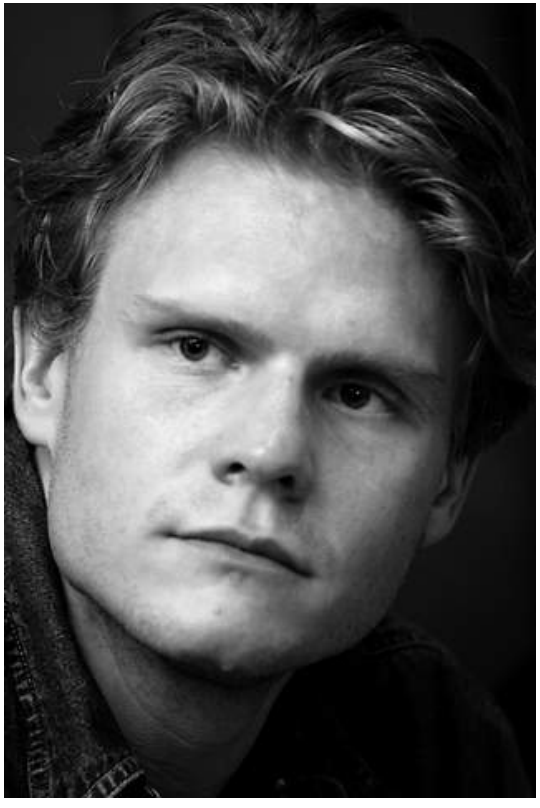
Highly accurate protein structure prediction with AlphaFold

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

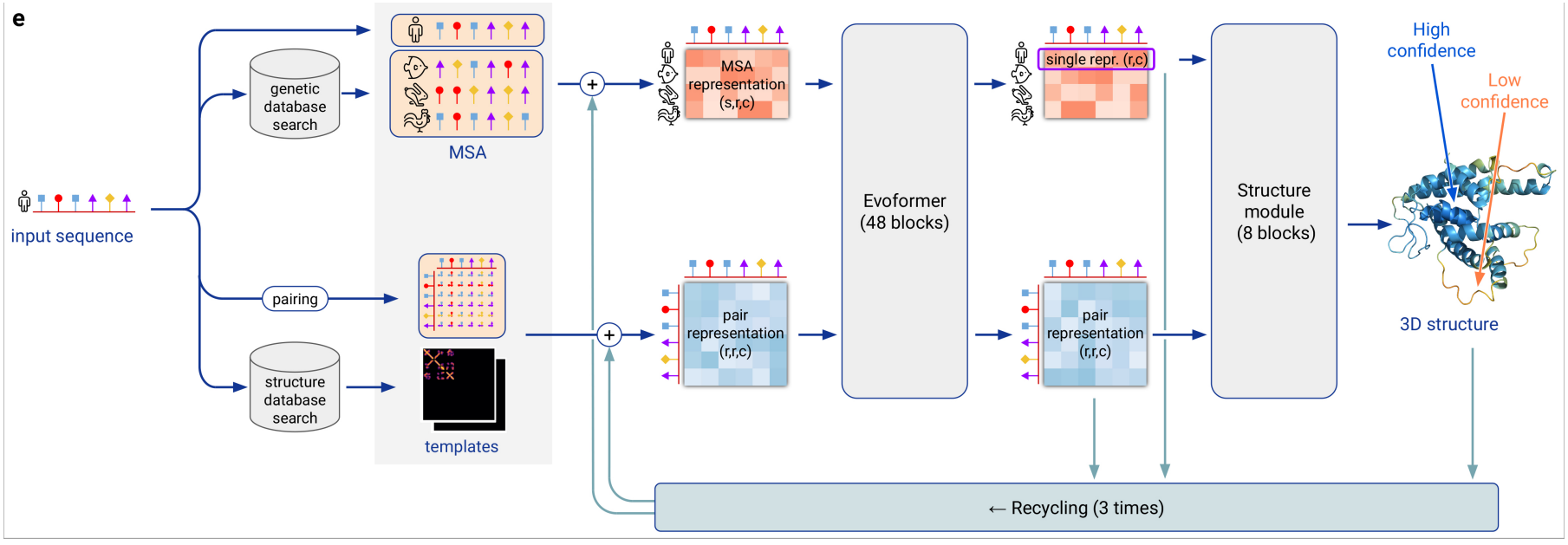
Introduction

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

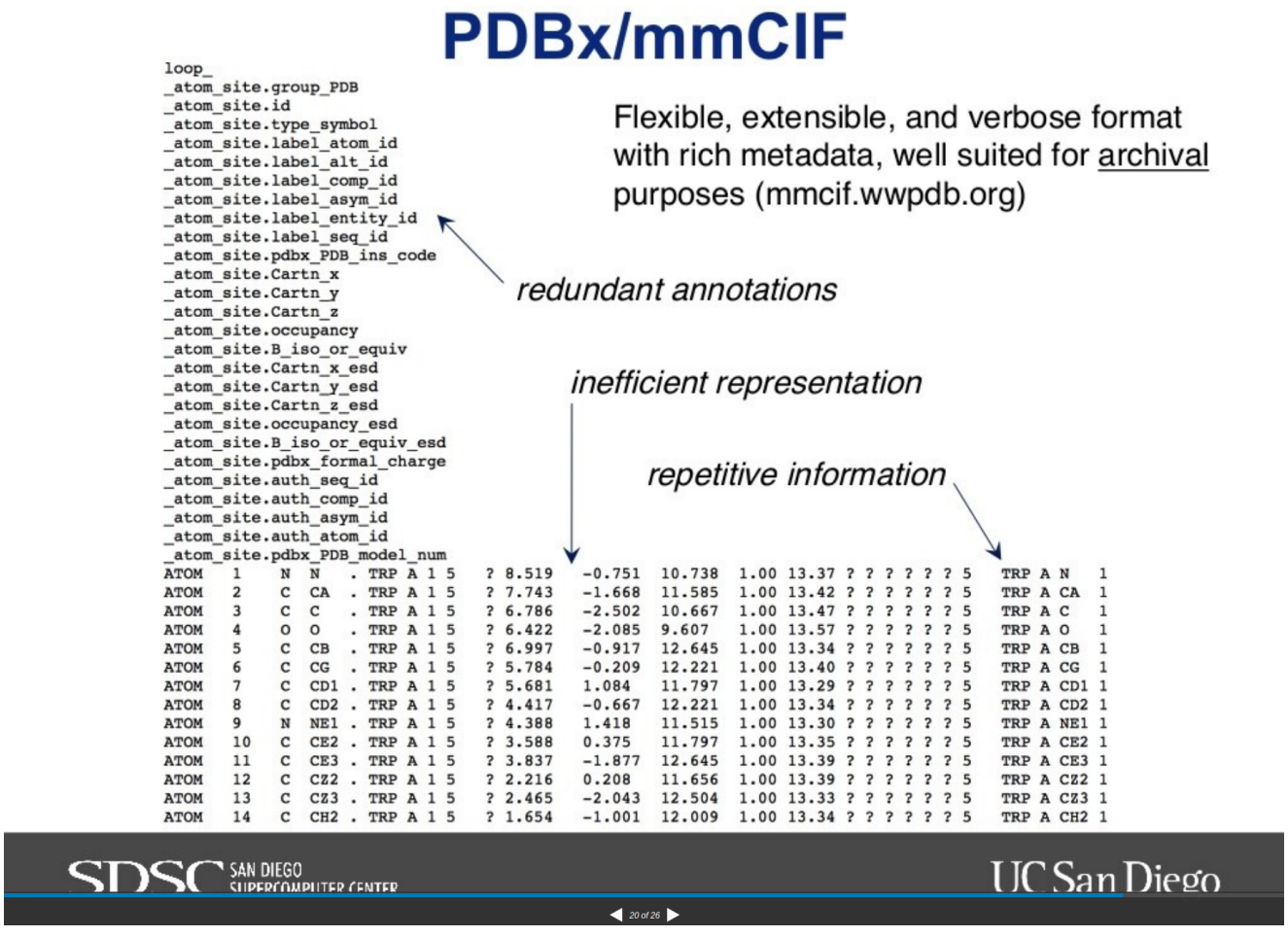


Figure: Example mmCIF file (see [2])

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
```

Figure: Example FASTA file

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$_{max}$ 100000 -maxfilt 100000 -min$_{prefilterhits}$ 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]

- 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]

Algorithm 1 MSA block deletion

def MSABlockDeletion(**msa**) :

1: block_size $= \lfloor 0.3 \cdot N_{\text{all\_seq}} \rfloor$

2: to_delete $= \{\}$

3: **for all** $j \in [1, \ldots, 5]$ **do**

4: block_start $\leftarrow$ uniform$(1, N_{\text{all_seq}})$

5: to_delete $\leftarrow$ to_delete $\cup$ [block_start, $\ldots$, block_start + block_size -1]

6: **end for**

7: keep $\leftarrow [1, \ldots, N_{\text{all\_seq}}] \setminus$ to_delete

8: **msa** $\leftarrow$ **msa**$_{\text{keep}}$

9: **return** **msa**

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

Clustering Algorithm:

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]

- $\text{target}_{\text{feat}}$
  This is a feature of size [Nres, 21] consisting of the "aatype" feature.
- $\text{residue}_{\text{index}}$
  Positional encoding constant tensor. 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 stacking (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 "$\text{template}_{\text{torsionanglesmask}}$".

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

Table 1 | Input features to the model. Feature dimensions: N_{res} is the number of residues, N_{clust} is the number of MSA clusters, $N_{\text{extra\_seq}}$ is the number of additional unclustered MSA sequences, and N_{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 ith sequence and jth 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 subsubsection 1.8.1.
