Question 2

(a) Briefly describe a computational biology topic of your choice, its brief history and provide a motivation for choosing it. (1/2 page) (5 marks)

(b) State some computational goal(s) or problem(s) most relevant to the topic. Aim to describe the computational goal(s) or formulate the problem(s) in a way understandable for computer scientists. (1/2 page) (10 marks)

(c) Describe how molecular biology would benefit from achieving the computational goal(s) or solving the stated problem(s). (1/2 page) (5 marks)

(d) Characterise some existing solutions (to date) to this problem available in the literature and clearly cite references.

(i) Describe the underlying ideas or algorithms associated with these solutions by creating your own schemes, pseudocodes, equations, illustrations of model architectures (as applicable) to demonstrate your understanding of the technical background related to these solutions. (2 pages) (20 marks)

(ii) Quantify and compare advantages and limitations of existing solutions using figures and tables (as applicable). If available, you may use data (e.g. on quantifying the performance of a given method) in the literature but you have to create your own informative (and carefully labeled) figures and tables. (1 page) (10 marks)

(e) Provide critical assessment of all solutions you consider and discuss their potential (future) impact. (1 page) (10 marks)

(f) Based on your understanding of the state-of-art in the field, propose two specific suggestions for future development to complement, replace or improve on existing solutions. Provide clear arguments to support your suggestions and describe what expected advantages and limitations your proposed developments would bring when compared to existing methods available in the literature. (1 page) (10 marks)

LAST PAGE

Structure prediction of protein-nucleic acid complexes via deep learning

Weiqi Feng

(a) Topic

Our topic is to predict the structure of protein-nucleic acid complexes based on their sequences, through the means of deep learning.

This problem originates from the "protein folding problem" in Biology literature, which has been an important research topic for more than 50 years.[1] Recently, with the development in deep learning, new computational approaches, such as AlphaFold[1] and RoseTTAFold[3], have been proposed to quickly predict the 3D structure of a protein using only its amino acid sequence. Achieving great success, researchers soon moved on to other biomolecules like nucleic acid, and more recently, their complexes. Given separate structures, some people tried to build up complexes using docking calculations, but AlphaFold3[2] and RoseTTAFoldNA[5] demonstrate a more general and promising methodology.

Biomolecules are essential to life, and understanding their structure can facilitate a mechanistic comprehension of their function.[1] As a result, I chose this topic because I believe this is the crucial step to truly understand and thus make use of these fundamental molecules, which can benefit many fields such as biology and medicine, and ultimately, human kind. Moreover, as a computer science student, I am intrigued by how these networks incorporate iconic deep learning models, such as the ideas of transformer and RNN recycling, and combine them with knowledge in biology, chemistry and physics.

(b) Problem definition

The problem of choice is quite straightforward: given the sequence of a certain protein-nucleic acid complex, we want to use a network to predict its 3D structure quickly and accurately. In terms of computer scientists, the input is the sequence and chemical structures of a complex, such as ligands and covalent bonds. From this, we extract token-level and pair-level multi-dimensional features, pass them through a trained network to construct 3D structure from its output, and maybe calculate the confidence estimate.

(c) Benefit

Biological complexes are essential to life, and understanding their 3D structures is the key to understanding cellular functions and the rational design of therapeutics.[3] Accurate computational approaches can largely accelerate the speed in determining their structure compared to experimental methods, and thus enable large-scale structural bioinformatics,[1] which greatly benefits molecular biology and relevant fields such as medicine and chemistry.

(d) Solutions

This essay focuses on two mainstream approaches, the AlphaFold family by DeepMind and the RoseTTAFold family.

(i) Algorithms

Rough architectures and pseudo codes of AlphaFold3[2] and RoseTTAFold All Atom[6] are drawn on the following two pages. Black denotes model and code, blue denotes important illustrations, and purple denotes major modifications from their former models: AF2[1] and RF2[4]. Due to restriction on paragraph, I can not demonstrate all architecture details and pseudo codes, but here are some highlights worth noting.

① AlphaFold3[1,2]

