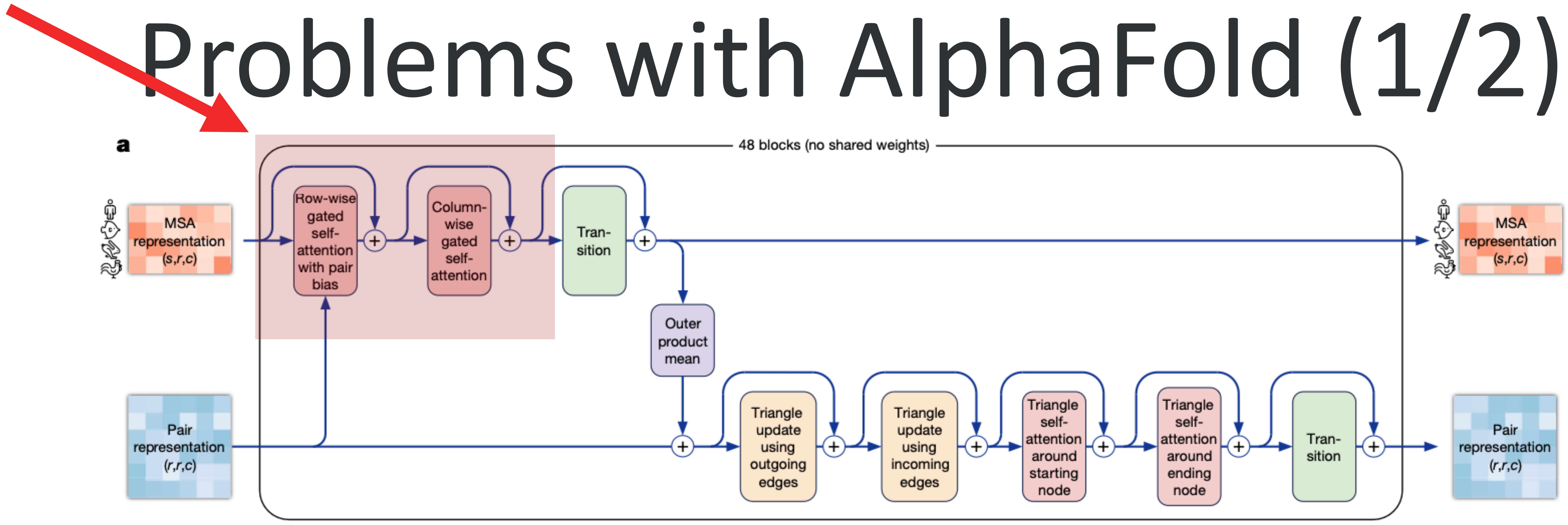
MambaFold

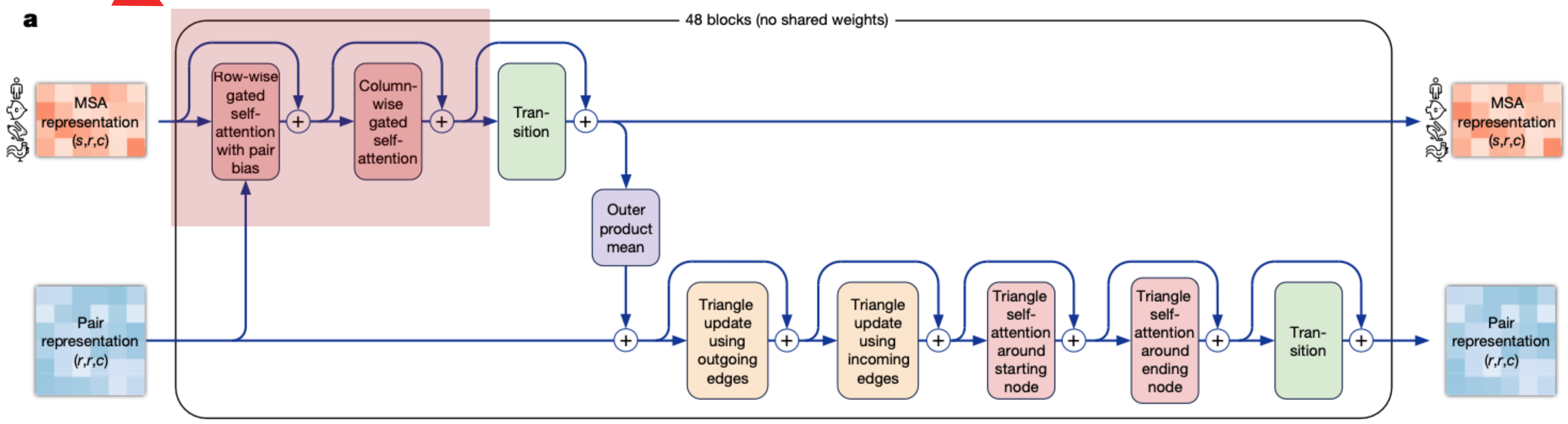
02.02.24

Problems with AlphaFold (1/2)



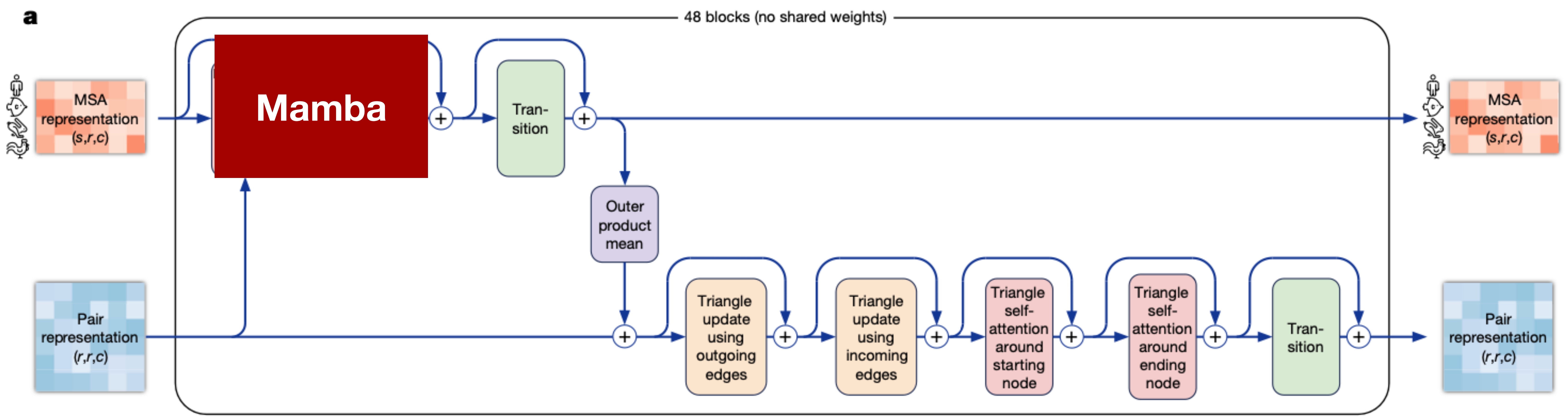
- Row-wise Attention is O(ML^2) memory
- Train: crop to 384 AA
    - contiguous crop or structure crop

Problems with AlphaFold (2/2)



- Structure cropping is ad hoc
- Pairing MSAs is ad hoc
  - e.g. group by species, sequence similarity

MambaFold



- low memory consumption
- joint row-column model
- no paired MSAs
- deeper MSAs
- longer sequences

Computational Complexity (A40)

Row Module	Train Time	Memory Usage	Params
Attention	12.2 sec/iter	9300 MiB	93.2 M
Mamba	6.7 sec/iter	8300 MiB	98.4 M
Mamba - No Fuse	10.3 sec/iter	8300 MiB	98.4 M
Mamba-Pair - No Fuse	10.5 sec/iter	8600 MiB	100 M

CAMEO validation set

- 200 proteins (all <700AA)
- (CASP15 test set)

	Training Iterations	lDDT-Cα
AlphaFold (Fig 2A,7)	7,000 (~1,500 GPU hr) 8gpu,7days	0.83
AlphaFold (Fig 1A)	90,000	0.91

Goal

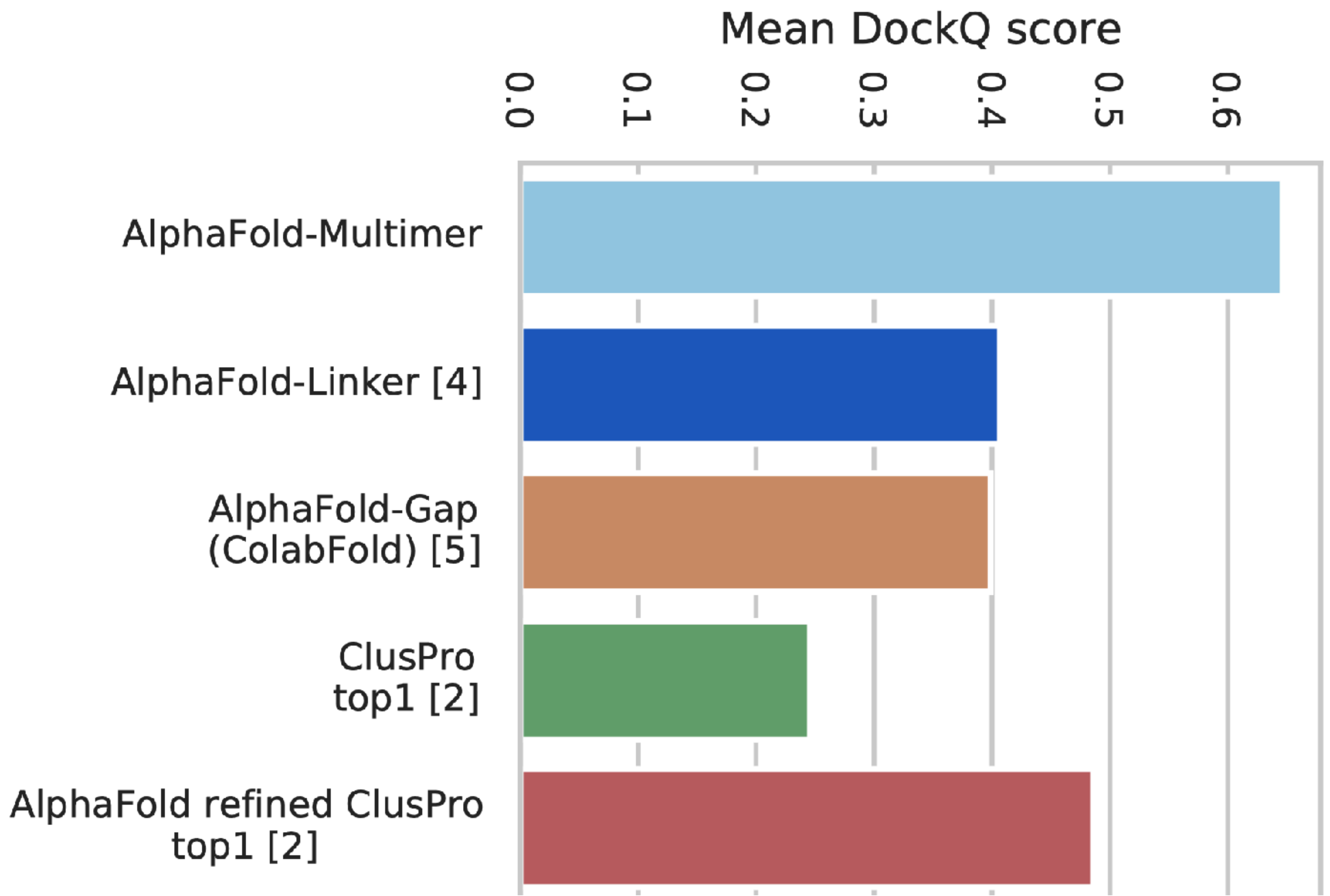
- Next week: Alphafold training in 3 days on 8 GPUs

AlphaFold History

2.2.24

AlphaFold history (1/3)

- 2.0: single chain prediction - July 2021
- 2.1-2.2: multimer - Nov 2021 - March 2022 (8mo)
  - paired MSA
  - multi-chain cropping
  - permutation handling loss

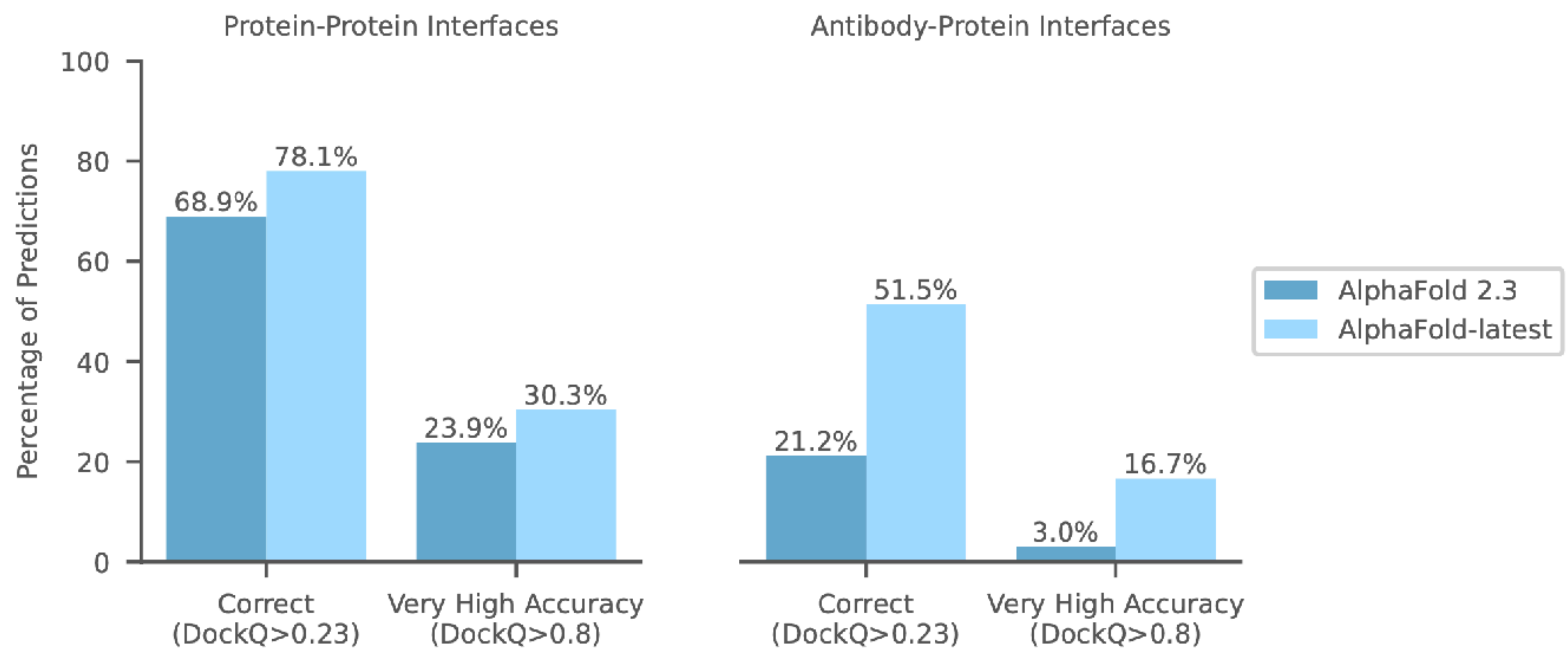


AlphaFold history (2/3)

- 2.3: larger-complexes - December 2022 (9mo)
  - 2021-09-30 cutoff (prev 2018-04-30)
  - identical to AF-multimer except
    - train 640 (prev 384) residues
    - train 20 chains, 2048 MSA (prev 1152)
    - TTA (20 seeds, 20 recycles)

AlphaFold history (3/3)

- 3.0: latest - October 2023 (10mo)
    - 2021-09-30 cutoff (same)
    - ligands (smiles), nucleic acids
    - post translational modifications



Alphafold++ at CASP15

- DeepMSA2
- Improved Template Search

teins and protein assemblies. Five groups were selected to give presentations, and at CASP15, they described how they outperformed AlphaFold. In short, there are two ways to improve over standard AlphaFold: to use templates more efficiently, increase the sampling by using alternative methods to generate the MSAs, or modify AlphaFold to use dropouts.

