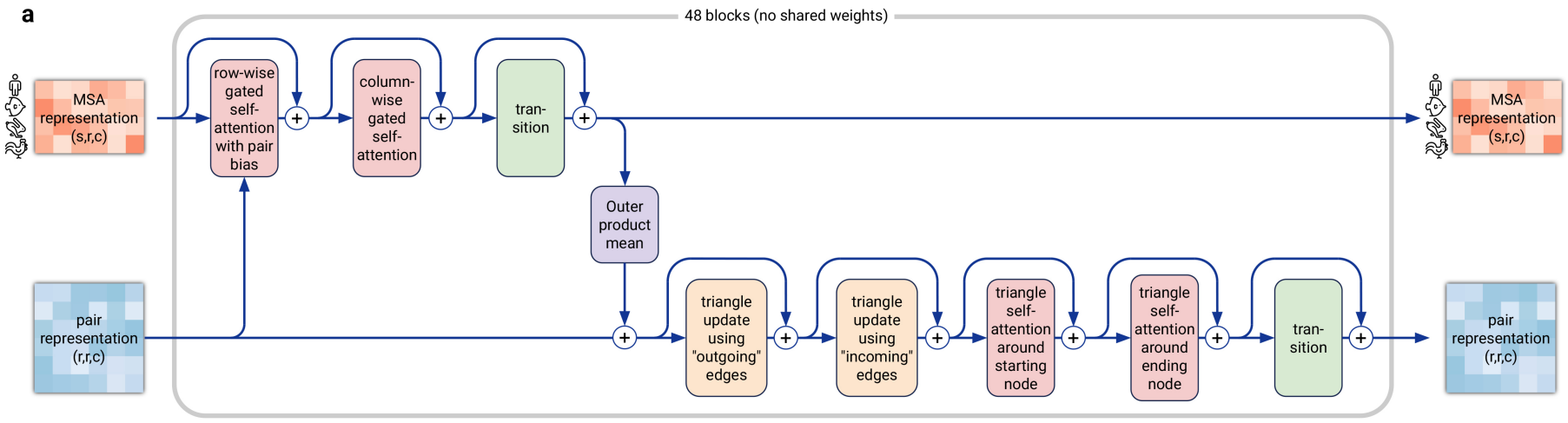
Self-distillation dataset [1]

- Build dataset (on unlabeled sequences):
  1. Make MSA for every cluster in Uniclust30
  2. Remove sequences that appear in another sequences MSA
  3. Keep sequences of $200 < \text{length} < 1024$
  4. Remove sequences where MSA < 200 alignments
- For predicted structures:
  - train 'undistlled' model on just PDB dataset
  - use this model to predict above set
  - for every residue pair, computer confidence metric using KL-divergence between distance distribution and a reference distribution
  - reference distribution
- self-distillation training took ~2 weeks

Input embeddings [1]



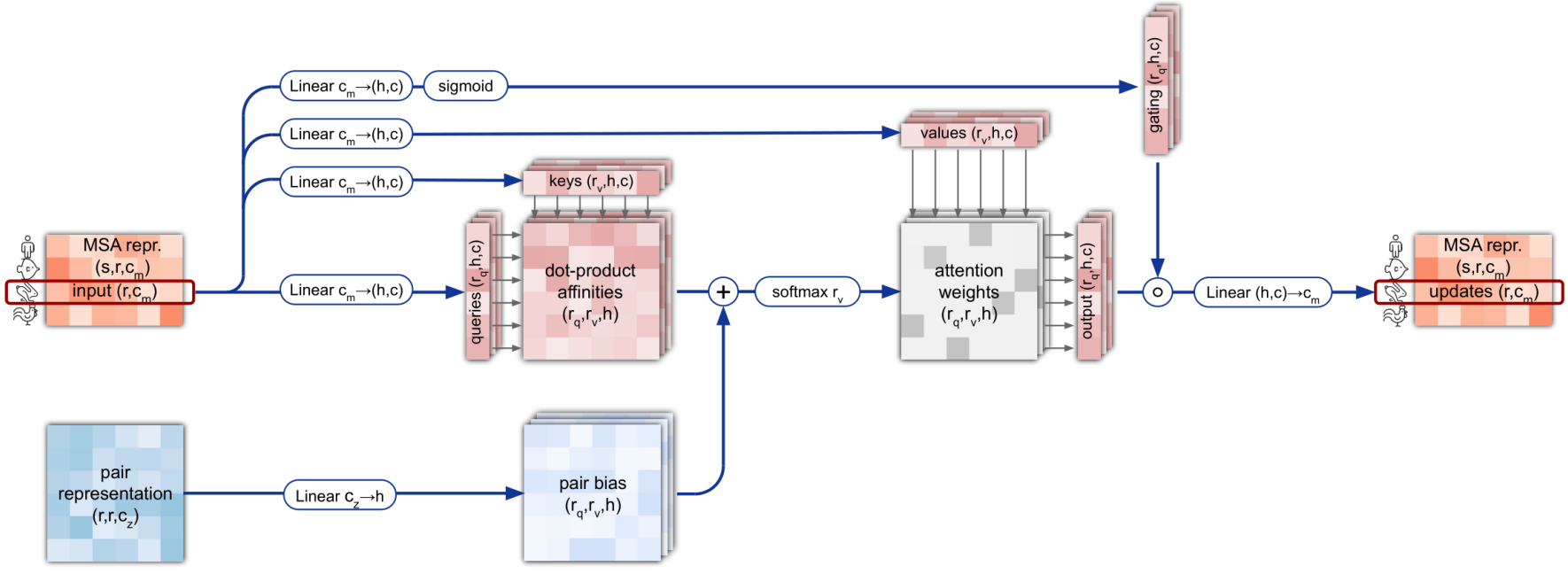
EvoFormer: Overview [1]



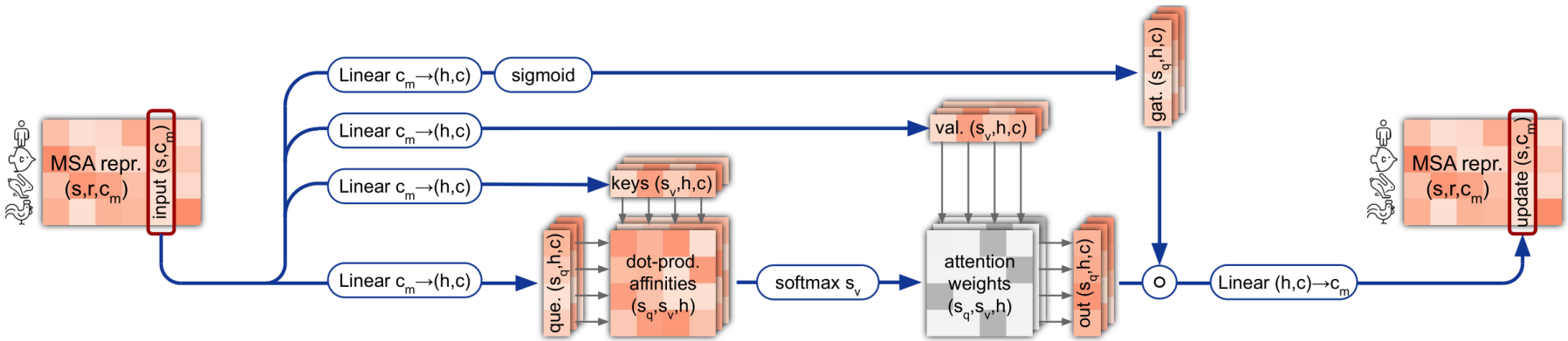
EvoFormer: Overview [1]