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^o-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 '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 roughly 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 $\Big(\{\mathbf{f}_i^{\text{target\_feat}}\}, \{f_i^{\text{residue\_index}}\}, \Big\{\{\mathbf{f}_{s_c i}^{\text{msa\_feat}}\}_{c,n}\Big\}, \Big\{\{\mathbf{f}_{s_e i}^{\text{extra\_msa\_feat}}\}_{c,n}\Big\},$
$\{\mathbf{f}_{s_t i}^{\text{template\_angle\_feat}}\}, \{\mathbf{f}_{s_t ij}^{\text{template\_pair\_feat}}\}, N_{\text{cycle}} = 4, N_{\text{ensemble}} = 3\Big)$:

 # Recycling iterations:

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

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

 # Average embeddings in ensemble:

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

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

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

$\{\mathbf{m}_{s_c i}\}, \{\mathbf{z}_{ij}\} = \text{InputEmbedder}(\{\mathbf{f}_i^{\text{target\_feat}}\}, \{f_i^{\text{residue\_index}}\}, \{\mathbf{f}_{s_c i}^{\text{msa\_feat}}\}_{c,n})$

 # Inject previous outputs for recycling:

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

 # Embed templates:

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

$\mathbf{m}_{si} = \text{concat}_s(\mathbf{m}_{s_c i}, \mathbf{a}_{s_t i})$

$\mathbf{t}_{s_t ij} = \text{Linear}(\mathbf{f}_{s_t ij}^{\text{template\_pair\_feat}})$ $\quad \mathbf{t}_{s_t ij} \in \mathbb{R}^{c_t}, c_t = 64$

for all $s_t \in [1, \ldots, N_{\text{templ}}]$ **do** *# shared weights*

$\{\mathbf{t}_{s_t ij}\} \leftarrow \text{TemplatePairStack}(\{\mathbf{t}_{s_t ij}\})$

end for

$\{\mathbf{z}_{ij}\}$ += $\text{TemplatePointwiseAttention}(\{\mathbf{t}_{s_t ij}\}, \{\mathbf{z}_{ij}\})$

Algorithm 2 Model Inference [1]

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

$\{\mathbf{a}_{s_e i}\} \leftarrow \{\mathbf{f}_{s_e i}^{\text{extra\_msa\_feat}}\}_{c,n}$

$\mathbf{e}_{s_e i} = \text{Linear}(\mathbf{a}_{s_e i})$ $\quad \mathbf{e}_{s_e i} \in \mathbb{R}^{c_e}, c_e = 64$

$\{\mathbf{z}_{ij}\} \leftarrow \text{ExtraMsaStack}(\{\mathbf{e}_{s_e i}\}, \{\mathbf{z}_{ij}\})$

 # Main trunk of the network:

$\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}, \{\mathbf{s}_i\} \leftarrow \text{EvoformerStack}(\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\})$

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

end for

$\{\hat{\mathbf{m}}_{1i}\}, \{\hat{\mathbf{z}}_{ij}\}, \{\hat{\mathbf{s}}_i\}$ /= N_{ensemble}

 # Structure and confidence prediction:

$\{\vec{\mathbf{x}}_i^a\}, \{r_i^{\text{pLDDT}}\} = \text{StructureModule}(\{\hat{\mathbf{s}}_i\}, \{\hat{\mathbf{z}}_{ij}\})$

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

end for

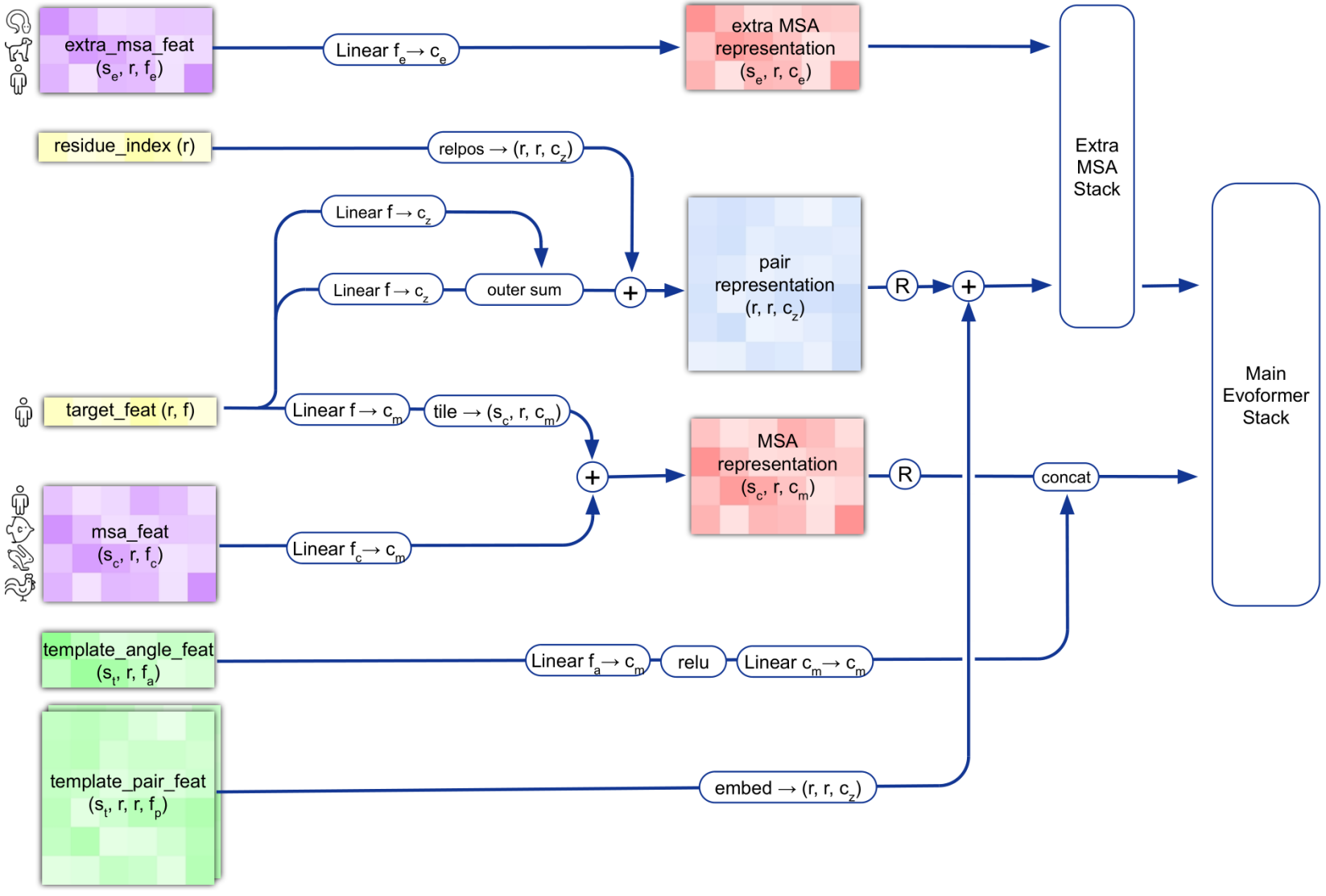
return $\{\vec{\mathbf{x}}_i^a\}, \{r_i^{\text{pLDDT}}\}$

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

Input embeddings [1]



Algorithm 3 Input embeddings [1]

Algorithm 3 Embeddings for initial representations

def InputEmbedder($\{\mathbf{f}_i^{\text{target\_feat}}\}, \{f_i^{\text{residue\_index}}\}, \{\mathbf{f}_{s_c i}^{\text{msa\_feat}}\}$) :