AlphaFold Interesting Tidbits

2.2.24

AF is not trained End-to-End

```python
# Enable grad iff we're training and it's the final recycling layer
is_final_iter = cycle_no == (num_iters - 1)
with torch.set_grad_enabled(is_grad_enabled and is_final_iter):
    if is_final_iter:
        # Sidestep AMP bug (PyTorch issue #65766)
        if torch.is_autocast_enabled():
            torch.clear_autocast_cache()
```

input_pipeline.py M ×

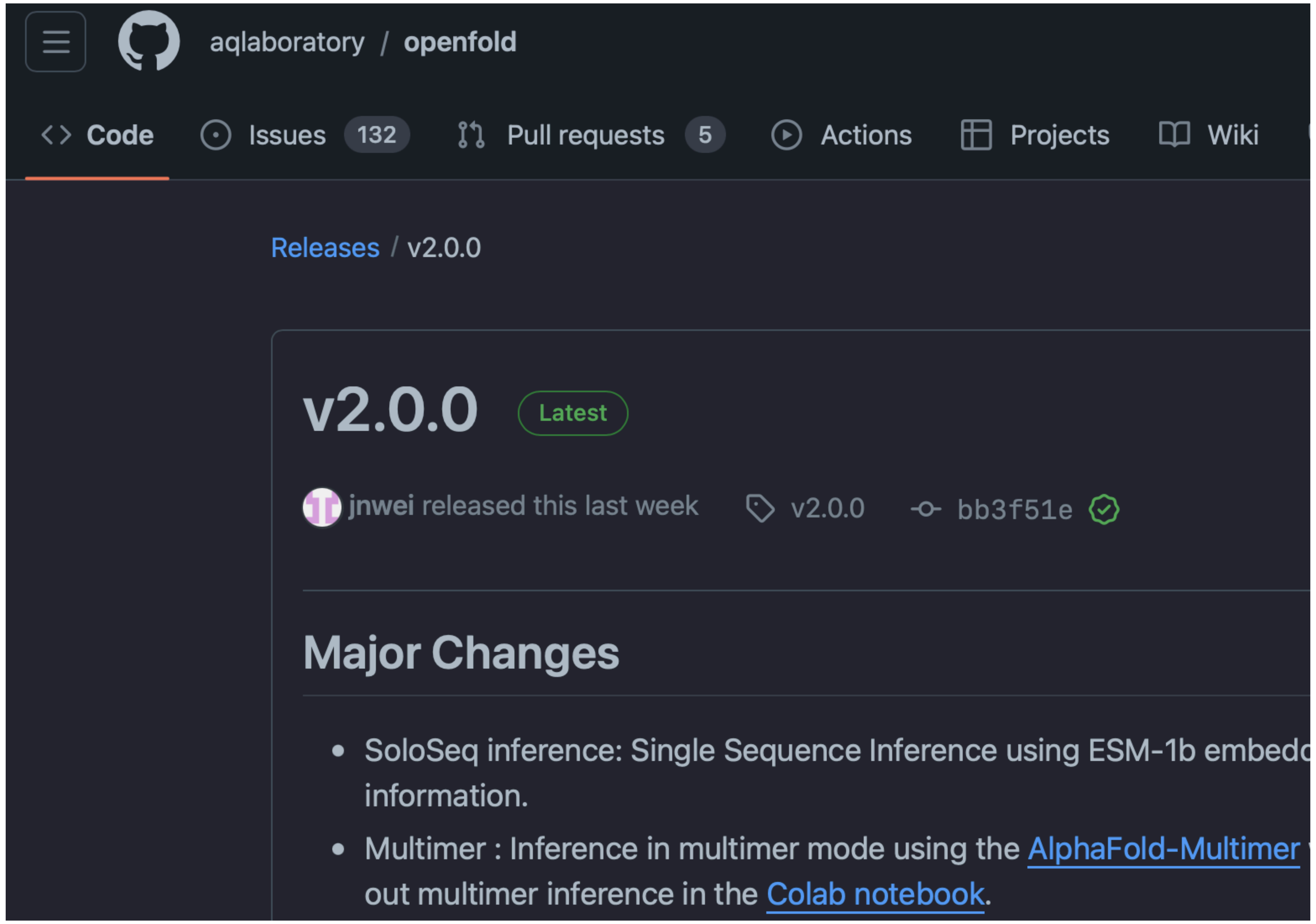
openfold > data > input_pipeline.py > process_tensors_from_config

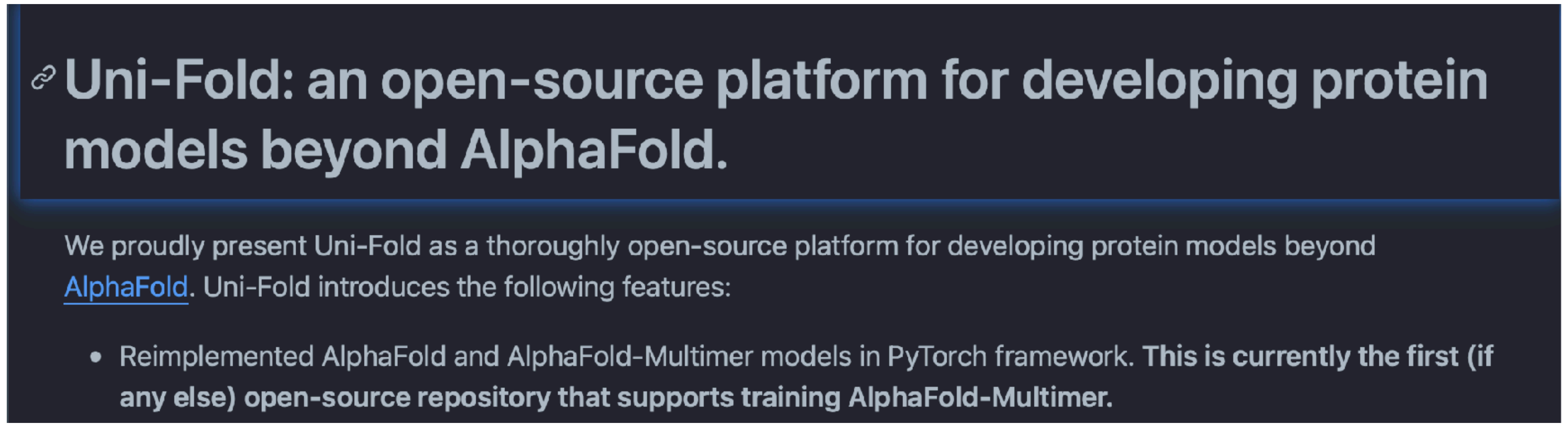
```python

    # TODO: why repeat data processing for each recycle?
    tensors = map_fn(
        lambda x: wrap_ensemble_fn(tensors, x), torch.arange(num_recycling + 1)
    )

    return tensors
```

Upgrade to OpenFold 2.0 to jumpstart multimer support





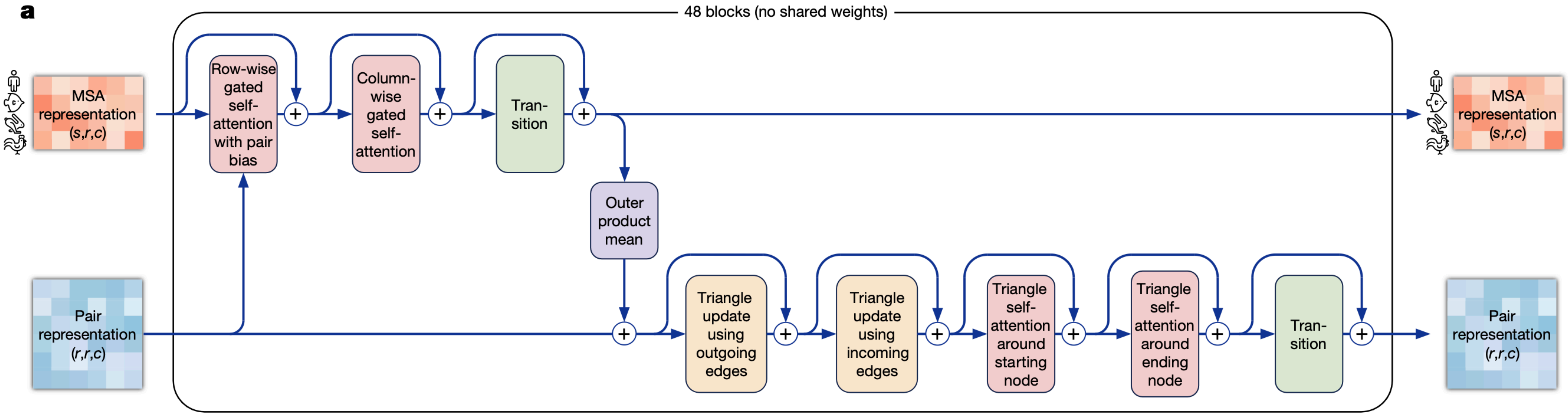
AlphaFold

02.22.24

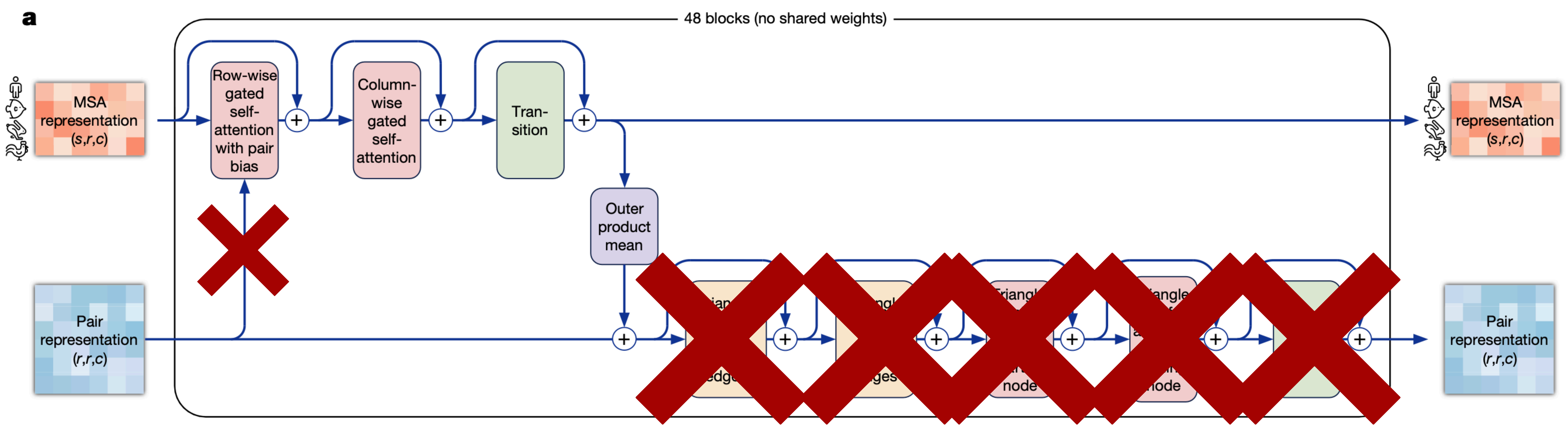
Subsample MSA at Test Time

MSA	Extra MSA: 4	Extra MSA: 64	Extra MSA: 1024
1	0.32	0.35	0.59
4	0.61	0.77	**0.86**
32	0.85	0.87	0.89
128	**0.88**	0.88	0.89

Have: Evoformer



Want: Transformer

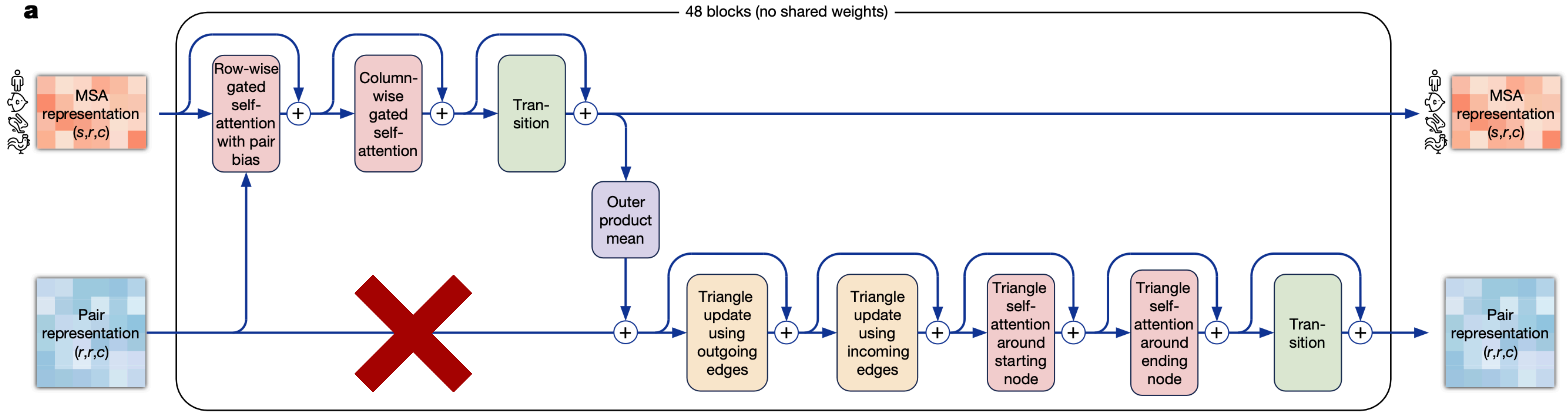


Shallow Pair Representation

	LDDT_CA	GDT_TS
Pair from MSA	0.69	0.49
Baseline	0.78	0.61

Missing

- Relative Positional Embedding
- ExtraMSAStack



Ablate RelPos

Inference Modification	LDDT_CA	GDT_TS
No Rel Pos	0.56	0.43
Baseline	0.89	0.75

Algorithm 4 Relative position encoding

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

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

2: $\mathbf{p}_{ij} = \text{Linear}(\text{one\_hot}(d_{ij}, \mathbf{v}_{\text{bins}}))$

3: **return** $\{\mathbf{p}_{ij}\}$

Likely very important

Should be approximated by ALiBi

Not clipped at 32 though…

Ablate RelPos

Inference Modification	LDDT_CA	GDT_TS
No Rel Pos	0.56	0.43
No Rel Pos - cycle_no <1	0.88	0.74
No Rel Pos - cycle_no <2	0.74	0.59
No Rel Pos - cycle_no <3	0.76	0.61
No Rel Pos - cycle_no =3	0.89	0.75
Baseline	0.89	0.75

Algorithm 4 Relative position encoding

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

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

2: $\mathbf{p}_{ij} = \text{Linear}(\text{one\_hot}(d_{ij}, \mathbf{v}_{\text{bins}}))$

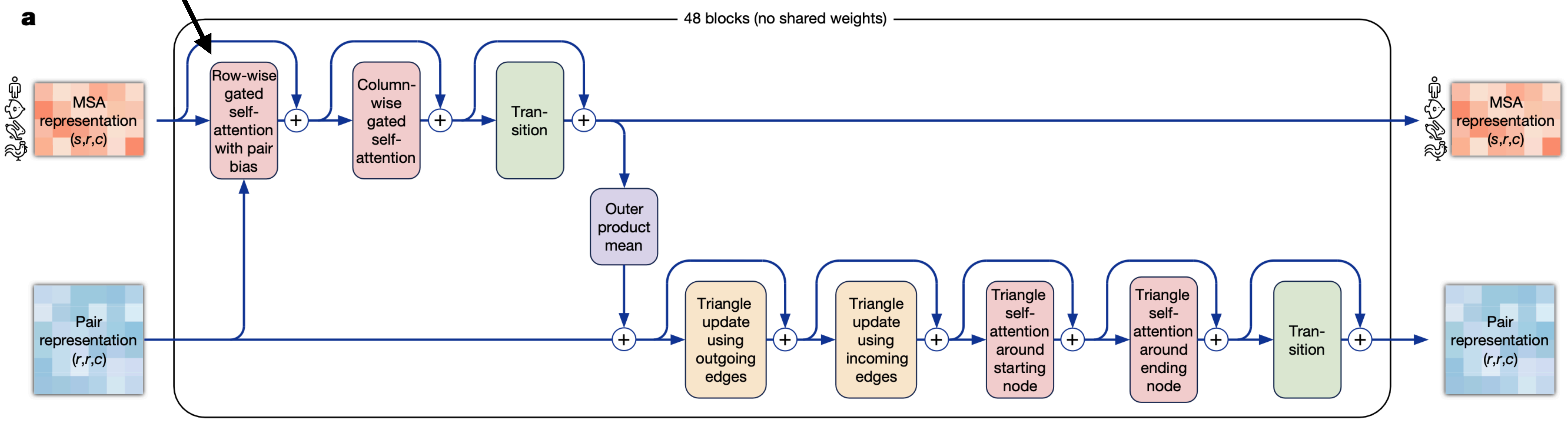
3: **return** $\{\mathbf{p}_{ij}\}$

which iteration needs relpos?