- cast as a graph inference problem
- cross-optimization and information flow between MSA representation and pair-wise representation
- layer normalization

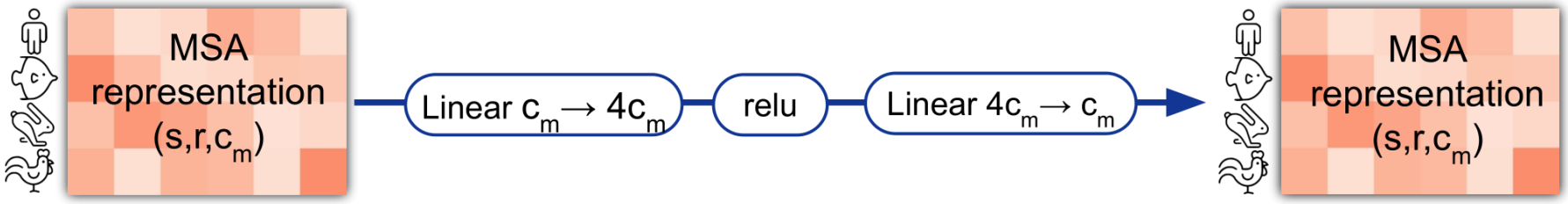
EvoFormer: Row wise Gated Attention [1]



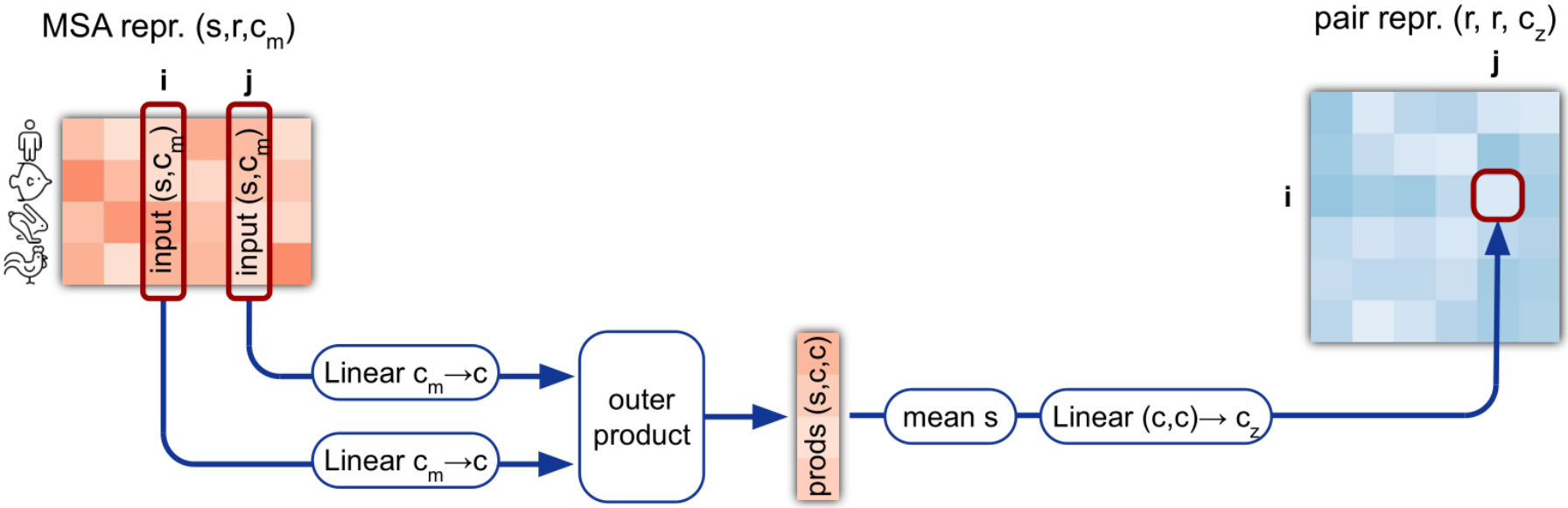
EvoFormer: Column wise Gated Attention [1]



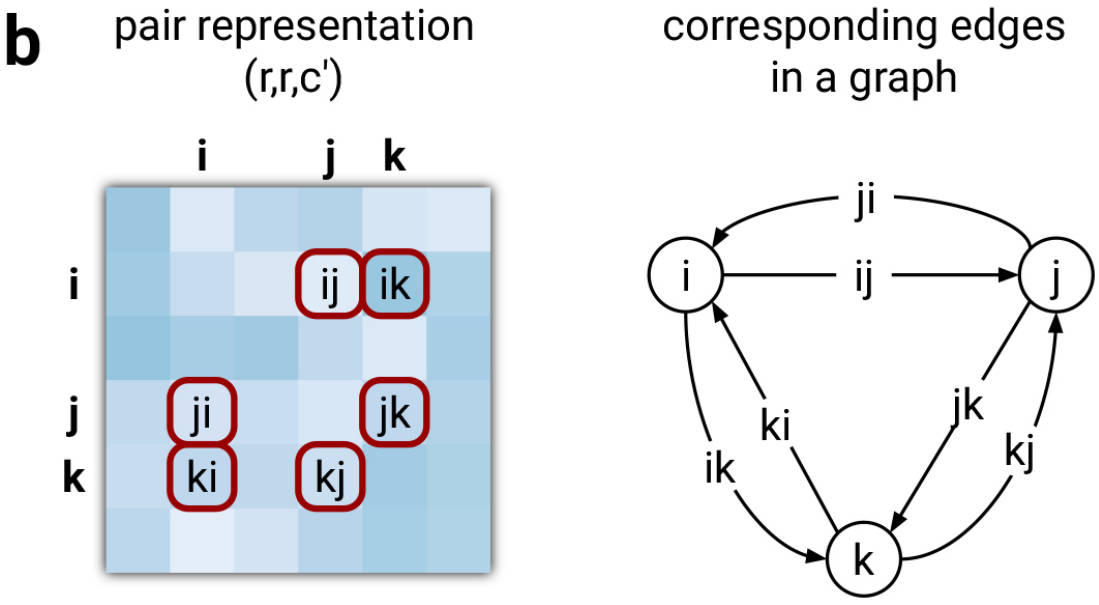
EvoFormer: MSA Translation Layer [1]



