

# Highly accurate protein structure prediction with AlphaFold

## AISC Healthcare Discussion Group

Willy Rempel

Saturday, July 24, 2021

# Introduction



Willy Rempel

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

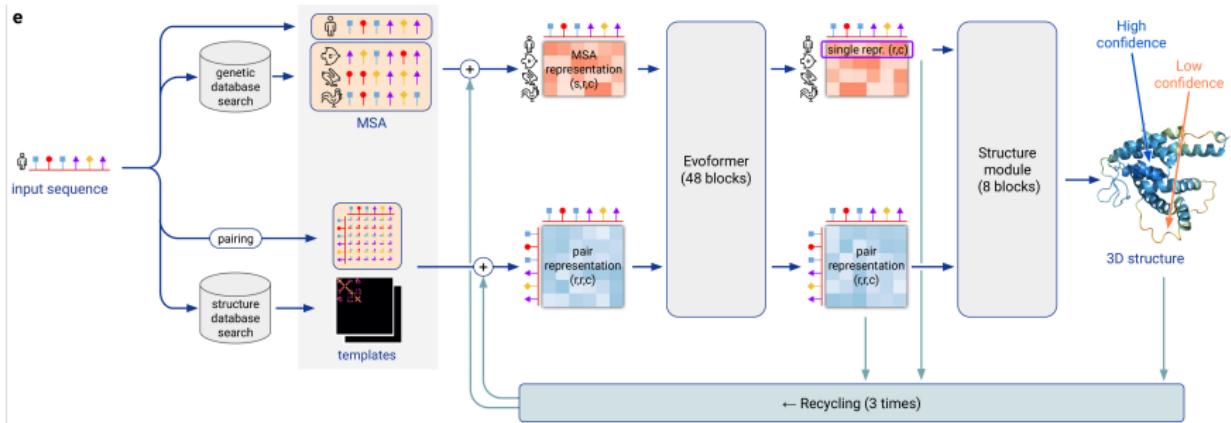
## Introduction

*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

```
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		TRP	A	1	5	?	8.519	-0.751	10.738	1.00	13.37	?	?	?	?	?	?	5	
ATOM	2	C	CA	.	TRP	A	1	5	?	7.743	-1.668	11.585	1.00	13.42	?	?	?	?	?	?	5
ATOM	3	C	C	.	TRP	A	1	5	?	6.786	-2.502	10.667	1.00	13.47	?	?	?	?	?	?	5
ATOM	4	O	O	.	TRP	A	1	5	?	6.422	-2.085	9.607	1.00	13.57	?	?	?	?	?	?	5
ATOM	5	C	CB	.	TRP	A	1	5	?	6.997	-0.917	12.645	1.00	13.34	?	?	?	?	?	?	5
ATOM	6	C	CG	.	TRP	A	1	5	?	5.784	-0.209	12.221	1.00	13.40	?	?	?	?	?	?	5
ATOM	7	C	CD1	.	TRP	A	1	5	?	5.681	1.084	11.797	1.00	13.29	?	?	?	?	?	?	5
ATOM	8	C	CD2	.	TRP	A	1	5	?	4.417	-0.667	12.221	1.00	13.34	?	?	?	?	?	?	5
ATOM	9	N	NE1	.	TRP	A	1	5	?	4.388	1.418	11.515	1.00	13.30	?	?	?	?	?	?	5
ATOM	10	C	CE2	.	TRP	A	1	5	?	3.588	0.375	11.797	1.00	13.35	?	?	?	?	?	?	5
ATOM	11	C	CE3	.	TRP	A	1	5	?	3.837	-1.877	12.645	1.00	13.39	?	?	?	?	?	?	5
ATOM	12	C	CZ2	.	TRP	A	1	5	?	2.216	0.208	11.656	1.00	13.39	?	?	?	?	?	?	5
ATOM	13	C	CZ3	.	TRP	A	1	5	?	2.465	-2.043	12.504	1.00	13.33	?	?	?	?	?	?	5
ATOM	14	C	CH2	.	TRP	A	1	5	?	1.654	-1.001	12.009	1.00	13.34	?	?	?	?	?	?	5

redundant annotations

inefficient representation

repetitive information

## Initial Input: mmCIF or FASTA files

```
;LCB0 - Prolactin precursor - Bovine
; a sample sequence in FASTA format
MDSKGSSQKGSRLLLLLVVSNLLCQGVVSTPVCPNGPGNCQVSLRDLFDRAVMVSHYIHDLSS
EMFNEFDKRYAQGKGFITMALNSCHTSSLPTPEDKEQAQQTHHEVLMMSLILGLLRSWNDPLYHL
VTEVRGMKGAPDAILSRAIEIEEENKRLLEGMEMIFGQVIPGAKETEPYPVWSGLPSLQTKDED
ARYSAFYNLLHCLRRDSSKIDTYLKLLNCRIYNNNC*

>MCHU - Calmodulin - Human, rabbit, bovine, rat, and chicken
MADQLTEEQIAEFKEAFSLFDKDGDTITTKELGTVMRSLGQNPTAEELQDMINEVDADGNGTID
FPEFLTMMARKMKDTDSEEEIREAFRVDKDGNGYISAAELRHVMTNLGEKLTDEEVDEMIREA
DIDGDGQVNYYEEFVQMMTAK*

>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAI PYIGTNLV
EWIWGGFSVDKATLNRRFFAHFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTWIGSQPVEYPYTIIGQMASILYFSIIILAFLPIAGX
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<sub>max</sub> 100000 -maxfilt 100000 -min<sub>prefilterhits</sub> 1000 -maxseq 100000.

## 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]

- Block deletion tends to remove similarities (ie. whole branch phylogeny) and promote diversity
  - Similar sequences are likely to be adjacent
  - Contiguous blocks in MSAs are deleted.
  - First MSAs are grouped by tool
  - Then sorted according to tool defaults (usually e-value)

## Algorithm 1 MSA Block deletion [1]

```
./imgs/algo1_blockdelete.png
```

# MSA clustering [1]

- Similarity clusters used to randomly select subset of MSA sequences
    - to reduce computational cost from attention modules, reduce  $N_{seq}$
  - Modified K-means is used, with the input sequence used as first cluster center
- 1  $N_{clust}$  centers selected from MSA
  - 2 Generate a mask where  $p=0.15$  that any position is selected by the mask
  - 3 Each center is modified for each mask selected residue according to:
    - 1  $p=0.1$  replaced with a uniformly sampled random amino acid
    - 2  $p=0.1$  replaced with an amino acid sampled from the MSA profile
    - 3  $p=0.1$  no replacement
    - 4  $p=0.7$  replaced with a special token ( $\text{masked}_{\text{msa token}}$ )
  - 4 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]

- **target<sub>feat</sub>** This is a feature of size [Nres, 21] consisting of the “aatype” feature.
- **residue<sub>index</sub>** This is a feature of size [Nres] consisting of the “residue<sub>index</sub>” feature.
- **msafeat** This is a feature of size [Nclust, Nres, 49] constructed by concatenating “cluster<sub>msa</sub>”, “cluster<sub>hasdeletion</sub>”, “cluster<sub>deletionvalue</sub>”, “cluster<sub>deletionmean</sub>”, “cluster<sub>profile</sub>”. We draw  $N_{cycle} \times N_{ensemble}$  random samples from this feature to provide each recycling/ensembling iteration of the network with a different sample (see subsubsection 1.11.2).
- **extra<sub>msafeat</sub>** This is a feature of size [ $N_{extra_{seq}}$ , Nres, 25] constructed by concatenating “extra<sub>msa</sub>”, “extra<sub>msahasdeletion</sub>”, “extra<sub>msadeletionvalue</sub>”. Together with ‘msafeat’ above we also draw  $N_{cycle} \times N_{ensemble}$  random samples from this feature (see subsubsection 1.11.2).

# Featurization and model inputs [1]

- **template<sub>pairfeat</sub>** This is a feature of size [Ntempl, Nres, Nres, 88] and consists of concatenation of the pair residue features “template<sub>distogram</sub>”, “template<sub>unitvector</sub>”, and also several residue features, which are transformed into pair features. The “template<sub>aatype</sub>” feature is included via tiling and stack- ing (this is done twice, in both residue directions). Also the mask features “template<sub>pseudobetamask</sub>” and “template<sub>backboneframemask</sub>” are included, where the feature  $f_{ij} = \text{mask}_i \cdot \text{mask}_j$ . -  
**template<sub>anglefeat</sub>** This is a feature of size [Ntempl, Nres, 51] constructed by concatenating the following features: “template<sub>aatype</sub>”, “template<sub>torsionangles</sub>”, “template<sub>alttorsionangles</sub>”, and “template<sub>torsionanglesmask</sub>”.

# Table 1 Input Features (1/2) [1]

**Table 1** | Input features to the model. Feature dimensions:  $N_{\text{res}}$  is the number of residues,  $N_{\text{clust}}$  is the number of MSA clusters,  $N_{\text{extra\_seq}}$  is the number of additional unclustered MSA sequences, and  $N_{\text{templ}}$  is the number of templates.