Row Attention -> Mamba

	LDDT_CA	GDT_TS
Mamba	0.77	0.59
Baseline	0.78	0.61

Mamba w/ Pair Bias



Future Work

- Entirely Remove Pair Branch
- Attn: combine row and col attn
- Mamba: combine row and col attn
- Mamba: train on longer sequences (multimer)

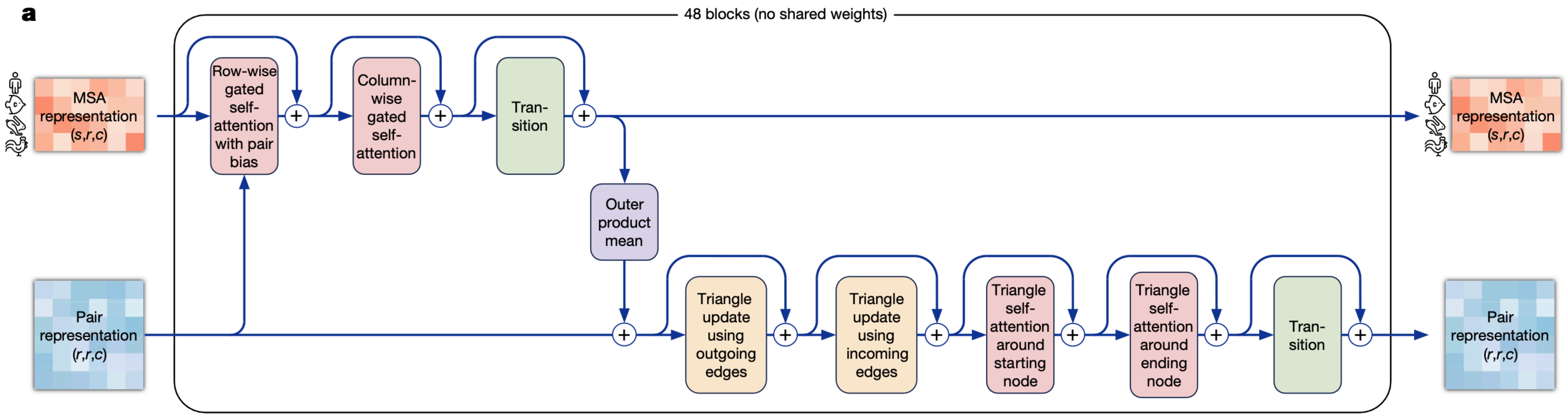
RF-AA Datasets

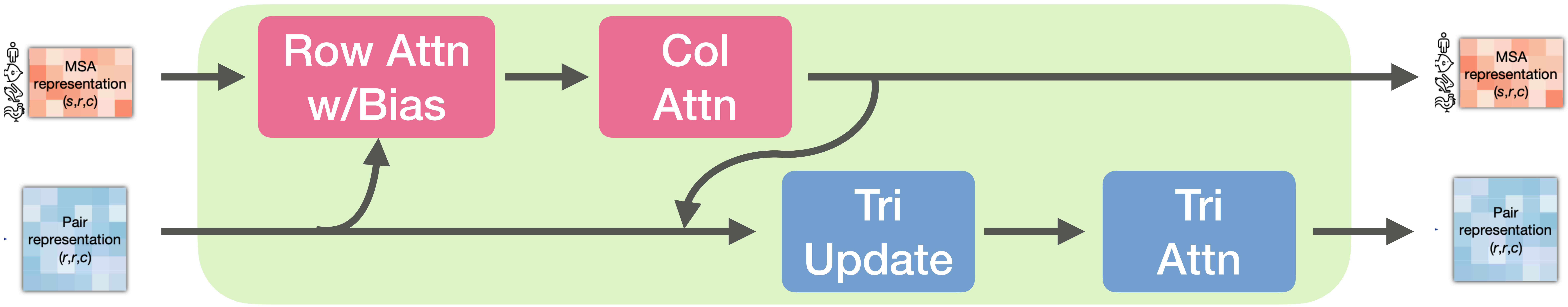
Dataset	Sequence Clusters	Examples
Protein Monomer	21,648	301,934
AF2 Distillation Set	1,036,080	3,605,951
Protein Heteromer	13,755	183,821
Protein Nucleic Acid	1,235	17,240
RNA	1,449	6,522
Protein Small Molecule	5,662	121,800
Protein Metal Complex	5324	112,456
Protein Multi-Residue Ligand	613	4,775
Protein Small Molecule Assembly	2,564	43,838
Covalent Modification	1,099	12,689

MambaFold

02.29.24

Evoformer

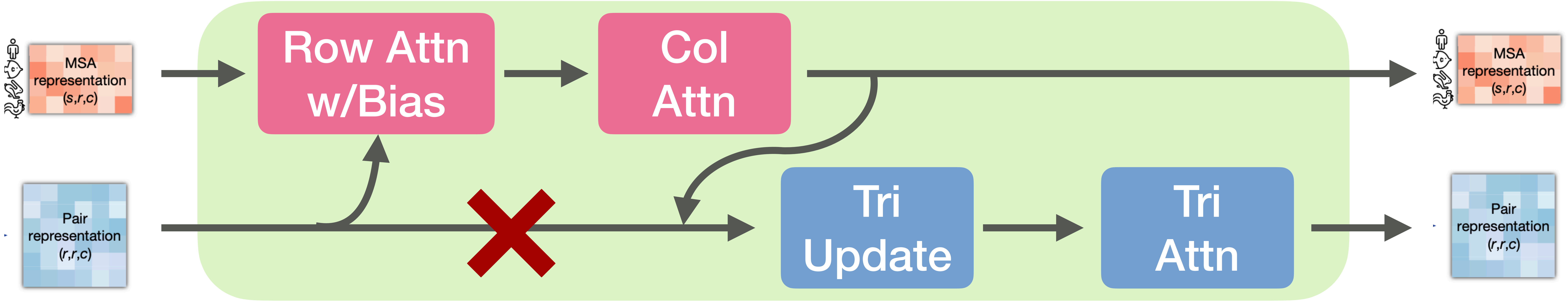

a
48 blocks (no shared weights)
MSA representation (s,r,c)
Pair representation (r,r,c)
Row-wise gated self-attention with pair bias
Column-wise gated self-attention
Tran-sition
Outer product mean
Triangle update using outgoing edges
Triangle update using incoming edges
Triangle self-attention around starting node
Triangle self-attention around ending node
Tran-sition
MSA representation (s,r,c)
Pair representation (r,r,c)


MSA representation (s,r,c)
Pair representation (r,r,c)
Row Attn w/Bias
Col Attn
Tri Update
Tri Attn
MSA representation (s,r,c)
Pair representation (r,r,c)

Trying to remove pair repr

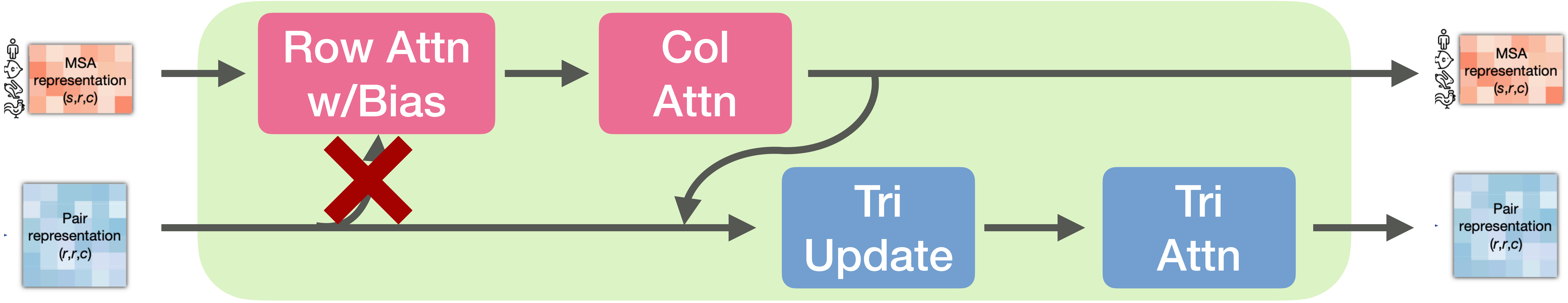
	Rel Pos Emb	LDDT_CA	GDT_TS
No residual	n	0.69	0.49
+ pos embed + OPM w/LN	y	0.73	0.53
Baseline	y	0.78	0.61

Performance drop from
- lack of ExtraMSAStack
- deep pair representation



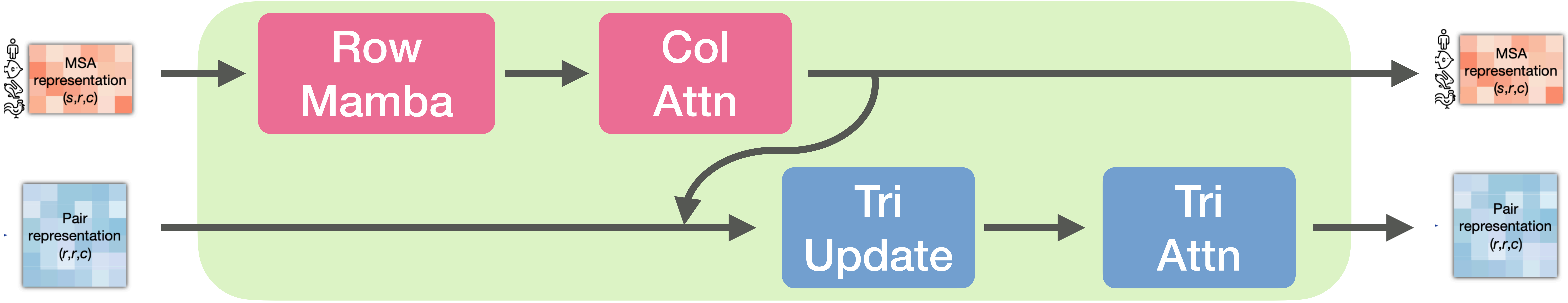
Removing pair bias?

	Inference Change	LDDT_CA	GDT_TS
AF	No Pair Bias	0.38	0.22
RowMambaWithPairBias	No Pair Bias	0.76	0.58
RowMambaWithPairBias	N/A	0.77	0.59



