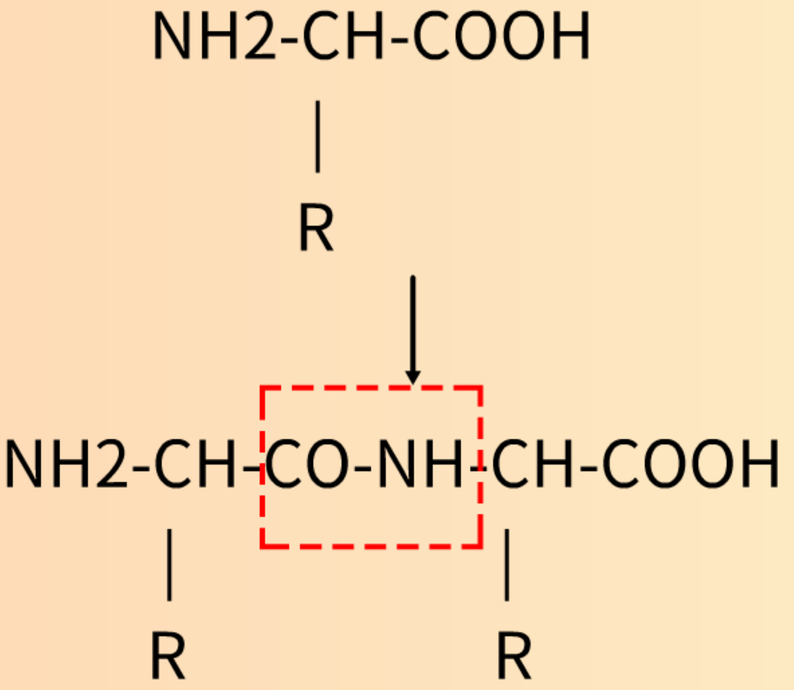
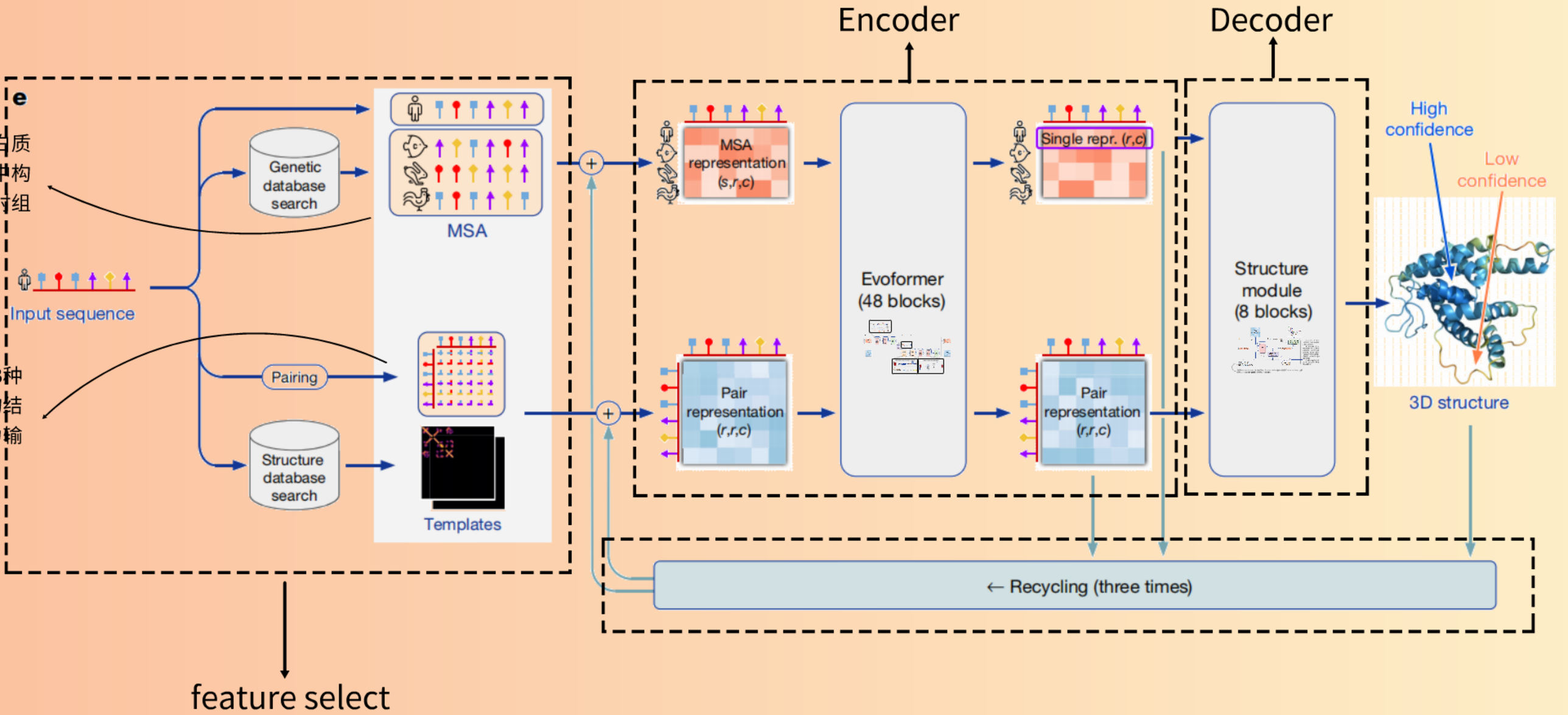
蛋白质基础知识

蛋白质是由氨基酸通过肽键连接而成的长链分子。每条蛋白质链称为一条多肽链（polypeptide）。蛋白质的结构和功能取决于其氨基酸序列、三维结构（四级）以及与其他分子的相互作用。

三级结构是指蛋白质链在三维空间中的整体折叠形式。三级结构由氨基酸侧链之间的相互作用（如氢键、疏水作用、离子键等）决定。



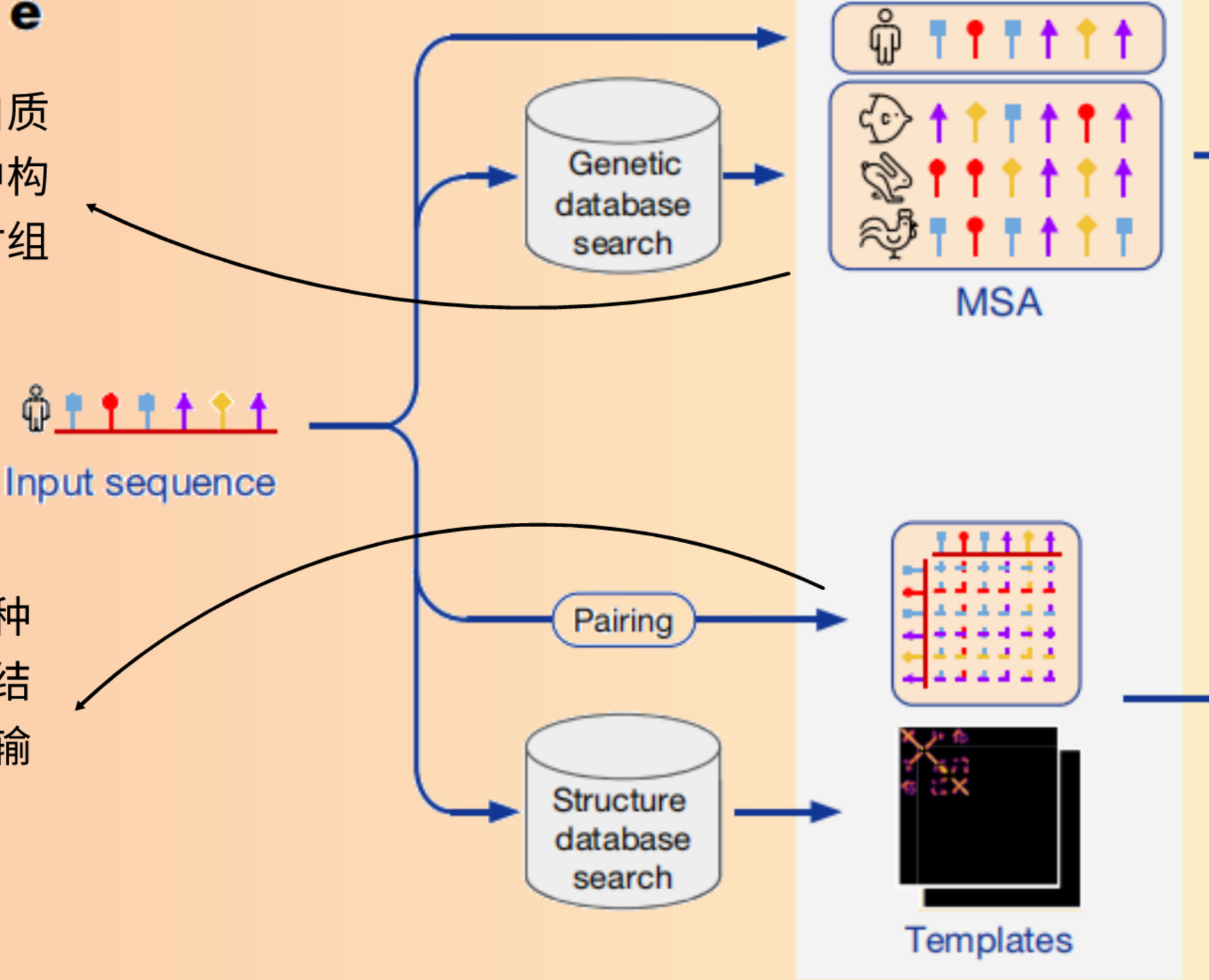

Encoder
Decoder
Genetic database search
MSA
Input sequence
Pairing
Structure database search
Templates
MSA representation (s,r,c)
Single repr. (r,c)
Evoformer (48 blocks)
Pair representation (r,r,c)
Pair representation (r,r,c)
Structure module (8 blocks)
High confidence
Low confidence
3D structure
← Recycling (three times)
feature select

e

根据输入的蛋白质序列从数据库中构建的多序列比对组（MSA）

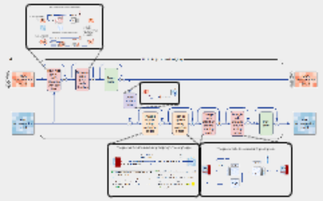
蛋白质序列

根据MSA在PDB种搜索同源蛋白的结构，可能不作为输入



MSA representation (*s*,*r*,*c*)