Feature & Shape	Description
aatype [ $N_{\text{res}}$ , 21]	One-hot representation of the input amino acid sequence (20 amino acids + unknown).
cluster_msa [ $N_{\text{clust}}$ , $N_{\text{res}}$ , 23]	One-hot representation of the msa cluster centre sequences (20 amino acids + unknown + gap + masked_msa_token).
cluster_has_deletion [ $N_{\text{clust}}$ , $N_{\text{res}}$ , 1]	A binary feature indicating if there is a deletion to the left of the residue in the MSA cluster centres.
cluster_deletion_value [ $N_{\text{clust}}$ , $N_{\text{res}}$ , 1]	The raw deletion counts (the number of deletions to the left of every position in the MSA cluster centres) are transformed to the range [0, 1] using $\frac{2}{\pi} \arctan \frac{d}{3}$ where $d$ are the raw counts.
cluster_deletion_mean [ $N_{\text{clust}}$ , $N_{\text{res}}$ , 1]	The mean deletions for every residue in every cluster are computed as $\frac{1}{n} \sum_{i=1}^n d_{ij}$ where $n$ is the number of sequences in the cluster and $d_{ij}$ is the number of deletions to the left of the $i$ th sequence and $j$ th residue. These are then transformed into the range [0, 1] in the same way as for the cluster_deletion_value feature above.
cluster_profile [ $N_{\text{clust}}$ , $N_{\text{res}}$ , 23]	The distribution across amino acid types for each residue in each MSA cluster (20 amino acids + unknown + gap + masked_msa_token).

# Table 1 Input Features (2/2) [1]

Feature & Shape	Description
extra_msa [ $N_{\text{extra\_seq}}$ , $N_{\text{res}}$ , 23]	One-hot representation of all MSA sequences not selected as cluster centres (20 amino acids + unknown + gap + masked_msa_token).
extra_msa_has_deletion [ $N_{\text{extra\_seq}}$ , $N_{\text{res}}$ , 1]	A binary feature indicating if there is a deletion to the left of the residue in the extra MSA sequences.
extra_msa_deletion_value [ $N_{\text{extra\_seq}}$ , $N_{\text{res}}$ , 1]	The raw deletion counts to the left of every residue in the extra_msa, converted to the range [0, 1] using the same formula as for cluster_deletion_value.
template_aatype [ $N_{\text{templ}}$ , $N_{\text{res}}$ , 22]	One-hot representation of the amino acid sequence (20 amino acids + unknown and gap).
template_mask [ $N_{\text{templ}}$ , $N_{\text{res}}$ ]	Mask indicating if a template residue exists and has coordinates.
template_pseudo_beta_mask [ $N_{\text{templ}}$ , $N_{\text{res}}$ ]	Mask indicating if the beta carbon (alpha carbon for glycine) atom has coordinates for the template at this residue.
template_backbone_frame_mask [ $N_{\text{templ}}$ , $N_{\text{res}}$ ]	A mask indicating if the coordinates of all the required atoms to compute the backbone frame (used in the template_unit_vector feature) exist.
template_distogram [ $N_{\text{templ}}$ , $N_{\text{res}}$ , $N_{\text{res}}$ , 39]	A one-hot pairwise feature indicating the distance between beta carbons (alpha carbon used for glycine) atoms. The pairwise distances are discretized into 38 bins of equal width between 3.25 Å and 50.75 Å; and one more bin contains any larger distances.
template_unit_vector [ $N_{\text{templ}}$ , $N_{\text{res}}$ , $N_{\text{res}}$ , 3]	The unit vector of the displacement of the alpha carbon atom of all residues within the local frame of each residue. These local frames are computed in the same way as for the target structure, see <a href="#">subsubsection 1.8.1</a> .
template_torsion_angles [ $N_{\text{templ}}$ , $N_{\text{res}}$ , 14]	The 3 backbone torsion angles and up to 4 side-chain torsion angles for each residue represented as sine and cosine encoding.
template_alt_torsion_angles [ $N_{\text{templ}}$ , $N_{\text{res}}$ , 14]	Alternative torsion angles for side chain parts with 180°-rotation symmetry.
template_torsion_angles_mask [ $N_{\text{templ}}$ , $N_{\text{res}}$ , 14]	A mask indicating if the torsion angle is present in the template structure.
residue_index [ $N_{\text{res}}$ ]	The index into the original amino acid sequence.

# 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 'undistilled' 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

# 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).

# Algorithm 2 Model Inference [1]

---

**Algorithm 2** AlphaFold Model Inference

---

```
def Inference  $\left( \{f_i^{\text{target\_feat}}\}, \{f_i^{\text{residue\_index}}\}, \left\{ \{f_{sc i}^{\text{msa\_feat}}\}_{c,n} \right\}, \left\{ \{f_{se i}^{\text{extra\_msa\_feat}}\}_{c,n} \right\}, \{f_{st i}^{\text{template\_angle\_feat}}\}, \{f_{st ij}^{\text{template\_pair\_feat}}\}, N_{\text{cycle}} = 4, N_{\text{ensemble}} = 3 \right) :$ 
```

# Recycling iterations:

1:  $\hat{\mathbf{m}}_{1i}^{\text{prev}}, \hat{\mathbf{z}}_{ij}^{\text{prev}}, \vec{\mathbf{x}}_i^{\text{prev,C}^\beta} = \mathbf{0}, \mathbf{0}, \vec{\mathbf{0}}$

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

# Average embeddings in ensemble:

3:  $\{\hat{\mathbf{m}}_{1i}\}, \{\hat{\mathbf{z}}_{ij}\}, \{\hat{\mathbf{s}}_i\} = \mathbf{0}, \mathbf{0}, \mathbf{0}$

4: **for all**  $n \in [1, \dots, N_{\text{ensemble}}]$  **do** # shared weights

# Embed clustered MSA (use different MSA samples in each iteration):

5:  $\{\mathbf{m}_{sc i}\}, \{\mathbf{z}_{ij}\} = \text{InputEmbedder}(\{f_i^{\text{target\_feat}}\}, \{f_i^{\text{residue\_index}}\}, \{f_{sc i}^{\text{msa\_feat}}\}_{c,n})$

# Inject previous outputs for recycling:

6:  $\{\mathbf{m}_{1i}\}, \{\mathbf{z}_{ij}\} += \text{RecyclingEmbedder}(\{\hat{\mathbf{m}}_{1i}^{\text{prev}}\}, \{\hat{\mathbf{z}}_{ij}^{\text{prev}}\}, \{\vec{\mathbf{x}}_i^{\text{prev,C}^\beta}\})$

# Embed templates:

7:  $\mathbf{a}_{st i} \leftarrow \text{Linear}(\text{relu}(\text{Linear}(f_{st i}^{\text{template\_angle\_feat}}))) \quad \mathbf{a}_{st i} \in \mathbb{R}^{c_m}, c_m = 256$

8:  $\mathbf{m}_{si} = \text{concat}_s(\mathbf{m}_{sc i}, \mathbf{a}_{st i})$

template\_pair\_feat

## Algorithm 2 Model Inference [1]

```
#      Embed extra MSA features (use different samples in each iteration):
14:      $\{\mathbf{a}_{se_i}\} \leftarrow \{\mathbf{f}_{se_i}^{\text{extra\_msa\_feat}}\}_{c,n}$ 
15:      $\mathbf{e}_{se_i} = \text{Linear}(\mathbf{a}_{se_i})$   $\mathbf{e}_{se_i} \in \mathbb{R}^{c_e}, c_e = 64$ 
16:      $\{\mathbf{z}_{ij}\} \leftarrow \text{ExtraMsaStack}(\{\mathbf{e}_{se_i}\}, \{\mathbf{z}_{ij}\})$ 

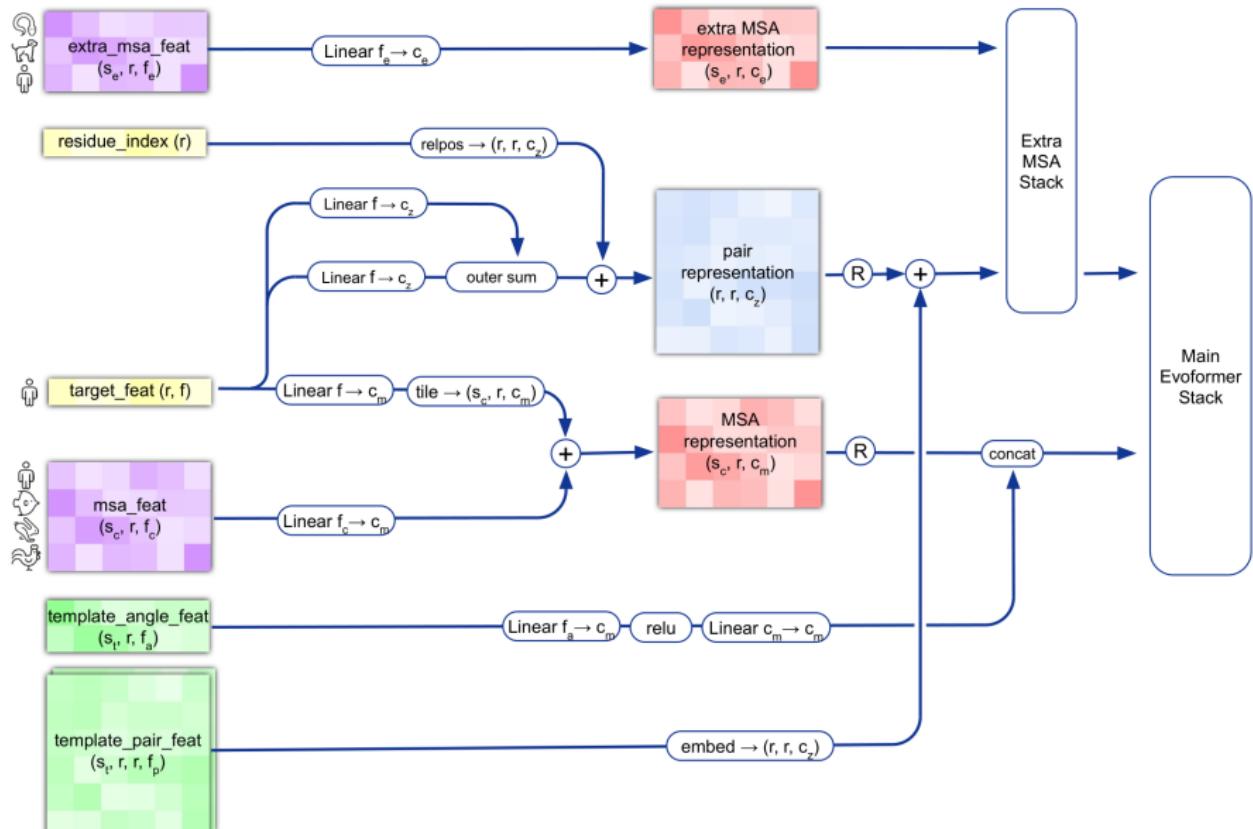
#      Main trunk of the network:
17:      $\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}, \{\mathbf{s}_i\} \leftarrow \text{EvoformerStack}(\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\})$ 
18:      $\{\hat{\mathbf{m}}_{1i}\}, \{\hat{\mathbf{z}}_{ij}\}, \{\hat{\mathbf{s}}_i\} += \{\mathbf{m}_{1i}\}, \{\mathbf{z}_{ij}\}, \{\mathbf{s}_i\}$ 
19:   end for
20:    $\{\hat{\mathbf{m}}_{1i}\}, \{\hat{\mathbf{z}}_{ij}\}, \{\hat{\mathbf{s}}_i\} /= N_{\text{ensemble}}$ 

#      Structure and confidence prediction:
21:      $\{\vec{\mathbf{x}}_i^a\}, \{r_i^{\text{PLDDT}}\} = \text{StructureModule}(\{\hat{\mathbf{s}}_i\}, \{\hat{\mathbf{z}}_{ij}\})$ 
22:      $\{\hat{\mathbf{m}}_{1i}^{\text{prev}}\}, \{\hat{\mathbf{z}}_{ij}^{\text{prev}}\}, \{\vec{\mathbf{x}}_i^{\text{prev}, C^\beta}\} \leftarrow \{\hat{\mathbf{m}}_{1i}\}, \{\hat{\mathbf{z}}_{ij}\}, \{\vec{\mathbf{x}}_i^{C^\beta}\}$ 
23:   end for
24:   return  $\{\vec{\mathbf{x}}_i^a\}, \{r_i^{\text{PLDDT}}\}$ 
```