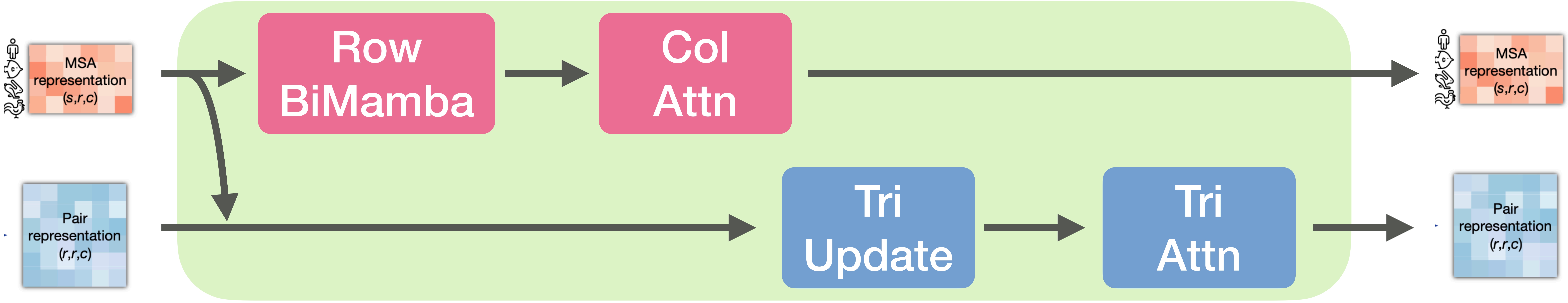
Row Attention -> Mamba

	Pair Bias	LDDT_CA	GDT_TS
Mamba	N	0.76	0.58
Mamba	Y	0.77	0.59
AF	Y	0.78	0.61

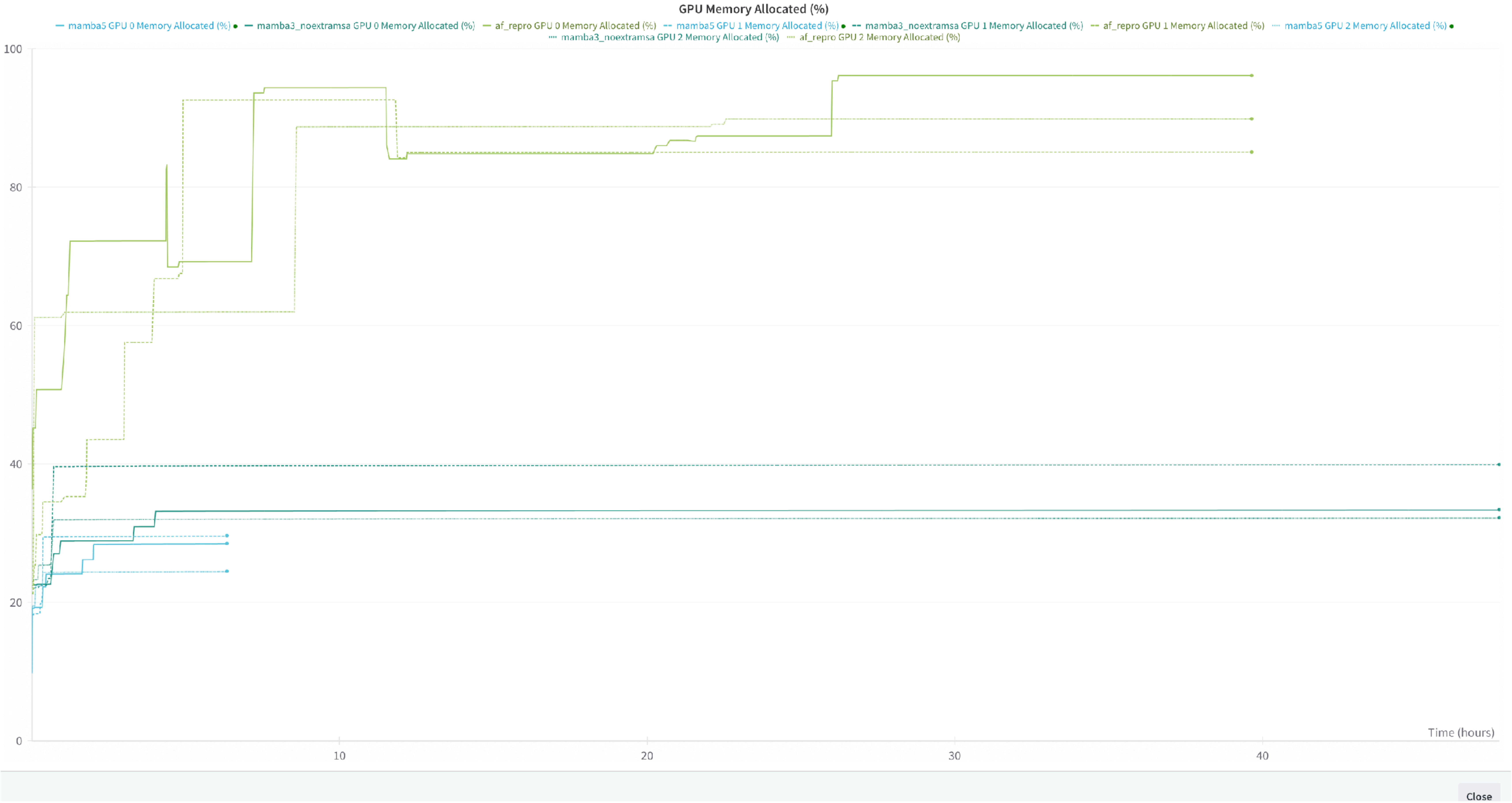


Bidirectional Mamba

	Bidirectional	LDDT_CA	GDT_TS
Mamba	N	0.76	0.58
Mamba	Y	0.77	0.59



40% memory

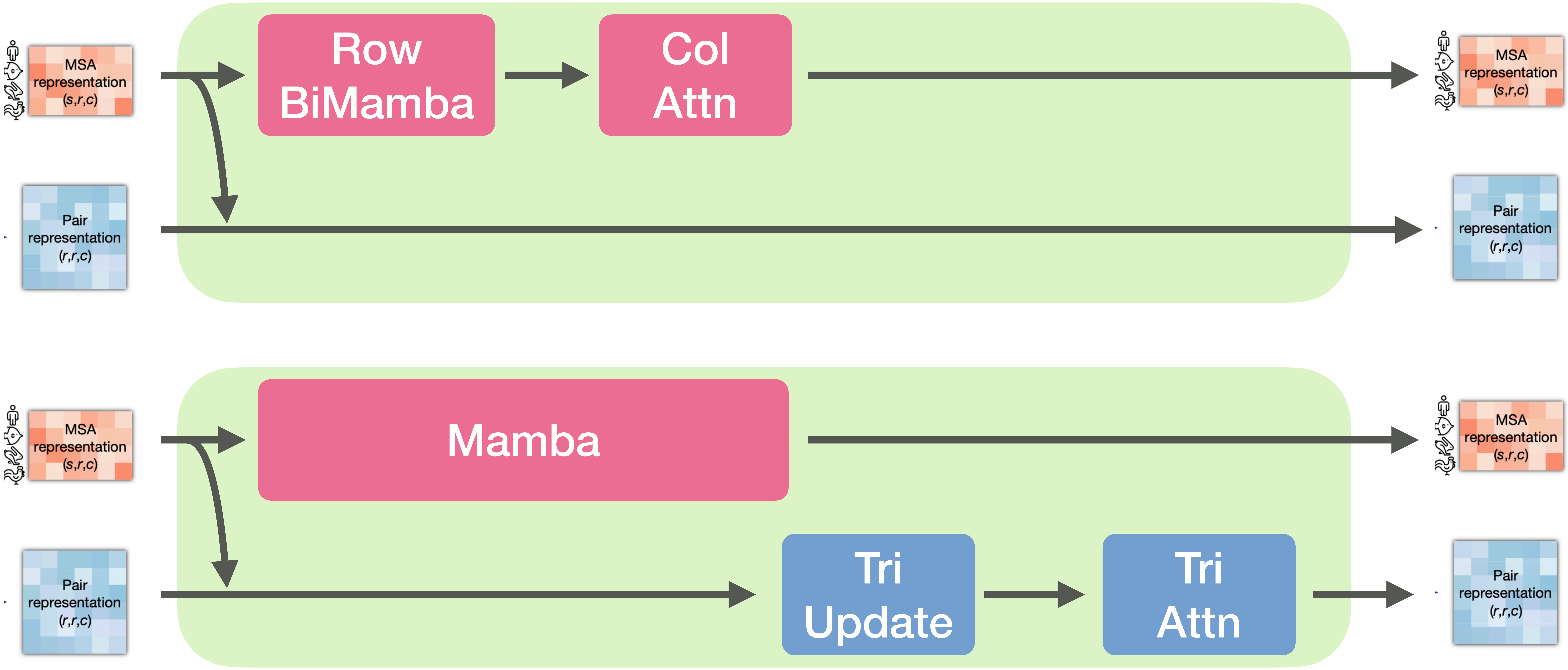


GPU Memory Allocated (%)
mamba5 GPU 0 Memory Allocated (%)
mamba3_noextramsa GPU 0 Memory Allocated (%)
af_repro GPU 0 Memory Allocated (%)
mamba5 GPU 1 Memory Allocated (%)
mamba3_noextramsa GPU 1 Memory Allocated (%)
af_repro GPU 1 Memory Allocated (%)
mamba5 GPU 2 Memory Allocated (%)
mamba3_noextramsa GPU 2 Memory Allocated (%)
af_repro GPU 2 Memory Allocated (%)
100
80
60
40
20
0
10
20
30
40
Time (hours)
Close

Speedups

- Baseline: 10s/iter
- Crop sequence length: 9s/iter
- Load cached pkl: 6s/iter
- More workers: TODO

current experiments



Intuition for GDT-TS

- Compare AF vs AF repro
- AF LDDT: 0.9
- AF repro LDDT: 0.7
- Where is the difference? loops, tertiary structures

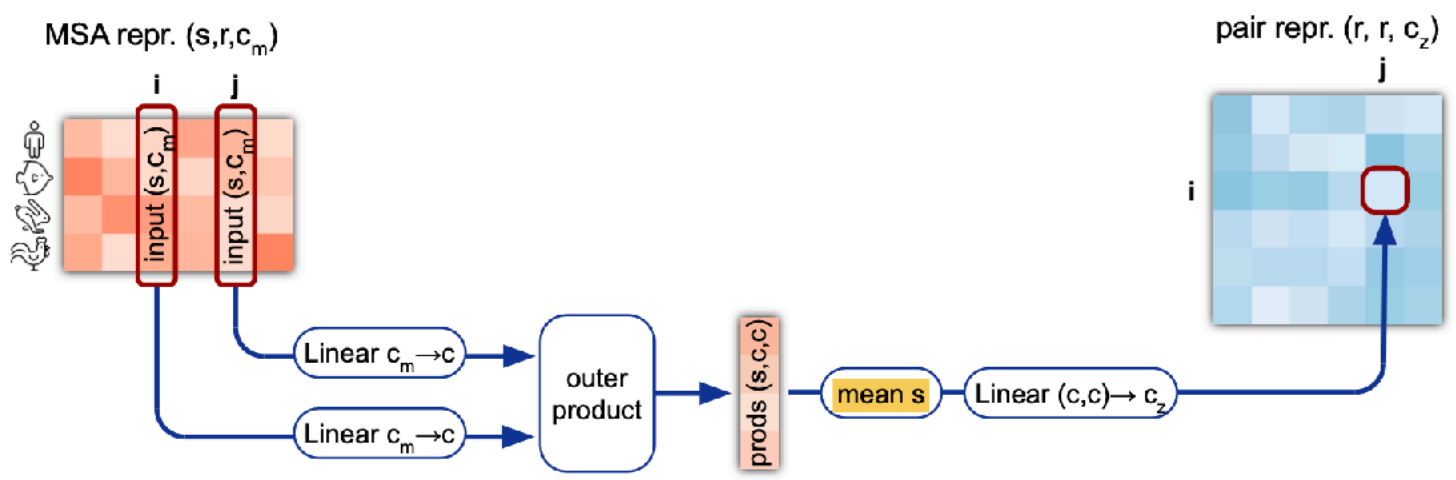
MambaFold

3.3.24

Why keep pair?

- Structure module needs good pair value features
  - (32 per residue is sufficient - can find which w/o pair bias)
  - Pair model only computes good pair value features
- Model skipping triangle ops sucks
  - H1: triangle ops are more expressive
  - H2: MSA -> Pair is lossy
  - H3: MSA ops are less expressive

Outer Product Mean



Supplementary Figure 5 | Outer product mean. Dimensions: s: sequences, r: residues, c: channels.

Algorithm 10 Outer product mean

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

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

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

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

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

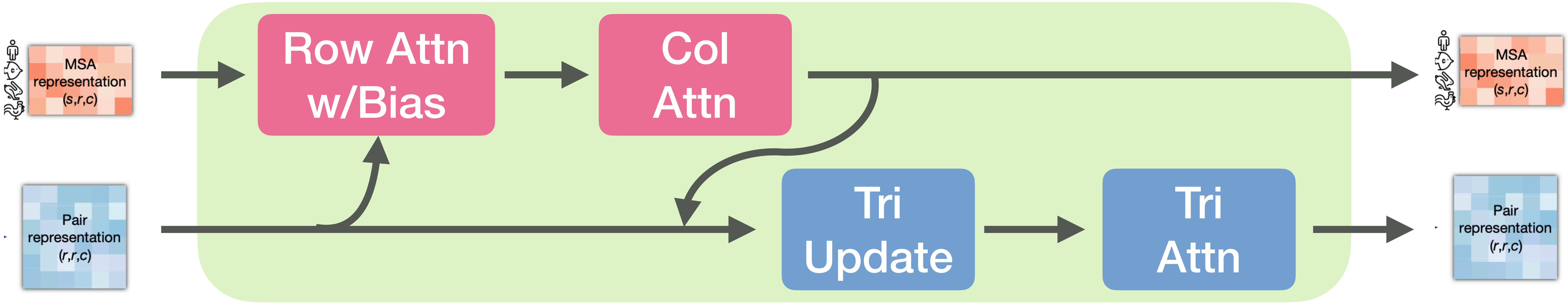
return $\{\mathbf{z}_{ij}\}$

- averaged across MSA
- unnormalized outputs
- no positional embedding

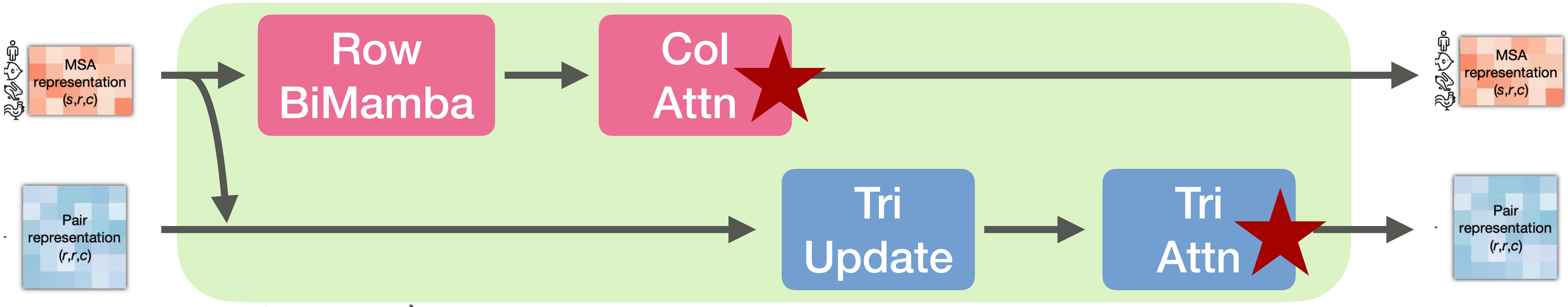
MambaFold

3.3.24

AlphaFold Backbone



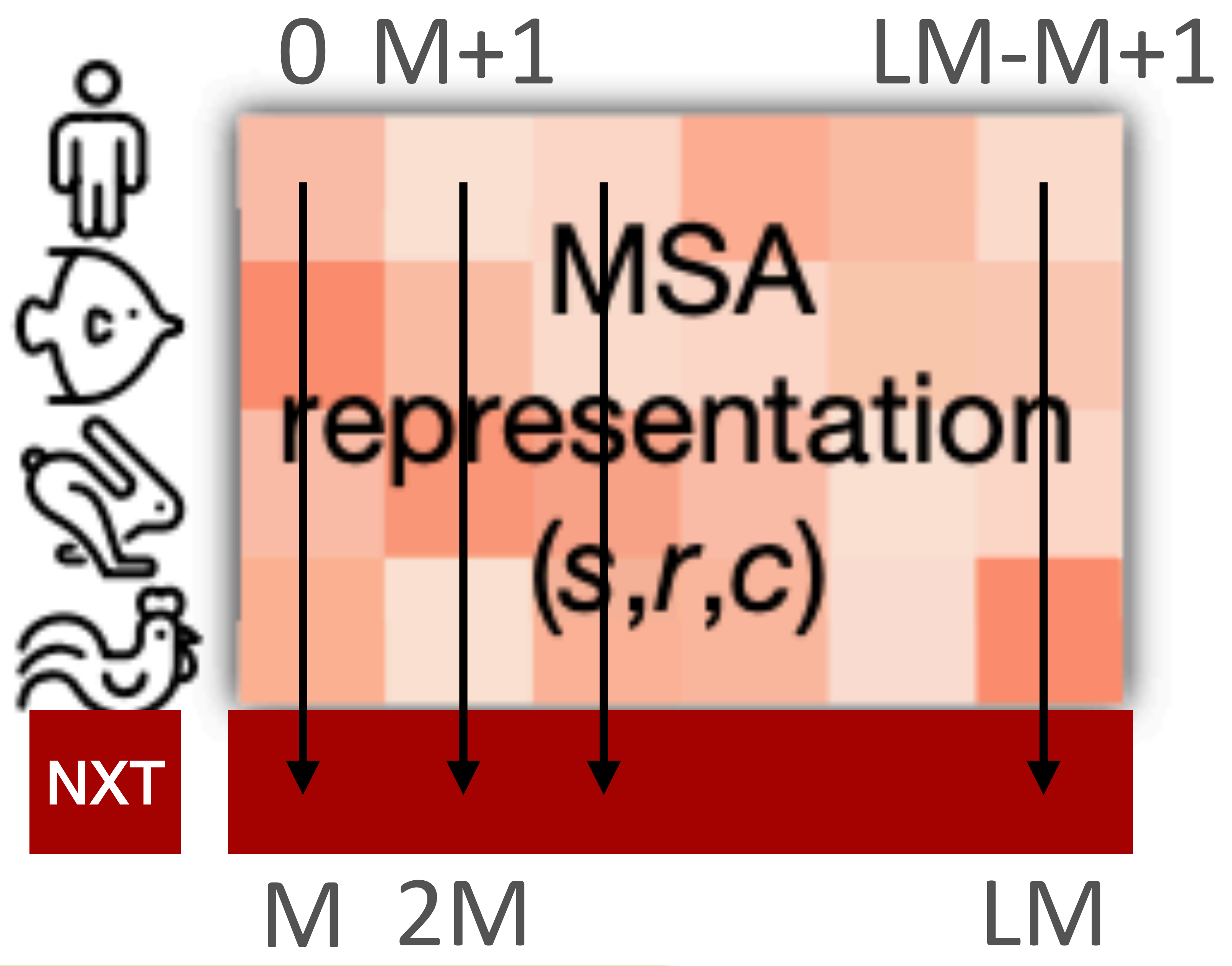
Current Model

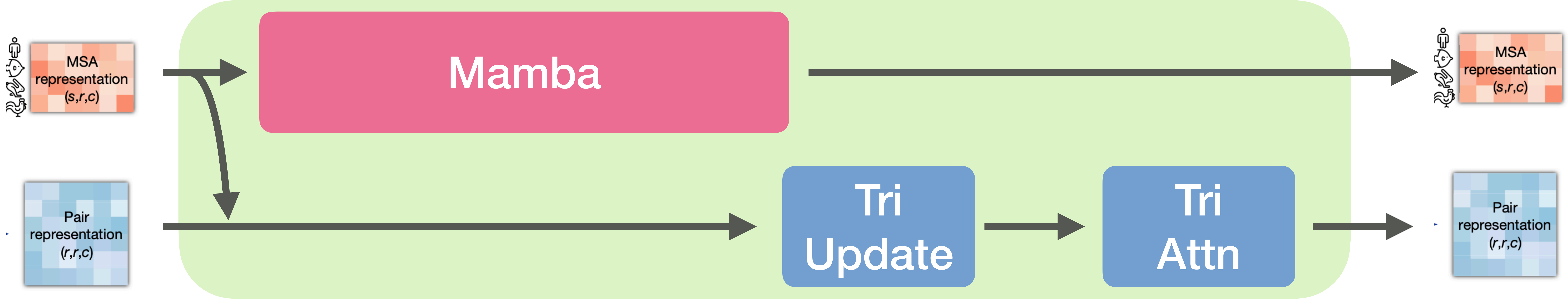


(Also No Extra MSA Input)

Merge Row/Col

	LDDT_CA	GDT_TS	Train Time
Row Mamba - Col Attn	0.77	0.59	2.5d
Mamba Combined	0.71 (-0.06)	0.51 (-0.08)	3d