1: $\mathbf{a}_i, \mathbf{b}_i = \text{Linear}(\mathbf{f}_i^{\text{target\_feat}})$ $\quad \mathbf{a}_i, \mathbf{b}_i \in \mathbb{R}^{c_z}, c_z = 128$

2: $\mathbf{z}_{ij} = \mathbf{a}_i + \mathbf{b}_j$ $\quad \mathbf{z}_{ij} \in \mathbb{R}^{c_z}, c_z = 128$

3: $\{\mathbf{z}_{ij}\}$ += $\text{relpos}(\{f_i^{\text{residue\_index}}\})$

4: $\mathbf{m}_{s_c i} = \text{Linear}(\mathbf{f}_{s_c i}^{\text{msa\_feat}}) + \text{Linear}(\mathbf{f}_i^{\text{target\_feat}})$ $\quad \mathbf{m}_{s_c i} \in \mathbb{R}^{c_m}, c_m = 256$

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

Algorithm 4 Relative positional encoding [1]

Algorithm 4 Relative position encoding

def $\text{relpos}(\{f_i^{\text{residue\_index}}\}, \mathbf{v}_{\text{bins}} = [-32, -31, \ldots, 32])$:

1: $d_{ij} = f_i^{\text{residue\_index}} - f_j^{\text{residue\_index}}$ $\quad d_{ij} \in \mathbb{Z}$

2: $\mathbf{p}_{ij} = \text{Linear}(\text{one\_hot}(d_{ij}, \mathbf{v}_{\text{bins}}))$ $\quad \mathbf{p}_{ij} \in \mathbb{R}^{c_z}$

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($x, \mathbf{v}_{\text{bins}}$) : $\quad x \in \mathbb{R}, \quad \mathbf{v}_{\text{bins}} \in \mathbb{R}^{N_{\text{bins}}}$

$\mathbf{p} \in \mathbb{R}^{N_{\text{bins}}}$

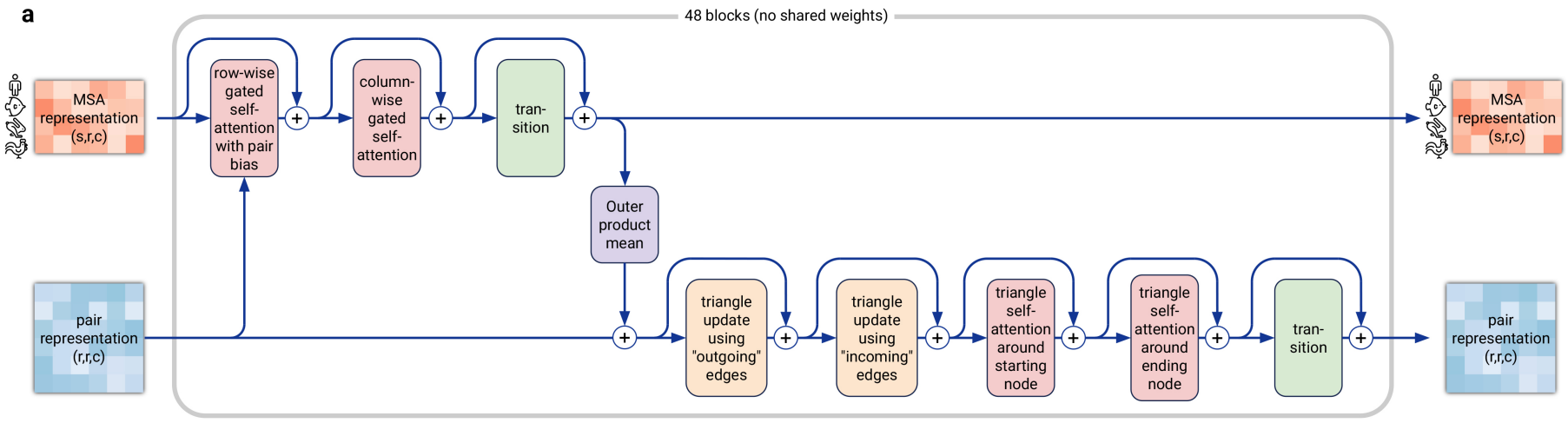
1: $\mathbf{p} = \mathbf{0}$

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($\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}, N_{\text{block}} = 48, c_s = 384$) :

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

 # MSA stack

$\{\mathbf{m}_{si}\}$ += $\text{DropoutRowwise}_{0.15}(\text{MSARowAttentionWithPairBias}(\{\mathbf{m}_{si}\}, \{\mathbf{z}_{ij}\}))$

$\{\mathbf{m}_{si}\}$ += $\text{MSAColumnAttention}(\{\mathbf{m}_{si}\})$

$\{\mathbf{m}_{si}\}$ += $\text{MSATransition}(\{\mathbf{m}_{si}\})$

 # Communication

$\{\mathbf{z}_{ij}\}$ += $\text{OuterProductMean}(\{\mathbf{m}_{si}\})$

 # Pair stack

$\{\mathbf{z}_{ij}\}$ += $\text{DropoutRowwise}_{0.25}(\text{TriangleMultiplicationOutgoing}(\{\mathbf{z}_{ij}\}))$

$\{\mathbf{z}_{ij}\}$ += $\text{DropoutRowwise}_{0.25}(\text{TriangleMultiplicationIncoming}(\{\mathbf{z}_{ij}\}))$

$\{\mathbf{z}_{ij}\}$ += $\text{DropoutRowwise}_{0.25}(\text{TriangleAttentionStartingNode}(\{\mathbf{z}_{ij}\}))$

$\{\mathbf{z}_{ij}\}$ += $\text{DropoutColumnwise}_{0.25}(\text{TriangleAttentionEndingNode}(\{\mathbf{z}_{ij}\}))$

$\{\mathbf{z}_{ij}\}$ += $\text{PairTransition}(\{\mathbf{z}_{ij}\})$

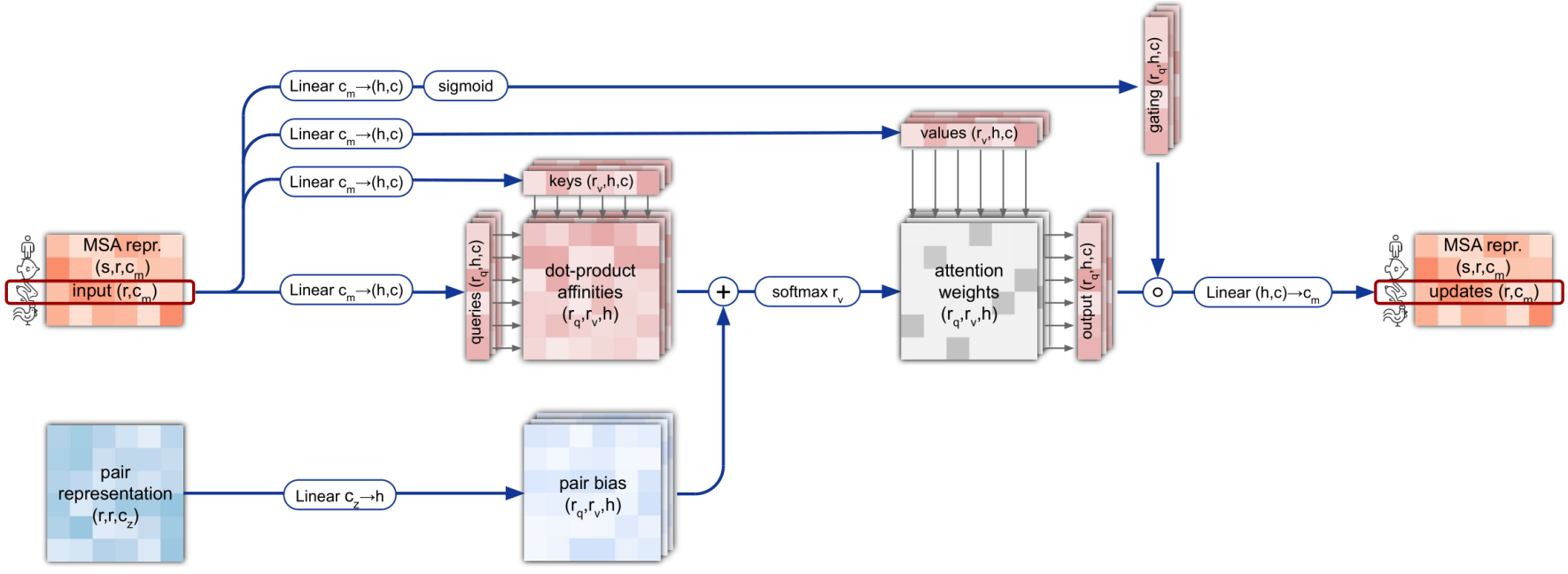
end for

 # Extract the single representation

$\mathbf{s}_i = \text{Linear}(\mathbf{m}_{1i})$ $\quad \mathbf{s}_i \in \mathbb{R}^{c_s}$

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

 # Input projections

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

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

$b^h_{ij} = \text{LinearNoBias}(\text{LayerNorm}(\mathbf{z}_{ij}))$

$\mathbf{g}^h_{si} = \text{sigmoid}\left(\text{Linear}(\mathbf{m}_{si})\right)$ $\qquad \mathbf{g}^h_{si} \in \mathbb{R}^c$

 # Attention

$a^h_{sij} = \text{softmax}_j\left(\frac{1}{\sqrt{c}}\, \mathbf{q}^{h\top}_{si} \mathbf{k}^h_{sj} + b^h_{ij}\right)$

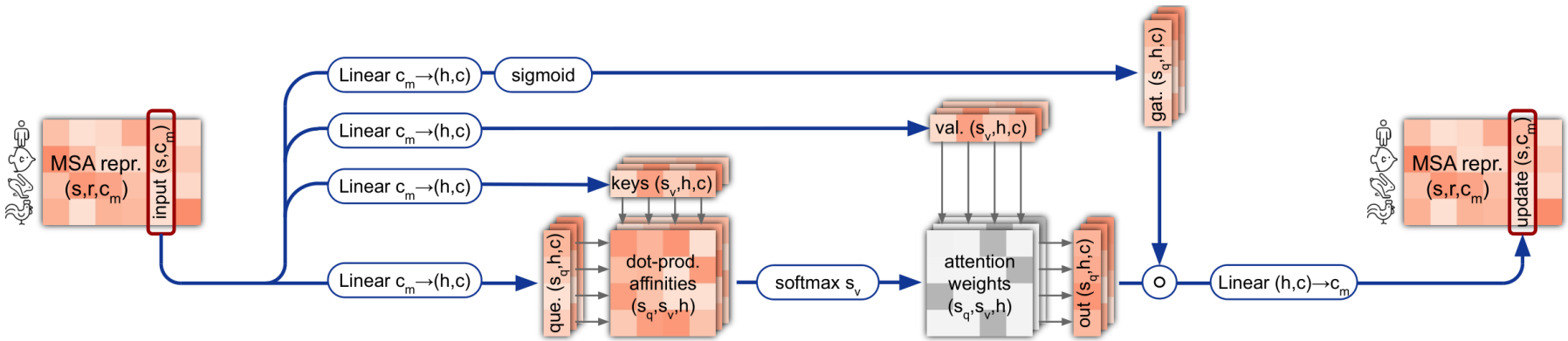
$\mathbf{o}^h_{si} = \mathbf{g}^h_{si} \odot \sum_j a^h_{sij} \mathbf{v}^h_{sj}$