# AlphaFold Training [1]

`./imgs/af-training-table.png`

# Input embeddings [1]



### Algorithm 3 Input embeddings [1]

---

## Algorithm 3 Embeddings for initial representations

---

```
def InputEmbedder( $\{\mathbf{f}_i^{\text{target\_feat}}\}$ ,  $\{f_i^{\text{residue\_index}}\}$ , {  
    1:  $\mathbf{a}_i, \mathbf{b}_i = \text{Linear}(\mathbf{f}_i^{\text{target\_feat}})$   
    2:  $\mathbf{z}_{ij} = \mathbf{a}_i + \mathbf{b}_j$   
    3:  $\{\mathbf{z}_{ij}\} += \text{relpos}(\{f_i^{\text{residue\_index}}\})$   
    4:  $\mathbf{m}_{sci} = \text{Linear}(\mathbf{f}_{sci}^{\text{msa\_feat}}) + \text{Linear}(\mathbf{f}_i^{\text{target\_feat}})$   
    5: return  $\{\mathbf{m}_{sci}\}, \{\mathbf{z}_{ij}\}$ 
```

Algorithm 4 Relative positional encoding [1]

---

## Algorithm 4 Relative position encod

---

```
def relpos( $\{f_i^{\text{residue\_index}}\}$ ,  $\mathbf{v}_{\text{bins}} = [$   
1:  $d_{ij} = f_i^{\text{residue\_index}} - f_j^{\text{residue\_index}}$   
2:  $\mathbf{p}_{ij} = \text{Linear}(\text{one\_hot}(d_{ij}, \mathbf{v}_{\text{bins}}))$   
3: return  $\{\mathbf{p}_{ij}\}$ 
```

## Algorithm 5 One-hot encoding with nearest bin [1]

---

### Algorithm 5 One-hot encoding with nearest bin

---

```
def one_hot( $x$ ,  $\mathbf{v}_{\text{bins}}$ ) :
```

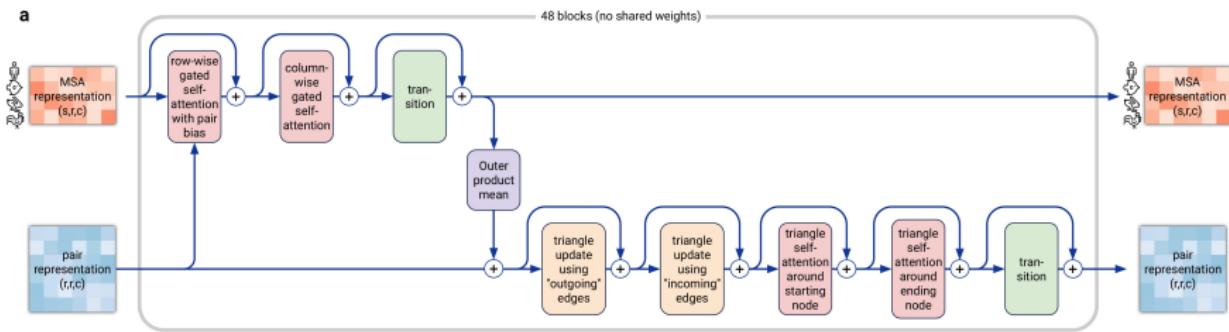
```
    1:  $\mathbf{p} = \mathbf{0}$ 
```

```
    2:  $b = \arg \min(|x - \mathbf{v}_{\text{bins}}|)$ 
```

```
    3:  $p_b = 1$ 
```

```
    4: return  $\mathbf{p}$ 
```

# 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

# Algorithm 6 EvoFormer stack [1]

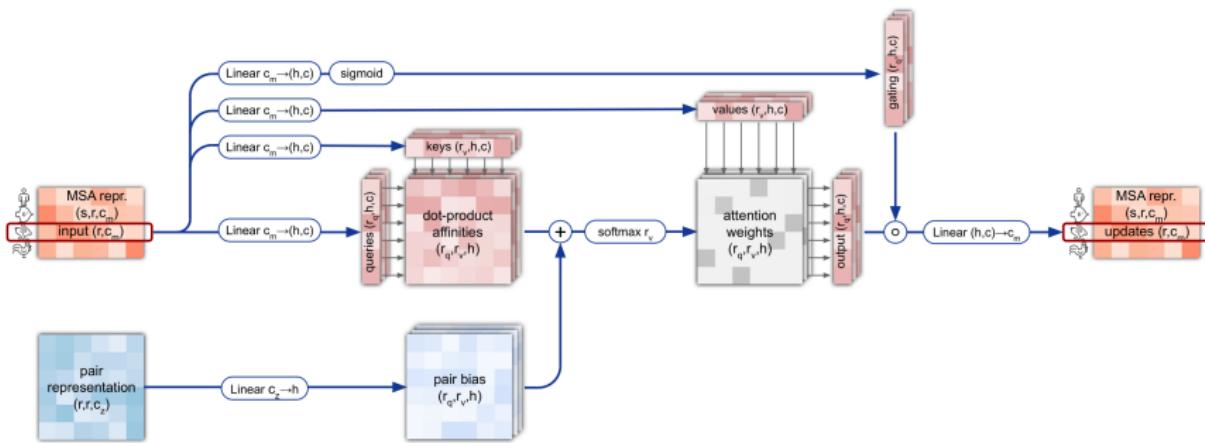
---

**Algorithm 6** Evoformer stack

---

```
def EvoformerStack({ $\{\mathbf{m}_{si}\}$ ,  $\{\mathbf{z}_{ij}\}$ ,  $N_{block} = 48$ ,  $c_s = 384$ } :  
1: for all  $l \in [1, \dots, N_{block}]$  do  
# MSA stack  
2:    $\{\mathbf{m}_{si}\} += \text{DropoutRowwise}_{0.15}(\text{MSARowAttentionWithPairBias}(\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}))$   
3:    $\{\mathbf{m}_{si}\} += \text{MSAColumnAttention}(\{\mathbf{m}_{si}\})$   
4:    $\{\mathbf{m}_{si}\} += \text{MSATransition}(\{\mathbf{m}_{si}\})$   
# Communication  
5:    $\{\mathbf{z}_{ij}\} += \text{OuterProductMean}(\{\mathbf{m}_{si}\})$   
# Pair stack  
6:    $\{\mathbf{z}_{ij}\} += \text{DropoutRowwise}_{0.25}(\text{TriangleMultiplicationOutgoing}(\{\mathbf{z}_{ij}\}))$   
7:    $\{\mathbf{z}_{ij}\} += \text{DropoutRowwise}_{0.25}(\text{TriangleMultiplicationIncoming}(\{\mathbf{z}_{ij}\}))$   
8:    $\{\mathbf{z}_{ij}\} += \text{DropoutRowwise}_{0.25}(\text{TriangleAttentionStartingNode}(\{\mathbf{z}_{ij}\}))$   
9:    $\{\mathbf{z}_{ij}\} += \text{DropoutColumnwise}_{0.25}(\text{TriangleAttentionEndingNode}(\{\mathbf{z}_{ij}\}))$   
10:   $\{\mathbf{z}_{ij}\} += \text{PairTransition}(\{\mathbf{z}_{ij}\})$   
11: end for  
# Extract the single representation  
12:  $\mathbf{s}_i = \text{Linear}(\mathbf{m}_{1i})$   $\mathbf{s}_i \in \mathbb{R}^{c_s}$   
13: return  $\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}, \{\mathbf{s}_i\}$ 
```