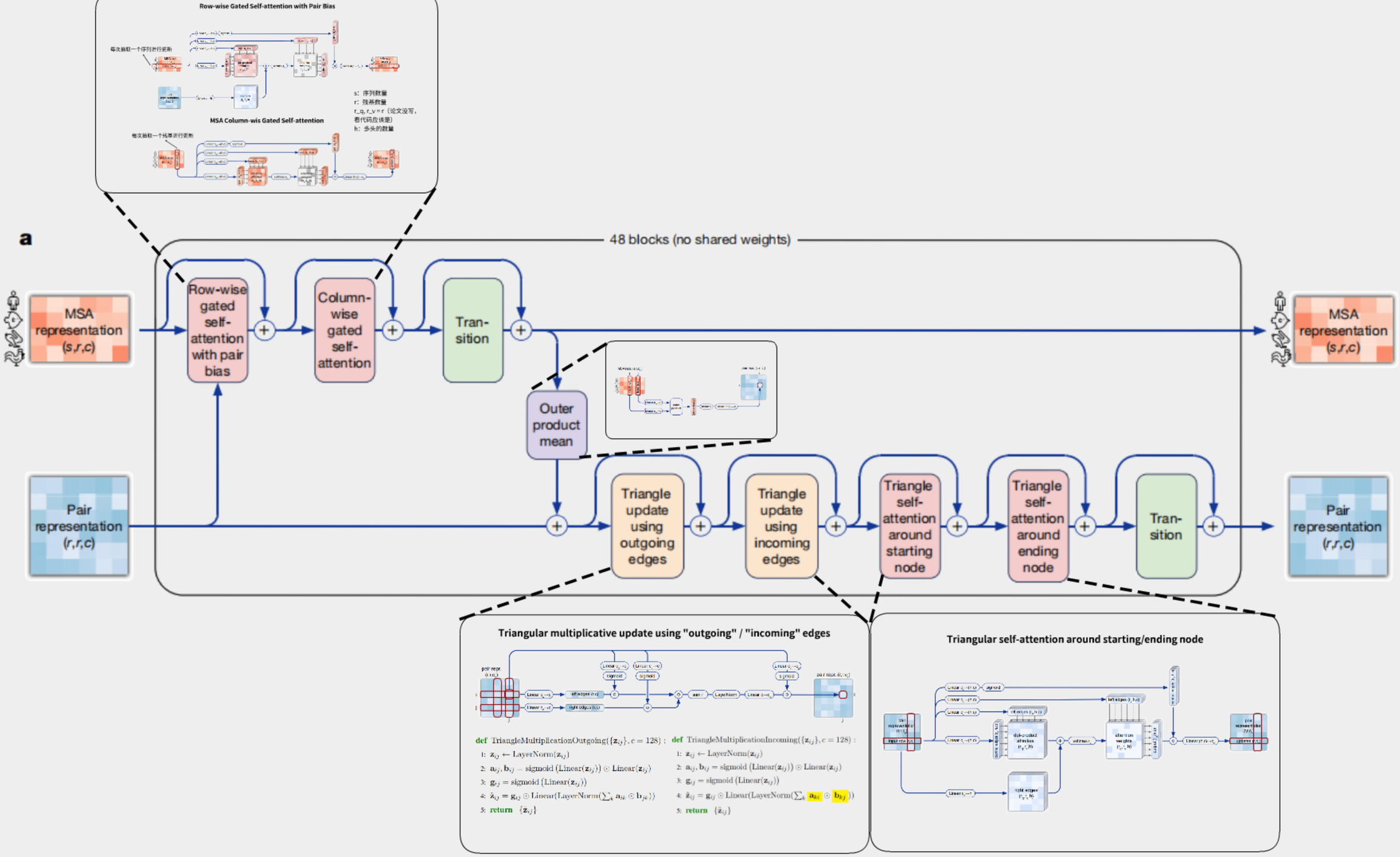
Evoformer (48 blocks)



Single repr. (*r*,*c*)

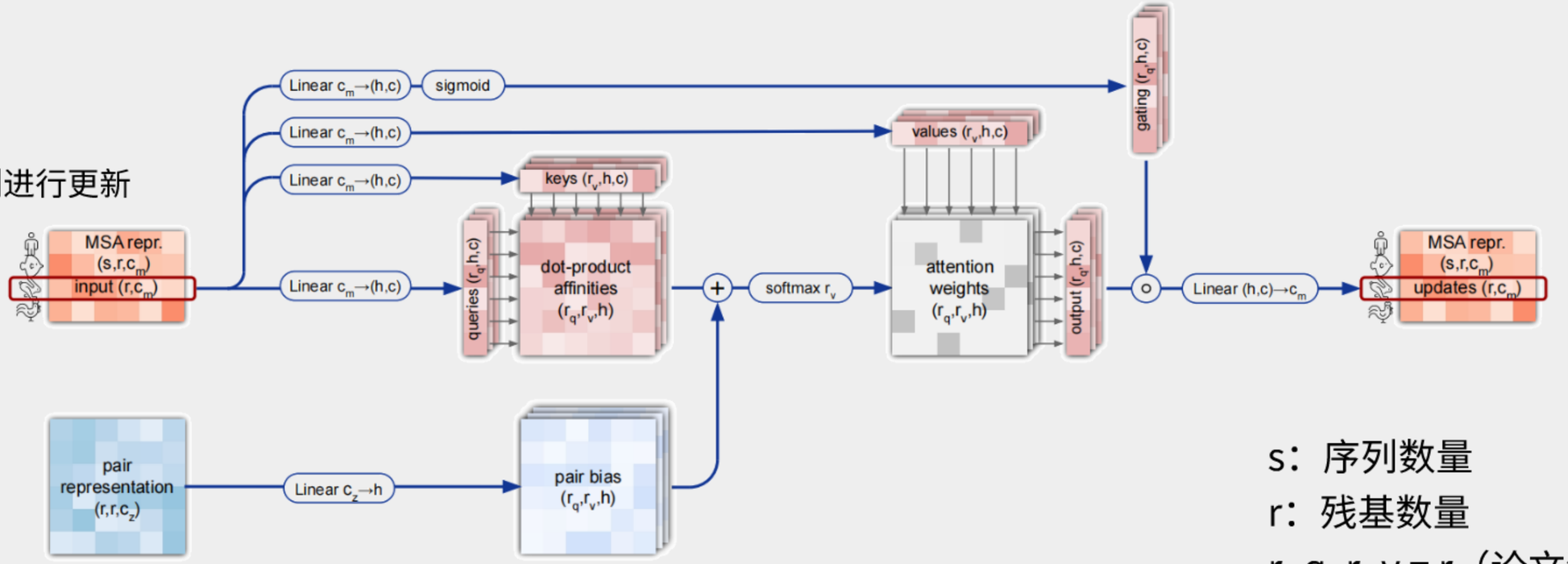
Pair representation (*r*,*r*,*c*)

Pair representation (*r*,*r*,*c*)


Row-wise Gated Self-attention with Pair Bias
每次抽取一个序列进行更新
s：序列数量
r：残基数量
r_q, r_v = r（论文没写，看代码应该是）
h：多头的数量
MSA Column-wis Gated Self-attention
每次抽取一个残基进行更新
a
48 blocks (no shared weights)
MSA representation (s,r,c)
Row-wise gated self-attention with pair bias
Column-wise gated self-attention
Tran-sition
Outer product mean
Pair representation (r,r,c)
Triangle update using outgoing edges
Triangle update using incoming edges
Triangle self-attention around starting node
Triangle self-attention around ending node
Tran-sition
MSA representation (s,r,c)
Pair representation (r,r,c)
Triangular multiplicative update using "outgoing" / "incoming" edges
def TriangleMultiplicationOutgoing($\{\mathbf{z}_{ij}\}, c = 128$) :
1: $\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$
2: $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$
3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$
4: $\hat{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ik} \odot \mathbf{b}_{jk}))$
5: return $\{\mathbf{z}_{ij}\}$
def TriangleMultiplicationIncoming($\{\mathbf{z}_{ij}\}, c = 128$) :
1: $\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$
2: $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$
3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$
4: $\hat{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ki} \odot \mathbf{b}_{kj}))$
5: return $\{\hat{\mathbf{z}}_{ij}\}$
Triangular self-attention around starting/ending node