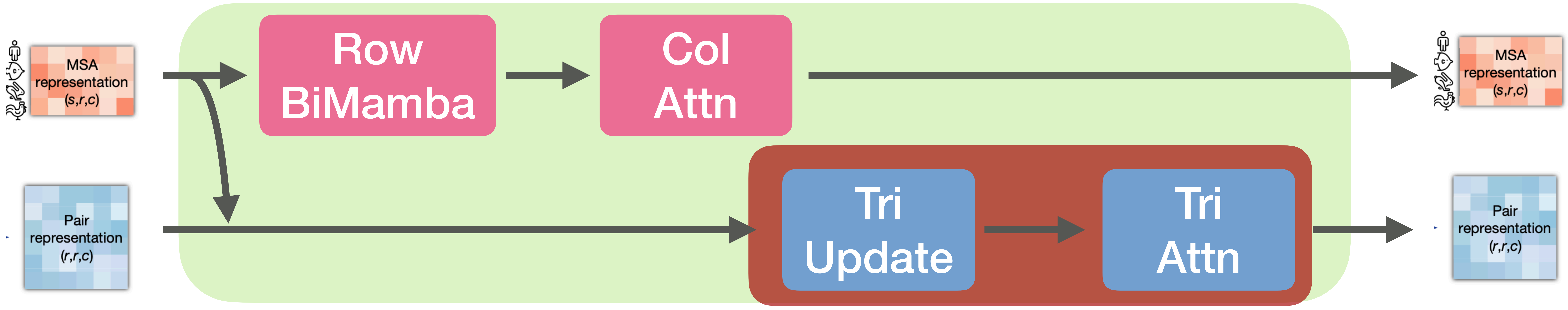




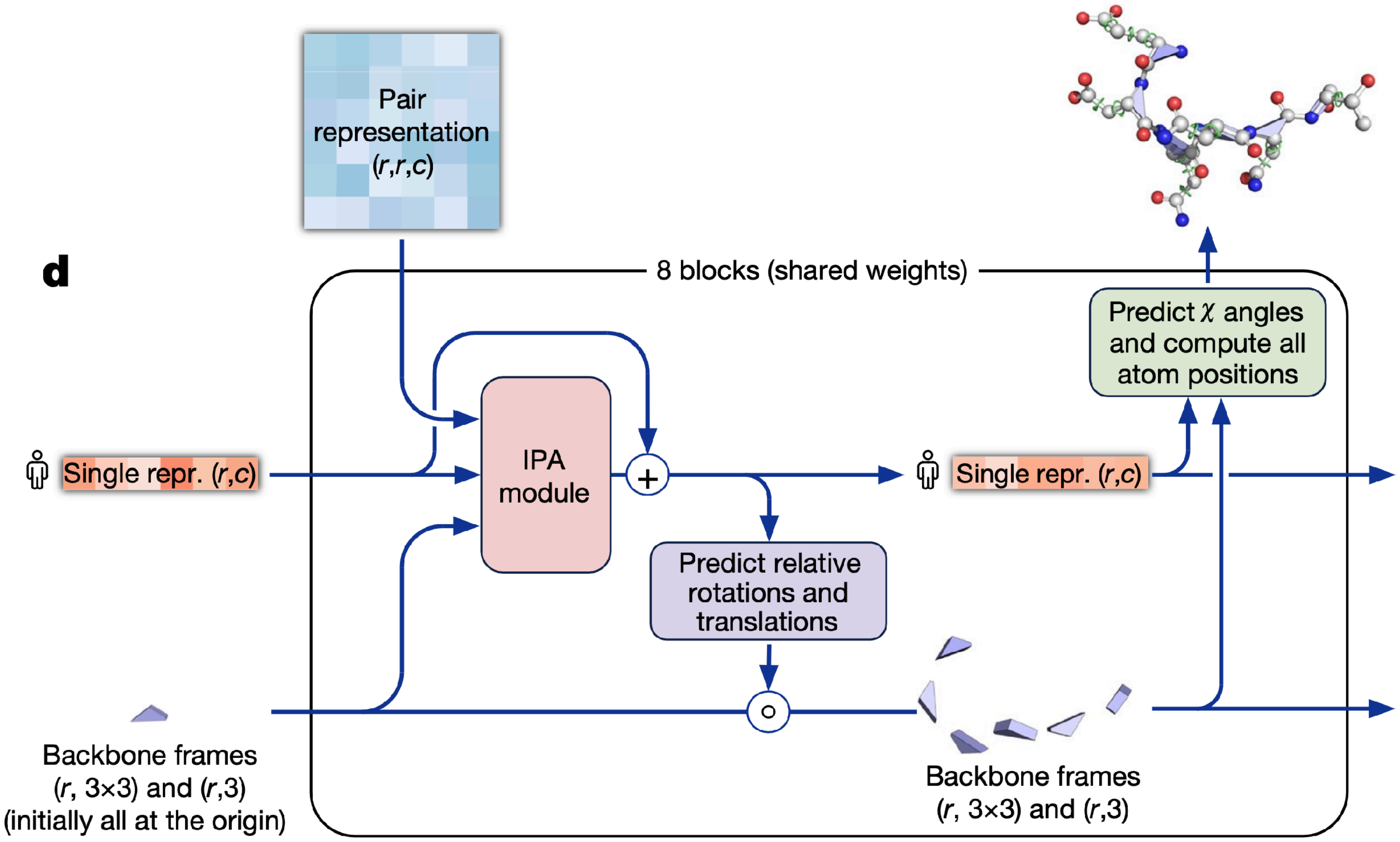
Simplify Pair Branch

Tri Update	Tri Attn	LDDT_CA	GDT_TS	Train Time
Y	Y	0.77	0.59	2.5d
N	Y	0.77	0.59	2d
Y	N	0.74	0.54	2d
N	N	0.35*	0.06	1.2d

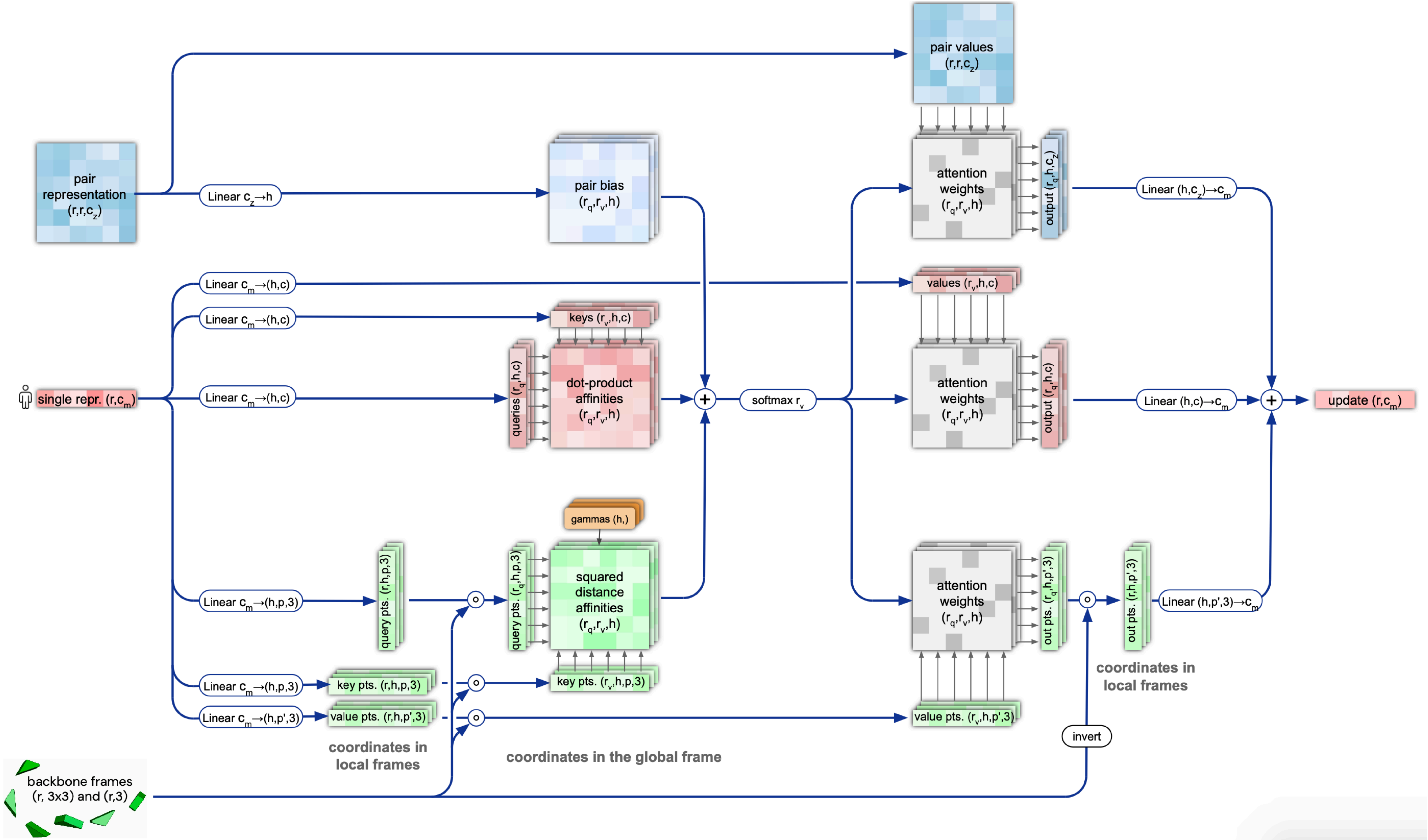
- Tried modifying the outer product as well



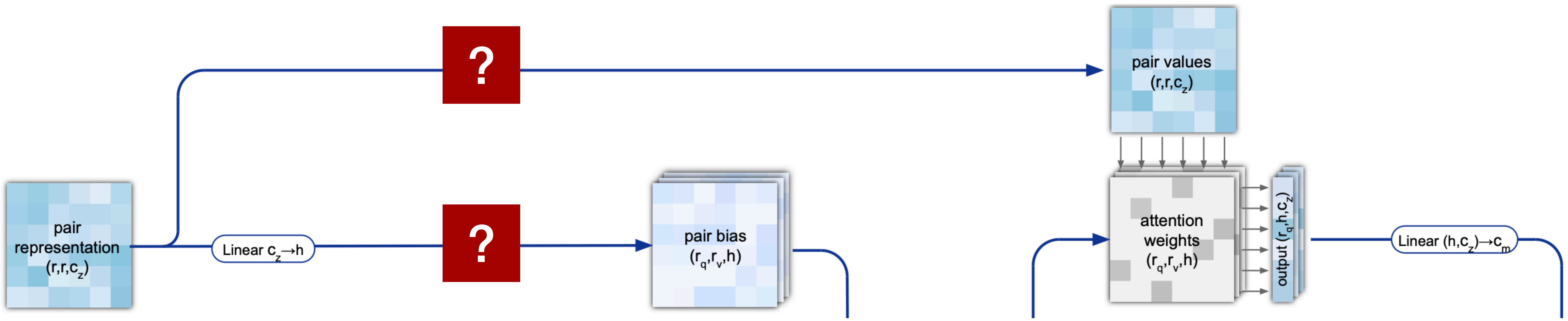
Structure Module



Invariant Point Attention



Invariant Point Attention



Model	Inference	LDDT_CA	GDT_TS
Mamba3	N/A	0.77	0.59
Mamba3	No Pair Bias	**0.76**	**0.58**
Mamba3	No Pair Values	0.06	0.05
AF (long)	N/A	0.89	0.75
AF (long)	No Pair Bias	0.11	0.02
AF (long)	No Pair Values	0.42	0.46

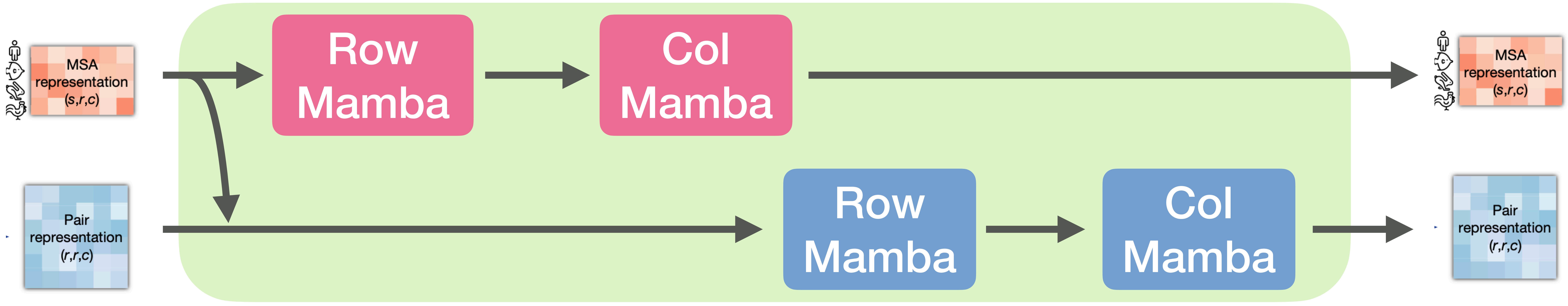
Pair Values are important!
No Pair bias w/Mamba

MambaFold

3.16.24

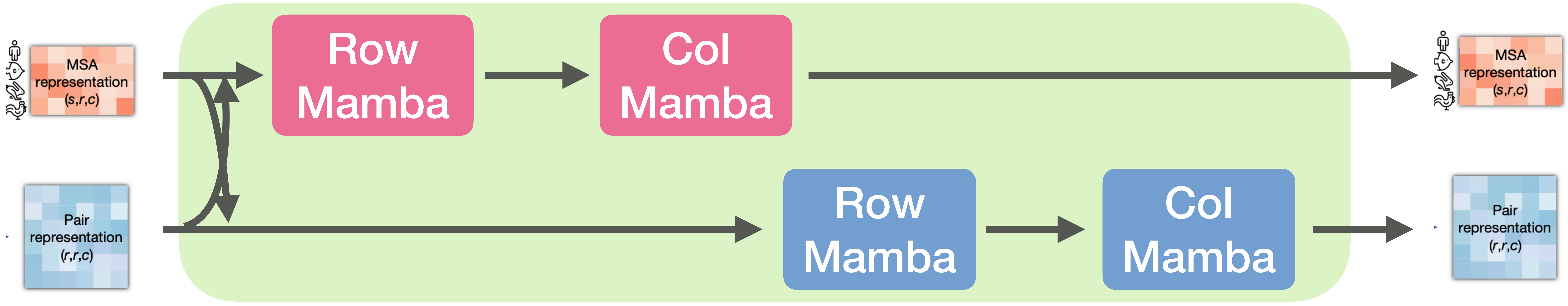
Mambafy

Col	Tri	LDDT_CA	GDT_TS	Train Time	Memory
Attn	Attn	0.77	0.59	2d	45%
Mamba	Attn	0.76	0.57	2.1d	33%
Mamba	Mamba	0.43			31%

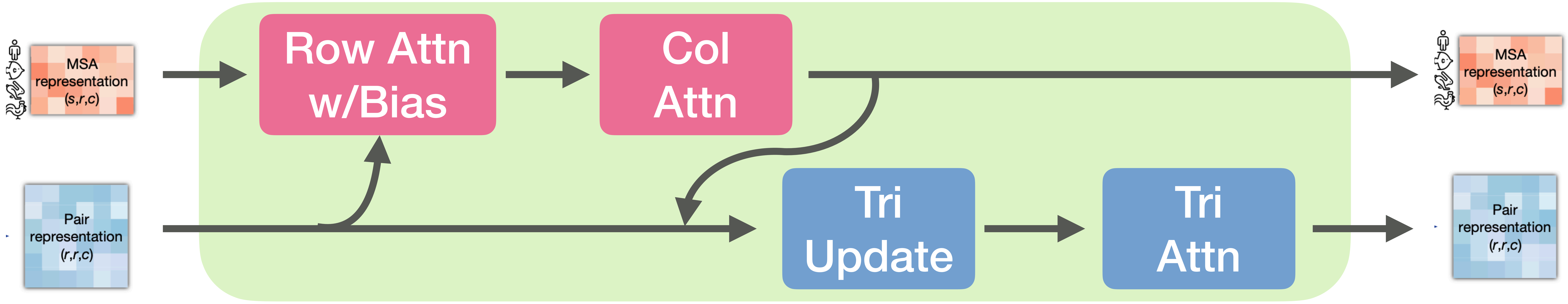


Pair to MSA

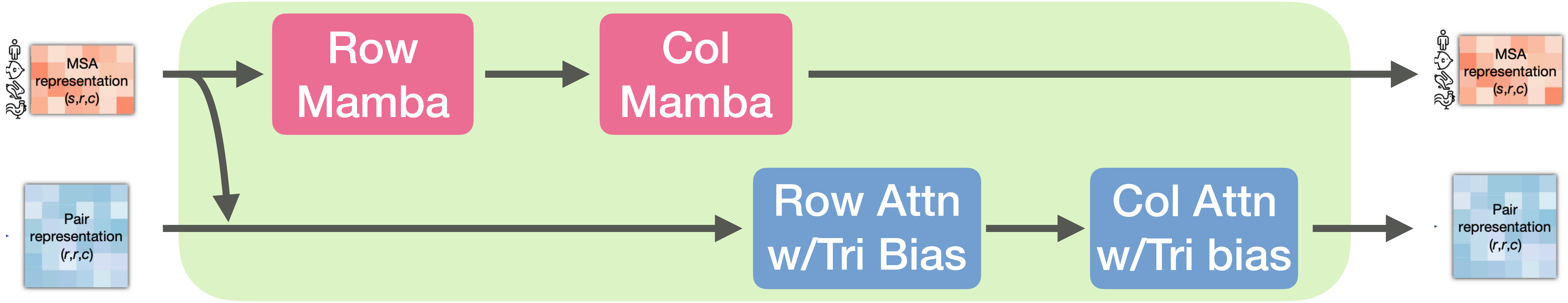
Col	Tri	LDDT_CA	GDT_TS	Train Time	Memory	Iter
Mamba	Attn	0.76	0.57	2.1d	33%	7000
		0.17	0.03			4200