# EvoFormer: Row wise Gated Attention [1]



## Algorithm 7 Row wise Gated Attention [1]

---

**Algorithm 7** MSA row-wise gated self-attention with pair bias

---

**def** MSARowAttentionWithPairBias( $\{\mathbf{m}_{si}\}$ ,  $\{\mathbf{z}_{ij}\}$ ,  $c = 32$ ,  $N_{\text{head}} = 8$ ) :

*# Input projections*

1:  $\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

2:  $\mathbf{q}_{si}^h, \mathbf{k}_{si}^h, \mathbf{v}_{si}^h = \text{LinearNoBias}(\mathbf{m}_{si})$   $\mathbf{q}_{si}^h, \mathbf{k}_{si}^h, \mathbf{v}_{si}^h \in \mathbb{R}^c$ ,

3:  $b_{ij}^h = \text{LinearNoBias}(\text{LayerNorm}(\mathbf{z}_{ij}))$

4:  $\mathbf{g}_{si}^h = \text{sigmoid}(\text{Linear}(\mathbf{m}_{si}))$

*# Attention*

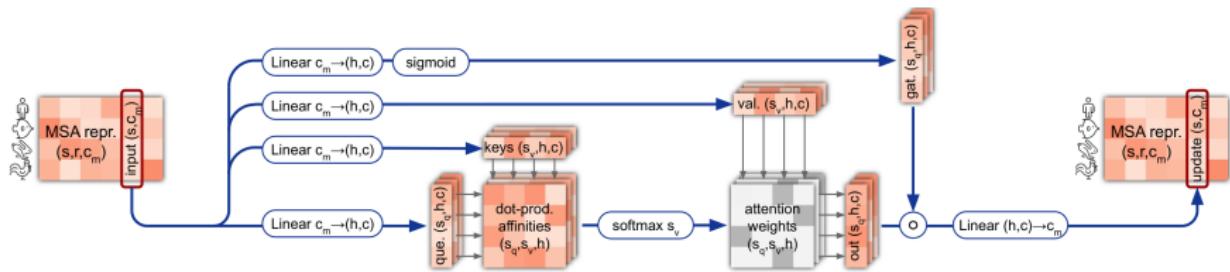
5:  $a_{sij}^h = \text{softmax}_j \left( \frac{1}{\sqrt{c}} \mathbf{q}_{si}^{h\top} \mathbf{k}_{sj}^h + b_{ij}^h \right)$

6:  $\mathbf{o}_{si}^h = \mathbf{g}_{si}^h \odot \sum_j a_{sij}^h \mathbf{v}_{sj}^h$

*# Output projection*

7:  $\tilde{\mathbf{m}}_{si} = \text{Linear} \left( \text{concat}_h(\mathbf{o}_{si}^h) \right)$

# EvoFormer: Column wise Gated Attention [1]



# Algorithm 8 Column wise Gated Attention [1]

---

**Algorithm 8** MSA column-wise gated self-attention

---

**def** MSAColumnAttention( $\{\mathbf{m}_{si}\}$ ,  $c = 32$ ,  $N_{\text{head}} = 8$ ) :

*# Input projections*

1:  $\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

2:  $\mathbf{q}_{si}^h, \mathbf{k}_{si}^h, \mathbf{v}_{si}^h = \text{LinearNoBias}(\mathbf{m}_{si})$

$\mathbf{q}_{si}^h, \mathbf{k}_{si}^h, \mathbf{v}_{si}^h \in \mathbb{R}^c$ ,  $h$

3:  $\mathbf{g}_{si}^h = \text{sigmoid}(\text{Linear}(\mathbf{m}_{si}))$

*# Attention*

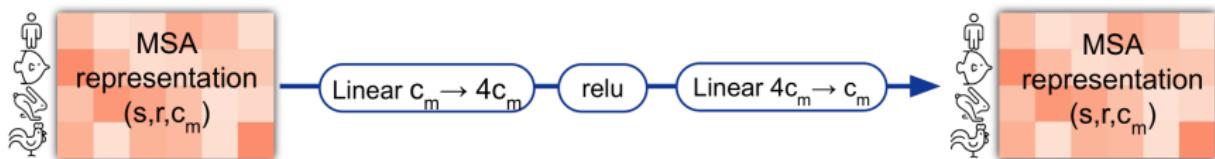
4:  $a_{sti}^h = \text{softmax}_t \left( \frac{1}{\sqrt{c}} \mathbf{q}_{si}^{h\top} \mathbf{k}_{ti}^h \right)$

5:  $\mathbf{o}_{si}^h = \mathbf{g}_{si}^h \odot \sum_t a_{sti}^h \mathbf{v}_{st}^h$

*# Output projection*

6:  $\tilde{\mathbf{m}}_{si} = \text{Linear} \left( \text{concat}_h(\mathbf{o}_{si}^h) \right)$

# EvoFormer: MSA Translation Layer [1]



## Algorithm 9 MSA Translation Layer [1]

---

### Algorithm 9 Transition layer in the MSA

---

**def** MSATransition( $\{\mathbf{m}_{si}\}$ ,  $n = 4$ ) :

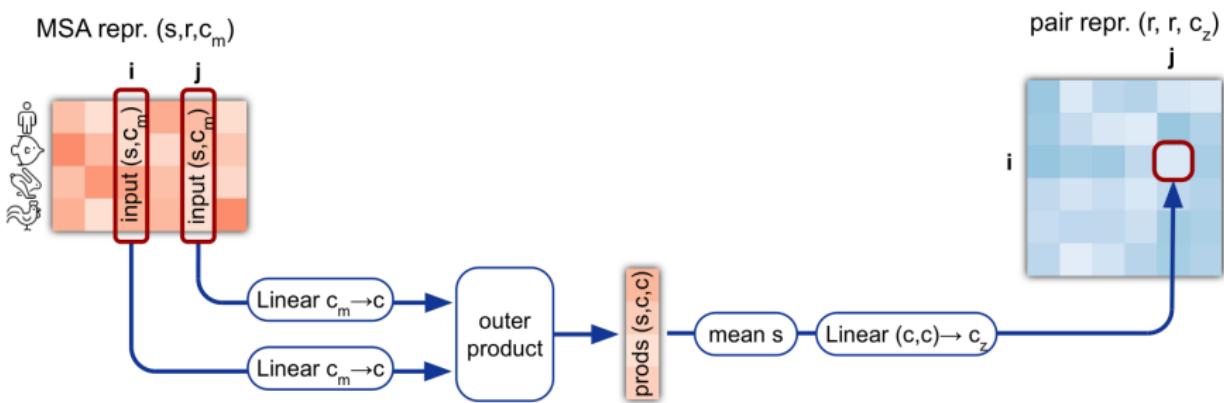
1:  $\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

2:  $\mathbf{a}_{si} = \text{Linear}(\mathbf{m}_{si})$

3:  $\mathbf{m}_{si} \leftarrow \text{Linear}(\text{relu}(\mathbf{a}_{si}))$

4: **return**  $\{\mathbf{m}_{si}\}$

# EvoFormer: Outer-Product Mean [1]



## Algorithm 10 Outer-Product Mean [1]

---

### Algorithm 10 Outer product mean

---

**def** OuterProductMean( $\{\mathbf{m}_{si}\}$ ,  $c = 32$ ) :

1:  $\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

2:  $\mathbf{a}_{si}, \mathbf{b}_{si} = \text{Linear}(\mathbf{m}_{si})$

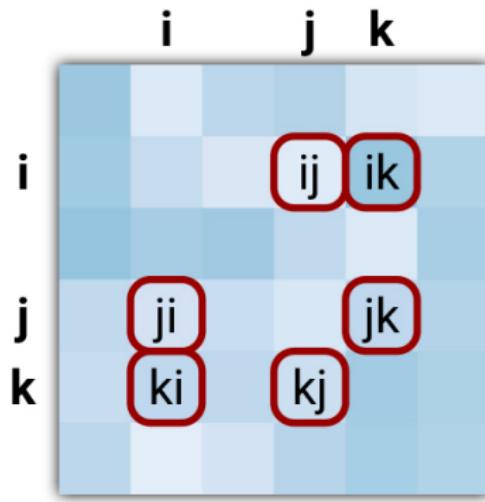
3:  $\mathbf{o}_{ij} = \text{flatten}(\text{mean}_s(\mathbf{a}_{si} \otimes \mathbf{b}_{sj}))$