Row-wise Gated Self-attention with Pair Bias

每次抽取一个序列进行更新



s：序列数量

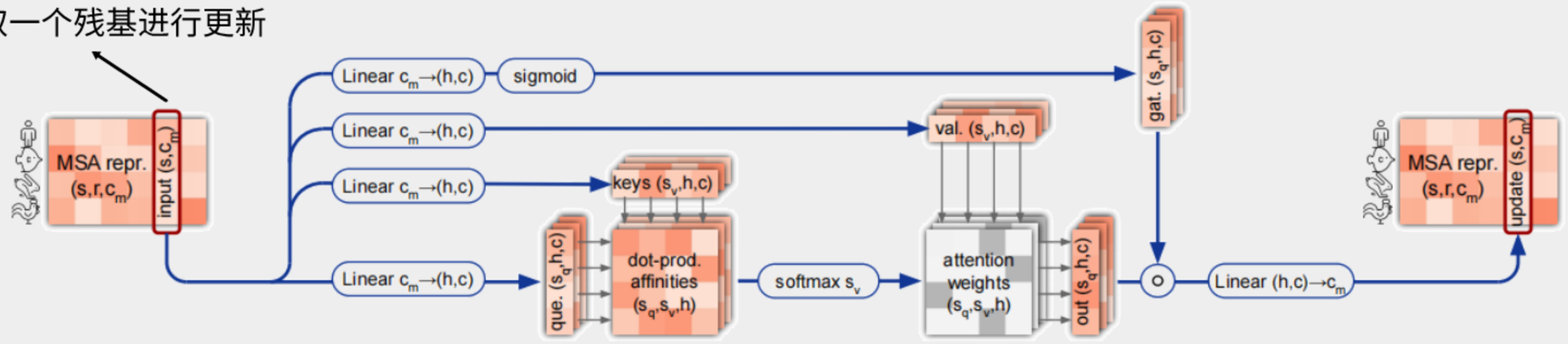
r：残基数量

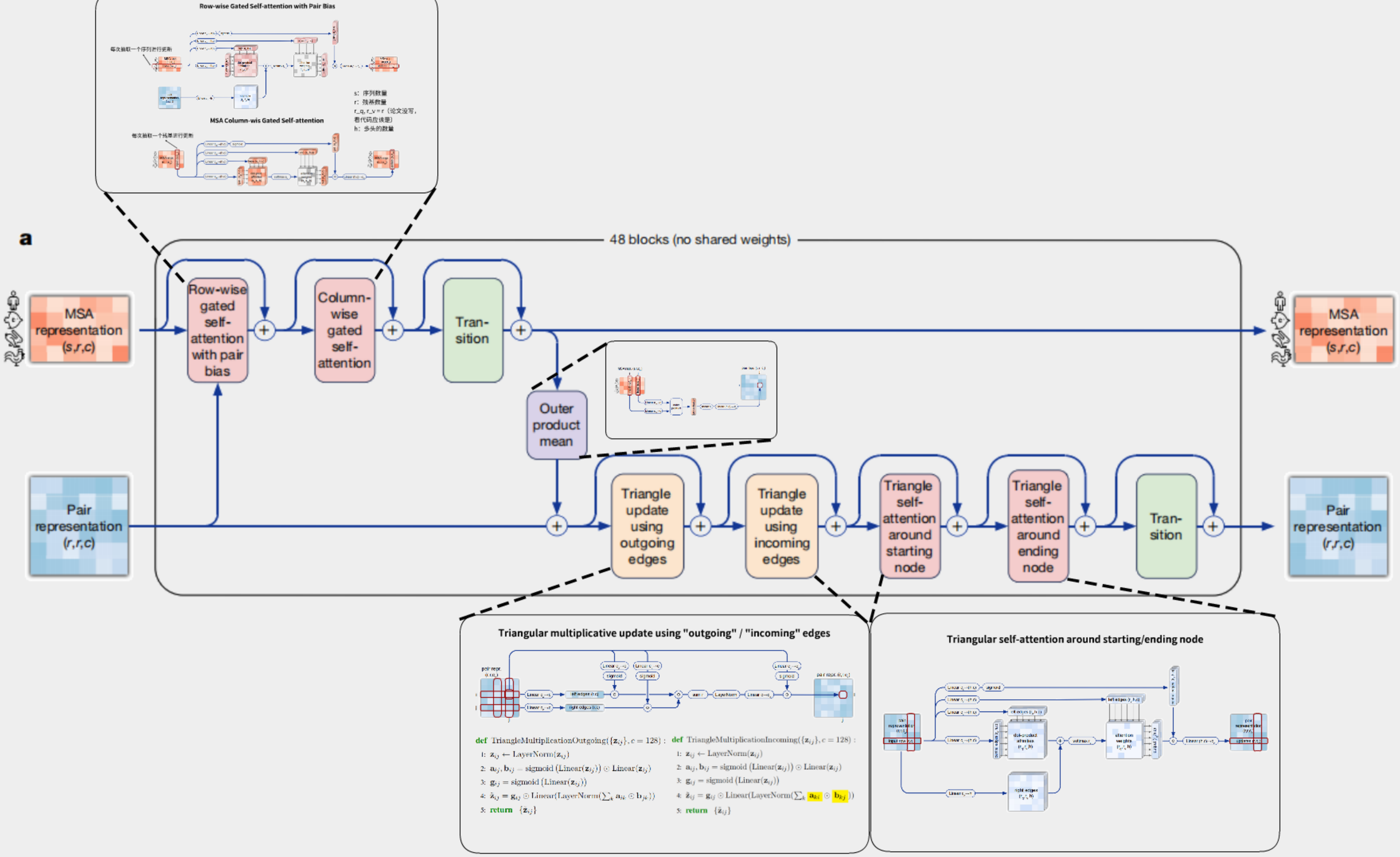
r_q, r_v = r（论文没写，看代码应该是）

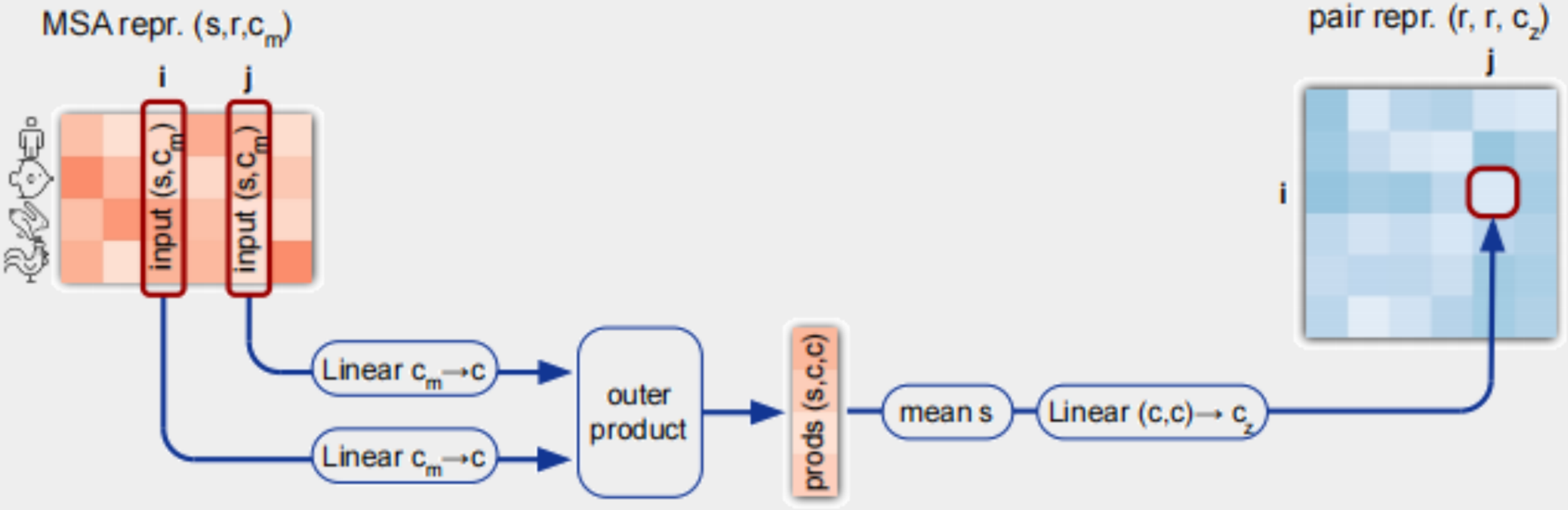
h：多头的数量

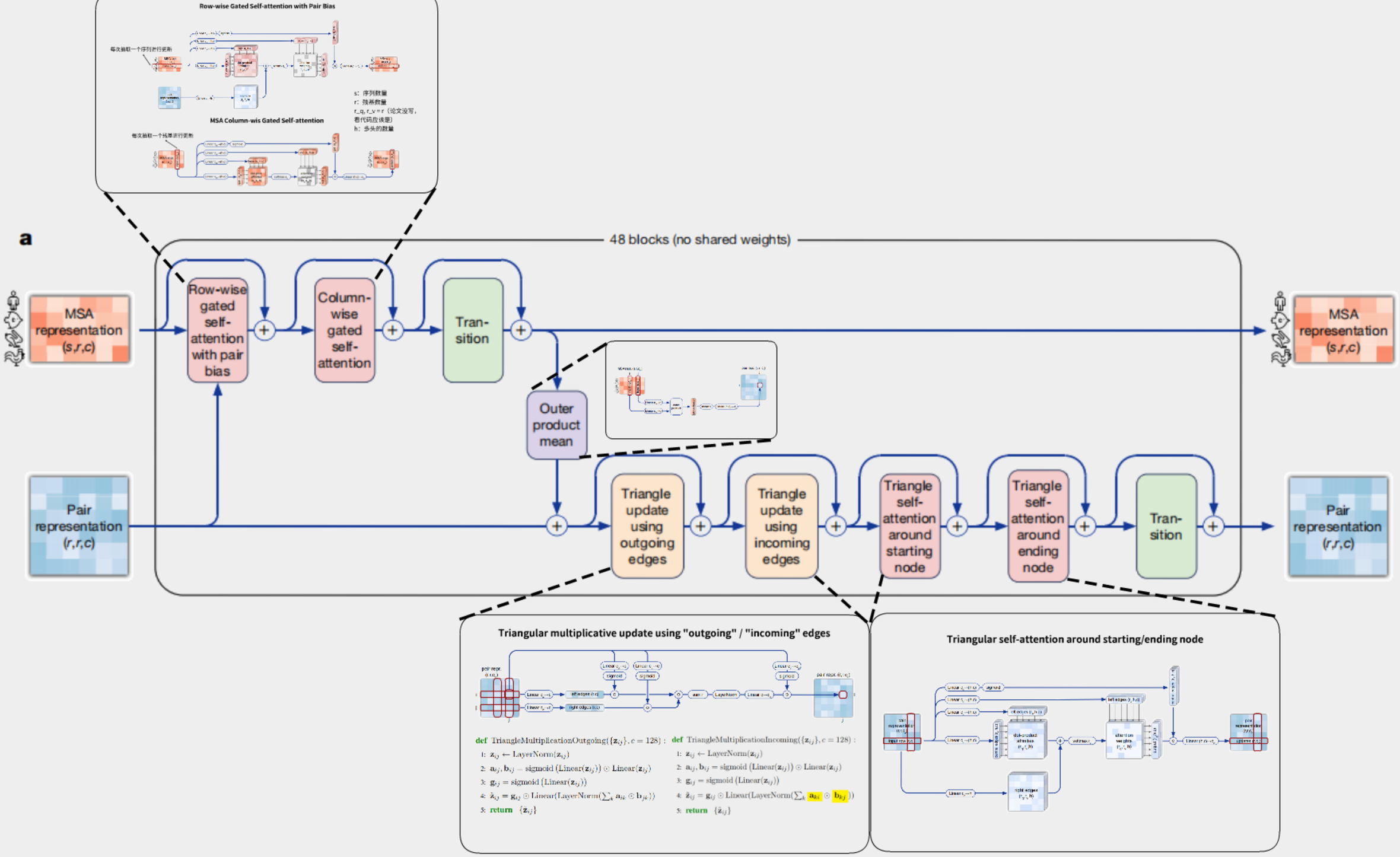
MSA Column-wis Gated Self-attention

每次抽取一个残基进行更新

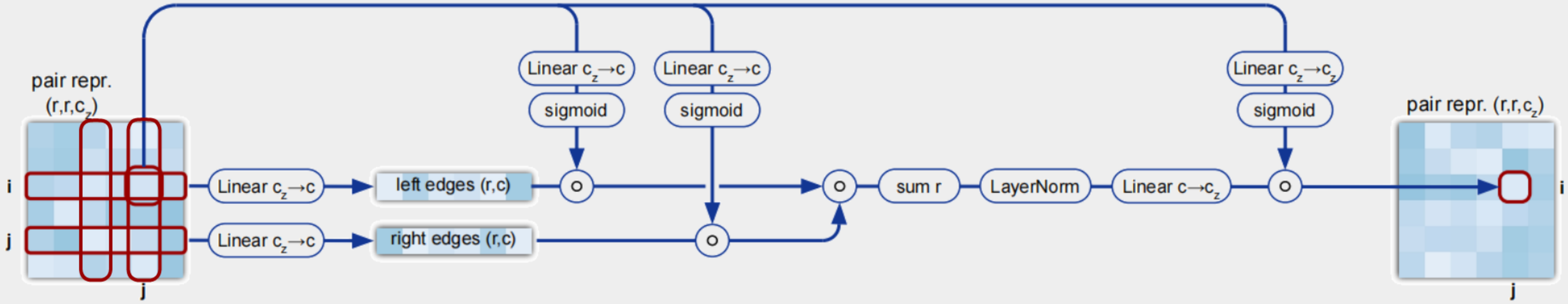



Row-wise Gated Self-attention with Pair Bias
每次抽取一个序列进行更新
s：序列数量
r：残基数量
r_q, r_v = r（论文没写，看代码应该是）
h：多头的数量
MSA Column-wis Gated Self-attention
每次抽取一个残基进行更新
a
48 blocks (no shared weights)
MSA representation (s,r,c)
Row-wise gated self-attention with pair bias
Column-wise gated self-attention
Tran-sition
Outer product mean
Pair representation (r,r,c)
Triangle update using outgoing edges
Triangle update using incoming edges
Triangle self-attention around starting node
Triangle self-attention around ending node
Tran-sition
MSA representation (s,r,c)
Pair representation (r,r,c)
Triangular multiplicative update using "outgoing" / "incoming" edges
def TriangleMultiplicationOutgoing($\{\mathbf{z}_{ij}\}, c = 128$) :
1: $\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$
2: $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$
3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$
4: $\hat{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ik} \odot \mathbf{b}_{jk}))$
5: return $\{\mathbf{z}_{ij}\}$
def TriangleMultiplicationIncoming($\{\mathbf{z}_{ij}\}, c = 128$) :
1: $\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$
2: $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$
3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$
4: $\hat{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ki} \odot \mathbf{b}_{kj}))$