AlphaFold Backbone



Current Model



(Also No Extra MSA Input)

can probably sparsify attn but do later

Improvements

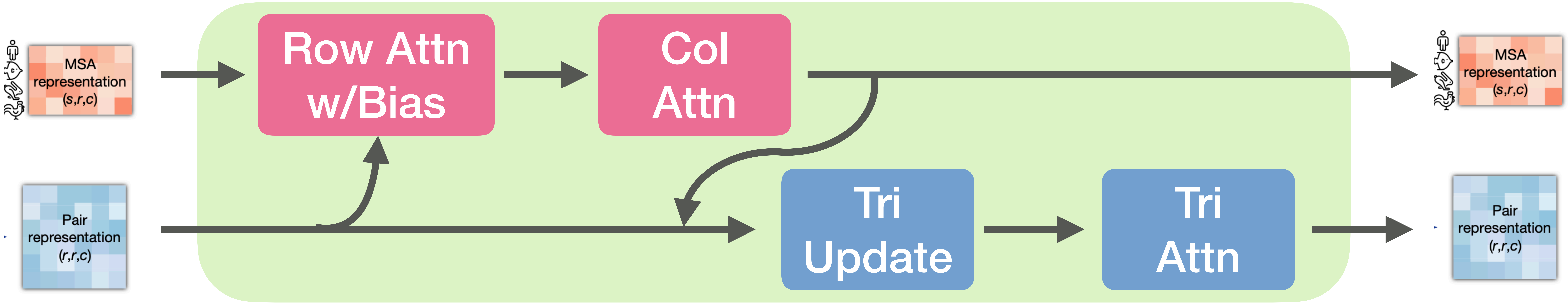
- Row Attn -> Mamba
- Col Attn -> Mamba
- Outer Product - first + LN
- No triangle multiplication

- No extra MSA
- No template
- Max pre MSA=2048
- LION lr1e-4

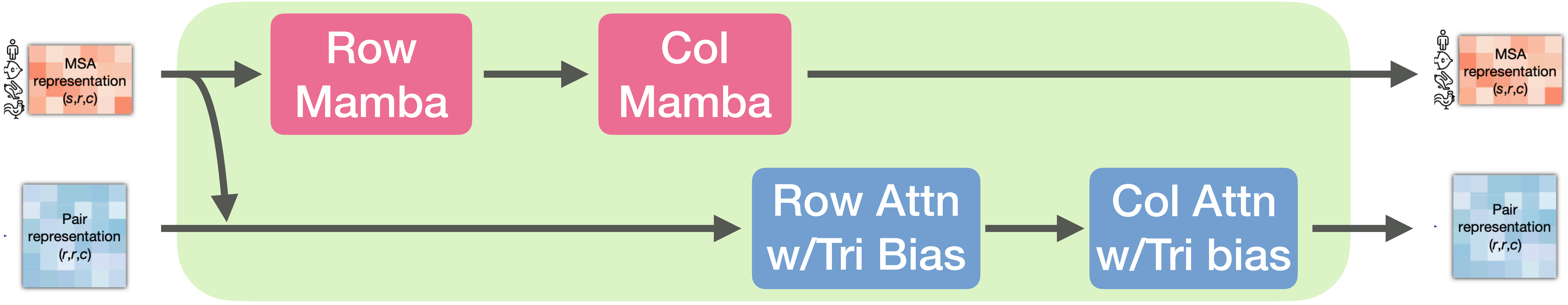
MambaFold

3.23.24

AlphaFold Backbone



Current Model

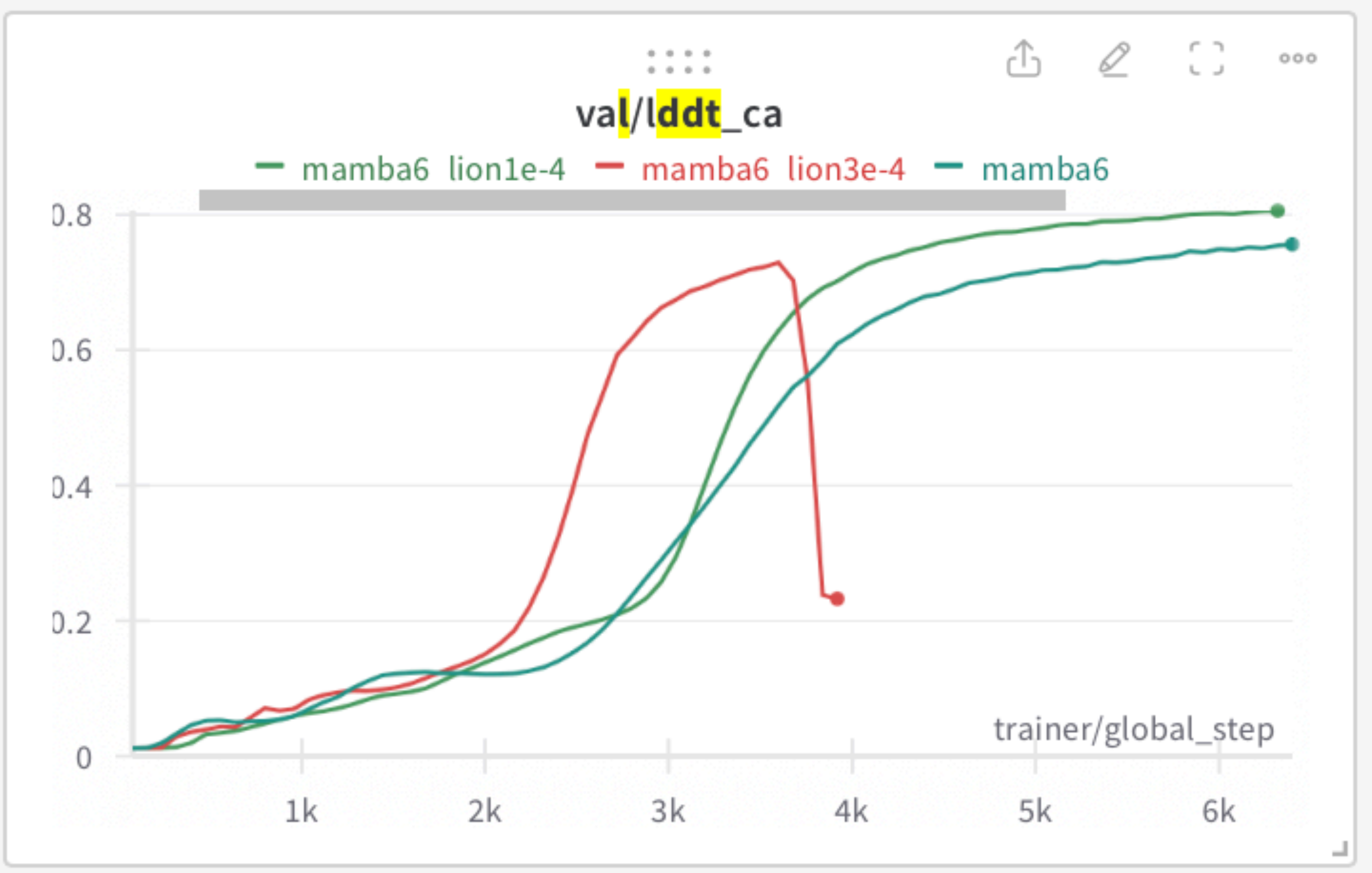


(Also No Extra MSA Input)

can probably sparsify attn but do later

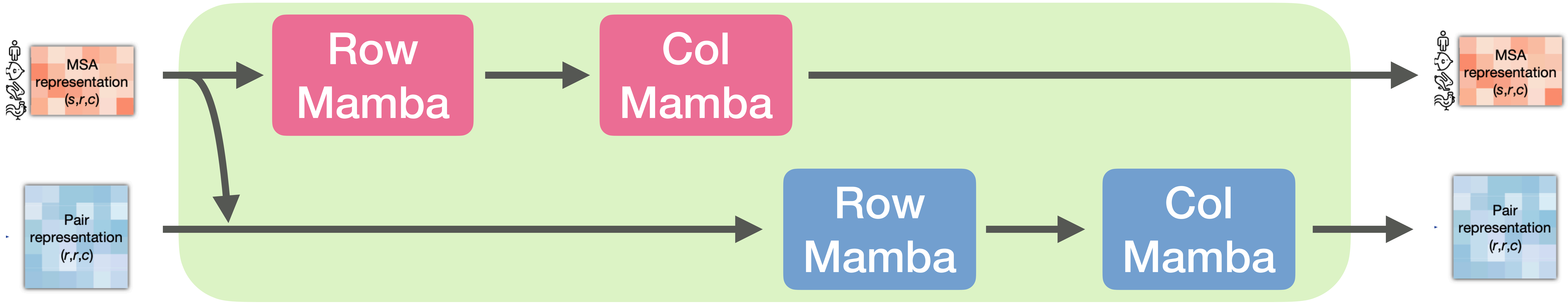
LIONify

Optimizer	LR	LDDT_CA	GDT_TS	Steps
AdamW	1E-03	0.77	0.59	7000
Lion	3E-03	NaN		
Lion	1E-04	**0.81**	**0.63**	6300



Mambafy Triangle v2

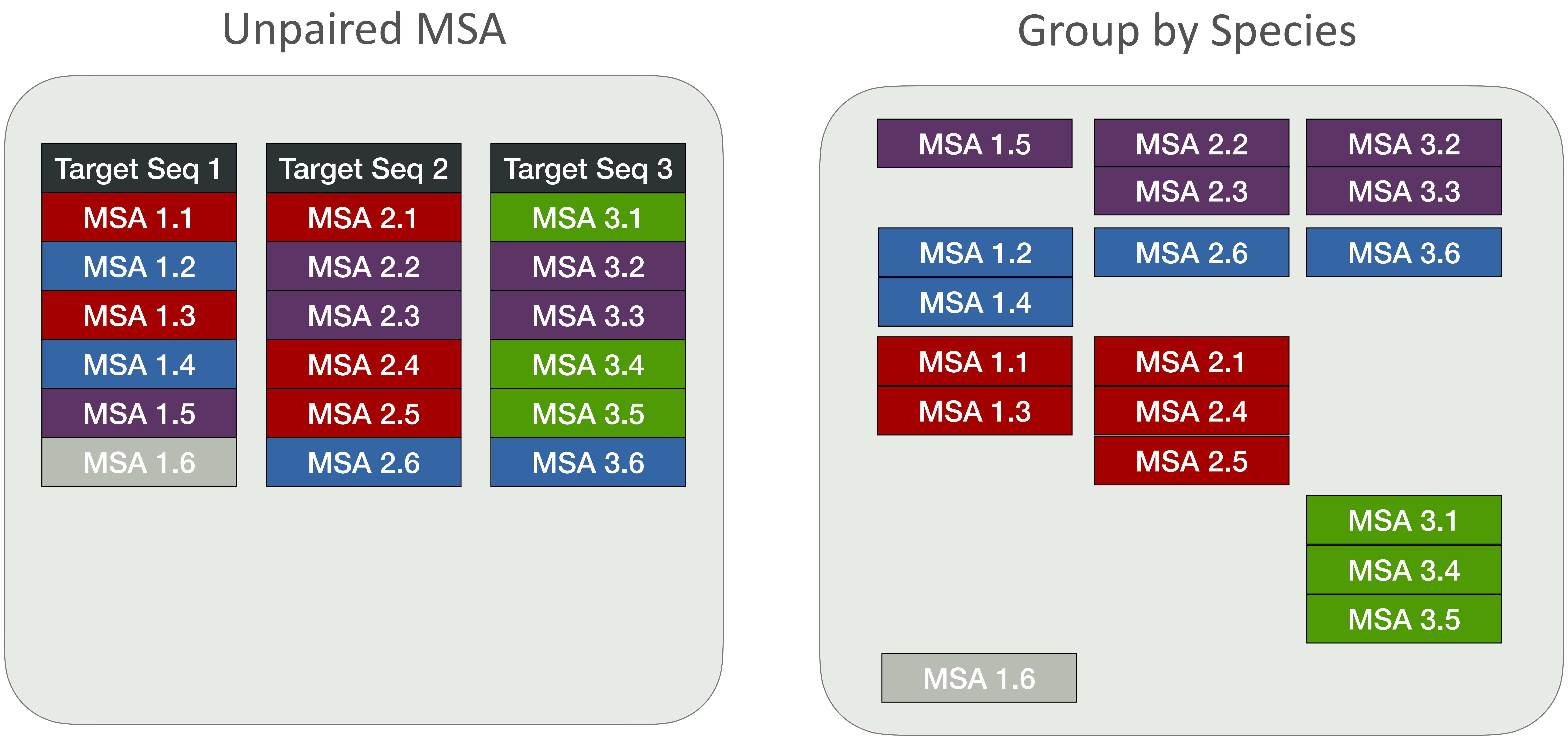
Tri	LDDT_CA	GDT_TS	Train Time
Attn	0.81	0.63	?
Mamba Plain	0.43*	0.15*	
Gating B	0.67*	0.44*	
Gating dt			
Gating B,C,dt			



MSA Pairing

Unpaired MSA
Target Seq 1
MSA 1.1
MSA 1.2
MSA 1.3
MSA 1.4
MSA 1.5
MSA 1.6
Target Seq 2
MSA 2.1
MSA 2.2
MSA 2.3
MSA 2.4
MSA 2.5
MSA 2.6
Group by Species
MSA 1.1
MSA 1.3
MSA 2.1
MSA 2.4
MSA 2.5
MSA 1.5
MSA 2.2
MSA 2.3
MSA 1.2
MSA 1.4
MSA 2.6
MSA 1.6
Filter to Species Min
MSA 1.1
MSA 1.3
MSA 2.1
MSA 2.4
MSA 1.5
MSA 2.2
MSA 1.2
MSA 2.6
Final MSA
Target Seq 1
MSA 1.1
MSA 1.3
MSA 1.5
MSA 1.2
MSA 1.4
MSA 1.6
Target Seq 2
MSA 2.1
MSA 2.4
MSA 2.2
MSA 2.6
MSA 2.3
MSA 2.5

MSA Pairing with more MSAs



RowAttn ignores pairing

MSA Pairing	LDDT-CA	GDT-TS	GDT-HA
Default	0.88	0.60	0.45
Shuffle paired	0.88	0.60	0.45
Asym mask	0.88	0.61	0.45
Sym mask	0.88	0.60	0.45

missing backbone angle loss

```
class AlphaFoldLoss(nn.Module):
    def forward(self, out, batch, _return_breakdown=False):
```