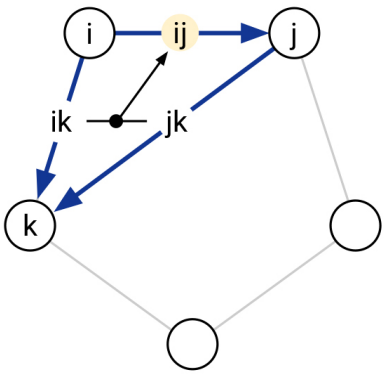
EvoFormer: Outer-Product Mean [1]



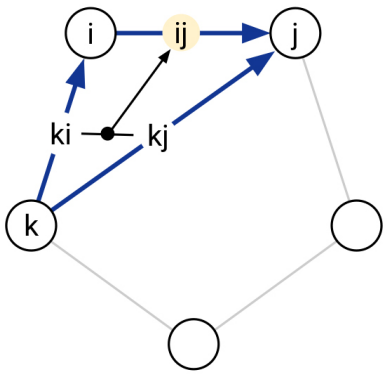
EvoFormer: Residue Pairs [1]



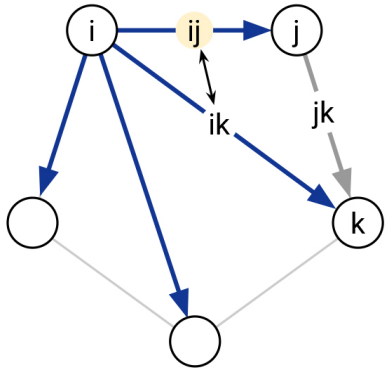
Triangle multiplicative update using "outgoing" edges



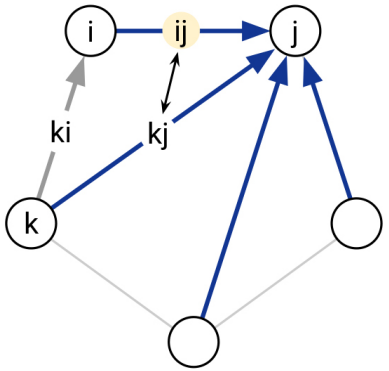
Triangle multiplicative update using "incoming" edges



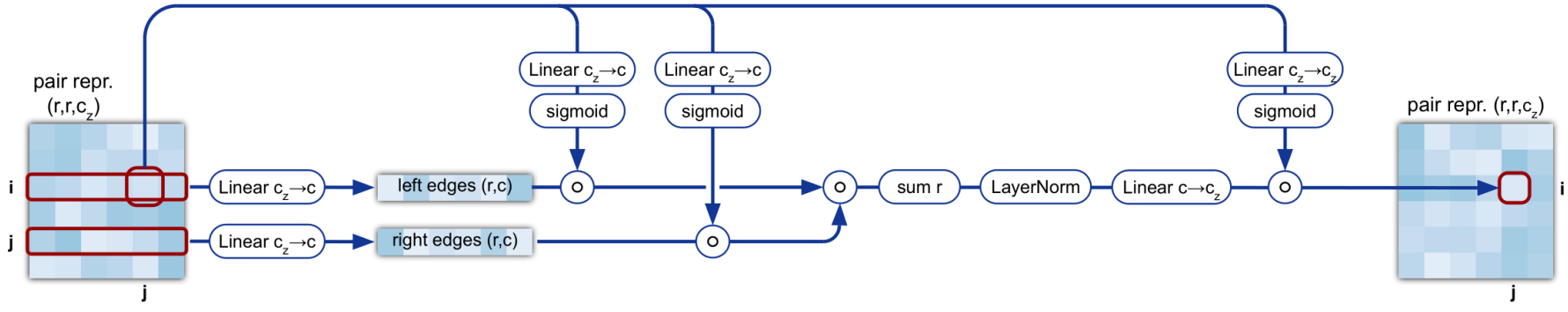
Triangle self-attention around starting node



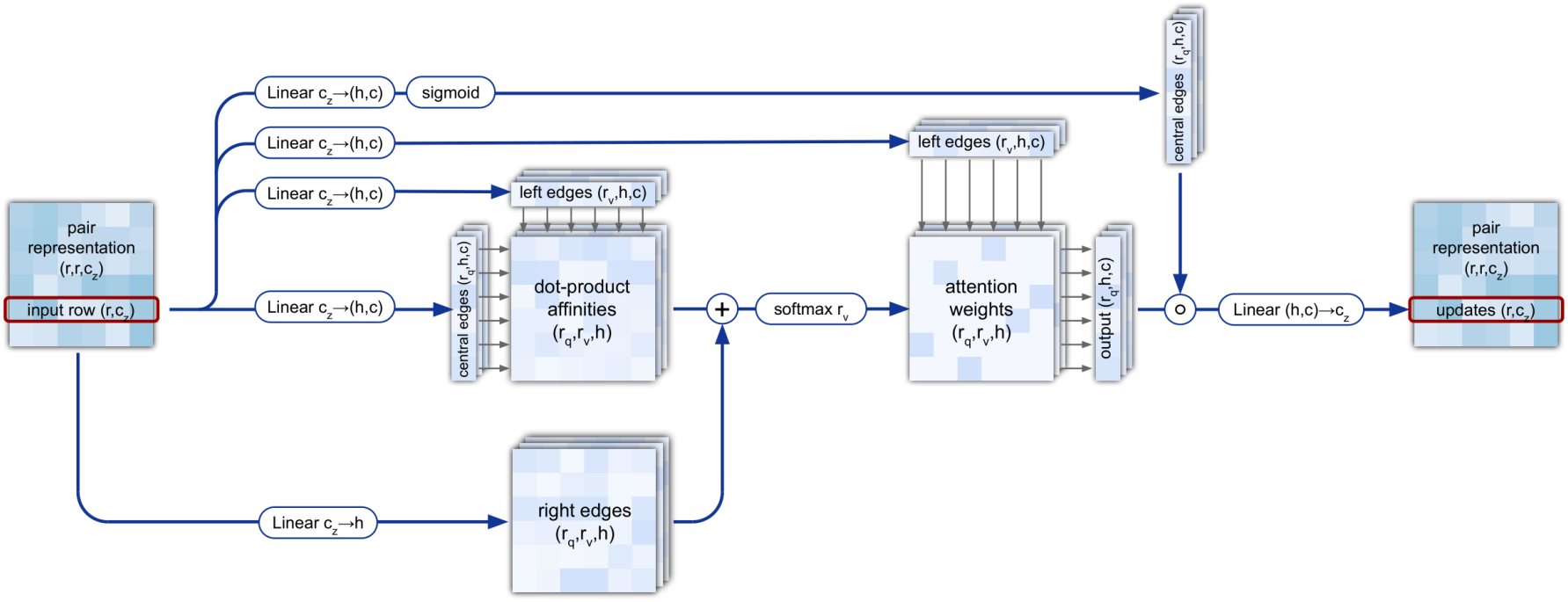
Triangle self-attention around ending node