5: return $\{\hat{\mathbf{z}}_{ij}\}$
Triangular self-attention around starting/ending node


MSA repr. (s,r,c_m)
i
j
input (s,c_m)
input (s,c_m)
Linear c_m→c
Linear c_m→c
outer product
prods (s,c,c)
mean s
Linear (c,c)→ c_z
pair repr. (r, r, c_z)
j
i


Row-wise Gated Self-attention with Pair Bias
每次抽取一个序列进行更新
s：序列数量
r：残基数量
r_q, r_v = r（论文没写，看代码应该是）
h：多头的数量
MSA Column-wis Gated Self-attention
每次抽取一个残基进行更新
a
48 blocks (no shared weights)
MSA representation (s,r,c)
Row-wise gated self-attention with pair bias
Column-wise gated self-attention
Tran-sition
Outer product mean
Pair representation (r,r,c)
Triangle update using outgoing edges
Triangle update using incoming edges
Triangle self-attention around starting node
Triangle self-attention around ending node
Tran-sition
MSA representation (s,r,c)
Pair representation (r,r,c)
Triangular multiplicative update using "outgoing" / "incoming" edges
def TriangleMultiplicationOutgoing($\{\mathbf{z}_{ij}\}, c = 128$) :
1: $\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$
2: $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$
3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$
4: $\hat{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ik} \odot \mathbf{b}_{jk}))$
5: return $\{\mathbf{z}_{ij}\}$
def TriangleMultiplicationIncoming($\{\mathbf{z}_{ij}\}, c = 128$) :
1: $\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$
2: $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$
3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$
4: $\hat{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ki} \odot \mathbf{b}_{kj}))$
5: return $\{\hat{\mathbf{z}}_{ij}\}$
Triangular self-attention around starting/ending node