 # Output projection

$\tilde{\mathbf{m}}_{si} = \text{Linear}\left(\text{concat}_h(\mathbf{o}^h_{si})\right)$ $\qquad \tilde{\mathbf{m}}_{si} \in \mathbb{R}^{c_m}$

return $\{\tilde{\mathbf{m}}_{si}\}$

EvoFormer: Column wise Gated Attention [1]



Algorithm 8 Column wise Gated Attention [1]

Algorithm 8 MSA column-wise gated self-attention

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

 # Input projections

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

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

$\mathbf{g}^h_{si} = \text{sigmoid}\left(\text{Linear}(\mathbf{m}_{si})\right)$ $\qquad \mathbf{g}^h_{si} \in \mathbb{R}^c$

 # Attention

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

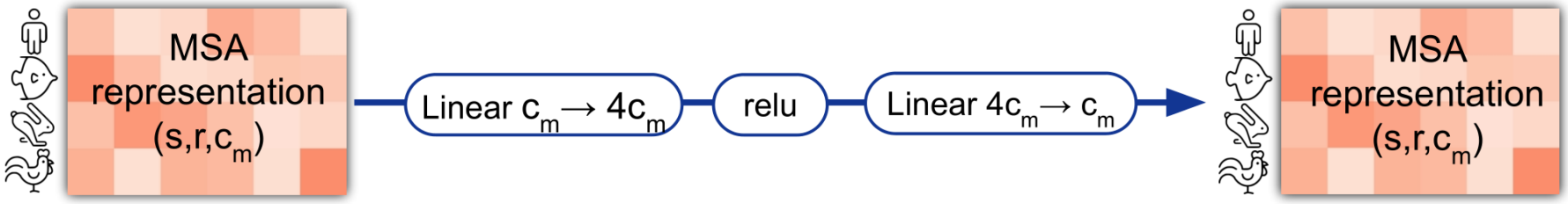
$\mathbf{o}^h_{si} = \mathbf{g}^h_{si} \odot \sum_t a^h_{sti} \mathbf{v}^h_{st}$

 # Output projection

$\tilde{\mathbf{m}}_{si} = \text{Linear}\left(\text{concat}_h(\mathbf{o}^h_{si})\right)$ $\qquad \tilde{\mathbf{m}}_{si} \in \mathbb{R}^{c_m}$

return $\{\tilde{\mathbf{m}}_{si}\}$

EvoFormer: MSA Translation Layer [1]



Algorithm 9 MSA Translation Layer [1]

Algorithm 9 Transition layer in the MSA stack

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

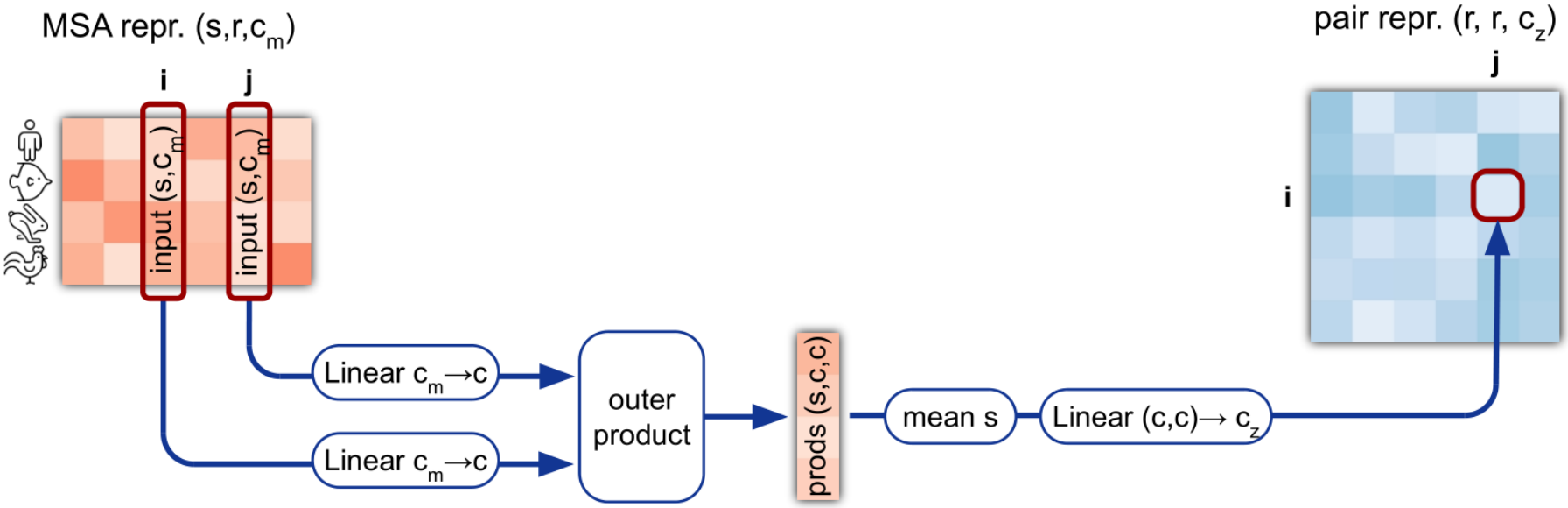
1: $\mathbf{m}_{si} \leftarrow \text{LayerNorm}(\mathbf{m}_{si})$

2: $\mathbf{a}_{si} = \text{Linear}(\mathbf{m}_{si})$ $\mathbf{a}_{si} \in \mathbb{R}^{n \cdot c_m}$

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

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

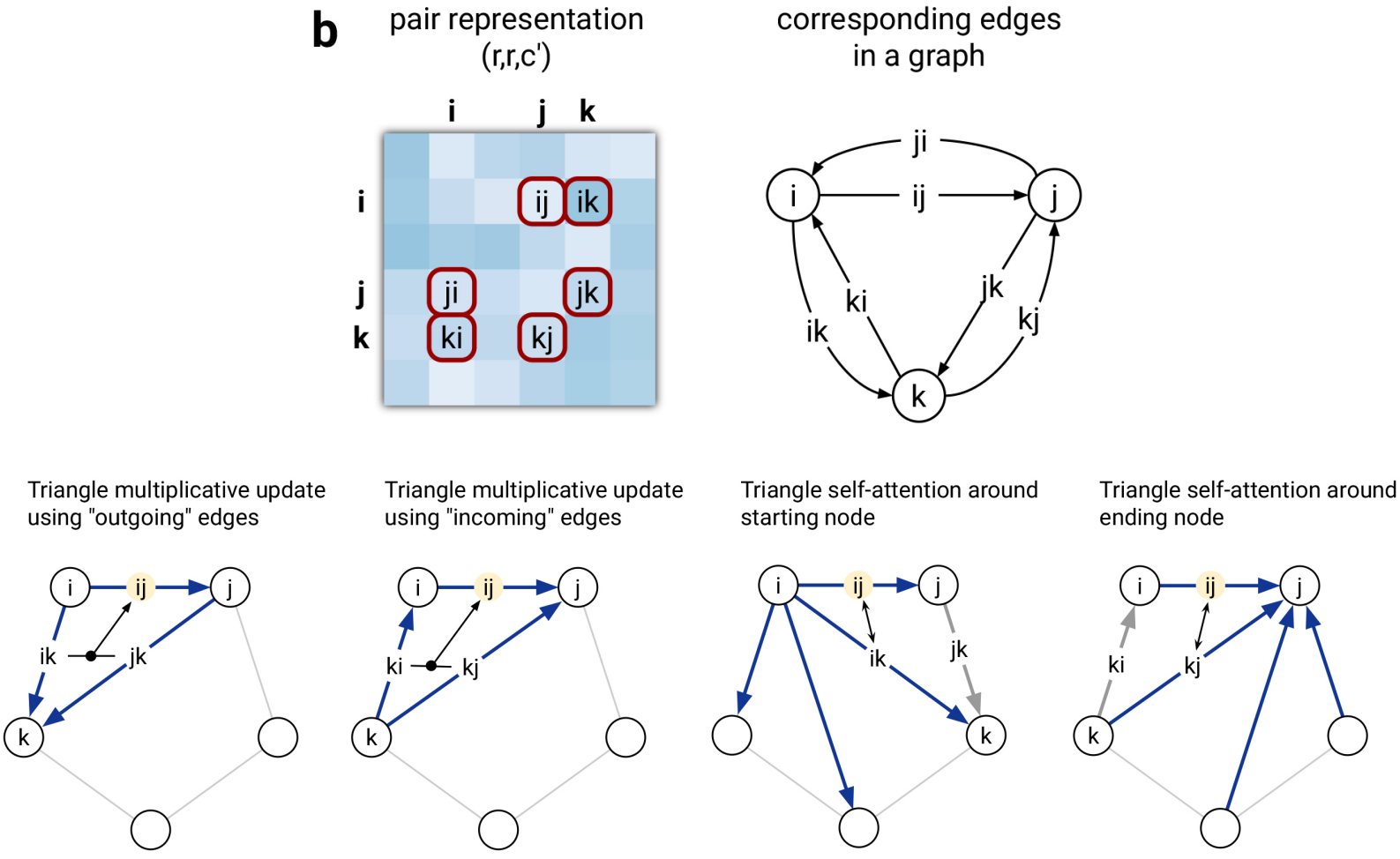
2: $\mathbf{a}_{si}, \mathbf{b}_{si} = \text{Linear}(\mathbf{m}_{si})$ $\mathbf{a}_{si}, \mathbf{b}_{si} \in \mathbb{R}^{c}$

3: $\mathbf{o}_{ij} = \text{flatten}\left(\text{mean}_s(\mathbf{a}_{si} \otimes \mathbf{b}_{sj})\right)$ $\mathbf{o}_{ij} \in \mathbb{R}^{c \cdot c}$

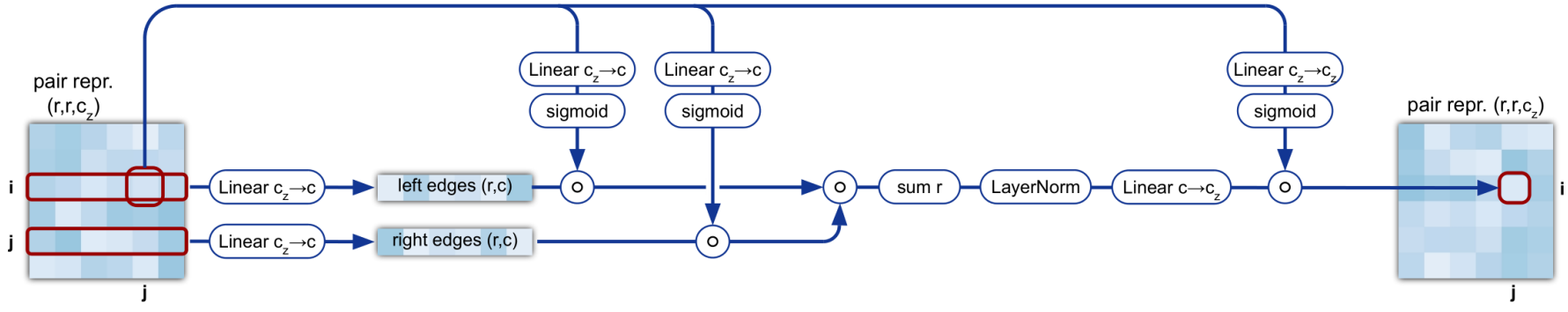
4: $\mathbf{z}_{ij} = \text{Linear}(\mathbf{o}_{ij})$ $\mathbf{z}_{ij} \in \mathbb{R}^{c_z}$

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

EvoFormer: Residue Pairs [1]



EvoFormer: Triangular Multiplicative Update [1]



Algorithm 11 Triangular Multiplicative Update: outward [1]

Algorithm 11 Triangular multiplicative update using "outgoing" edges

def TriangleMultiplicationOutgoing($\{\mathbf{z}_{ij}\}, c = 128$) :

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

2: $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$ $\quad \mathbf{a}_{ij}, \mathbf{b}_{ij} \in \mathbb{R}^{c}$

3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$ $\quad \mathbf{g}_{ij} \in \mathbb{R}^{c_z}$

4: $\tilde{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ik} \odot \mathbf{b}_{jk}))$ $\quad \tilde{\mathbf{z}}_{ij} \in \mathbb{R}^{c_z}$

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

Algorithm 12 Triangular Multiplicative Update: inward [1]

Algorithm 12 Triangular multiplicative update using "incoming" edges

def TriangleMultiplicationIncoming($\{\mathbf{z}_{ij}\}, c = 128$) :

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

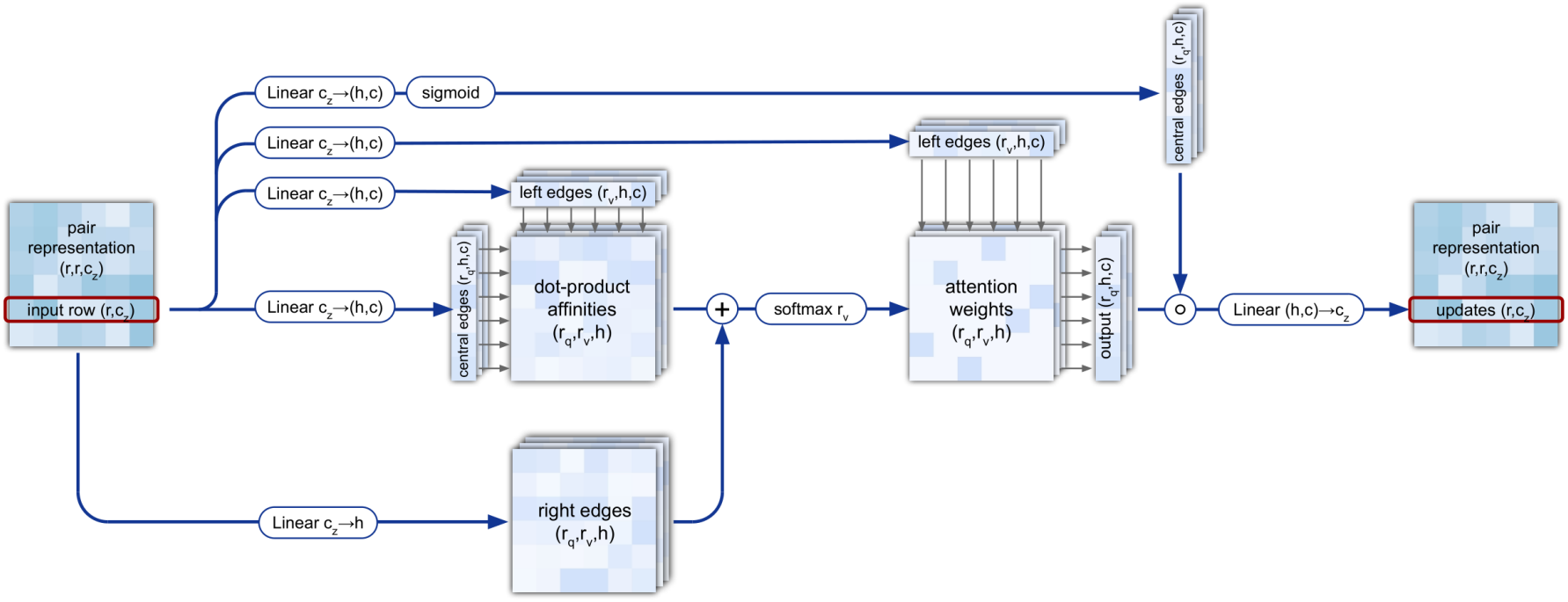
2: $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$ $\quad \mathbf{a}_{ij}, \mathbf{b}_{ij} \in \mathbb{R}^{c}$