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

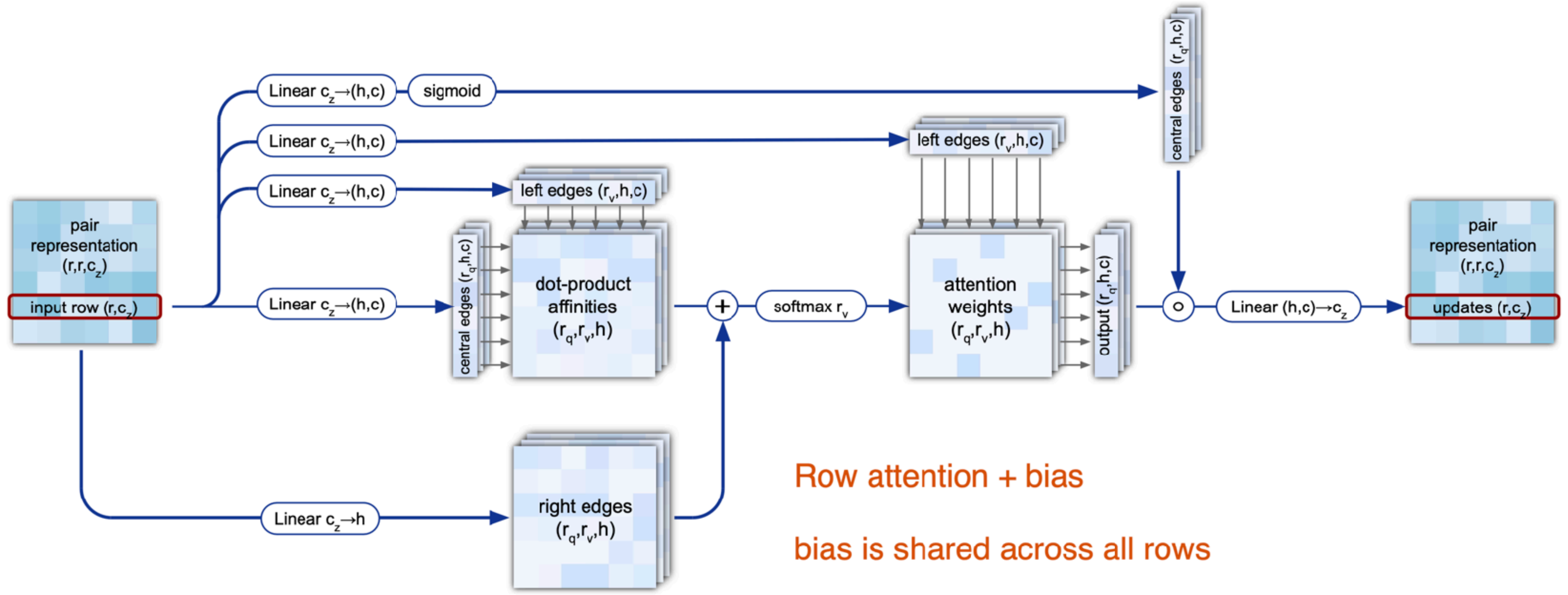
5: **return** $\mathcal{L}_{\text{torsion}} + 0.02\mathcal{L}_{\text{anglenorm}}$

```
            "supervised_chi": lambda: supervised_chi_loss(
                out["sm"]["angles"],
                out["sm"]["unnormalized_angles"],
                **{**batch, **self.config.supervised_chi},
            ),
```

MambaFold

4.4.24

Mambify Triangle v2

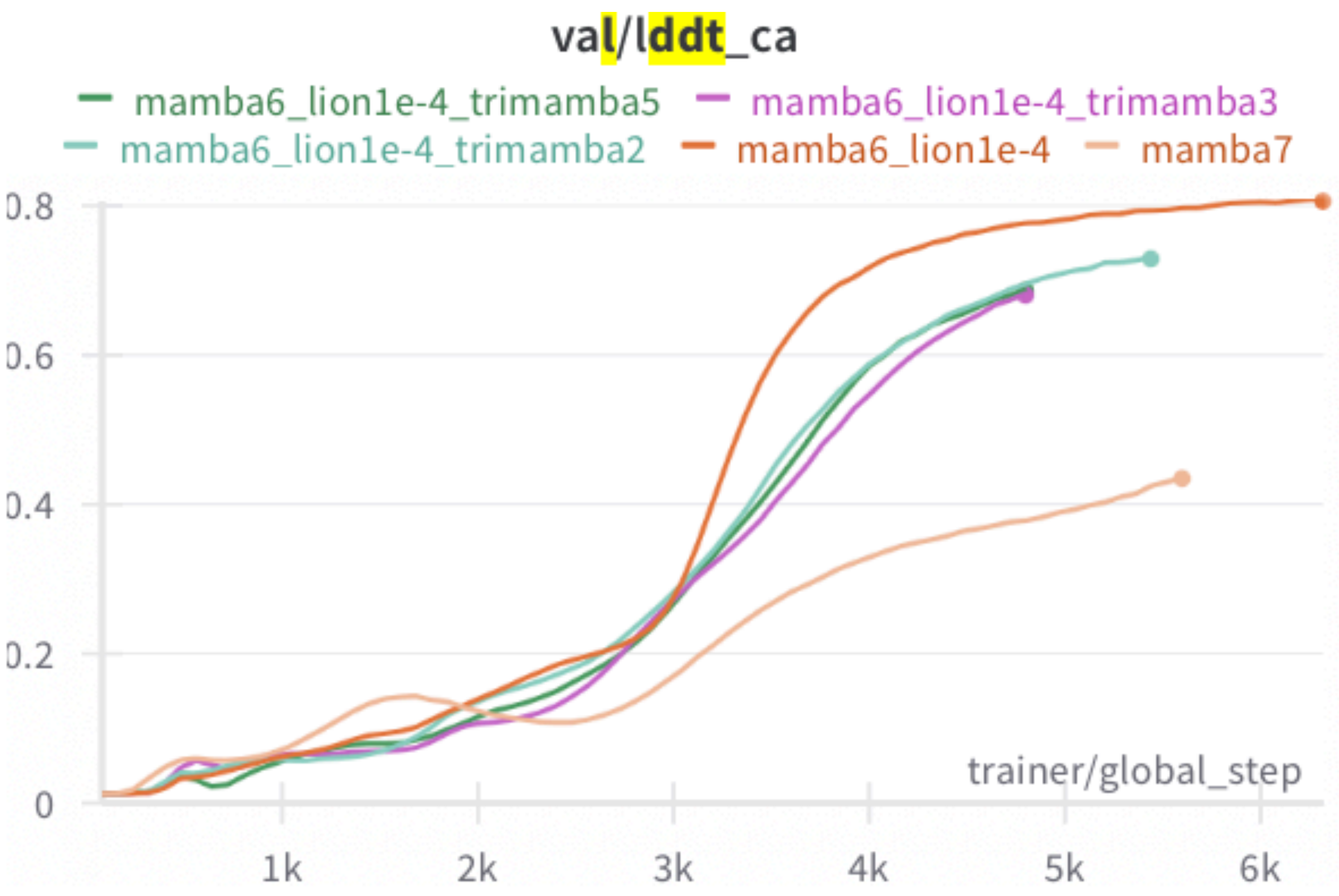


$$\overline{A} = \exp(\Delta A)$$

$$\overline{B} = (\Delta A)^{-1}(\exp(\Delta A) - I) \cdot \Delta B$$

$$h_t = \overline{A} h_{t-1} + \overline{B} x_t$$

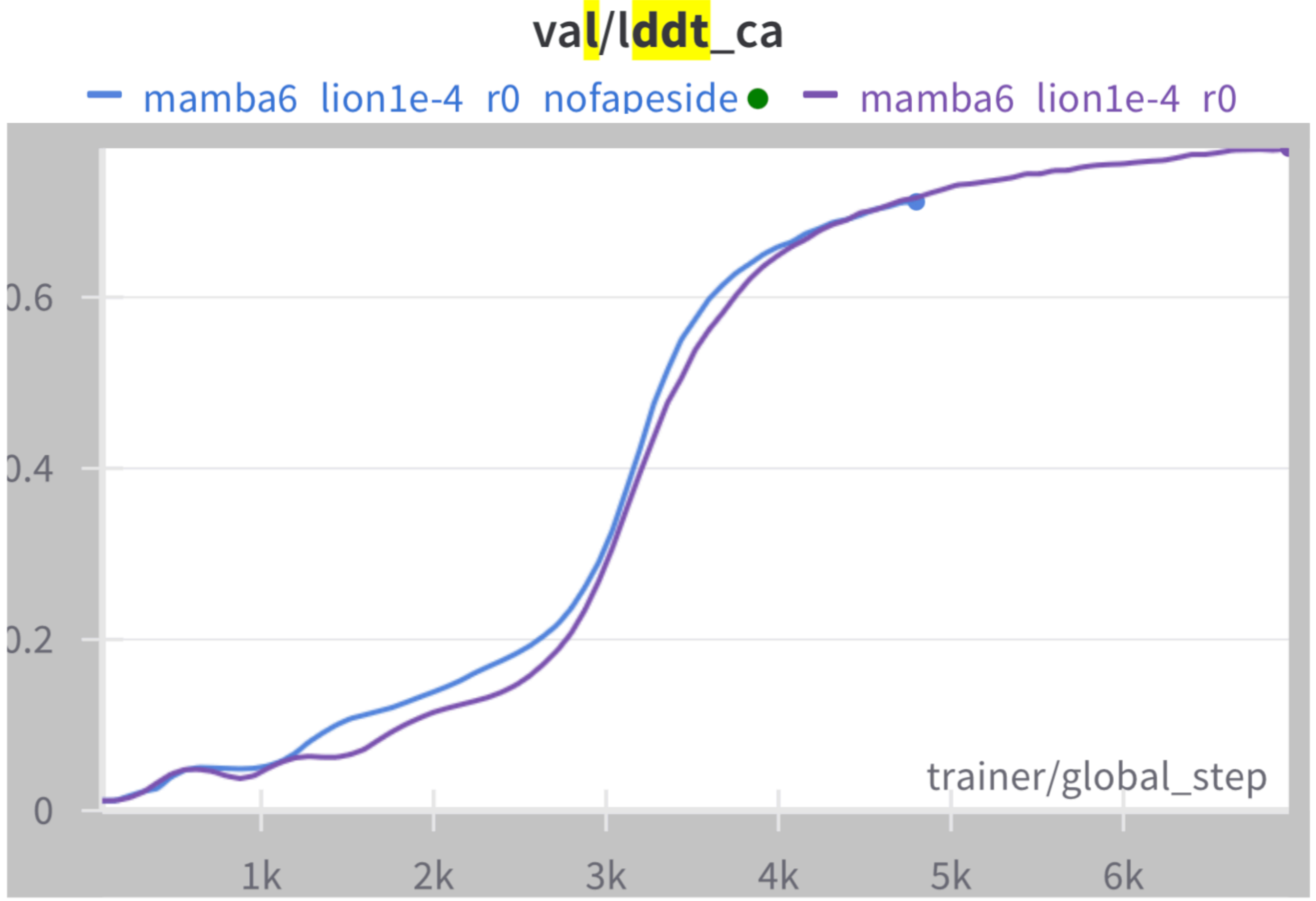
$$y_t = C h_t$$



FAPE on side chain not needed

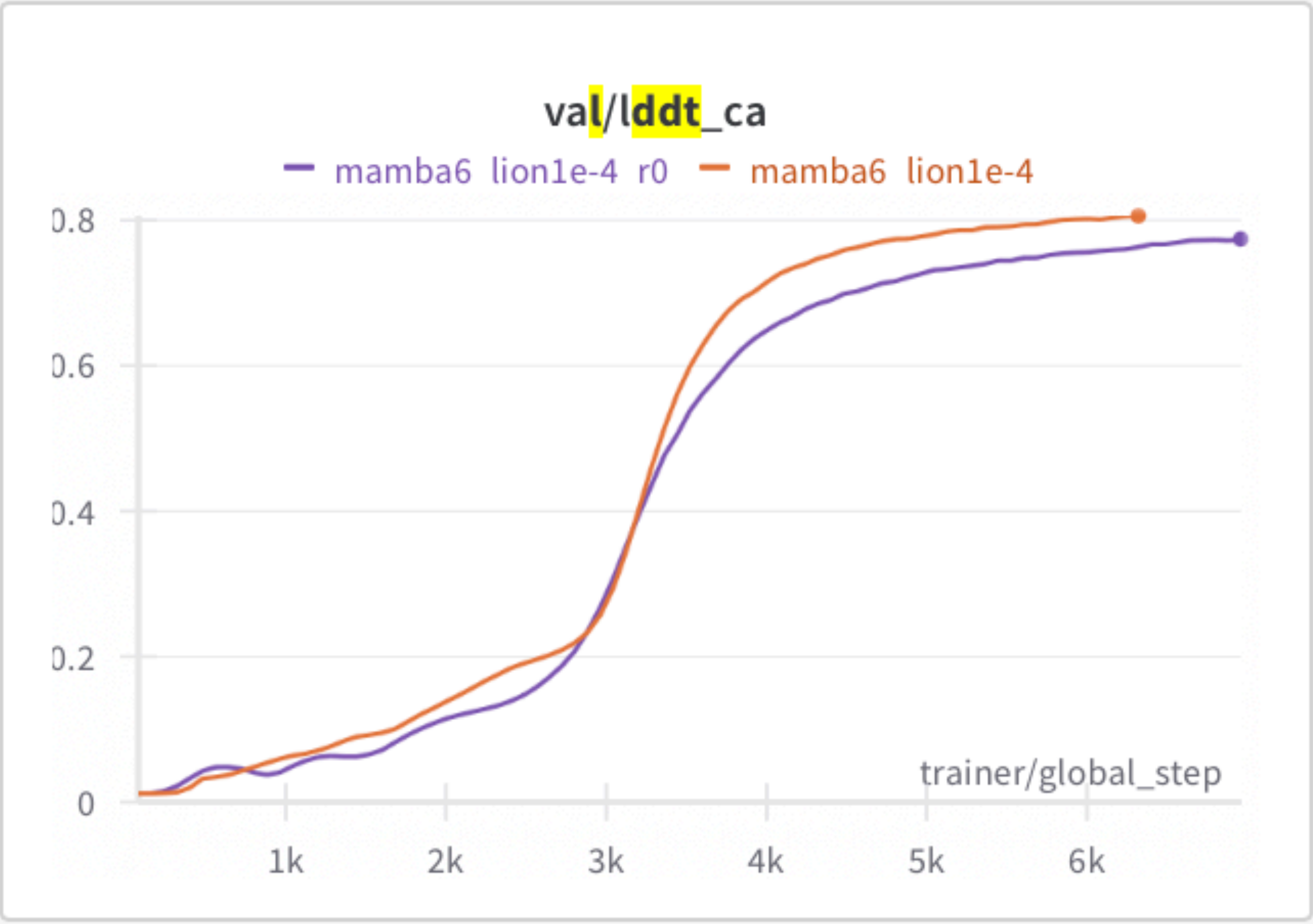
$$\sum_{i,j} \left\| T_i^{-1} \circ \vec{\mathbf{x}}_j - \tilde{T}_i^{-1} \circ \vec{\tilde{\mathbf{x}}}_j \right\|$$

- $T_i \in SE(3)$ for residue i
- $x_j \in \mathbb{R}(3)$ is atom j



No Recycling

- n_recycle=0: 40hr
    - 25% faster
- n_recycle=3: 52hr

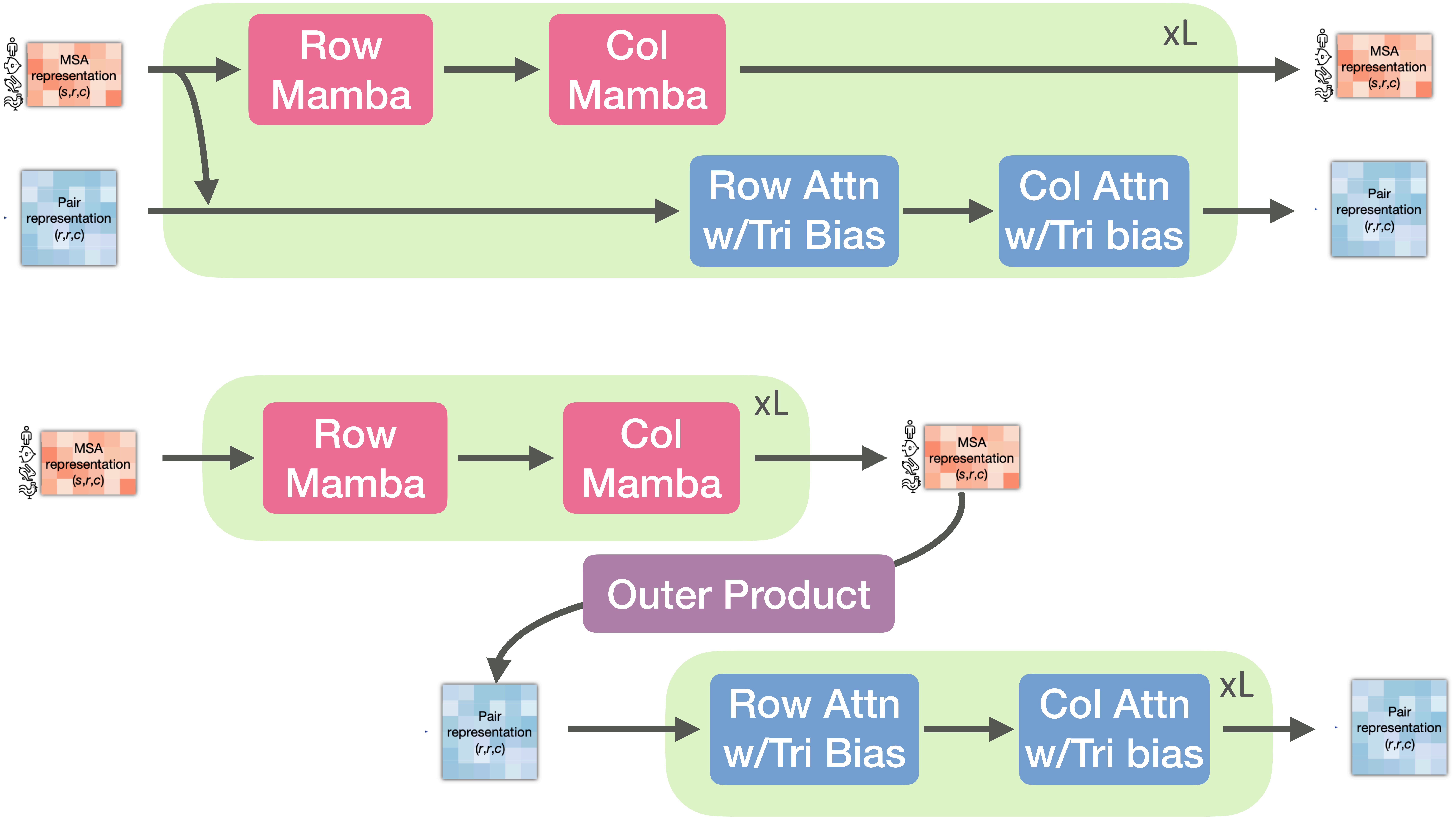


Re-Implementation

- From: 7x3 A100 40hr
- To: 2x8 A40 30hr

MambaFold

4.11.24


MSA representation (s,r,c)
Pair representation (r,r,c)
Row Mamba
Col Mamba
xL
Row Attn w/Tri Bias
Col Attn w/Tri bias
MSA representation (s,r,c)
Pair representation (r,r,c)
MSA representation (s,r,c)
Row Mamba
Col Mamba
xL
MSA representation (s,r,c)
Outer Product
Pair representation (r,r,c)
Row Attn w/Tri Bias
Col Attn w/Tri bias
xL
Pair representation (r,r,c)