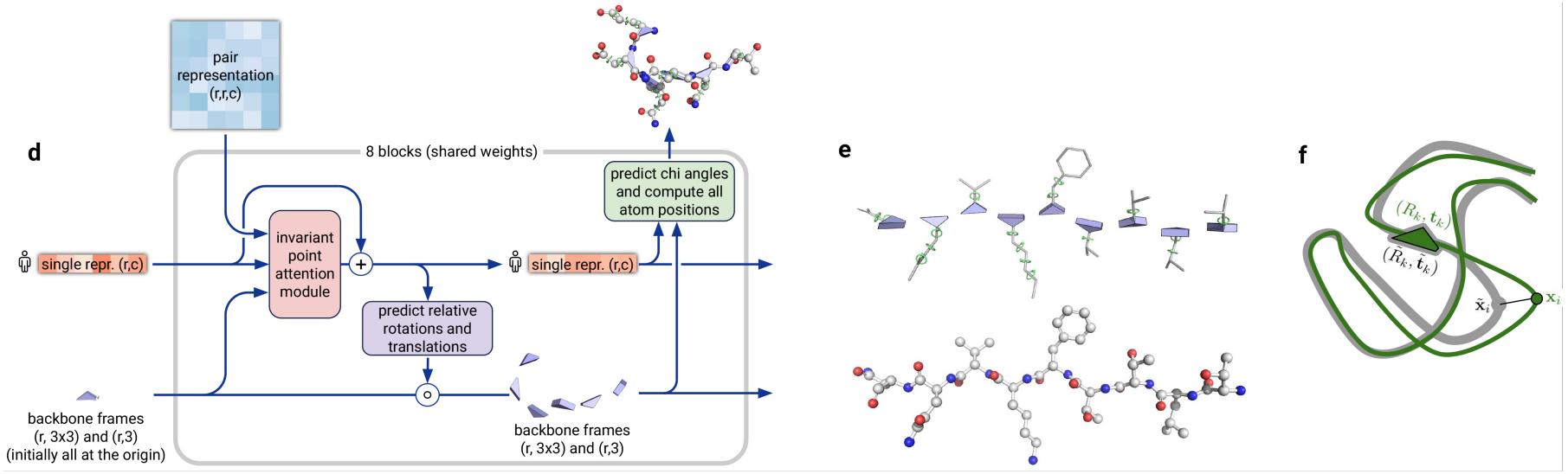
EvoFormer: Triangular Multiplicative Update [1]



EvoFormer: Triangular Self-Attention [1]



Structure Module: Overview [1]



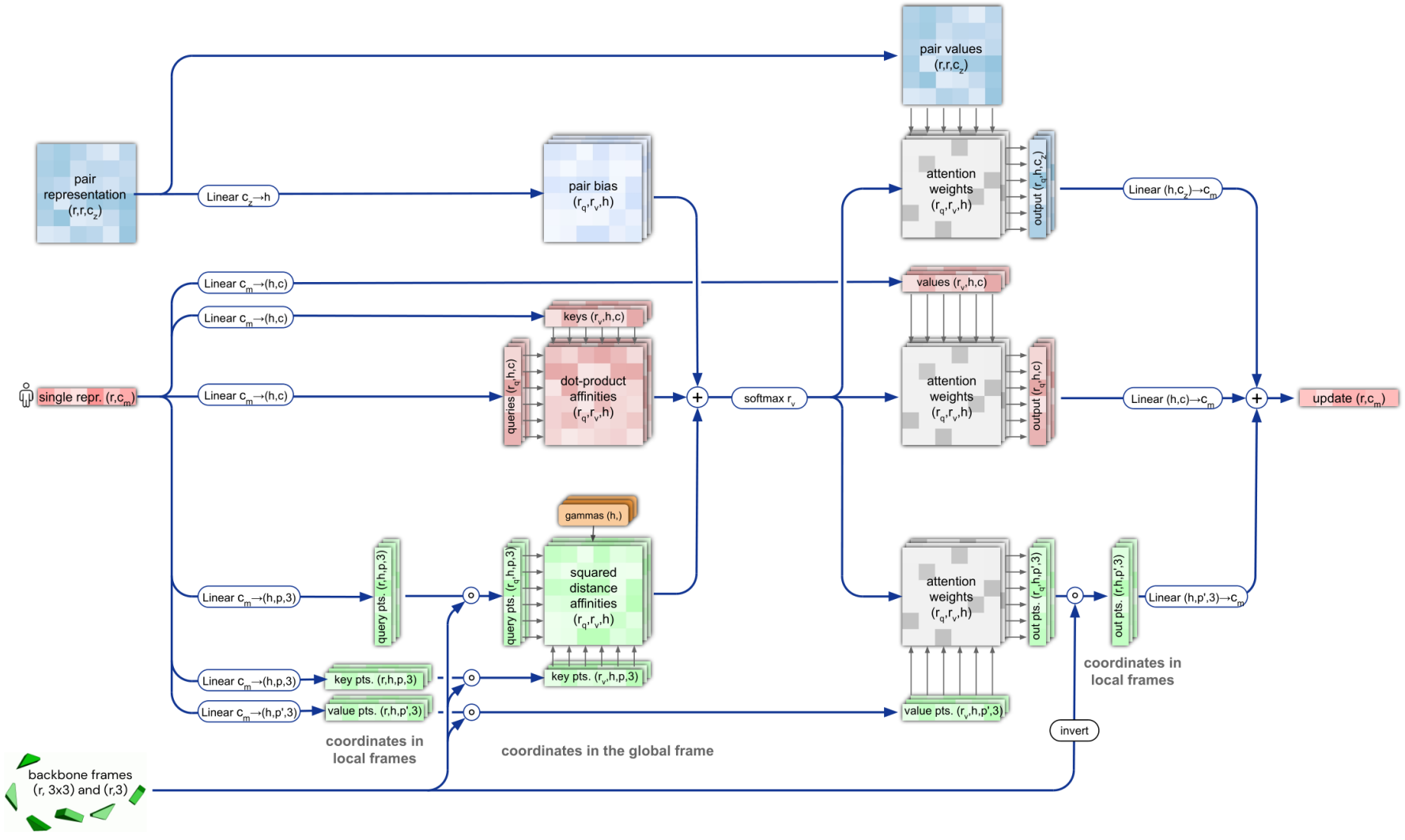
Structure Module: Frame Representation

$$\mathbf{M} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Example transform cite:SpatialTransformationMatrices

- rotation + translation transforms $T_i := (R_i, t_i)$
- no reflection, scaling, or shear

Structure Module: IPA [1]



Structure Module: Algorithm Part 1 [1]

Algorithm 20 Structure module

def StructureModule $\Big(\{\mathbf{s}_i^{\text{initial}}\}, \{\mathbf{z}_{ij}\}, N_{\text{layer}} = 8, c = 128,$ $\mathbf{s}_i^{\text{initial}} \in \mathbb{R}^{c_s}$