3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$ $\quad \mathbf{g}_{ij} \in \mathbb{R}^{c_z}$

4: $\tilde{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ki} \odot \mathbf{b}_{kj}))$ $\quad \tilde{\mathbf{z}}_{ij} \in \mathbb{R}^{c_z}$

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

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

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

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

$\mathbf{g}_{ij}^h = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$ $\qquad \mathbf{g}_{ij}^h \in \mathbb{R}^c$

 # Attention

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

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

 # Output projection

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

return $\{\tilde{\mathbf{z}}_{ij}\}$

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

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

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

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

$\mathbf{g}_{ij}^h = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$ $\quad \mathbf{g}_{ij}^h \in \mathbb{R}^c$

 # Attention

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

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

 # Output projection

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

return $\{\tilde{\mathbf{z}}_{ij}\}$

Algorithm 15 Transition layer in the pair stack [1]

Algorithm 15 Transition layer in the pair stack

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

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

2: $\mathbf{a}_{ij} = \text{Linear}(\mathbf{z}_{ij})$ $\mathbf{a}_{ij} \in \mathbb{R}^{n \cdot c_z}$

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

for all $l \in [1, \ldots, N_{\text{block}}]$ **do**
  $\{\mathbf{t}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleAttentionStartingNode($\{\mathbf{t}_{ij}\}, c = 64, N_{\text{head}} = 4$))
  $\{\mathbf{t}_{ij}\}$ += DropoutColumnwise$_{0.25}$(TriangleAttentionEndingNode($\{\mathbf{t}_{ij}\}, c = 64, N_{\text{head}} = 4$))
  $\{\mathbf{t}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleMultiplicationOutgoing($\{\mathbf{t}_{ij}\}, c = 64$))
  $\{\mathbf{t}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleMultiplicationIncoming($\{\mathbf{t}_{ij}\}, c = 64$))
  $\{\mathbf{t}_{ij}\}$ += PairTransition($\{\mathbf{t}_{ij}\}, n = 2$)
end for
return LayerNorm $\left(\{\mathbf{t}_{ij}\}\right)$

Algorithm 17 Template pointwise attention [1]

Algorithm 17 Template pointwise attention

def $\text{TemplatePointwiseAttention}(\{\mathbf{t}_{s_t ij}\}, \{\mathbf{z}_{ij}\}, c = 64, N_{\text{head}} = 4)$:

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

2: $\mathbf{k}_{s_t ij}^h, \mathbf{v}_{s_t ij}^h = \text{LinearNoBias}(\mathbf{t}_{s_t ij})$ $\qquad \mathbf{k}_{s_t ij}^h, \mathbf{v}_{s_t ij}^h \in \mathbb{R}^c$

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

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

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

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

Algorithm 18 Extra MSA stack [1]

Algorithm 18 Extra MSA stack

def ExtraMsaStack($\{\mathbf{e}_{s_e i}\}, \{\mathbf{z}_{ij}\}, N_{\text{block}} = 4$) :