Triangular multiplicative update using "outgoing" / "incoming" edges



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

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

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

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

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

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

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

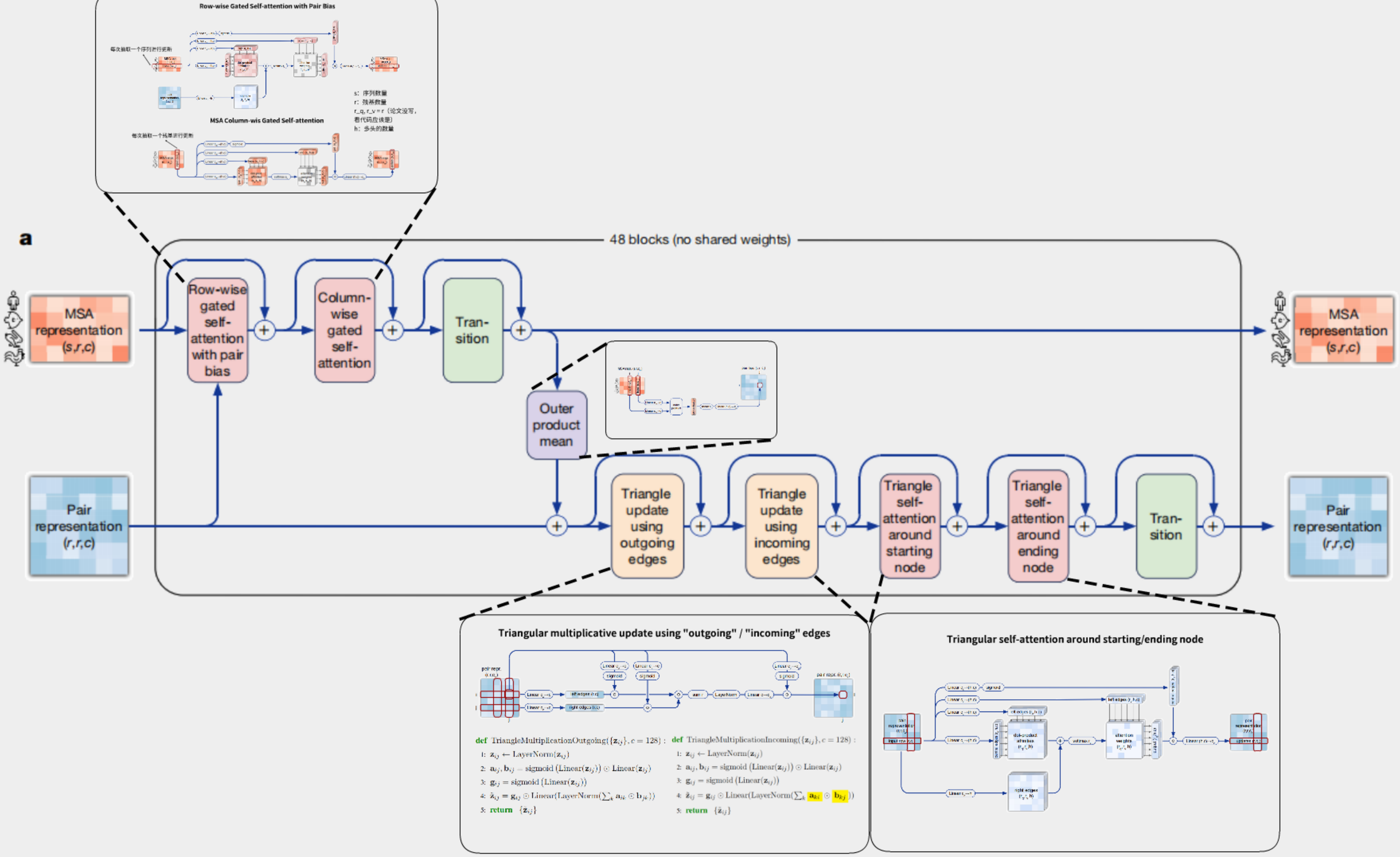
1: $\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$

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

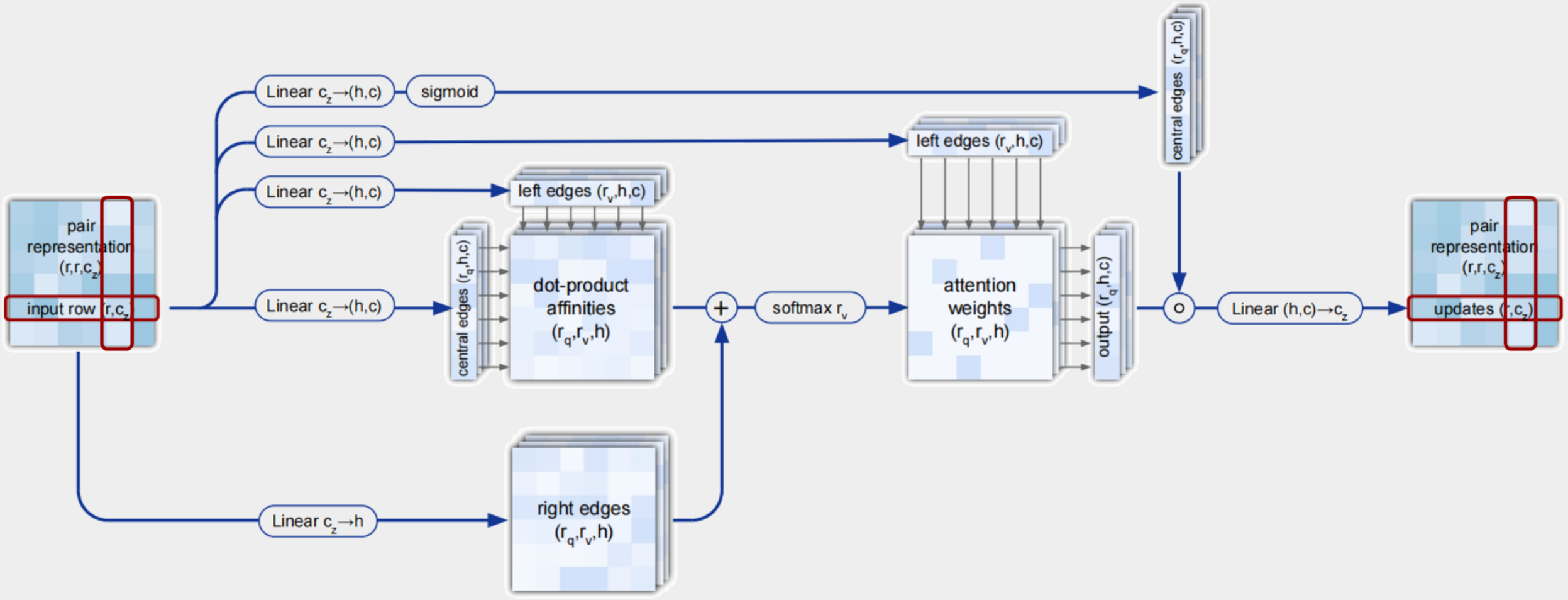
3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$

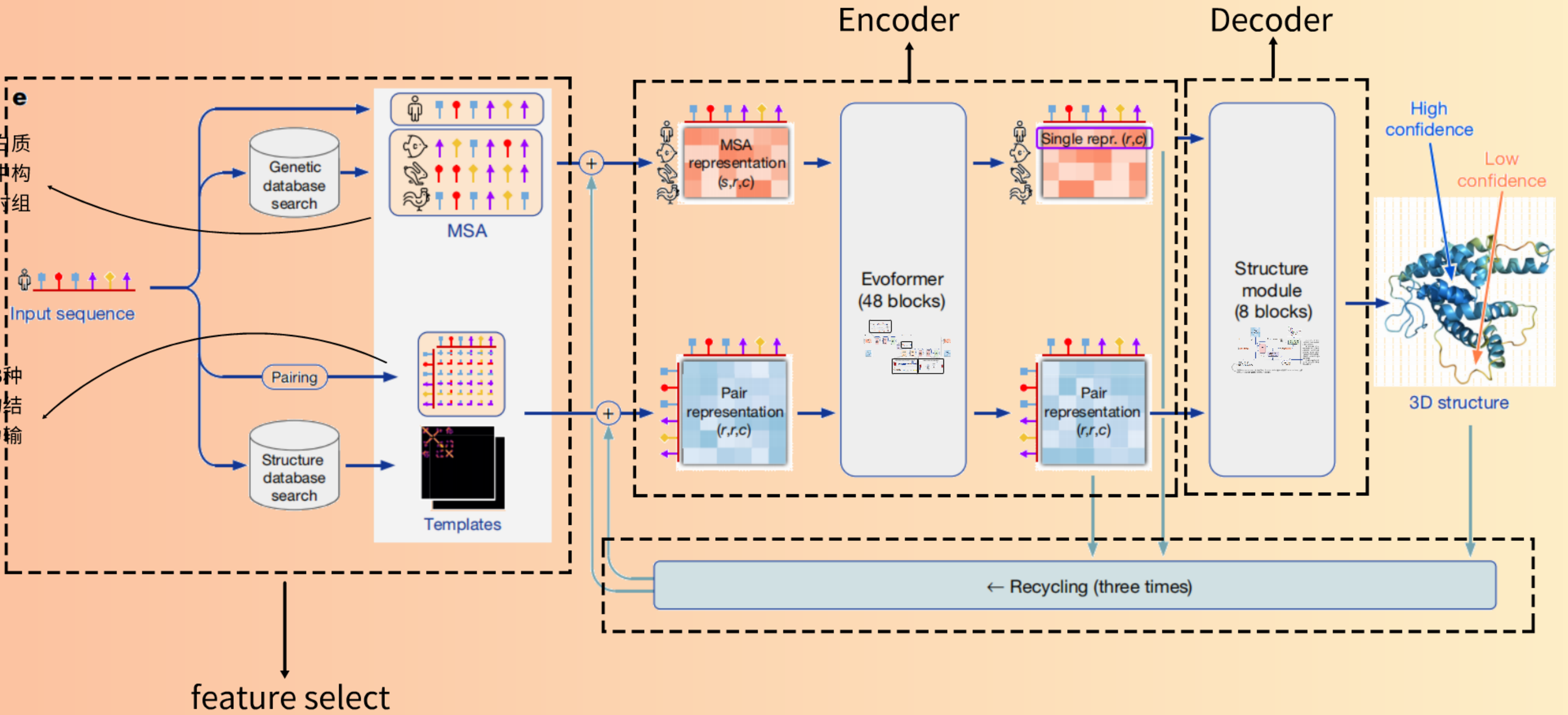
4: $\tilde{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ki} \odot \mathbf{b}_{kj}))$