$\{T_i^{\text{true},f}\}, \{T_i^{\text{alt truth},f}\}, \{\vec{\alpha}_i^{\text{true},f}\}, \{\vec{\alpha}_i^{\text{alt truth},f}\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\}, \{\vec{\mathbf{x}}_i^{\text{alt truth},a}\}\Big)$:

1: $\mathbf{s}_i^{\text{initial}} \leftarrow \text{LayerNorm}(\mathbf{s}_i^{\text{initial}})$

2: $\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$

3: $\mathbf{s}_i = \text{Linear}(\mathbf{s}_i^{\text{initial}})$ $\mathbf{s}_i \in \mathbb{R}^{c_s}$

4: $T_i = (\mathbf{I}, \vec{\mathbf{0}})$ $\mathbf{I} \in \mathbb{R}^{3\times3}, \vec{\mathbf{0}} \in \mathbb{R}^3$

Structure Module: Algorithm Part 2 [1]

5: **for all** $l \in [1, \ldots, N_{\text{layer}}]$ **do** *# shared weights*

6: $\{\mathbf{s}_i\}$ += InvariantPointAttention$(\{\mathbf{s}_i\}, \{\mathbf{z}_{ij}\}, \{T_i\})$

7: $\mathbf{s}_i \leftarrow \text{LayerNorm}(\text{Dropout}_{0.1}(\mathbf{s}_i))$

 # Transition.

8: $\mathbf{s}_i \leftarrow \mathbf{s}_i + \text{Linear}(\text{relu}(\text{Linear}(\text{relu}(\text{Linear}(\mathbf{s}_i)))))$ all intermediate activations $\in \mathbb{R}^{c_s}$

9: $\mathbf{s}_i \leftarrow \text{LayerNorm}(\text{Dropout}_{0.1}(\mathbf{s}_i))$

 # Update backbone.

10: $T_i \leftarrow T_i \circ \text{BackboneUpdate}(\mathbf{s}_i)$

 # Predict side chain and backbone torsion angles $\omega, \phi, \psi, \chi_1, \chi_2, \chi_3, \chi_4$

11: $\mathbf{a}_i = \text{Linear}(\mathbf{s}_i) + \text{Linear}(\mathbf{s}_i^{\text{initial}})$ $\mathbf{a}_i \in \mathbb{R}^c$

12: $\mathbf{a}_i \leftarrow \mathbf{a}_i + \text{Linear}(\text{relu}(\text{Linear}(\text{relu}(\mathbf{a}_i))))$ all intermediate activations $\in \mathbb{R}^c$

13: $\mathbf{a}_i \leftarrow \mathbf{a}_i + \text{Linear}(\text{relu}(\text{Linear}(\text{relu}(\mathbf{a}_i))))$ all intermediate activations $\in \mathbb{R}^c$

14: $\vec{\alpha}_i^f = \text{Linear}(\text{relu}(\mathbf{a}_i))$ $\vec{\alpha}_i^f \in \mathbb{R}^2, f \in \mathcal{S}_{\text{torsion names}}$

 # Auxiliary losses in every iteration.

15: $(R_i, \vec{\mathbf{t}}_i) = T_i$

16: $\vec{\mathbf{x}}_i^{\text{C}\alpha} = \vec{\mathbf{t}}_i$

17: $\mathcal{L}_{\text{aux}}^l = \Big(\text{computeFAPE}(\{T_i\}, \{\vec{\mathbf{x}}_i^{\text{C}\alpha}\}, \{T_i^{\text{true}}\}, \{\vec{\mathbf{x}}_i^{\text{true,C}\alpha}\}, \epsilon = 10^{-12})$

18: $\quad + \text{torsionAngleLoss}(\{\vec{\alpha}_i^f\}, \{\vec{\alpha}_i^{\text{true},f}\}, \{\vec{\alpha}_i^{\text{alt truth},f}\}) \Big)$

Structure Module: Algorithm Part 3 [1]

23: $\mathcal{L}_{\text{aux}} = \text{mean}_l(\{\mathcal{L}^l_{\text{aux}}\})$

24: $T^f_i, \vec{\mathbf{x}}^a_i = \text{computeAllAtomCoordinates}(T_i, \vec{\alpha}^f_i)$ $\quad a \in \mathcal{S}_{\text{atom names}}$

25: $T^f_i \leftarrow \text{concat}(T_i, T^f_i)$

 # Final loss on all atom coordinates.

26: $\{T^{\text{true},f}_i\}, \{\vec{\mathbf{x}}^{\text{true},a}_i\} \leftarrow \text{renameSymmetricGroundTruthAtoms}($

27: $\{T^f_i\}, \{\vec{\mathbf{x}}^a_i\}, \{T^{\text{true},f}_i\}, \{T^{\text{alt truth},f}_i\}, \{\vec{\mathbf{x}}^{\text{true},a}_i\}, \{\vec{\mathbf{x}}^{\text{alt truth},a}_i\})$

28: $\mathcal{L}_{\text{FAPE}} = \text{computeFAPE}(\{T^f_i\}, \{\vec{\mathbf{x}}^a_i\}, \{T^{\text{true},f}_i\}, \{\vec{\mathbf{x}}^{\text{true},a}_i\}, \epsilon = 10^{-4})$

 # Predict model confidence.

29: $\{r^{\text{true LDDT}}_i\} = \text{perResidueLDDT\_Ca}(\{\vec{\mathbf{x}}^a_i\}, \{\vec{\mathbf{x}}^{\text{true},a}_i\})$

30: $\{r^{\text{pLDDT}}_i\}, \mathcal{L}_{\text{conf}} = \text{predictPerResidueLDDT\_Ca}(\{\mathbf{s}_i\}, \{r^{\text{true LDDT}}_i\})$

31: **return** $\{\vec{\mathbf{x}}^a_i\}, \{r^{\text{pLDDT}}_i\}, \mathcal{L}_{\text{FAPE}}, \mathcal{L}_{\text{conf}}, \mathcal{L}_{\text{aux}}$

Loss Functions [1]

$$\mathcal{L} = \begin{cases} 0.5\mathcal{L}_{\text{FAPE}} + 0.5\mathcal{L}_{\text{aux}} + 0.3\mathcal{L}_{\text{dist}} + 2.0\mathcal{L}_{\text{msa}} + 0.01\mathcal{L}_{\text{conf}} & \text{training} \\ 0.5\mathcal{L}_{\text{FAPE}} + 0.5\mathcal{L}_{\text{aux}} + 0.3\mathcal{L}_{\text{dist}} + 2.0\mathcal{L}_{\text{msa}} + 0.01\mathcal{L}_{\text{conf}} + 0.01\mathcal{L}_{\text{exp resolved}} + 1.0\mathcal{L}_{\text{viol}} & \text{fine-tuning} \end{cases}, \tag{7}$$