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

 # MSA stack

$\{\mathbf{e}_{s_e i}\}$ += DropoutRowwise$_{0.15}$(MSARowAttentionWithPairBias($\{\mathbf{e}_{s_e i}\}, \{\mathbf{z}_{ij}\}, c = 8$))

$\{\mathbf{e}_{s_e i}\}$ += MSAColumnGlobalAttention($\{\mathbf{e}_{s_e i}\}$

$\{\mathbf{e}_{s_e i}\}$ += MSATransition($\{\mathbf{e}_{s_e i}\}$)

 # Communication

$\{\mathbf{z}_{ij}\}$ += OuterProductMean($\{\mathbf{e}_{s_e i}\}$)

 # Pair stack

$\{\mathbf{z}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleMultiplicationOutgoing($\{\mathbf{z}_{ij}\}$))

$\{\mathbf{z}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleMultiplicationIncoming($\{\mathbf{z}_{ij}\}$))

$\{\mathbf{z}_{ij}\}$ += DropoutRowwise$_{0.25}$(TriangleAttentionStartingNode($\{\mathbf{z}_{ij}\}$))

$\{\mathbf{z}_{ij}\}$ += DropoutColumnwise$_{0.25}$(TriangleAttentionEndingNode($\{\mathbf{z}_{ij}\}$))

$\{\mathbf{z}_{ij}\}$ += PairTransition($\{\mathbf{z}_{ij}\}$)

end for

return $\{\mathbf{z}_{ij}\}$

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

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

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

$\mathbf{q}^h_i = \text{mean}_s\, \mathbf{q}^h_{si}$

$\mathbf{g}^h_{si} = \text{sigmoid}\left(\text{Linear}(\mathbf{m}_{si})\right)$ $\quad \mathbf{g}^h_{si} \in \mathbb{R}^c$

 # Attention

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

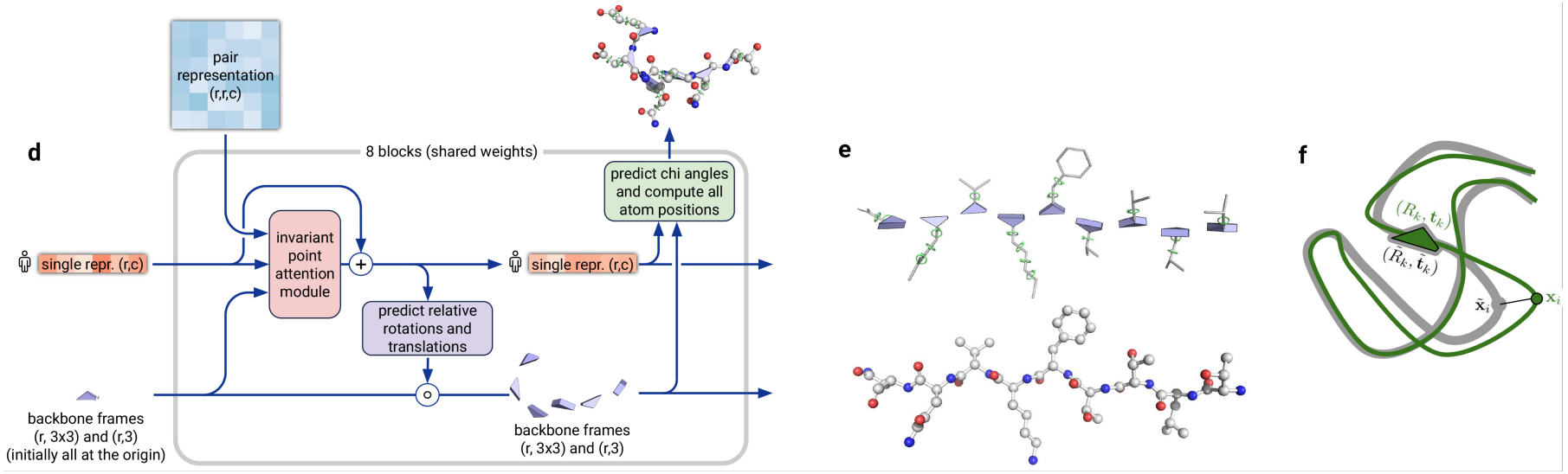
$\mathbf{o}^h_{si} = \mathbf{g}^h_{si} \odot \sum_t a^h_{ti} \mathbf{v}_{ti}$

 # Output projection

$\tilde{\mathbf{m}}_{si} = \text{Linear}\left(\text{concat}_h\left(\mathbf{o}^h_{si}\right)\right)$ $\quad \tilde{\mathbf{m}}_{si} \in \mathbb{R}^{c_m}$

return $\{\tilde{\mathbf{m}}_{si}\}$

Structure Module: Overview [1]



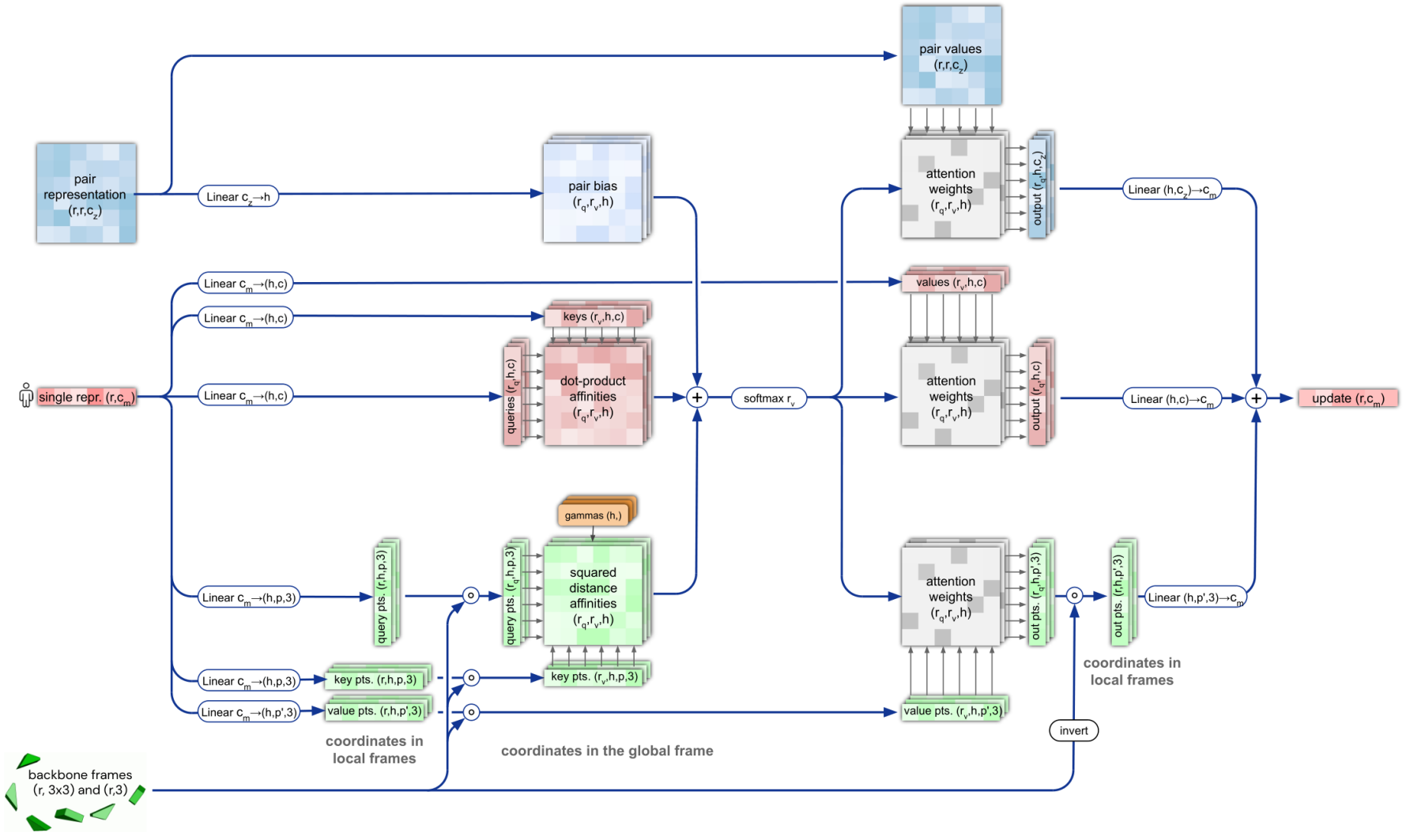
Structure Module: Frame Representation

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

Figure: Example transform [5]

- rotation + translation transforms $T_i := (R_i, t_i)$
- no reflection, scaling, or shear
- they construct ground truth frames using the position of three atoms from the ground truth PDB structures using a Gram–Schmidt process (Algorithm 21)

Structure Module: Invariant point attention (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]

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

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

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

 # Transition.

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

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

 # Update backbone.

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

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

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

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

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

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

 # Auxiliary losses in every iteration.

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

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

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

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