4:  $\mathbf{z}_{ij} = \text{Linear}(\mathbf{o}_{ij})$

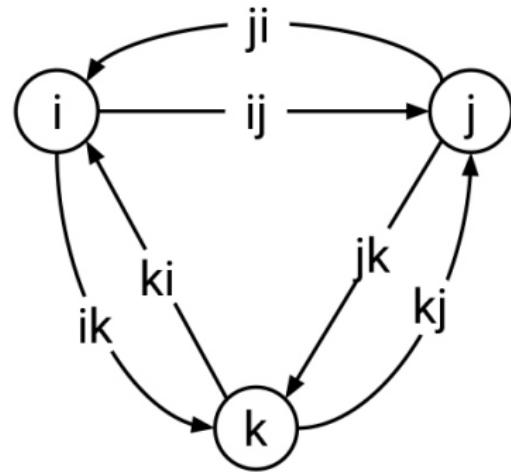
5: **return**  $\{\mathbf{z}_{ij}\}$

# EvoFormer: Residue Pairs [1]

**b** pair representation  
( $r, r, c'$ )



corresponding edges  
in a graph



Triangle multiplicative update  
using "outgoing" edges



Triangle multiplicative update  
using "incoming" edges



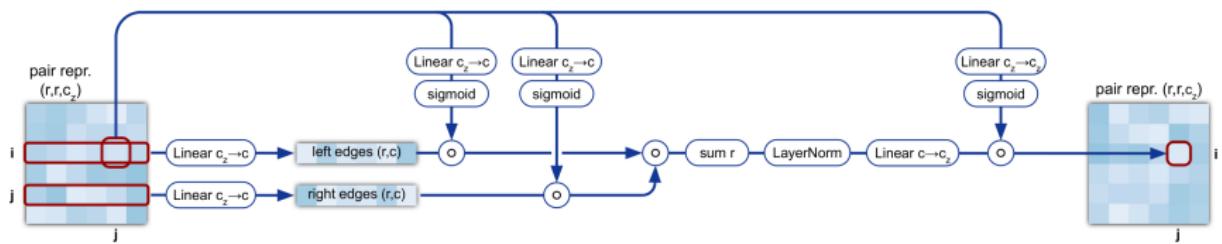
Triangle self-attention around  
starting node



Triangle self-attention around  
ending node



# EvoFormer: Triangular Multiplicative Update [1]



Algorithm 11 Triangular Multiplicative Update: outward [1]

---

**Algorithm 11** Triangular multiplicative update u

---

**def** TriangleMultiplicationOutgoing( $\{\mathbf{z}_{ij}\}$ ,  $c$ ) =

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

2:  $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}(\text{Linear}(\mathbf{z}_{ij})) \odot \text{Linear}(\mathbf{z}_{ij})$

3:  $\mathbf{g}_{ij} = \text{sigmoid}(\text{Linear}(\mathbf{z}_{ij}))$

4:  $\tilde{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ik} \odot \mathbf{b}_{ik}))$

5: **return**  $\{\tilde{\mathbf{z}}_{ij}\}$

## Algorithm 12 Triangular Multiplicative Update: inward [1]

---

### Algorithm 12 Triangular multiplicative update using

---

**def** TriangleMultiplicationIncoming( $\{\mathbf{z}_{ij}\}$ ,  $c = 12$ )

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

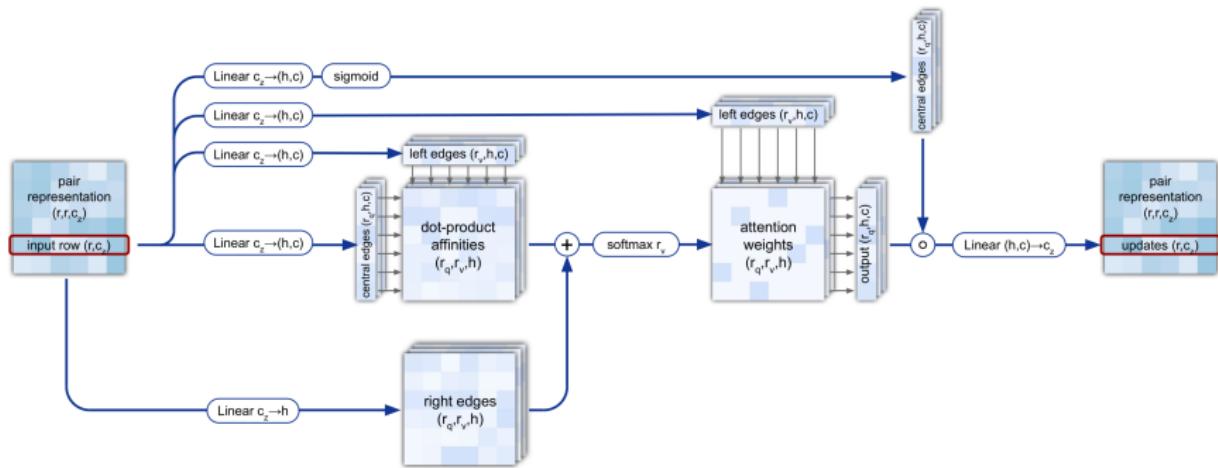
2:  $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}(\text{Linear}(\mathbf{z}_{ij})) \odot \text{Linear}(\mathbf{z}_{ij})$

3:  $\mathbf{g}_{ij} = \text{sigmoid}(\text{Linear}(\mathbf{z}_{ij}))$

4:  $\tilde{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ki} \odot \mathbf{b}_k))$

5: **return**  $\{\tilde{\mathbf{z}}_{ij}\}$

# EvoFormer: Triangular Self-Attention [1]



# Algorithm 13 Triangular Self-Attention: start [1]

---

**Algorithm 13** Triangular gated self-attention around starting node

---

**def** TriangleAttentionStartingNode( $\{\mathbf{z}_{ij}\}$ ,  $c = 32$ ,  $N_{\text{head}} = 4$ ) :

*# Input projections*

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

2:  $\mathbf{q}_{ij}^h, \mathbf{k}_{ij}^h, \mathbf{v}_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$

$\mathbf{q}_{ij}^h, \mathbf{k}_{ij}^h, \mathbf{v}_{ij}^h \in \mathbb{R}^c$ ,  $h \in \{1, \dots, N_{\text{head}}\}$

3:  $b_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$

4:  $\mathbf{g}_{ij}^h = \text{sigmoid}(\text{Linear}(\mathbf{z}_{ij}))$

*# Attention*

5:  $a_{ijk}^h = \text{softmax}_k \left( \frac{1}{\sqrt{c}} \mathbf{q}_{ij}^{h\top} \mathbf{k}_{ik}^h + b_{jk}^h \right)$

6:  $\mathbf{o}_{ij}^h = \mathbf{g}_{ij}^h \odot \sum_k a_{ijk}^h \mathbf{v}_{ik}^h$

*# Output projection*

7:  $\tilde{\mathbf{z}}_{ij} = \text{Linear} \left( \text{concat}_h(\mathbf{o}_{ij}^h) \right)$

## Algorithm 14 Triangular Self-Attention: end [1]

**Algorithm 14** Triangular gated self-attention around ending node

**def** TriangleAttentionEndingNode( $\{\mathbf{z}_{ij}\}$ ,  $c = 32$ ,  $N_{\text{head}} = 4$ ) :

# Input projections

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

2:  $\mathbf{q}_{ij}^h, \mathbf{k}_{ij}^h, \mathbf{v}_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$   $\mathbf{q}_{ij}^h, \mathbf{k}_{ij}^h, \mathbf{v}_{ij}^h \in \mathbb{R}^c$ ,  $h \in \{1, \dots, N_{\text{head}}\}$

3:  $b_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$

4:  $\mathbf{g}_{ij}^h = \text{sigmoid}(\text{Linear}(\mathbf{z}_{ij}))$   $\mathbf{g}_{ij}^h \in \mathbb{R}$

# Attention

5:  $a_{ijk}^h = \text{softmax}_k \left( \frac{1}{\sqrt{c}} \mathbf{q}_{ij}^{h\top} \mathbf{k}_{kj}^h + b_{ki}^h \right)$

6:  $\mathbf{o}_{ij}^h = \mathbf{g}_{ij}^h \odot \sum_k a_{ijk}^h \mathbf{v}_{kj}^h$

# Output projection

7:  $\tilde{\mathbf{z}}_{ij} = \text{Linear} \left( \text{concat}_h (\mathbf{o}_{ij}^h) \right)$   $\tilde{\mathbf{z}}_{ij} \in \mathbb{R}^{c_z}$

Algorithm 15 Transition layer in the pair stack [1]

---

## Algorithm 15 Transition layer in the pair

---

**def** PairTransition( $\{\mathbf{z}_{ij}\}$ ,  $n = 4$ ) :

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

2:  $\mathbf{a}_{ij} = \text{Linear}(\mathbf{z}_{ij})$

3:  $\mathbf{z}_{ij} \leftarrow \text{Linear}(\text{relu}(\mathbf{a}_{ij}))$

4: **return**  $\{\mathbf{z}_{ij}\}$

## Algorithm 16 Template pair stack [1]

---

### Algorithm 16 Template pair stack

---

**def** TemplatePairStack( $\{\mathbf{t}_{ij}\}$ ,  $N_{\text{block}} = 2$ ) :

1: **for all**  $l \in [1, \dots, N_{\text{block}}]$  **do**

2:    $\{\mathbf{t}_{ij}\} += \text{DropoutRowwise}_{0.25}(\text{TriangleAttentionStarting})$

3:    $\{\mathbf{t}_{ij}\} += \text{DropoutColumnwise}_{0.25}(\text{TriangleAttentionEnding})$

4:    $\{\mathbf{t}_{ij}\} += \text{DropoutRowwise}_{0.25}(\text{TriangleMultiplicationOutput})$

5:    $\{\mathbf{t}_{ij}\} += \text{DropoutRowwise}_{0.25}(\text{TriangleMultiplicationInput})$

6:    $\{\mathbf{t}_{ij}\} += \text{PairTransition}(\{\mathbf{t}_{ij}\}, n = 2)$

7: **end for**

8: **return** LayerNorm( $\{\mathbf{t}_{ij}\}$ )