- weighted sum
- weighted to reduce importance of short sequences

Loss Functions & Auxillary Heads [1]

1. Side chain and backbone torsion angle loss
2. Frame aligned point error (FAPE)
   - Configurations with FAPE(X,Y) = 0
   - Metric properties of FAPE
3. Chiral properties of AlphaFold and its loss
4. Model confidence prediction (pLDDT)
5. TM-score prediction
6. Distogram prediction
7. Masked MSA prediction
8. "Experimentally resolved" prediction
9. Structural violations

Loss Functions: FAPE

Algorithm 28 Compute the Frame aligned point error

def computeFAPE($\{T_i\}, \{\vec{\mathbf{x}}_j\}, \{T_i^{\text{true}}\}, \{\vec{\mathbf{x}}_j^{\text{true}}\}, Z = 10\text{Å}, d_{\text{clamp}} = 10\text{Å}, \epsilon = 10^{-4}\text{Å}^2$) :

$T_i, T_i^{\text{true}} \in (\mathbb{R}^{3\times3}, \mathbb{R}^3)$
$\vec{\mathbf{x}}_j, \vec{\mathbf{x}}_j^{\text{true}} \in \mathbb{R}^3,$
$i \in \{1, ..., N_{\text{frames}}\}, j \in \{1, ..., N_{\text{atoms}}\}$

1: $\vec{\mathbf{x}}_{ij} = T_i^{-1} \circ \vec{\mathbf{x}}_j$ $\quad \vec{\mathbf{x}}_{ij} \in \mathbb{R}^3$

2: $\vec{\mathbf{x}}_{ij}^{\text{true}} = T_i^{\text{true}\,-1} \circ \vec{\mathbf{x}}_j^{\text{true}}$ $\quad \vec{\mathbf{x}}_{ij}^{\text{true}} \in \mathbb{R}^3$

3: $d_{ij} = \sqrt{\|\vec{\mathbf{x}}_{ij} - \vec{\mathbf{x}}_{ij}^{\text{true}}\|^2 + \epsilon}$ $\quad d_{ij} \in \mathbb{R}$

4: $\mathcal{L}_{\text{FAPE}} = \frac{1}{Z}\,\text{mean}_{i,j}(\text{minimum}(d_{\text{clamp}}, d_{ij}))$

5: **return** $\mathcal{L}_{\text{FAPE}}$

[1]

- Variation of commonly used root-mean-squared deviation (RMSD) of atomic positions
- not invariant to reflections, preventing proteins of the wrong chirality. [3], [1]

AlphaFold Inference [1]

- AlphaFold receives input features derived from:
  - the amino-acid sequence
  - MSA
  - templates (see subsubsection 1.2.9)
- outputs features:
  - atom coordinates
  - the distogram
  - per-residue confidence scores.
- Recycling x3
  - initial recycled inputs are zero

Algorithm 2 outlines the main steps (see also Fig 1e and the corresponding description in the main article).

AlphaFold Training [1]

Table 4 | AlphaFold training protocol. We train each stage until convergence with the approximate timings and number of samples provided.

Model	**Initial training**	**Fine-tuning**
Number of templates N_{templ}	4	4
Sequence crop size N_{res}	256	384
Number of sequences N_{seq}	128	512
Number of extra sequences $N_{\text{extra\_seq}}$	1024	5120
Parameters initialized from	Random	Initial training
Initial learning rate	10^{-3}	$5 \cdot 10^{-4}$
Learning rate linear warm-up samples	128000	0
Structural violation loss weight	0.0	1.0
Training samples ($\cdot 10^6$)	≈ 10	≈ 1.5
Training time	≈ 7 days	≈ 4 days

Results [1]

They did well

Results [1]

They did well

Results: Positional Encodings [1]

Novel Folds

They did well

Ablation Studies [1]

Baseline for all ablation models: Full model without noisy-student self-attention
Ablations:

1. With noisy-student self-distillation training
2. No templates
3. No raw MSA (use MSA pairwise frequencies)
4. No triangles, biasing, or gating (use axial attention)
5. No recycling
6. No IPA (use direct projection)
7. No invariant IPA & no recycling
8. No end-to-end structure gradients (keep auxiliary heads)
9. No auxiliary distogram head
10. No auxiliary masked MSA head

Network Probing [1]

todo

Attention Visualization [1]

todo

Algorithm 24 Compute all atom coordinates [1]

Algorithm 24 Compute all atom coordinates

def computeAllAtomCoordinates($T_i, \vec{\alpha}_i^f, F_i^{\text{aatype}}$) :

1: $\hat{\vec{\alpha}}_i^f = \vec{\alpha}_i^f / \|\vec{\alpha}_i^f\|$

2: $(\vec{\omega}_i, \vec{\phi}_i, \vec{\psi}_i, \vec{\chi}_{1_i}, \vec{\chi}_{2_i}, \vec{\chi}_{3_i}, \vec{\chi}_{4_i}) = \hat{\vec{\alpha}}_i^f$

 # Make extra backbone frames.

3: $r_i = F_i^{\text{aatype}}$

4: $T_{i1} = T_i \circ T^{\text{lit}}_{r_i,(\omega\to\text{bb})} \circ \text{makeRotX}(\vec{\omega}_i)$

5: $T_{i2} = T_i \circ T^{\text{lit}}_{r_i,(\phi\to\text{bb})} \circ \text{makeRotX}(\vec{\phi}_i)$

6: $T_{i3} = T_i \circ T^{\text{lit}}_{r_i,(\psi\to\text{bb})} \circ \text{makeRotX}(\vec{\psi}_i)$

 # Make side chain frames (chain them up along the side chain).

7: $T_{i4} = T_i \circ T^{\text{lit}}_{r_i,(\chi_1\to\text{bb})} \circ \text{makeRotX}(\vec{\chi}_{1_i})$

8: $T_{i5} = T_{i4} \circ T^{\text{lit}}_{r_i,(\chi_2\to\chi_1)} \circ \text{makeRotX}(\vec{\chi}_{2_i})$

9: $T_{i6} = T_{i5} \circ T^{\text{lit}}_{r_i,(\chi_3\to\chi_2)} \circ \text{makeRotX}(\vec{\chi}_{3_i})$

10: $T_{i7} = T_{i6} \circ T^{\text{lit}}_{r_i,(\chi_4\to\chi_3)} \circ \text{makeRotX}(\vec{\chi}_{4_i})$

 # Map atom literature positions to the global frame.

11: $\vec{\mathbf{X}}_i^a = \text{concat}_{\text{out}}\left(\{T_i^f \circ \vec{\mathbf{X}}^{\text{lit}}_{\ldots}\}\right)$

Algorithm 23 Backbone update [1]

Algorithm 23 Backbone update

def BackboneUpdate($\mathbf{s}_i$) :

1: $b_i, c_i, d_i, \vec{\mathbf{t}}_i = \text{Linear}(\mathbf{s}_i)$ $\qquad b_i, c_i, d_i \in \mathbb{R},\ \vec{\mathbf{t}}_i \in \mathbb{R}^3$