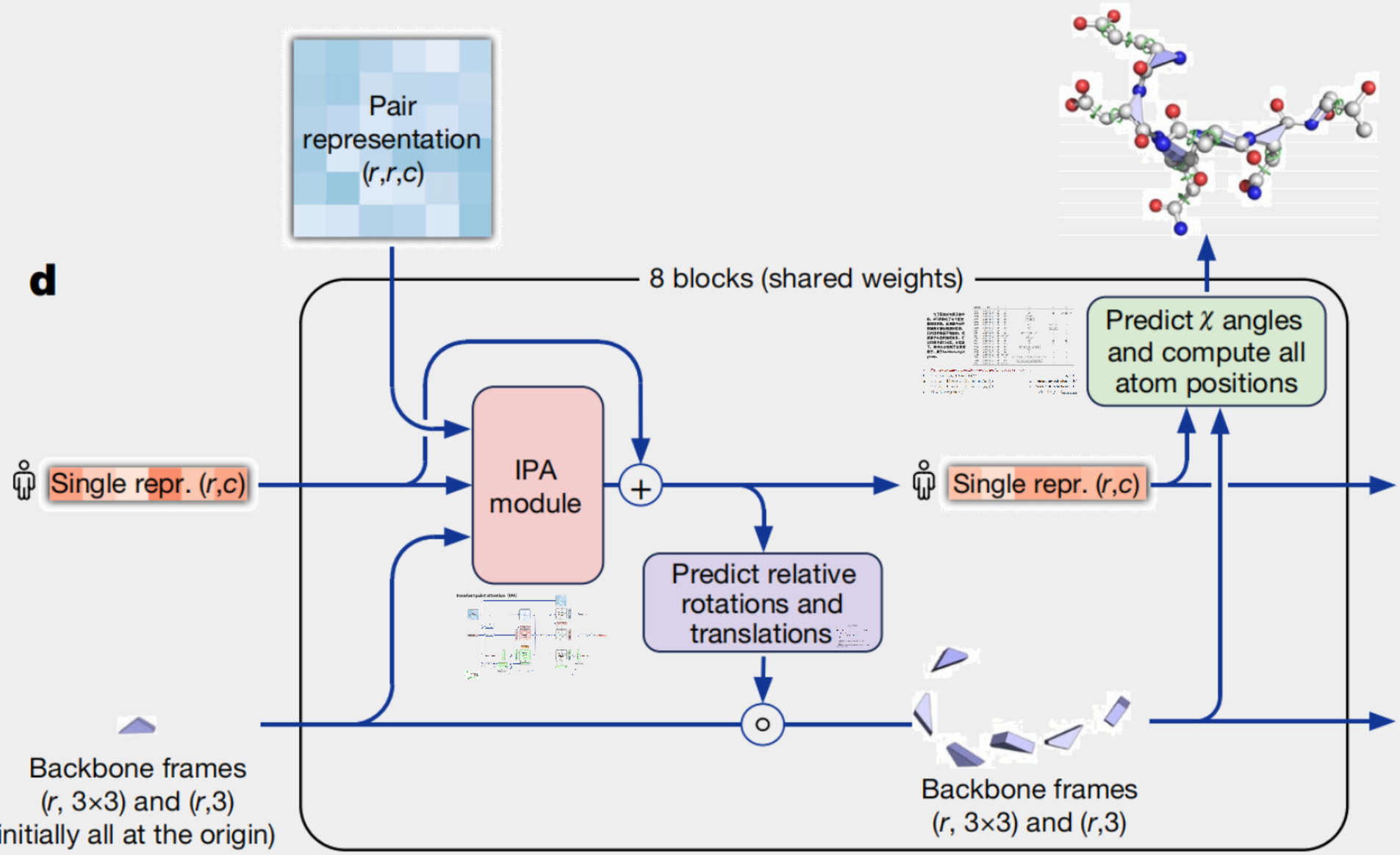
5: **return** $\{\tilde{\mathbf{z}}_{ij}\}$


Row-wise Gated Self-attention with Pair Bias
每次抽取一个序列进行更新
s：序列数量
r：残基数量
r_q, r_v = r（论文没写，看代码应该是）
h：多头的数量
MSA Column-wis Gated Self-attention
每次抽取一个残基进行更新
a
48 blocks (no shared weights)
MSA representation (s,r,c)
Row-wise gated self-attention with pair bias
Column-wise gated self-attention
Tran-sition
Outer product mean
Pair representation (r,r,c)
Triangle update using outgoing edges
Triangle update using incoming edges
Triangle self-attention around starting node
Triangle self-attention around ending node
Tran-sition
MSA representation (s,r,c)
Pair representation (r,r,c)
Triangular multiplicative update using "outgoing" / "incoming" edges
def TriangleMultiplicationOutgoing($\{\mathbf{z}_{ij}\}, c = 128$) :
1: $\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$
2: $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$
3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$
4: $\hat{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ik} \odot \mathbf{b}_{jk}))$
5: return $\{\mathbf{z}_{ij}\}$
def TriangleMultiplicationIncoming($\{\mathbf{z}_{ij}\}, c = 128$) :
1: $\mathbf{z}_{ij} \leftarrow \text{LayerNorm}(\mathbf{z}_{ij})$
2: $\mathbf{a}_{ij}, \mathbf{b}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right) \odot \text{Linear}(\mathbf{z}_{ij})$
3: $\mathbf{g}_{ij} = \text{sigmoid}\left(\text{Linear}(\mathbf{z}_{ij})\right)$
4: $\hat{\mathbf{z}}_{ij} = \mathbf{g}_{ij} \odot \text{Linear}(\text{LayerNorm}(\sum_k \mathbf{a}_{ki} \odot \mathbf{b}_{kj}))$
5: return $\{\hat{\mathbf{z}}_{ij}\}$
Triangular self-attention around starting/ending node

Triangular self-attention around starting/ending node




Encoder
Decoder
Genetic database search
MSA
Input sequence
Pairing
Structure database search
Templates
MSA representation (s,r,c)
Single repr. (r,c)
Evoformer (48 blocks)
Pair representation (r,r,c)
Pair representation (r,r,c)
Structure module (8 blocks)
High confidence
Low confidence
3D structure
← Recycling (three times)
feature select