## Algorithm 17 Template pointwise attention [1]

---

### Algorithm 17 Template pointwise attention

---

**def** TemplatePointwiseAttention( $\{\mathbf{t}_{stij}\}, \{\mathbf{z}_{ij}\}, c = 64, N_{he}$

1:  $\mathbf{q}_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$

2:  $\mathbf{k}_{stij}^h, \mathbf{v}_{stij}^h = \text{LinearNoBias}(\mathbf{t}_{stij})$

3:  $a_{stij}^h = \text{softmax}_{st} \left( \frac{1}{\sqrt{c}} \mathbf{q}_{ij}^{h\top} \mathbf{k}_{stij}^h \right)$

4:  $\mathbf{o}_{ij}^h = \sum_{st} a_{stij}^h \mathbf{v}_{stij}^h$

5:  $\{\tilde{\mathbf{z}}_{ij}\} = \text{Linear} \left( \text{concat}_h(\{\mathbf{o}_{ij}^h\}) \right)$

6: **return**  $\{\tilde{\mathbf{z}}_{ij}\}$

# Algorithm 18 Extra MSA stack [1]

---

**Algorithm 18** Extra MSA stack

---

```
def ExtraMsaStack({esei}, {zij}, Nblock = 4) :
    1: for all l ∈ [1, ..., Nblock] do
        # MSA stack
        2: {esei} += DropoutRowwise0.15(MSARowAttentionWithPairBias({esei}, {zij}, c = 8))
        3: {esei} += MSAColumn Global Attention({esei})
        4: {esei} += MSATransition({esei})

        # Communication
        5: {zij} += OuterProductMean({esei})

        # Pair stack
        6: {zij} += DropoutRowwise0.25(TriangleMultiplicationOutgoing({zij}))
        7: {zij} += DropoutRowwise0.25(TriangleMultiplicationIncoming({zij}))
        8: {zij} += DropoutRowwise0.25(TriangleAttentionStartingNode({zij}))
        9: {zij} += DropoutColumnwise0.25(TriangleAttentionEndingNode({zij}))
        10: {zij} += PairTransition({zij})

    11: end for
    12: return {zij}
```

# Algorithm 19 MSA global column-wise gated self-attention [1]

---

**Algorithm 19** MSA **global** column-wise gated self-attention

---

**def** MSAColumnGlobalAttention( $\{\mathbf{m}_{si}\}$ ,  $c = 8$ ,  $N_{\text{head}} = 8$ ) :

# Input projections

1:  $\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

2:  $\mathbf{q}_{si}^h, \mathbf{k}_{si}, \mathbf{v}_{si} = \text{LinearNoBias}(\mathbf{m}_{si})$   $\mathbf{q}_{si}^h, \mathbf{k}_{si}, \mathbf{v}_{si} \in \mathbb{R}^c$ ,  $h \in \{1, \dots, N_{\text{head}}\}$

3:  $\mathbf{q}_i^h = \text{mean}_s \mathbf{q}_{si}^h$

4:  $\mathbf{g}_{si}^h = \text{sigmoid}(\text{Linear}(\mathbf{m}_{si}))$   $\mathbf{g}_{si}^h \in \mathbb{R}^c$

# Attention

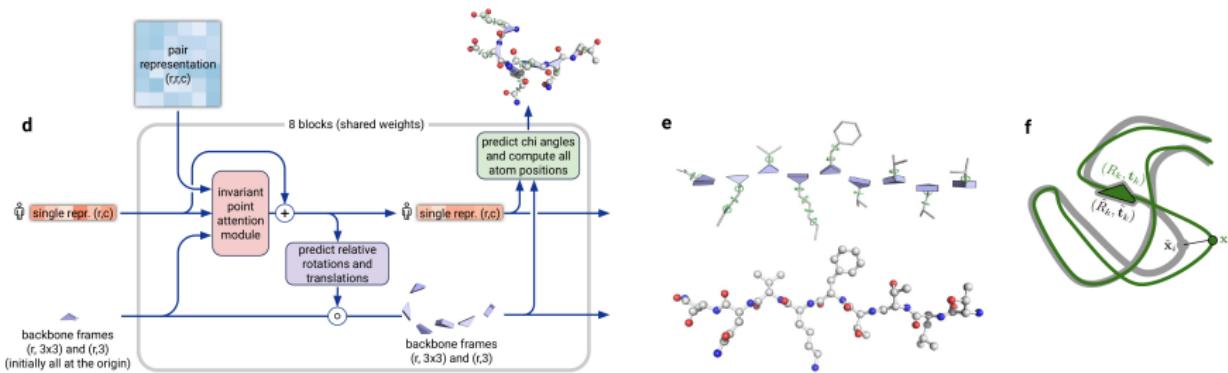
5:  $a_{ti}^h = \text{softmax}_t \left( \frac{1}{\sqrt{c}} \mathbf{q}_i^h \top \mathbf{k}_{ti} \right)$

6:  $\mathbf{o}_{si}^h = \mathbf{g}_{si}^h \odot \sum_t a_{ti}^h \mathbf{v}_{ti}$

# Output projection

7:  $\tilde{\mathbf{m}}_{si} = \text{Linear} \left( \text{concat}_h \left( \mathbf{o}_{si}^h \right) \right)$   $\tilde{\mathbf{m}}_{si} \in \mathbb{R}^c$

# Structure Module: Overview [1]



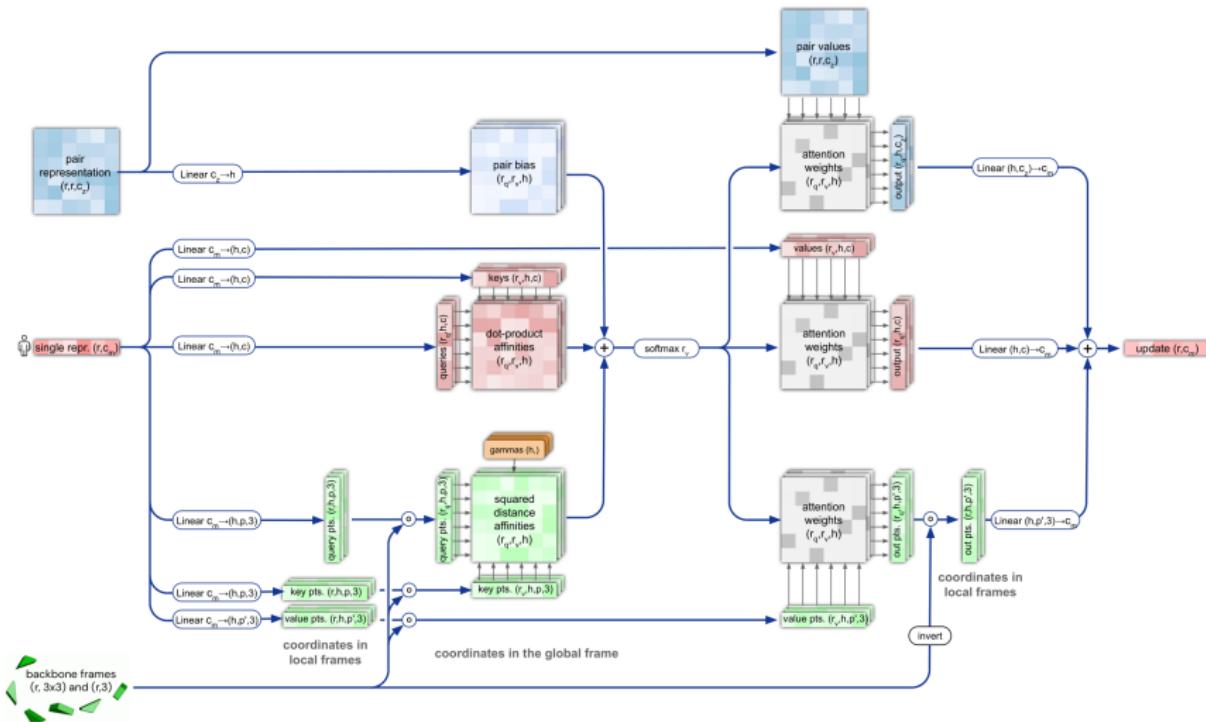
## Structure Module: Frame Representation

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

Example transform cite:SpatialTransformationMatrices

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

# Structure Module: Invariant point attention (IPA) [1]



# Structure Module: Algorithm Part 1 [1]

---

**Algorithm 20** Structure module

---

**def** StructureModule  $\left( \{\mathbf{s}_i^{\text{initial}}\}, \{\mathbf{z}_{ij}\}, N_{\text{layer}} = 8, c = 128, \mathbf{s}_i^{\text{initial}} \in \mathbb{R}^{c_s} \right.$   
$$\left. \{T_i^{\text{true},f}\}, \{T_i^{\text{alt truth},f}\}, \{\vec{\alpha}_i^{\text{true},f}\}, \{\vec{\alpha}_i^{\text{alt truth},f}\}, \{\vec{\mathbf{x}}_i^{\text{true},a}\}, \{\vec{\mathbf{x}}_i^{\text{alt truth},a}\} \right) :$$

- 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:  $\mathbf{s}_i = \text{Linear}(\mathbf{s}_i^{\text{initial}}) \quad \mathbf{s}_i \in \mathbb{R}^{c_s}$
- 4:  $T_i = (\mathbf{I}, \vec{\mathbf{0}}) \quad \mathbf{I} \in \mathbb{R}^{3 \times 3}, \vec{\mathbf{0}} \in \mathbb{R}^3$