 # Convert (non-unit) quaternion to rotation matrix.

2: $(a_i, b_i, c_i, d_i) \leftarrow (1, b_i, c_i, d_i)/\sqrt{1 + b_i^2 + c_i^2, +d_i^2}$

3: $R_i = \begin{pmatrix} a_i^2 + b_i^2 - c_i^2 - d_i^2 & 2b_ic_i - 2a_id_i & 2b_id_i + 2a_ic_i \\ 2b_ic_i + 2a_id_i & a_i^2 - b_i^2 + c_i^2 - d_i^2 & 2c_id_i - 2a_ib_i \\ 2b_id_i - 2a_ic_i & 2c_id_i + 2a_ib_i & a_i^2 - b_i^2 - c_i^2 + d_i^2 \end{pmatrix}$

4: $T_i = (R_i, \vec{\mathbf{t}}_i)$

5: **return** T_i

Algorithm 29 Predict model confidence pLDDT [1]

Algorithm 29 Predict model confidence pLDDT

def predictPerResidueLDDT_Ca($\{\mathbf{s}_i\}, \mathbf{v}_{\text{bins}} = [1, 3, 5, \ldots, 99]^\top, \{r_i^{\text{true LDDT}}\}, c = 128$) :

$\mathbf{a}_i = \text{relu}(\text{Linear}(\text{relu}(\text{Linear}(\text{LayerNorm}(\mathbf{s}_i)))))$ $\mathbf{a}_i$, and intermediate activations $\in \mathbb{R}^c$

$\mathbf{p}_i^{\text{pLDDT}} = \text{softmax}(\text{Linear}(\mathbf{a}_i))$ $\mathbf{p}_i^{\text{pLDDT}} \in \mathbb{R}^{|\mathbf{v}_{\text{bins}}|}$

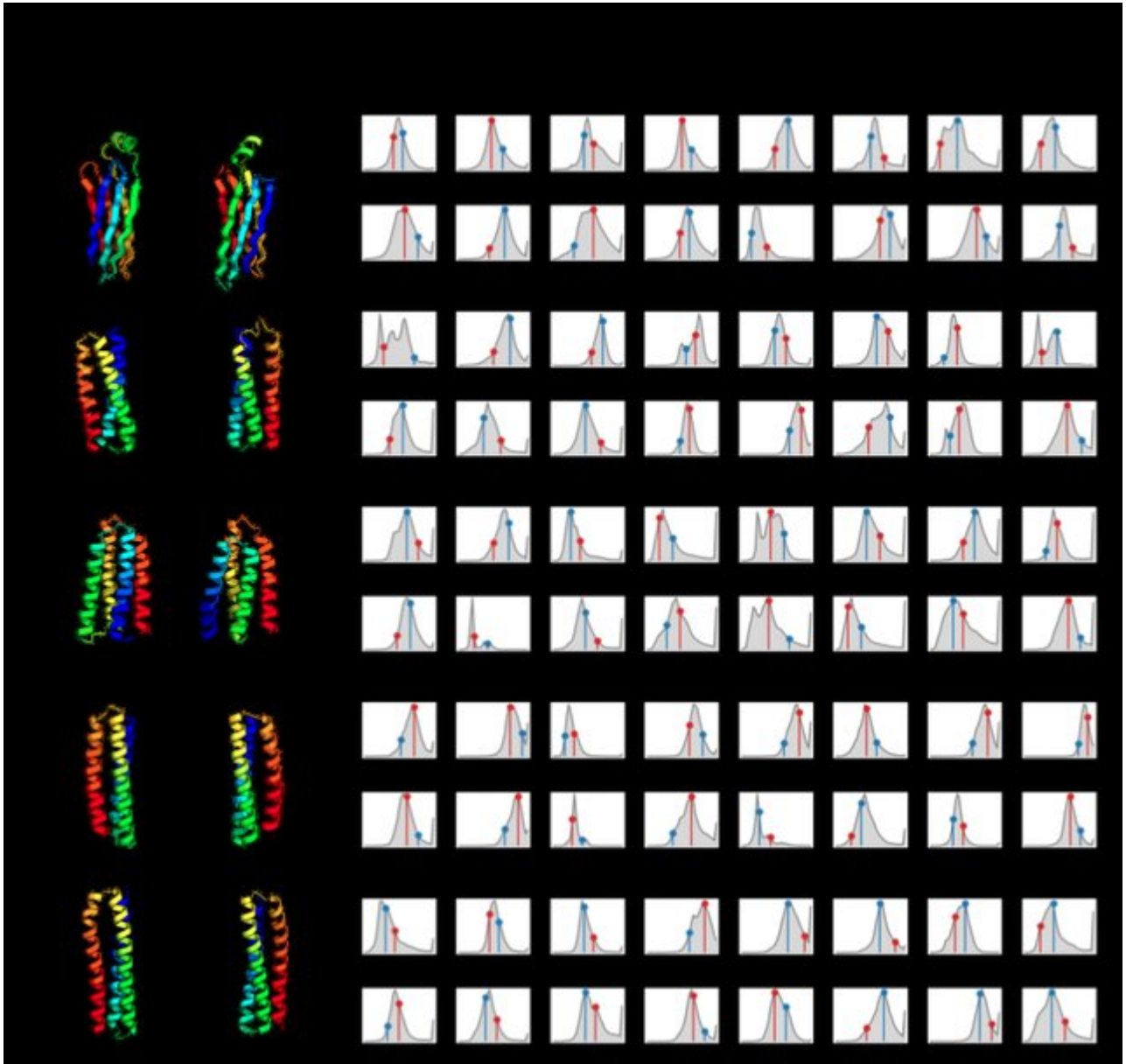
$\mathbf{p}_i^{\text{true LDDT}} = \text{one\_hot}(r_i^{\text{true LDDT}}, \mathbf{v}_{\text{bins}})$

$\mathcal{L}_{\text{conf}} = \text{mean}_i(\mathbf{p}_i^{\text{true LDDT}^\top} \log \mathbf{p}_i^{\text{pLDDT}})$

$r_i^{\text{pLDDT}} = \mathbf{p}_i^{\text{pLDDT}^\top} \mathbf{v}_{\text{bins}}$ $r_i^{\text{pLDDT}} \in \mathbb{R}$

return $\{r_i^{\text{pLDDT}}\}, \mathcal{L}_{\text{conf}}$

Distograms [1]



Algorithm 31 Generic recycling training procedure [1]

Algorithm 31 Generic recycling training procedure

def RecyclingTraining($\{\text{inputs}_c\}, N_{\text{cycle}}$) :

$N' = \text{uniform}(1, N_{\text{cycle}})$ *# shared value across the batch*

outputs $= \mathbf{0}$

 # Recycling iterations

for all $c \in [1, \ldots, N']$ **do** *# shared weights*

  outputs $\leftarrow$ stopgrad(outputs) *# no gradients between iterations*

  outputs $\leftarrow$ Model(inputs_c, outputs)

end for

return loss(outputs) *# only the final iteration's outputs are used*

Algorithm 30 Generic recycling inference procedure [1]