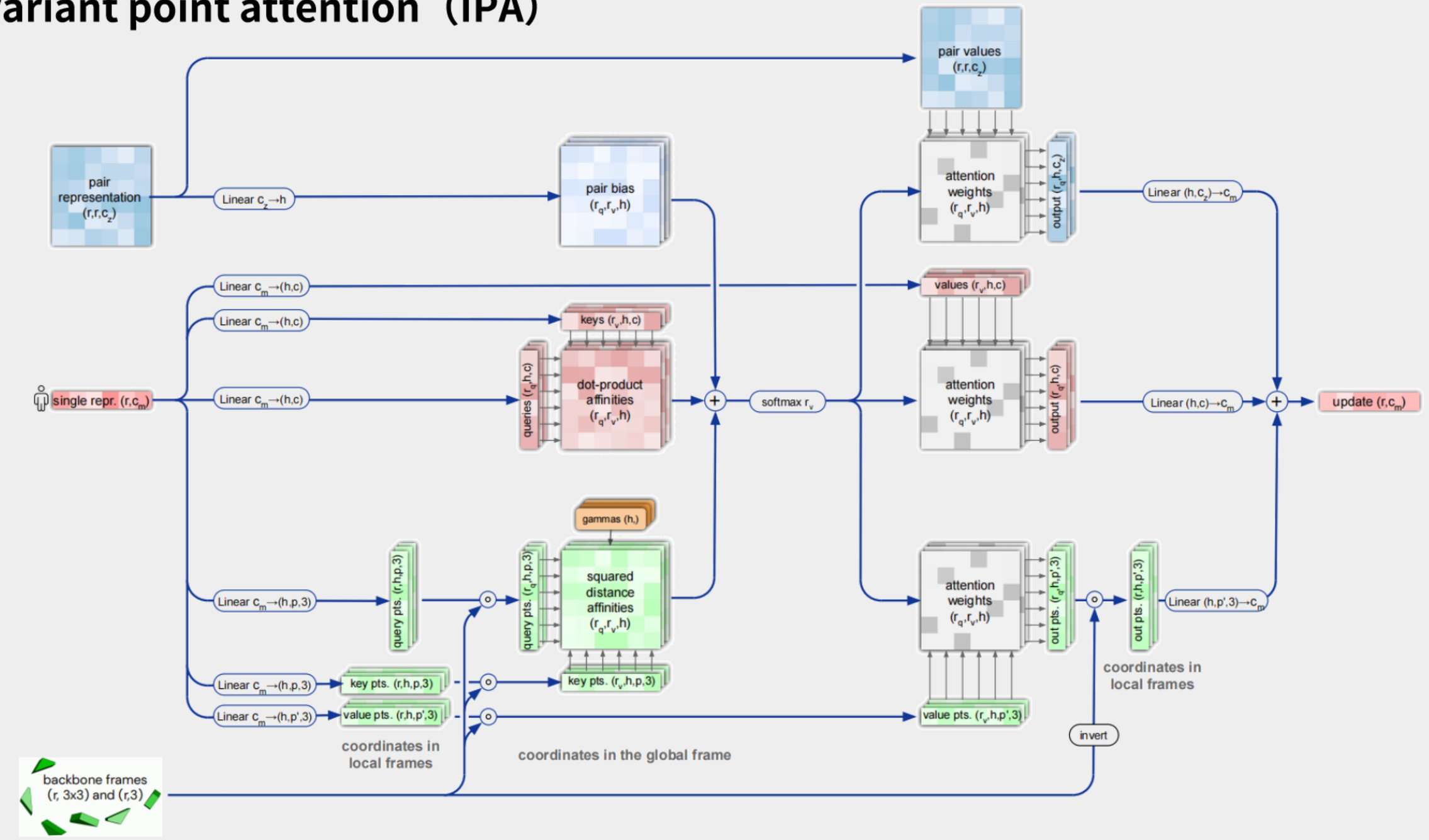
在AlphaFold中，蛋白质结构预测分为主干网络预测和分支结构预测。

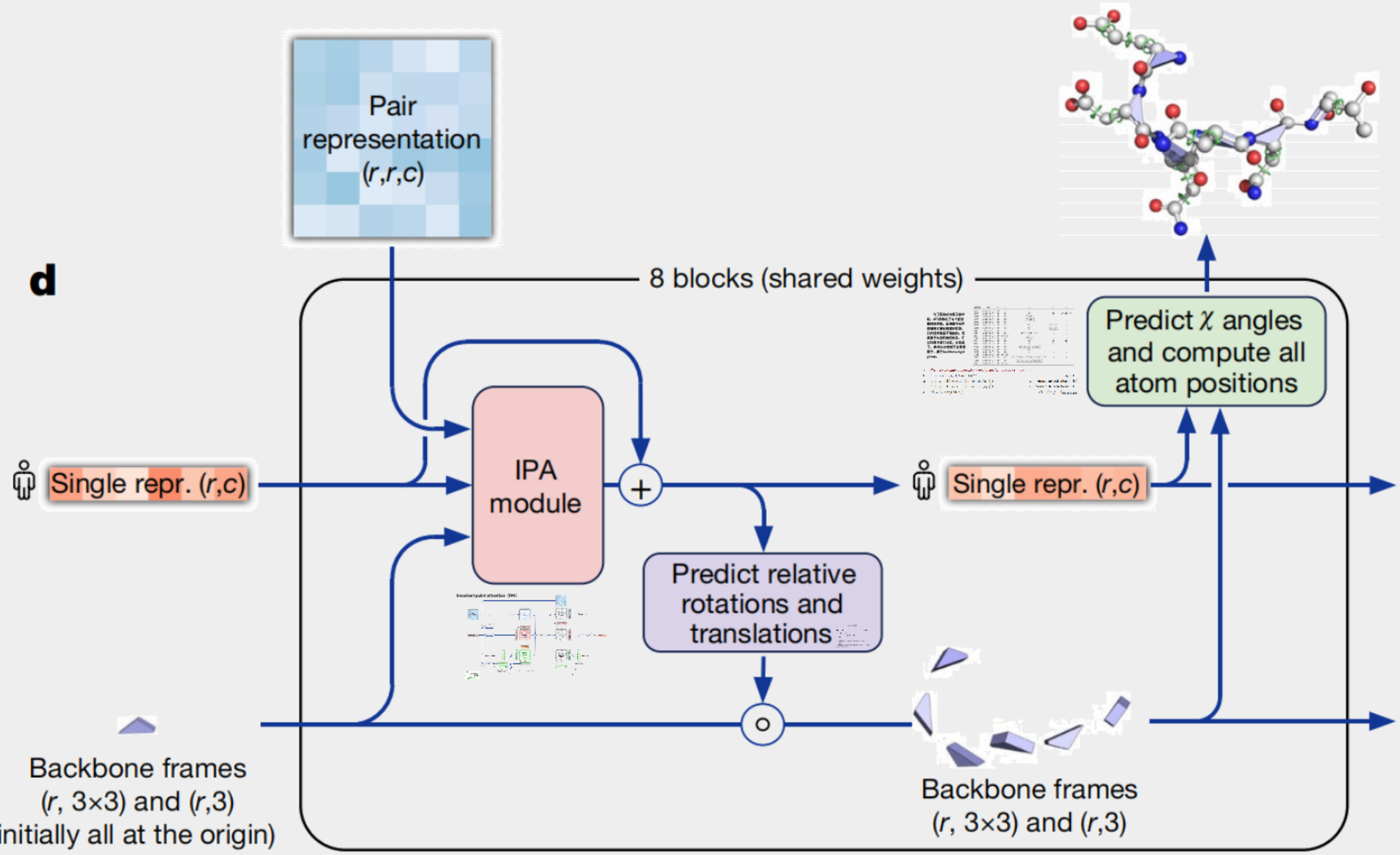
预测相对位置：蛋白质结构是各种原子，分子相互作用的结构，预测相对位置更加友好。可以通过相互可采用乘以一个3x3的旋转矩阵R再加上一个 3x1的平移向量相互转换。

对于每一个氨基酸内部的原子，只需要表达出其旋转的角度即可。

·初始化 backbone frame： 对于每个残基，Structure Module 使用一个初始的 backbone frame，其旋转矩阵 R 和平移向量 t 都设置为单位矩阵 I 和零向量 0。

Invariant point attention（IPA）





在AlphaFold中，蛋白质结构预测分为主干网络预测和分支结构预测。

预测相对位置：蛋白质结构是各种原子，分子相互作用的结构，预测相对位置更加友好。可以通过相互可采用乘以一个3x3的旋转矩阵R再加上一个 3x1的平移向量相互转换。

对于每一个氨基酸内部的原子，只需要表达出其旋转的角度即可。

·初始化 backbone frame： 对于每个残基，Structure Module 使用一个初始的 backbone frame，其旋转矩阵 R 和平移向量 t 都设置为单位矩阵 I 和零向量 0。

更新主干位置预测

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

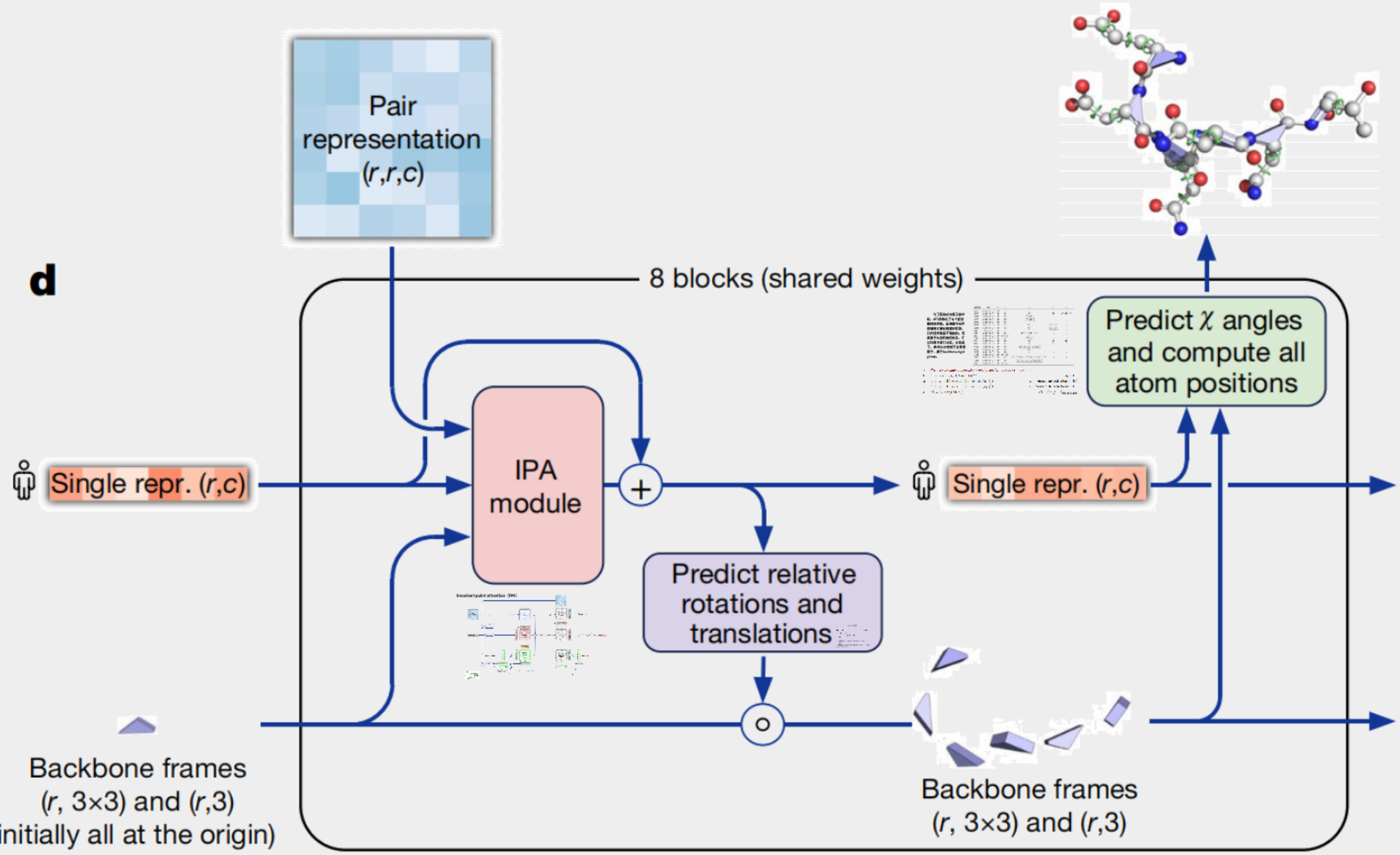
5: **return** T_i

主干网络上预测的是相邻氨基酸的相对旋转变换矩阵R，则必须有性质：

1. 正交阵

2. |R| = 1。

AlphaFold的做法是不直接输出矩阵各个维度的值，而是输出几个常量，并在其基础上构造出合法的单位旋转矩阵来。



在AlphaFold中，蛋白质结构预测分为主干网络预测和分支结构预测。

预测相对位置：蛋白质结构是各种原子，分子相互作用的结构，预测相对位置更加友好。可以通过相互可采用乘以一个3x3的旋转矩阵R再加上一个 3x1的平移向量相互转换。

对于每一个氨基酸内部的原子，只需要表达出其旋转的角度即可。

·初始化 backbone frame： 对于每个残基，Structure Module 使用一个初始的 backbone frame，其旋转矩阵 R 和平移向量 t 都设置为单位矩阵 I 和零向量 0。