 # No rotation gradients between iterations to stabilize training.

if $l < N_{\text{layer}}$ **then**

$T_i \leftarrow (\text{stopgrad}(R_i), \vec{\mathbf{t}}_i)$

end if

end for

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

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	ψ	χ_1	χ_2	χ_3	χ_4
ALA	N, C^{α}, C, C^{β}	O	-	-	-	-
ARG	N, C^{α}, C, C^{β}	O	C^{γ}	C^{δ}	N^{ϵ}	$N^{\eta 1}$, $N^{\eta 2}$, C^{ζ}
ASN	N, C^{α}, C, C^{β}	O	C^{γ}	$N^{\delta 2}$, $O^{\delta 1}$	-	-
ASP	N, C^{α}, C, C^{β}	O	C^{γ}	$\boxed{O^{\delta 1}, O^{\delta 2}}$	-	-
CYS	N, C^{α}, C, C^{β}	O	S^{γ}	-	-	-
GLN	N, C^{α}, C, C^{β}	O	C^{γ}	C^{δ}	$N^{\epsilon 2}$, $O^{\epsilon 1}$	-
GLU	N, C^{α}, C, C^{β}	O	C^{γ}	C^{δ}	$\boxed{O^{\epsilon 1}, O^{\epsilon 2}}$	-
GLY	N, C^{α}, C	O	-	-	-	-
HIS	N, C^{α}, C, C^{β}	O	C^{γ}	$C^{\delta 2}$, $N^{\delta 1}$, $C^{\epsilon 1}$, $N^{\epsilon 2}$	-	-
ILE	N, C^{α}, C, C^{β}	O	$C^{\gamma 1}$, $C^{\gamma 2}$	$C^{\delta 1}$	-	-
LEU	N, C^{α}, C, C^{β}	O	C^{γ}	$C^{\delta 1}$, $C^{\delta 2}$	-	-
LYS	N, C^{α}, C, C^{β}	O	C^{γ}	C^{δ}	C^{ϵ}	N^{ζ}
MET	N, C^{α}, C, C^{β}	O	C^{γ}	S^{δ}	C^{ϵ}	-
PHE	N, C^{α}, C, C^{β}	O	C^{γ}	$\boxed{C^{\delta 1}, C^{\delta 2}, C^{\epsilon 1}, C^{\epsilon 2}, C^{\zeta}}$	-	-
PRO	N, C^{α}, C, C^{β}	O	C^{γ}	C^{δ}	-	-
SER	N, C^{α}, C, C^{β}	O	O^{γ}	-	-	-
THR	N, C^{α}, C, C^{β}	O	$C^{\gamma 2}$, $O^{\gamma 1}$	-	-	-
TRP	N, C^{α}, C, C^{β}	O	C^{γ}	$C^{\delta 1}$, $C^{\delta 2}$, $C^{\epsilon 2}$, $C^{\epsilon 3}$, $N^{\epsilon 1}$, $C^{\eta 2}$, $C^{\zeta 2}$, $C^{\zeta 3}$	-	-
TYR	N, C^{α}, C, C^{β}	O	C^{γ}	$\boxed{C^{\delta 1}, C^{\delta 2}, C^{\epsilon 1}, C^{\epsilon 2}, O^{\eta}, C^{\zeta}}$	-	-
VAL	N, C^{α}, C, C^{β}	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$(\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \vec{\mathbf{x}}_3)$: $\vec{\mathbf{x}}_1, \vec{\mathbf{x}}_2, \vec{\mathbf{x}}_3 \in \mathbb{R}^3$

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

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

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

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

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

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

7: $R = \text{concat}(\vec{\mathbf{e}}_1, \vec{\mathbf{e}}_2, \vec{\mathbf{e}}_3)$ $R \in \mathbb{R}^{3 \times 3}$

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

9: **return** $(R, \vec{\mathbf{t}})$

Algorithm 22 Invariant point attention (IPA) [1]

Algorithm 22 Invariant point attention (IPA)

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

$\mathbf{q}_i^h, \mathbf{k}_i^h, \mathbf{v}_i^h = \text{LinearNoBias}(\mathbf{s}_i)$ $\qquad \mathbf{q}_i^h, \mathbf{k}_i^h, \mathbf{v}_i^h \in \mathbb{R}^c,\ h \in \{1, \ldots, N_{\text{head}}\}$

$\vec{\mathbf{q}}_i^{hp}, \vec{\mathbf{k}}_i^{hp} = \text{LinearNoBias}(\mathbf{s}_i)$ $\qquad \vec{\mathbf{q}}_i^{hp}, \vec{\mathbf{k}}_i^{hp}, \in \mathbb{R}^3,\ p \in \{1, \ldots, N_{\text{query points}}\},$ units: nanometres

$\vec{\mathbf{v}}_i^{hp} = \text{LinearNoBias}(\mathbf{s}_i)$ $\qquad \vec{\mathbf{v}}_i^{hp} \in \mathbb{R}^3,\ p \in \{1, \ldots, N_{\text{point values}}\},$ units: nanometres

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

$w_C = \sqrt{\frac{2}{9N_{\text{query points}}}},$

$w_L = \sqrt{\frac{1}{3}}$

$a_{ij}^h = \text{softmax}_k \left(w_L \left(\frac{1}{\sqrt{c}}\, \mathbf{q}_i^{h\top} \mathbf{k}_j^h \quad + b_{ij}^h \quad - \frac{\gamma^h w_C}{2} \sum_p \left\| T_i \circ \vec{\mathbf{q}}_i^{hp} - T_j \circ \vec{\mathbf{k}}_j^{hp} \right\|^2 \right) \right)$

$\tilde{\mathbf{o}}_i^h = \sum_j a_{ij}^h \mathbf{z}_{ij}$

$\mathbf{o}_i^h = \sum_j a_{ij}^h \mathbf{v}_j^h$

$\vec{\mathbf{o}}_i^{hp} = T_i^{-1} \circ \sum_j a_{ij}^h \left(T_j \circ \vec{\mathbf{v}}_j^{hp} \right)$

$\tilde{\mathbf{s}}_i = \text{Linear} \left(\text{concat}_{h,p} (\tilde{\mathbf{o}}_i^h, \mathbf{o}_i^h, \vec{\mathbf{o}}_i^{hp}, \left\| \vec{\mathbf{o}}_i^{hp} \right\|) \right)$

return $\{\tilde{\mathbf{s}}_i\}$

Algorithm 23 Backbone update [1]

Algorithm 23 Backbone update

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

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

 # 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 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: $\vec{\mathbf{x}}_i^a = \text{concat}_{f,a'} \left(\{ T_i^f \circ \vec{\mathbf{x}}^{\text{lit}}_{r_i,f,a'} \} \right)$

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

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