MambaFold

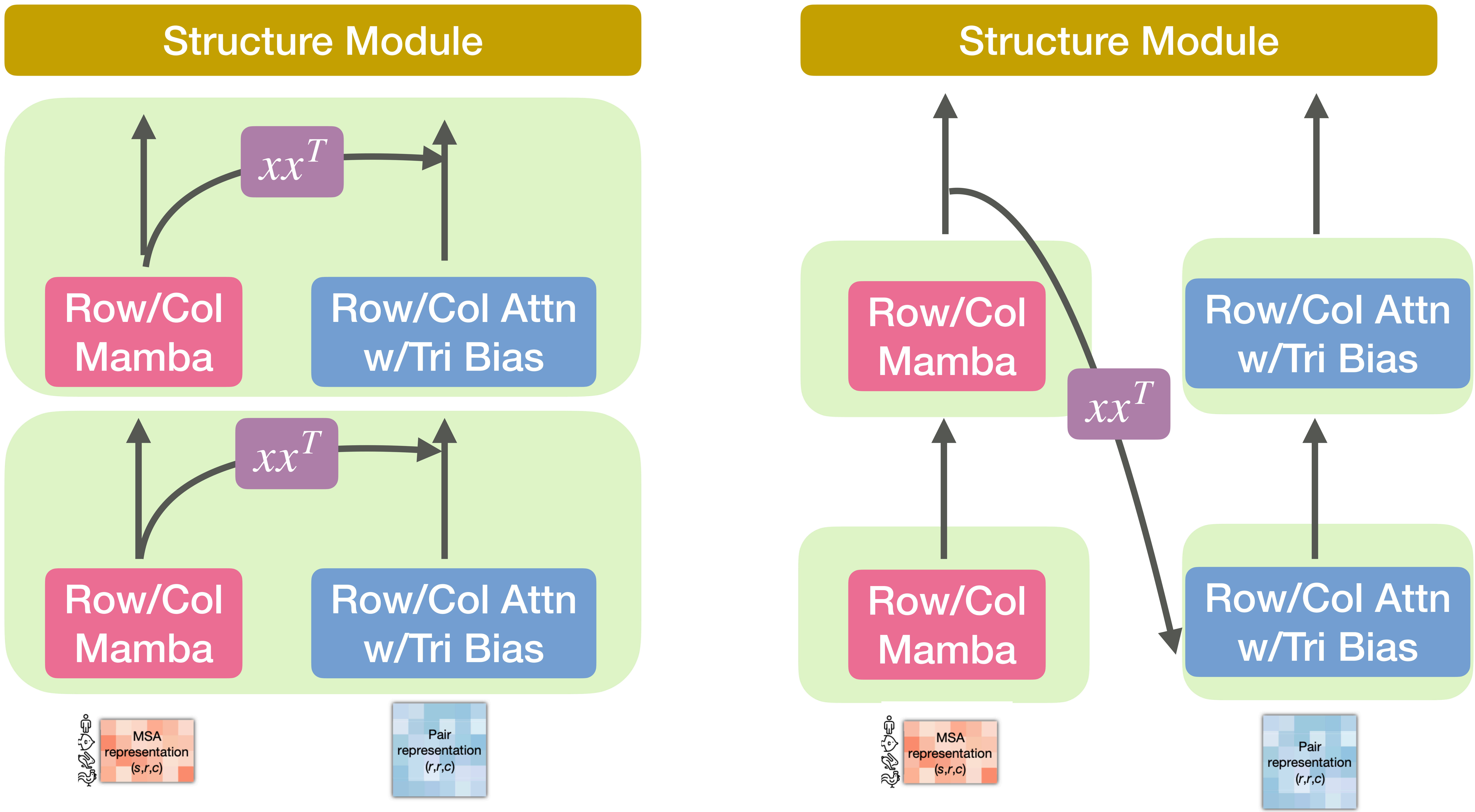
4.19.24

Multimer trains too slow

- Speedup x3 faster
  - Recycles = 0
  - No extraMSA or template stack
  - Init from monomer model
  - Architecture (splitformer, mamba)

SplitFormer

Layers	Runtime	LDDT-CA	GDT-TS
Joint MSA+Pair	48hr	0.77	0.59
MSA then Triangle	42hr	0.78	0.60


Structure Module
xx^T
Row/Col Mamba
Row/Col Attn w/Tri Bias
xx^T
Row/Col Mamba
Row/Col Attn w/Tri Bias
MSA representation (s,r,c)
Pair representation (r,r,c)
Structure Module
Row/Col Mamba
xx^T
Row/Col Attn w/Tri Bias
Row/Col Mamba
Row/Col Attn w/Tri Bias
MSA representation (s,r,c)
Pair representation (r,r,c)

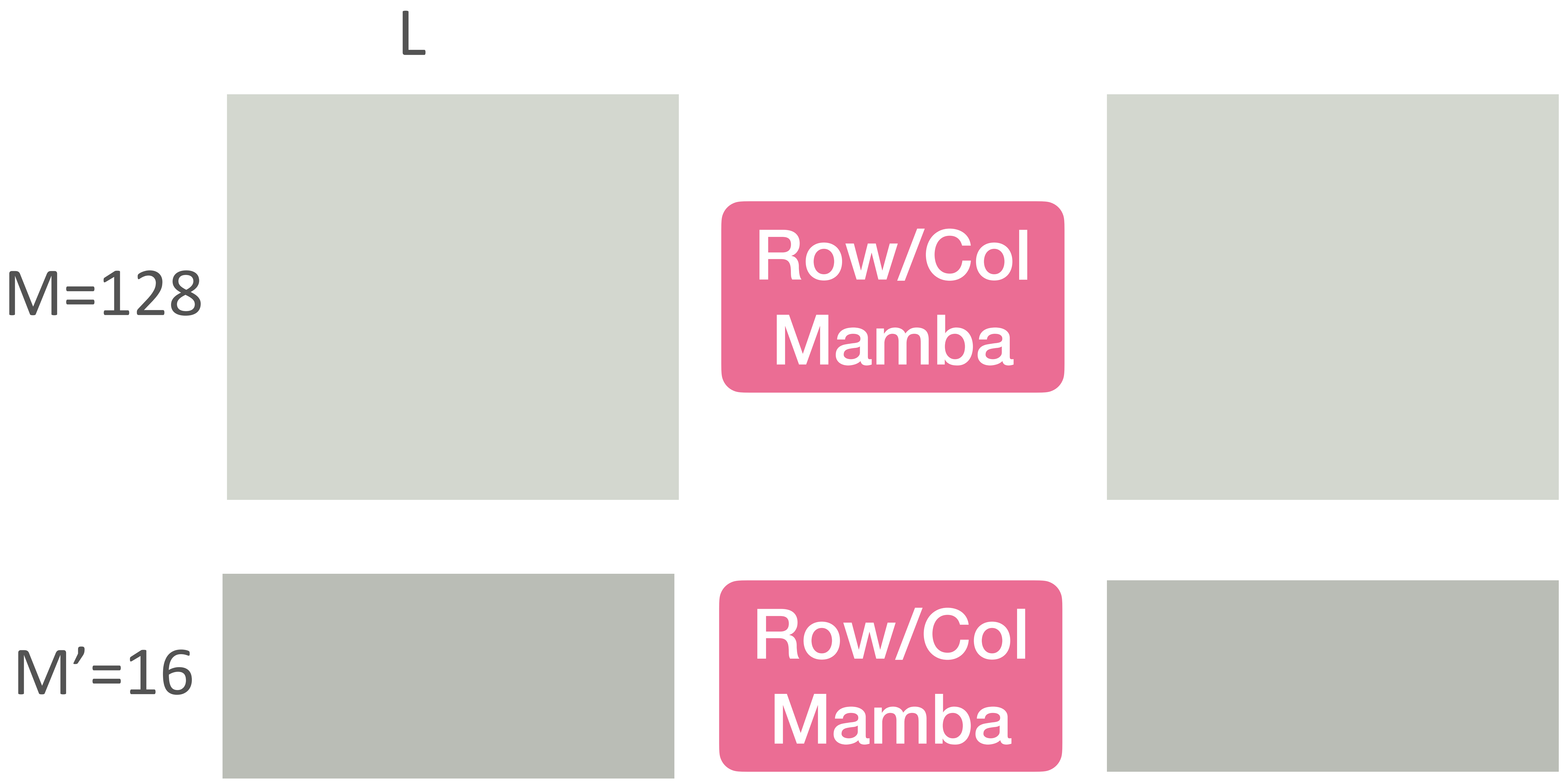
MambaFold

4.25.24

Multimer

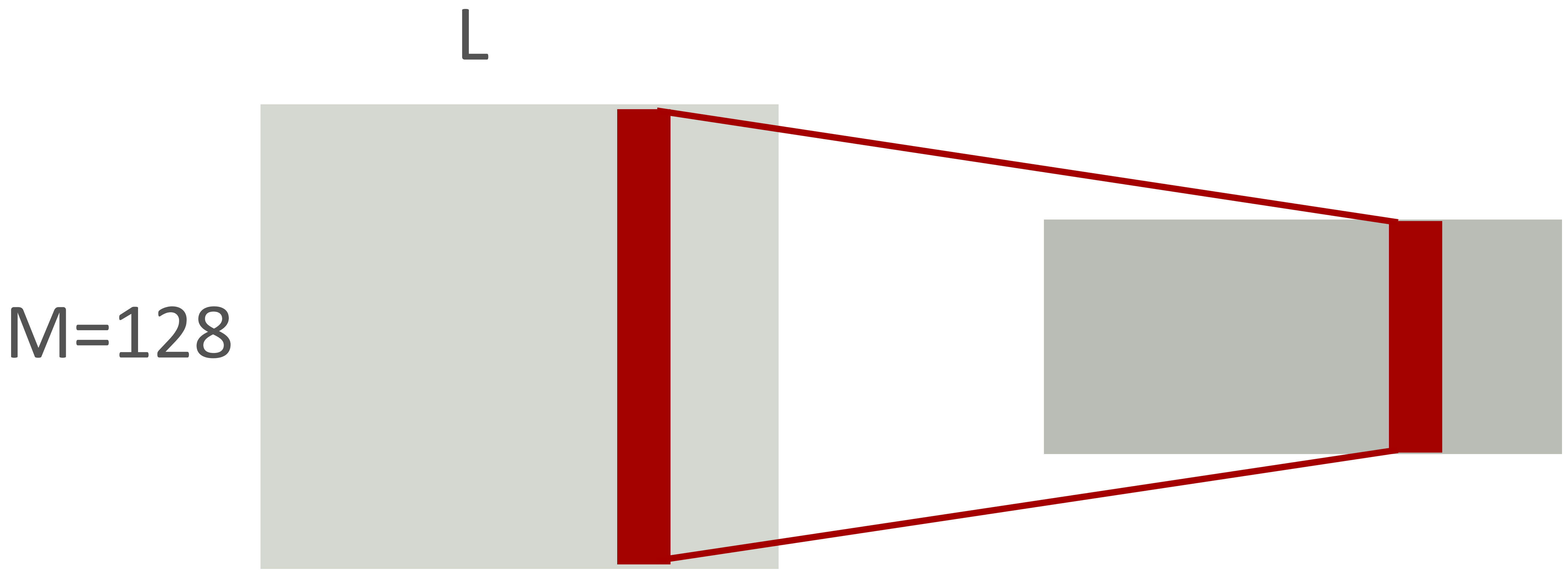
	iterations	LDDT-CA	GDT-TS	GDT-HA	TM	GDT-TS >1000 AA	GDT-HA >1000 AA
Ours	4000	0.72	0.38	0.23	0.60	0.24	0.14
Ours	6750	0.75	0.42	0.27	0.63	0.28	0.19
Ours	7500 >6750: crop768	0.76	0.45	0.29	0.65	0.29	0.19
Ours	9000	0.76	0.45	0.29	0.67	0.31	0.21
AlphaFold Multimer v2.3	80,000	0.87	0.62	0.46	0.78		

Latent MSAs



- Implicitly pairs MSAs
- Runs 25% faster
- Leverage ExtraMSAs

Latent MSAs



MambaFold

5.3.24

Long Schedule - Monomer

	Training Iterations	lDDT-Cα	GDT-TS
Ours	9,000	0.79	0.62
Ours	18,000	0.85	0.69
AlphaFold	90,000	0.89	0.75

- Model plateaued - increase MSA depth, seqlen

Table 5 | Training protocol for CASP14 models. The models in **bold** (i.e. **1.1.1** – **1.2.3**) were used in the assessment. We report the number of training samples and the training time (in days and hours) until the best validation score. Three dots ($\cdots$) indicate the same value as in the former column.

	initial training	first fine-tuning		second fine-tuning				
Model	1	1.1	1.2	**1.1.1**	**1.1.2**	**1.2.1**	**1.2.2**	**1.2.3**
Parameters initialized from	Random	Model 1	$\cdots$	Model 1.1	$\cdots$	Model 1.2	$\cdots$	$\cdots$
Number of templates N_{templ}	4	4	0	4	$\cdots$	0	$\cdots$	$\cdots$
Sequence crop size N_{res}	256	$\cdots$	$\cdots$	384	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Number of sequences N_{seq}	128	512	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Number of extra sequences $N_{\mathrm{extra\_seq}}$	1024	$\cdots$	$\cdots$	5120	1024	5120	$\cdots$	1024
Initial learning rate	10^{-3}	$5 \cdot 10^{-4}$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Learning rate linear warm-up samples	128000	0	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Structural violation loss weight	0.0	1.0	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
"Experimentally resolved" loss weight	0.0	0.01	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Training samples ($\cdot 10^6$)	9.2	1.1	1.7	0.3	0.6	1.4	1.1	2.4
Training time	6d 6h	1d 10h	2d 3h	20h	1d 13h	4d 1h	3d	5d 12h

Short Schedule - Monomer

- MSA cache
  - Data bug: 4TB -> 400GB (int64)
  - Presample sequences: 400GB -> 40GB
- Cif cache
  - Filter chains/bad: 225GB -> 34GB
- Iteration time (8 A40)
  - Smaller model/inputs
  - 5 days -> 1 day

	lDDT-Cα	GDT-TS	lDDT-Cα (Train)
Short	0.60	0.37	**0.86**
Long	0.79	0.62	0.80

why train metric is higher?
train metrics are on short sequences