# Structure Module: Algorithm Part 2 [1]

```
5: for all  $l \in [1, \dots, N_{\text{layer}}]$  do      # shared weights
6:    $\{\mathbf{s}_i\} += \text{InvariantPointAttention}(\{\mathbf{s}_i\}, \{\mathbf{z}_{ij}\}, \{T_i\})$ 
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:   $\mathcal{L}_{\text{aux}}^l = \left( \text{computeFAPE}(\{T_i\}, \{\vec{\mathbf{x}}_i^{\text{C}\alpha}\}, \{T_i^{\text{true}}\}, \{\vec{\mathbf{x}}_i^{\text{true,C}\alpha}\}, \epsilon = 10^{-12}) \right.$ 
18:     $\left. + \text{torsionAngleLoss}(\{\vec{\alpha}_i^f\}, \{\vec{\alpha}_i^{\text{true},f}\}, \{\vec{\alpha}_i^{\text{alt truth},f}\}) \right)$ 
```

# Structure Module: Algorithm Part 3 [1]

```
23:  $\mathcal{L}_{\text{aux}} = \text{mean}_l(\{\mathcal{L}_{\text{aux}}^l\})$ 
24:  $T_i^f, \vec{x}_i^a = \text{computeAllAtomCoordinates}(T_i, \vec{\alpha}_i^f)$   $a \in \mathcal{S}_{\text{atom names}}$ 
25:  $T_i^f \leftarrow \text{concat}(T_i, T_i^f)$ 

# Final loss on all atom coordinates.
26:  $\{T_i^{\text{true}, f}\}, \{\vec{x}_i^{\text{true}, a}\} \leftarrow \text{renameSymmetricGroundTruthAtoms}($ 
27:  $\{T_i^f\}, \{\vec{x}_i^a\}, \{T_i^{\text{true}, f}\}, \{T_i^{\text{alt truth}, f}\}, \{\vec{x}_i^{\text{true}, a}\}, \{\vec{x}_i^{\text{alt truth}, a}\})$ 
28:  $\mathcal{L}_{\text{FAPE}} = \text{computeFAPE}(\{T_i^f\}, \{\vec{x}_i^a\}, \{T_i^{\text{true}, f}\}, \{\vec{x}_i^{\text{true}, a}\}, \epsilon = 10^{-4})$ 

# Predict model confidence.
29:  $\{r_i^{\text{true LDDT}}\} = \text{perResidueLDDT_Ca}(\{\vec{x}_i^a\}, \{\vec{x}_i^{\text{true}, a}\})$ 
30:  $\{r_i^{\text{pLDDT}}\}, \mathcal{L}_{\text{conf}} = \text{predictPerResidueLDDT_Ca}(\{\mathbf{s}_i\}, \{r_i^{\text{true LDDT}}\})$ 
31: return  $\{\vec{x}_i^a\}, \{r_i^{\text{pLDDT}}\}, \mathcal{L}_{\text{FAPE}}, \mathcal{L}_{\text{conf}}, \mathcal{L}_{\text{aux}}$ 
```

---

# Table 2 Rigid atomic groups from torsion angles [1]

**Table 2** | Rigid groups for constructing all atoms from given torsion angles. Boxes highlight groups that are symmetric under 180° rotations.

aatype	bb	$\psi$	$\chi_1$	$\chi_2$	$\chi_3$	$\chi_4$
ALA	N, C $^\alpha$ , C, C $^\beta$	O	-	-	-	-
ARG	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	C $^\delta$	N $^\epsilon$	N $^{\eta 1}$ , N $^{\eta 2}$ , C $^\zeta$
ASN	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	N $^{\delta 2}$ , O $^{\delta 1}$	-	-
ASP	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	O $^{\delta 1}$ , O $^{\delta 2}$	-	-
CYS	N, C $^\alpha$ , C, C $^\beta$	O	S $^\gamma$	-	-	-
GLN	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	C $^\delta$	N $^{\epsilon 2}$ , O $^{\epsilon 1}$	-
GLU	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	C $^\delta$	O $^{\epsilon 1}$ , O $^{\epsilon 2}$	-
GLY	N, C $^\alpha$ , C	O	-	-	-	-
HIS	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	C $^{\delta 2}$ , N $^{\delta 1}$ , C $^{\epsilon 1}$ , N $^{\epsilon 2}$	-	-
ILE	N, C $^\alpha$ , C, C $^\beta$	O	C $^{\gamma 1}$ , C $^{\gamma 2}$	C $^{\delta 1}$	-	-
LEU	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	C $^{\delta 1}$ , C $^{\delta 2}$	-	-
LYS	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	C $^\delta$	C $^\epsilon$	N $^\zeta$
MET	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	S $^\delta$	C $^\epsilon$	-
PHE	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	C $^{\delta 1}$ , C $^{\delta 2}$ , C $^{\epsilon 1}$ , C $^{\epsilon 2}$ , C $^\zeta$	-	-
PRO	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	C $^\delta$	-	-
SER	N, C $^\alpha$ , C, C $^\beta$	O	O $^\gamma$	-	-	-
THR	N, C $^\alpha$ , C, C $^\beta$	O	C $^{\gamma 2}$ , O $^{\gamma 1}$	-	-	-
TRP	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	C $^{\delta 1}$ , C $^{\delta 2}$ , C $^{\epsilon 2}$ , C $^{\epsilon 3}$ , N $^{\epsilon 1}$ , C $^{\eta 2}$ , C $^{\zeta 2}$ , C $^{\zeta 3}$	-	-
TYR	N, C $^\alpha$ , C, C $^\beta$	O	C $^\gamma$	C $^{\delta 1}$ , C $^{\delta 2}$ , C $^{\epsilon 1}$ , C $^{\epsilon 2}$ , O $^\eta$ , C $^\zeta$	-	-
VAL	N, C $^\alpha$ , C, C $^\beta$	O	C $^{\gamma 1}$ , C $^{\gamma 2}$	-	-	-

## Algorithm 21 Frame construction from ground truth atom positions [1]

---

**Algorithm 21** Rigid from 3 points using the Gram–Schmidt process

---

**def** rigidFrom3Points( $\vec{x}_1, \vec{x}_2, \vec{x}_3$ ) :

1:  $\vec{v}_1 = \vec{x}_3 - \vec{x}_2$

2:  $\vec{v}_2 = \vec{x}_1 - \vec{x}_2$

3:  $\vec{e}_1 = \vec{v}_1 / \|\vec{v}_1\|$

4:  $\vec{u}_2 = \vec{v}_2 - \vec{e}_1 (\vec{e}_1^\top \vec{v}_2)$

5:  $\vec{e}_2 = \vec{u}_2 / \|\vec{u}_2\|$

6:  $\vec{e}_3 = \vec{e}_1 \times \vec{e}_2$

7:  $R = \text{concat}(\vec{e}_1, \vec{e}_2, \vec{e}_3)$

8:  $\vec{t} = \vec{x}_2$

## Algorithm 22 Invariant point attention (IPA) [1]

---

**Algorithm 22** Invariant point attention (IPA)

**def** InvariantPointAttention( $\{\mathbf{s}_i\}, \{\mathbf{z}_{ij}\}, \{T_i\}, N_{\text{head}} = 12, c = 16, N_{\text{query points}} = 4, N_{\text{point values}} = 8$ ) :

- 1:  $\mathbf{q}_i^h, \mathbf{k}_i^h, \mathbf{v}_i^h = \text{LinearNoBias}(\mathbf{s}_i)$   $\mathbf{q}_i^h, \mathbf{k}_i^h, \mathbf{v}_i^h \in \mathbb{R}^c, h \in \{1, \dots, N_{\text{head}}\}$
- 2:  $\vec{\mathbf{q}}_i^{hp}, \vec{\mathbf{k}}_i^{hp} = \text{LinearNoBias}(\mathbf{s}_i)$   $\vec{\mathbf{q}}_i^{hp}, \vec{\mathbf{k}}_i^{hp} \in \mathbb{R}^3, p \in \{1, \dots, N_{\text{query points}}\}, \text{ units: nanometres}$
- 3:  $\vec{\mathbf{v}}_i^{hp} = \text{LinearNoBias}(\mathbf{s}_i)$   $\vec{\mathbf{v}}_i^{hp} \in \mathbb{R}^3, p \in \{1, \dots, N_{\text{point values}}\}, \text{ units: nanometres}$
- 4:  $b_{ij}^h = \text{LinearNoBias}(\mathbf{z}_{ij})$
- 5:  $w_C = \sqrt{\frac{2}{9N_{\text{query points}}}},$
- 6:  $w_L = \sqrt{\frac{1}{3}}$
- 7:  $a_{ij}^h = \text{softmax}_k \left( w_L \left( \frac{1}{\sqrt{c}} \mathbf{q}_i^h \mathbf{k}_j^h + b_{ij}^h - \frac{\gamma^h w_C}{2} \sum_p \|T_i \circ \vec{\mathbf{q}}_i^{hp} - T_j \circ \vec{\mathbf{k}}_j^{hp}\|^2 \right) \right)$
- 8:  $\tilde{\mathbf{o}}_i^h = \sum_j a_{ij}^h \mathbf{z}_{ij}$
- 9:  $\mathbf{o}_i^h = \sum_j a_{ij}^h \mathbf{v}_j^h$
- 10:  $\vec{\mathbf{o}}_i^{hp} = T_i^{-1} \circ \sum_j a_{ij}^h (T_j \circ \vec{\mathbf{v}}_j^{hp})$
- 11:  $\tilde{\mathbf{s}}_i = \text{Linear} \left( \text{concat}_{h,p}(\tilde{\mathbf{o}}_i^h, \mathbf{o}_i^h, \vec{\mathbf{o}}_i^{hp}, \|\vec{\mathbf{o}}_i^{hp}\|) \right)$
- 12: **return**  $\{\tilde{\mathbf{s}}_i\}$