Table 3 Ambiguous atom renaming swaps [1]

Table 3 | Ambiguous atom names due to $180°$-rotation-symmetry for some of the rigid groups

aatype	renaming swaps
ASP	$O^{\delta 1} \leftrightarrow O^{\delta 2}$
GLU	$O^{\epsilon 1} \leftrightarrow O^{\epsilon 2}$
PHE	$C^{\delta 1} \leftrightarrow C^{\delta 2}$, $C^{\epsilon 1} \leftrightarrow C^{\epsilon 2}$
TYR	$C^{\delta 1} \leftrightarrow C^{\delta 2}$, $C^{\epsilon 1} \leftrightarrow C^{\epsilon 2}$

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

Distograms [1]

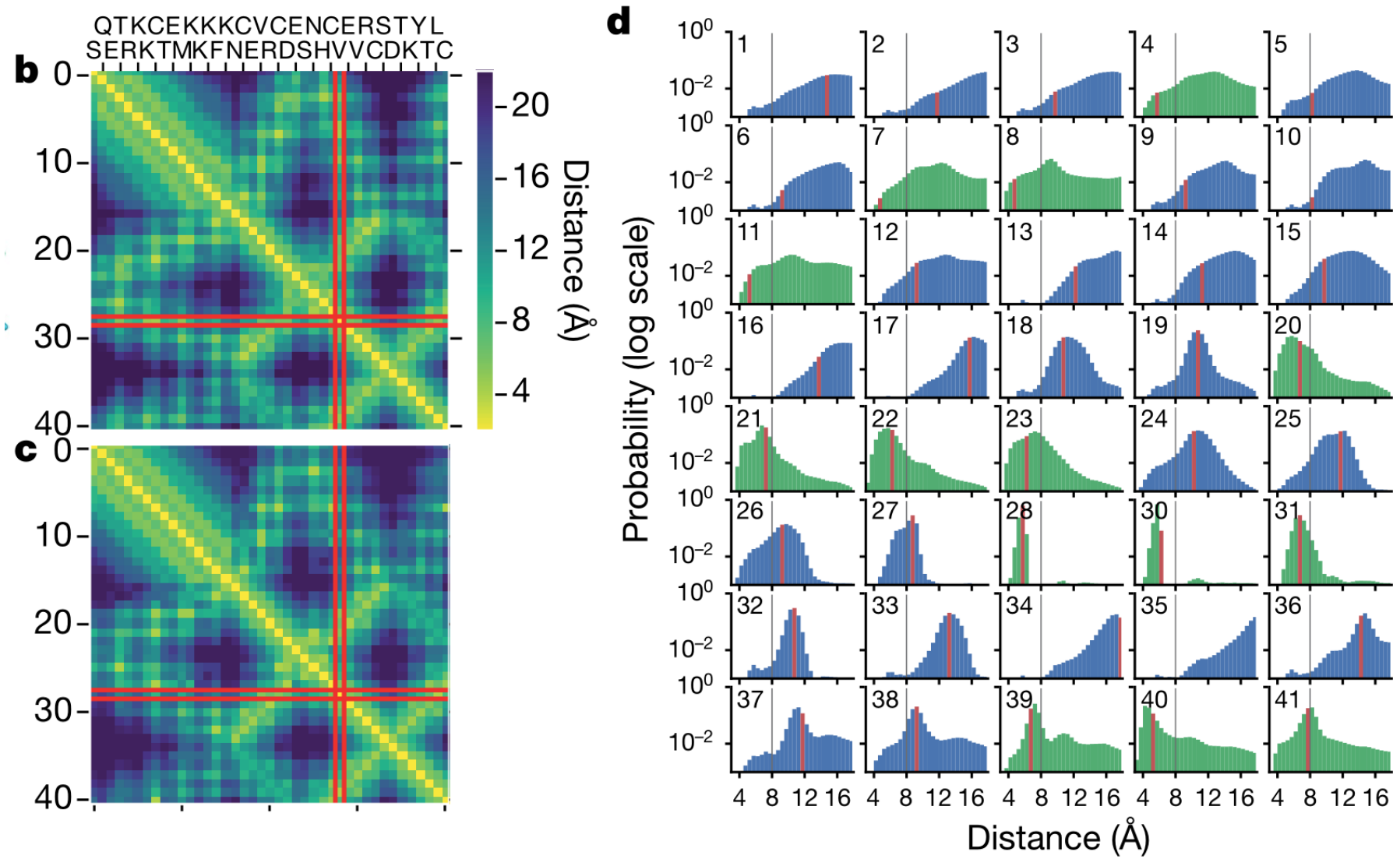


Figure: Example Distogram [4]

Amber Relaxation

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 (sec. 1.9.1)
2. Frame aligned point error (FAPE) (sec. 1.9.2)
   - Configurations with FAPE(X,Y) = 0 (sec. 1.9.4)
   - Metric properties of FAPE (sec. 1.9.5)
3. Chiral properties of AlphaFold and its loss (sec. 1.9.3)
   - transforms T_i are variant under reflection (see eq. 11 to 17)
   - atom positions via backbone frames and χ angles
4. Model confidence prediction (pLDDT) (sec. 1.9.6)
5. TM-score prediction (sec. 1.9.7)
6. Distogram prediction (sec. 1.9.8)
7. Masked MSA prediction (sec. 1.9.9)
8. "Experimentally resolved" prediction (sec. 1.9.10)
9. Structural violations (sec. 1.9.11)

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

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** $\mathcal{L}_{\text{FAPE}}$

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]

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

TM-score prediction [1]

- Global superposition metric of C_α atoms (eqs. 31 - 33)
  1. approximated (eqs. 34-36)
  2. Probabilistic lower-bound maximum-of-expectation score (eqs. 37-38)
  3. approximated TM-score using pairwise C_α based computation (e_{ij} matrix) and above (see eq. 39 and adjacent text)
  4. TM-score of any residue subset D can be computed (eq. 40)
     - can also be used to estimate GDT, FAPE, RMSD (using e_{ij}) matrix (not done)
     - used for confident domain packing visualizations

Distogram prediction [1]

- (eq. 41)

Masked MSA prediction [1]

"Experimentally resolved" prediction (fine tuning) [1]

Structural violations (fine tuning) [1]

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

Training stages [1]

MSA resampling and ensembling [1]

Optimization details [1]

- Adam $lr == 10^{-3}, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-6}$
  - lr warm-up for $0.128\dot{1}0^6$ samples, increase again by $0.65 after 6.4\dot{1}0^6$ samples
- batch: 128
- gradient clipping by global norm (per parameter*sample) of 0.1

Parameters initialization [1]

Loss clamping details [1]

Dropout details [1]

Evaluator setup [1]

Reducing the memory consumption [1]

CASP14 Assessment [1]

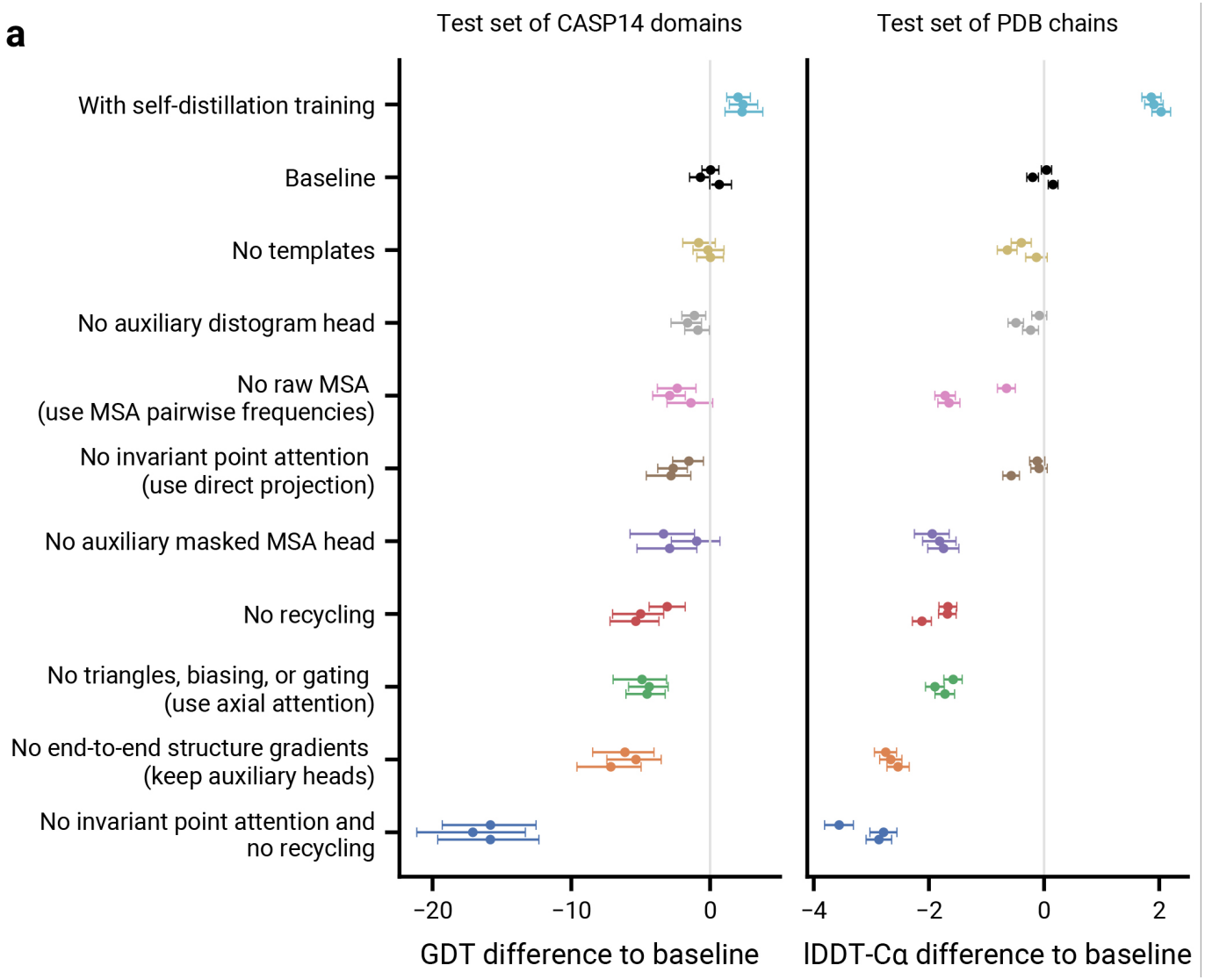
They did well

Ablation Studies [1]

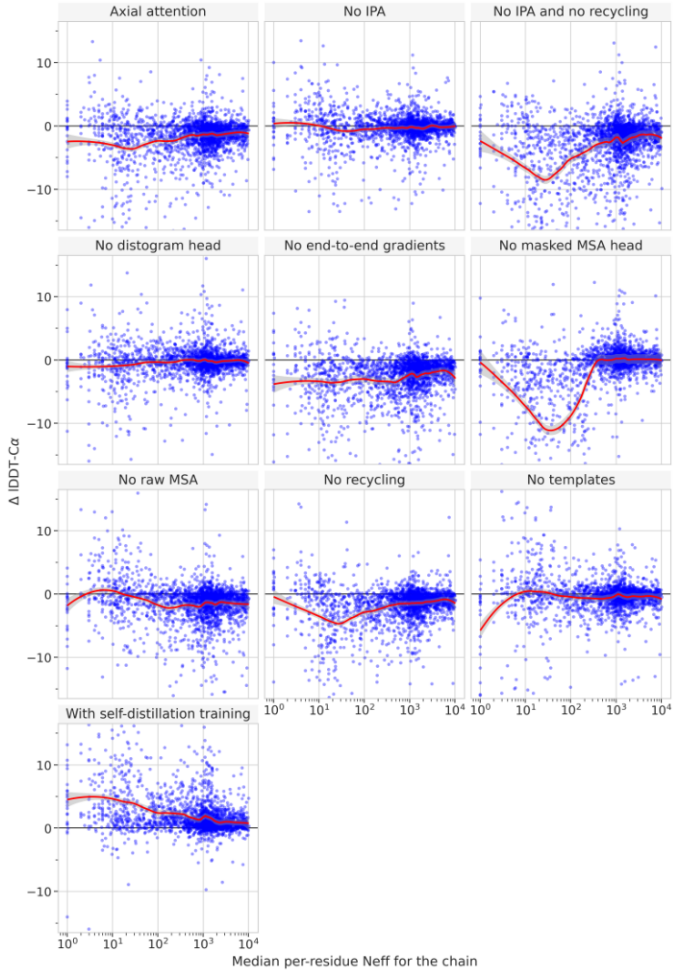
Baseline for all ablation models: Full model without noisy-student self-attention
Ablations:

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

Ablation Results (in main paper) [1]



Ablation Results [1]



Supplementary Figure 10 | Ablations accuracy relative to the baseline for different values of the MSA depth on the recent PDB set of chains, filtered by template coverage $\leq 30\%$ at 30% identity ($N = 2261$ protein chains). MSA depth is computed by counting the number of non-gap residues for each position in the MSA (using the Neff weighting scheme with a threshold of 80% identity measured on the region that is non-gap in either sequence) and taking the median across residues. Plots are restricted to the range [-15, 15]; the red lines represent LOESS mean estimates with 95% confidence intervals for the LOESS.

Network Probing [1]

todo

Novel Folds

They did well

Attention Visualization [1]

todo

Additional Results [1]

todo

Bibliography I

[1] John Jumper et al. "Highly Accurate Protein Structure Prediction with AlphaFold". en. In: *Nature* (July 2021), pp. 1–11. ISSN: 1476-4687. DOI: 10/gk7nfp.

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

[3] Carlos Outeiral Rubiera. *AlphaFold 2 Is Here: What's behind the Structure Prediction Miracle | Oxford Protein Informatics Group*. en-US.

[4] Andrew W. Senior et al. "Improved Protein Structure Prediction Using Potentials from Deep Learning". en. In: *Nature* 577.7792 (Jan. 2020), pp. 706–710. ISSN: 1476-4687. DOI: 10/ggjfcc.

[5] *Spatial Transformation Matrices*. https://www.brainvoyager.com/bv/doc/UsersGuide/CoordsAndTransforms/Spatia