Algorithm 30 Generic recycling inference procedure

def RecyclingInference($\{\text{inputs}_c\}, N_{\text{cycle}}$) :

1: outputs $= \mathbf{0}$

 # Recycling iterations

2: **for all** $c \in [1, \ldots, N_{\text{cycle}}]$ **do** *# shared weights*

3: outputs $\leftarrow$ Model(inputs_c, outputs)

4: **end for**

5: **return** outputs

Algorithm 32 Embedding of evoformer and structure module outputs for recycling [1]

Algorithm 32 Embedding of Evoformer and Structure module outputs for recycling

def RecyclingEmbedder($\{\mathbf{m}_{1i}\}, \{\mathbf{z}_{ij}\}, \{\vec{\mathbf{x}}_i^{\mathrm{C}^\beta}\}$) :

 # Embed pair distances of backbone atoms:

$d_{ij} = \left\| \vec{\mathbf{x}}_i^{\mathrm{C}^\beta} - \vec{\mathbf{x}}_j^{\mathrm{C}^\beta} \right\|$ — C^α used for glycin

$\mathbf{d}_{ij} = \mathrm{Linear}(\mathrm{one\_hot}(d_{ij}, \mathbf{v}_{\mathrm{bins}} = [3\text{⅜ Å}, 5\text{⅛ Å}, \ldots, 21\text{⅜ Å}]))$ — $\mathbf{d}_{ij} \in \mathbb{R}^{c_z}$

 # Embed output Evoformer representations:

$\tilde{\mathbf{z}}_{ij} = \mathbf{d}_{ij} + \mathrm{LayerNorm}(\mathbf{z}_{ij})$

$\tilde{\mathbf{m}}_{1i} = \mathrm{LayerNorm}(\mathbf{m}_{1i})$

return $\{\tilde{\mathbf{m}}_{1i}\}, \{\tilde{\mathbf{z}}_{ij}\}$

Algorithm 26 Rename symmetric ground truth atoms [1]

Algorithm 26 Rename symmetric ground truth atoms

def renameSymmetricGroundTruthAtoms($\{T_i^f\}, \{\vec{\mathbf{x}}_i^a\}, \{T_i^{\text{true},f}\}, \{T_i^{\text{alt truth},f}\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\}, \{\vec{\mathbf{x}}_i^{\text{alt truth},a}\}$) :

1: $d_{(i,a),(j,b)} = \left\| \vec{\mathbf{x}}_i^a - \vec{\mathbf{x}}_j^b \right\|$ $\quad \forall (j,b) \in \mathcal{S}_{\text{non-ambiguous atoms}}$

2: $d^{\text{true}}_{(i,a),(j,b)} = \left\| \vec{\mathbf{x}}_i^{\text{true},a} - \vec{\mathbf{x}}_j^{\text{true},b} \right\|$ $\quad \forall (j,b) \in \mathcal{S}_{\text{non-ambiguous atoms}}$

3: $d^{\text{alt truth}}_{(i,a),(j,b)} = \left\| \vec{\mathbf{x}}_i^{\text{alt truth},a} - \vec{\mathbf{x}}_j^{\text{true},b} \right\|$ $\quad \forall (j,b) \in \mathcal{S}_{\text{non-ambiguous atoms}}$

4: **for all** $i \in [1 \dots N_{\text{res}}]$ **do**

5: **if** $\text{sum}_{a,(j,b)} \left(\left| d_{(i,a),(j,b)} - d^{\text{alt truth}}_{(i,a),(j,b)} \right| \right) < \text{sum}_{a,(j,b)} \left(\left| d_{(i,a),(j,b)} - d^{\text{true}}_{(i,a),(j,b)} \right| \right)$ **then**

6: $\vec{\mathbf{x}}_i^{\text{true},a} \leftarrow \vec{\mathbf{x}}_i^{\text{alt truth},a}$

7: $T_i^{\text{true},f} \leftarrow T_i^{\text{alt truth},f}$

8: **end if**

9: **end for**

10: **return** $\{T_i^{\text{true},f}\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\}$

Algorithm 27 Side chain and backbone torsion angle loss [1]

Algorithm 27 Side chain and backbone torsion angle loss

def $\text{torsionAngleLoss}(\{\vec{\alpha}_i^f\}, \{\vec{\alpha}_i^{\text{true},f}\}, \{\vec{\alpha}_i^{\text{alt truth},f}\})$:

1: $\ell_i^f = \|\vec{\alpha}_i^f\|$

2: $\hat{\vec{\alpha}}_i^f = \vec{\alpha}_i^f / \ell_i^f$

3: $\mathcal{L}_{\text{torsion}} = \text{mean}_{i,f}(\text{minimum}(\|\hat{\vec{\alpha}}_i^f - \vec{\alpha}_i^{\text{true},f}\|^2,\ \|\hat{\vec{\alpha}}_i^f - \vec{\alpha}_i^{\text{alt truth},f}\|^2))$

4: $\mathcal{L}_{\text{anglenorm}} = \text{mean}_{i,f}(\{|\ell_i^f - 1|\})$

5: **return** $\mathcal{L}_{\text{torsion}} + 0.02\mathcal{L}_{\text{anglenorm}}$

Algorithm 25 Make a transformation that rotates around the x-axis [1]

Algorithm 25 Make a transformation that rotates around the x-axis

def makeRotX($\vec{\alpha}$) : $\vec{\alpha} \in R^2$ with $\|\vec{\alpha}\| = 1$

1: $R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \alpha_1 & -\alpha_2 \\ 0 & \alpha_2 & \alpha_1 \end{pmatrix}$

2: $\vec{\mathbf{t}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$

3: $T = (R, \vec{\mathbf{t}})$

4: **return** T

Bibliography I

[1] John Jumper et al. "Highly Accurate Protein Structure Prediction with AlphaFold". In: *Nature* (July 15, 2021), pp. 1–11. ISSN: 1476-4687. DOI: 10/gk7nfp. URL: https://www.nature.com/articles/s41586-021-03819-2 (visited on 07/15/2021).

[2] *PDB101: Learn: Guide to Understanding PDB Data: Beginner's Guide to PDB Structures and the PDBx/mmCIF Format*. RCSB: PDB-101. URL: https://pdb101.rcsb.org/learn/guide-to-understanding-pdb-data/beginner%E2%80%99s-guide-to-pdb-structures-and-the-pdbx-mmcif-format (visited on 07/24/2021).

[3] Carlos Outeiral Rubiera. *AlphaFold 2 Is Here: What's behind the Structure Prediction Miracle | Oxford Protein Informatics Group*. URL: https://www.blopig.com/blog/2021/07/alphafold-2-is-here-whats-behind-the-structure-prediction-miracle/ (visited on 07/24/2021).