## Algorithm 23 Backbone update [1]

---

**Algorithm 23** Backbone update

---

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

1:  $b_i, c_i, d_i, \vec{\mathbf{t}}_i = \text{Linear}(\mathbf{s}_i)$

$b_i, c_i, d_i \in \mathbb{R}, \vec{\mathbf{t}}_i \in \mathbb{R}^3$

# 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:  $R_i = \begin{pmatrix} a_i^2 + b_i^2 - c_i^2 - d_i^2 & 2b_i c_i - 2a_i d_i & 2b_i d_i + 2a_i c_i \\ 2b_i c_i + 2a_i d_i & a_i^2 - b_i^2 + c_i^2 - d_i^2 & 2c_i d_i - 2a_i b_i \\ 2b_i d_i - 2a_i c_i & 2c_i d_i + 2a_i b_i & a_i^2 - b_i^2 - c_i^2 + d_i^2 \end{pmatrix}$

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

5: **return**  $T_i$

---

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

# Make extra backbone frames.

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

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

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

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

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

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

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

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

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

# Map atom literature positions to the global frame.

11:  $\vec{\mathbf{x}}^a = \text{concat}_{\vec{\mathbf{x}}^f, \vec{\mathbf{x}}^{\text{lit}}} \left( \{T_i^f \circ \vec{\mathbf{x}}^{\text{lit}}_{\cdot, \cdot, \cdot, \cdot}\} \right)$

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

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

4: **return**  $T$

---

# Algorithm 26 Rename symmetric ground truth atoms [1]

---

**Algorithm 26** Rename symmetric ground truth atoms

---

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

- 1:  $d_{(i,a),(j,b)} = \left\| \vec{x}_i^a - \vec{x}_j^b \right\| \quad \forall (j, b) \in \mathcal{S}_{\text{non-ambiguous atoms}}$
- 2:  $d_{(i,a),(j,b)}^{\text{true}} = \left\| \vec{x}_i^{\text{true},a} - \vec{x}_j^{\text{true},b} \right\| \quad \forall (j, b) \in \mathcal{S}_{\text{non-ambiguous atoms}}$
- 3:  $d_{(i,a),(j,b)}^{\text{alt truth}} = \left\| \vec{x}_i^{\text{alt truth},a} - \vec{x}_j^{\text{true},b} \right\| \quad \forall (j, b) \in \mathcal{S}_{\text{non-ambiguous atoms}}$
- 4: **for all**  $i \in [1 \dots N_{\text{res}}]$  **do**
- 5:   **if**  $\text{sum}_{a,(j,b)} \left( \left| d_{(i,a),(j,b)} - d_{(i,a),(j,b)}^{\text{alt truth}} \right| \right) < \text{sum}_{a,(j,b)} \left( \left| d_{(i,a),(j,b)} - d_{(i,a),(j,b)}^{\text{true}} \right| \right)$  **then**
- 6:      $\vec{x}_i^{\text{true},a} \leftarrow \vec{x}_i^{\text{alt truth},a}$
- 7:      $T_i^{\text{true},f} \leftarrow T_i^{\text{alt truth},f}$
- 8:   **end if**
- 9: **end for**
- 10: **return**  $\{T_i^{\text{true},f}\}$ ,  $\{\vec{x}_i^{\text{true},a}\}$

---

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

---

**Algorithm 27** Side chain and backbone torsion angle loss

---

```
def 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 29 Predict model confidence pLDDT [1]

---

**Algorithm 29** Predict model confidence pLDDT

---

```
def predictPerResidueLDDT_Ca({si}, vbins = [1, 3, 5, ..., 99]T, {ritrue LDDT}, c = 128) :
    1: ai = relu(Linear(relu(Linear(LayerNorm(si)))))
        ai, and intermediate activations ∈ Rc
    2: pipLDDT = softmax(Linear(ai))
        pipLDDT ∈ R|vbins|
    3: pitrue LDDT = one_hot(ritrue LDDT, vbins)
    4: Lconf = meani(pitrue LDDTT log pipLDDT)
    5: ripLDDT = pipLDDTT vbins
        ripLDDT ∈ R
    6: return {ripLDDT}, Lconf
```

---

## Algorithm 30 Generic recycling inference procedure [1]

---

**Algorithm 30** Generic recycling inference procedure

---

```
def RecyclingInference( {inputsc} , Ncycle ) :  
    1: outputs = 0  
    # Recycling iterations  
    2: for all c ∈ [1, . . . , Ncycle] do      # shared weights  
    3:     outputs ← Model(inputsc, outputs)  
    4: end for  
    5: return outputs
```

---

# Algorithm 31 Generic recycling training procedure [1]

---

**Algorithm 31** Generic recycling training procedure

---

```
def RecyclingTraining( $\{\text{inputs}_c\}$ ,  $N_{\text{cycle}}$ ) :  
    1:  $N' = \text{uniform}(1, N_{\text{cycle}})$       # shared value across the batch  
    2: outputs = 0  
    # Recycling iterations  
    3: for all  $c \in [1, \dots, N']$  do      # shared weights  
        4: outputs  $\leftarrow \text{stopgrad}(\text{outputs})$       # no gradients between iterations  
        5: outputs  $\leftarrow \text{Model}(\text{inputs}_c, \text{outputs})$   
    6: end for  
    7: return loss(outputs)      # only the final iteration's outputs are used
```

---

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

# *Embed pair distances of backbone atoms:*

1:  $d_{ij} = \left\| \vec{\mathbf{x}}_i^{\text{C}^\beta} - \vec{\mathbf{x}}_j^{\text{C}^\beta} \right\|$   $C^\alpha$  used for glycine

2:  $\mathbf{d}_{ij} = \text{Linear}(\text{one\_hot}(d_{ij}, \mathbf{v}_{\text{bins}} = [3\frac{1}{8}\text{\AA}, 5\frac{1}{8}\text{\AA}, \dots, 21\frac{1}{8}\text{\AA}]))$   $\mathbf{d}_{ij} \in \mathbb{R}^{cz}$

# *Embed output Evoformer representations:*

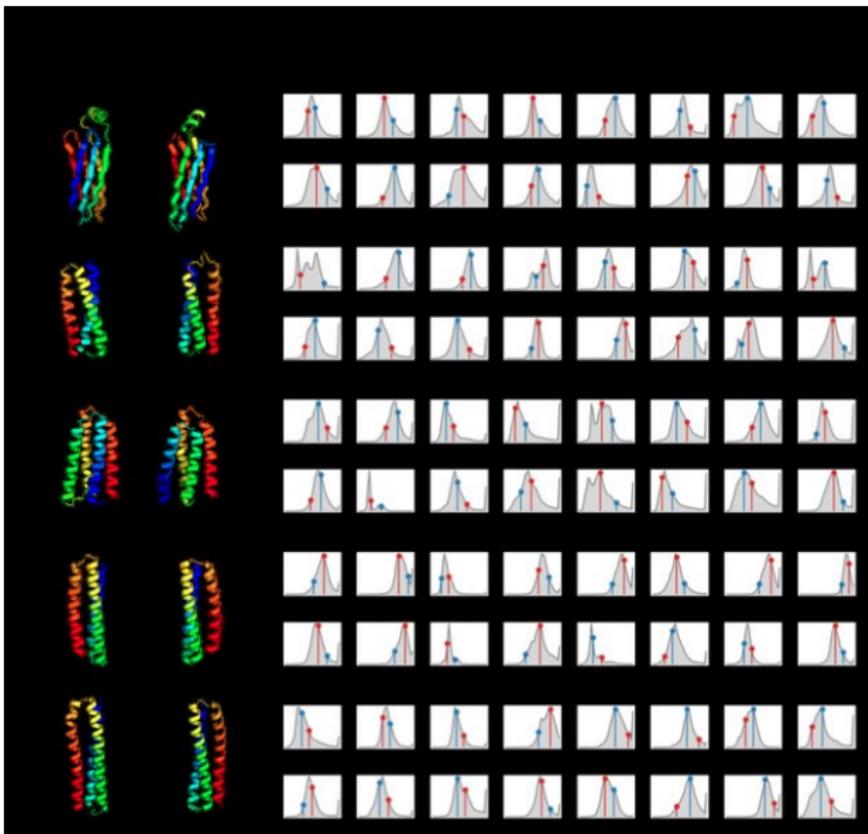
3:  $\tilde{\mathbf{z}}_{ij} = \mathbf{d}_{ij} + \text{LayerNorm}(\mathbf{z}_{ij})$

4:  $\tilde{\mathbf{m}}_{1i} = \text{LayerNorm}(\mathbf{m}_{1i})$

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

---

# Distograms [1]



## Loss Functions [1]

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

- 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  $\text{FAPE}(X,Y) = 0$
  - 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( $\{T_i\}, \{\vec{x}_j\}, \{T_i^{\text{true}}\}, \{\vec{x}_j^{\text{true}}\}, Z = 10\text{\AA}, d_{\text{clamp}} = 10\text{\AA}, \epsilon = 10^{-4}\text{\AA}^2$ ) :

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

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

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

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

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

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

---

Algorithm 28 cite:jumperHighlyAccurateProtein2021

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

## 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

# 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%80%99s-guide-to-pdb-structures-and-the-pdbx-mm cif-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).