为了获得所有原子的坐标，AF2参数化了每个氨基酸的扭转角。氨基酸中所有的键角和键长都是刚性的，只有扭转角是不确定的。根据原子和扭转角的关系，可以将原子进行分类，分类如下，其中bb仅依赖于主骨架原子，属于backbone rigid group。

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	$O^{\delta 1}$, $O^{\delta 2}$	-	-
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	$O^{\epsilon 1}$, $O^{\epsilon 2}$	-
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	$C^{\delta 1}$, $C^{\delta 2}$, $C^{\epsilon 1}$, $C^{\epsilon 2}$, C^ζ	-	-
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	$C^{\delta 1}$, $C^{\delta 2}$, $C^{\epsilon 1}$, $C^{\epsilon 2}$, O^η, C^ζ	-	-
VAL	N, C^α, C, C^β	O	$C^{\gamma 1}$, $C^{\gamma 2}$	-	-	-

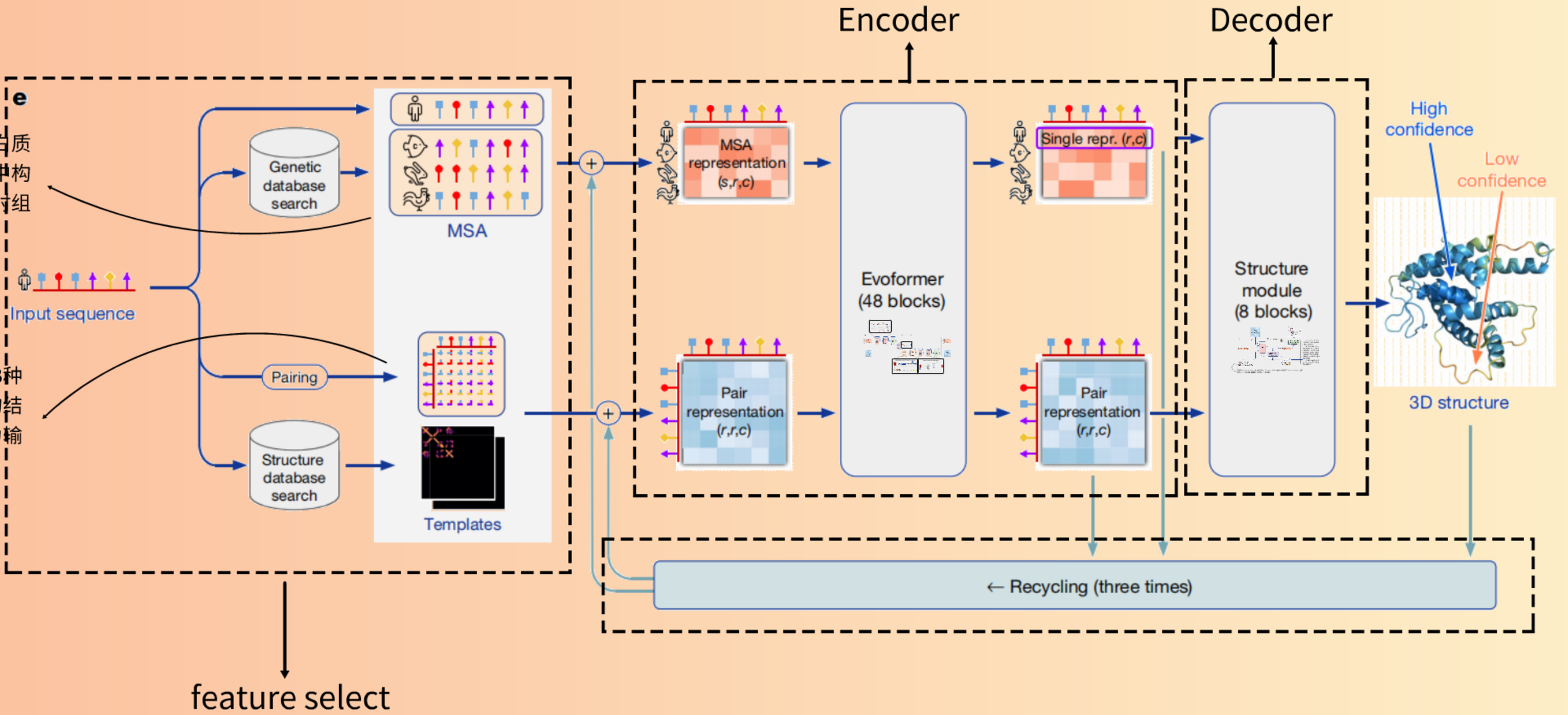
 # Predict side chain and backbone torsion angles $\omega, \phi, \psi, \chi_1, \chi_2, \chi_3, \chi_4$

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

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

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

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


Encoder
Decoder
Genetic database search
MSA
Input sequence
Pairing
Structure database search
Templates
MSA representation (s,r,c)
Single repr. (r,c)
Evoformer (48 blocks)
Pair representation (r,r,c)
Pair representation (r,r,c)
Structure module (8 blocks)
High confidence
Low confidence
3D structure
← Recycling (three times)
feature select

Loss function

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

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

step1/2：计算pred和label的local coord

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

step3：计算pred和label相差的欧式距离

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

step4：归一化

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

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: $\ell_i^f = \|\vec{\alpha}_i^f\|$ step1/2：归一化（AlphaFold用单位圆上的坐标

2: $\hat{\vec{\alpha}}_i^f = \vec{\alpha}_i^f / \ell_i^f$ 表示扭转角度）

3: $\mathcal{L}_{\text{torsion}} = \text{mean}_{i,f}(\text{minimum}(\|\hat{\vec{\alpha}}_i^f - \vec{\alpha}_i^{\text{true},f}\|^2, \|\hat{\vec{\alpha}}_i^f - \vec{\alpha}_i^{\text{alt truth},f}\|^2))$ step3：计算预测与真实值的MSE

4: $\mathcal{L}_{\text{anglenorm}} = \text{mean}_{i,f}(\{|\ell_i^f - 1|\})$ step4：辅助损失鼓励网络预测的扭转角尽量在单位圆上

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

Algorithm 29 Predict model confidence pLDDT

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

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

2: $\mathbf{p}_i^{\text{pLDDT}} = \text{softmax}(\text{Linear}(\mathbf{a}_i))$ step1/2：线性层预测lDDT-Cα分数概率（50bin） $\mathbf{p}_i^{\text{pLDDT}} \in \mathbb{R}^{|\mathbf{v}_{\text{bins}}|}$

3: $\mathbf{p}_i^{\text{true LDDT}} = \text{one\_hot}(r_i^{\text{true LDDT}}, \mathbf{v}_{\text{bins}})$ step3：通过pred和label计算真实的lDDT-Cα分数（维度为50的one-hot编码）

4: $\mathcal{L}_{\text{conf}} = \text{mean}_i(\mathbf{p}_i^{\text{true LDDT}^\top} \log \mathbf{p}_i^{\text{pLDDT}})$ step4：CEloss计算预测的lDDT-Cα分数和真实分数

5: $r_i^{\text{pLDDT}} = \mathbf{p}_i^{\text{pLDDT}^\top} \mathbf{v}_{\text{bins}}$ step5：最终预测的置信分数 $r_i^{\text{pLDDT}} \in \mathbb{R}$

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

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

L_FAPE:
类型：FAPELoss
最终输出的所有的预测的frame和原子坐标与真实frame和原子坐标计算loss，这个loss会评估主链和侧链的所有原子坐标。
L_aux:
类型：FAPELoss+torsionAngleLoss
在structure module中的每一层都需要计算主链的Cα的FAPE损失+主侧链的扭转角损失。
L_dist
类型：CrossEntropyLoss
logits：将pair representation中的对称表征（zij + zji） 通过线性层投影到 64 个距离 bin 中。
label：one-hot编码的分箱残基距离，该距离是根据除甘氨酸以外的所有氨基酸的真实 β 碳位置计算得出的，其中甘氨酸使用 α 碳代替。
L_msa
类型：CrossEntropyLoss
logits：将MSA representation中的表征通过线性层映射到23个类别上（20个标准氨基酸+未知+掩码+缺口）。
label：真实的one-hot编码的氨基酸类型。
L_conf
类型：pLDDTLoss