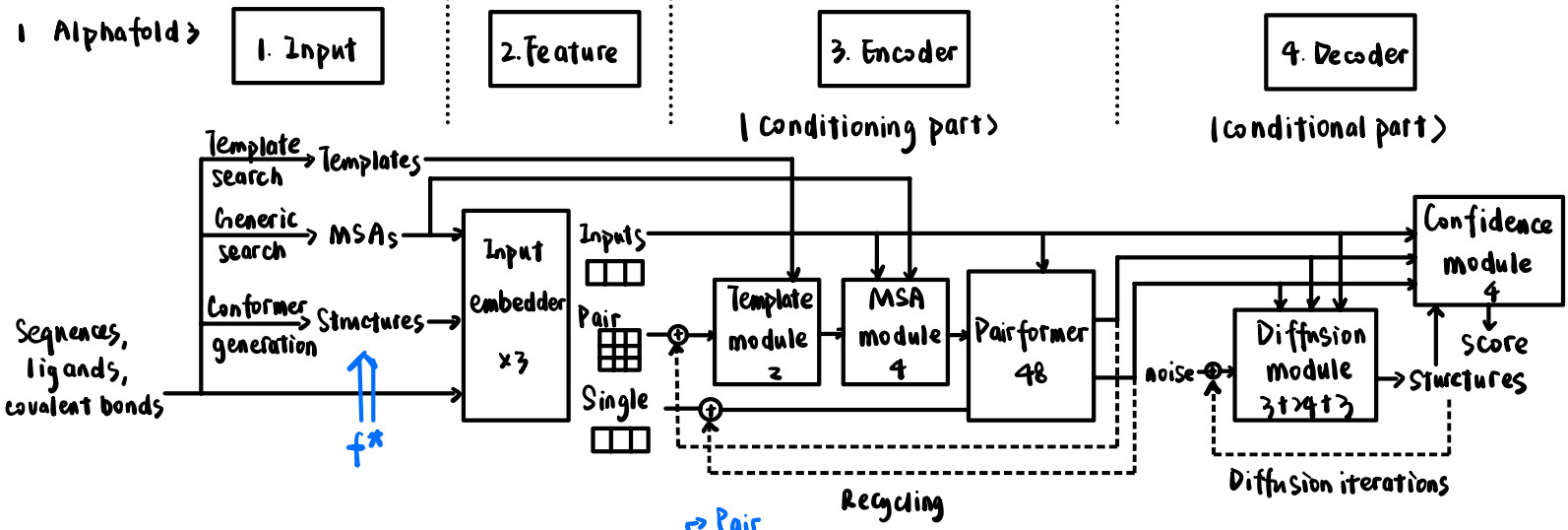
The key improvements of AF3 are that it is adapted to more general chemical structures and has higher learning efficiency. This is achieved through following modifications. Firstly, in the encoder, AF3 uses Pairformer instead of Evoformer, and isolates MSA module to reduce MSA-relevant dependency and calculations, more specifically, it replaces row-wise self-attention with pair-weighted average and single attention. Triangle update and self-attention remain the same. Secondly, in the decoder, it uses Diffusion Module on atoms and tokens instead of Structure Module, and ignores invariant point constraint in AF2.

② RoseTTAFoldNA[3,4,5,6]

Because RF2[4], RFNA[5] and RFAA[6] have very similar architecture and pipeline, this essay refers to all three papers. RFNA is the first network to predict the structures of protein – nucleic acid complexes without homology, a problem unsolved by AF and RF.[5] Its key features are that it uses a 3D track architecture of iterative updates while sharing information among them, and uses SE(3)-Transformer with equivariance in rotation and translation.

Question 2 (d)(i)

1 Alphafold3



def MainInferenceLoop({f*}, Ncycle=4, C_S=384, C_Z=128): ↳ Pair

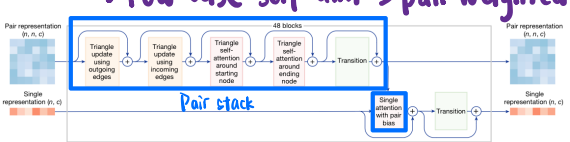
↳ Token : Inputs & Pair

2. Feature

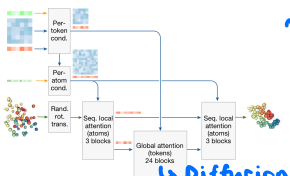
- $\{s_i^{inputs}\}$ = InputFeatureEmbedder({f*}) — $s_i^{inputs} \in \mathbb{R}^{C_S}$ Inputs for every token
- s_i^{init} = LinearNoBias(s_i^{inputs}) — $s_i^{init} \in \mathbb{R}^{C_S}$ Single for every token
- z_{ij}^{init} = LinearNoBias(s_i^{inputs}) + LinearNoBias(s_j^{inputs})
- z_{ij}^{init} += RelativePositionEncoding({f*}) position and order information
- z_{ij}^{init} += LinearNoBias($f_{ij}^{token\text{-}bonds}$) bond info — $z_{ij}^{init} \in \mathbb{R}^{C_Z}$ Pair for every pair of tokens

3. Encoder

- $\{\hat{z}_{ij}\}, \{\hat{s}_i\} = \vec{0}, \vec{0}$ (↳ Pair, ↳ Single)
- for all c ∈ {1, ..., Ncycle} do — Recycling 4 times
  - $z_{ij} = z_{ij}^{init}$ + LinearNoBias(LayerNorm($\hat{z}_{ij}$))
  - $\{z_{ij}\}$ += TemplateEmbedder({f*}, $\{z_{ij}\}$) — only 4 MSA blocks in recycle:
  - $\{z_{ij}\}$ += MsaModule($\{f_{s_i}^{msa}\}$, $\{z_{ij}\}$, $\{s_i^{inputs}\}$) — less dependency on MSA & quicker calculation
    - ↳ MSA stack & Pair stack
    - ↳ row-wise self-attn → pair weighted averaging



~ single row-wise



atom & token attn

↳ Diffusion Transformer

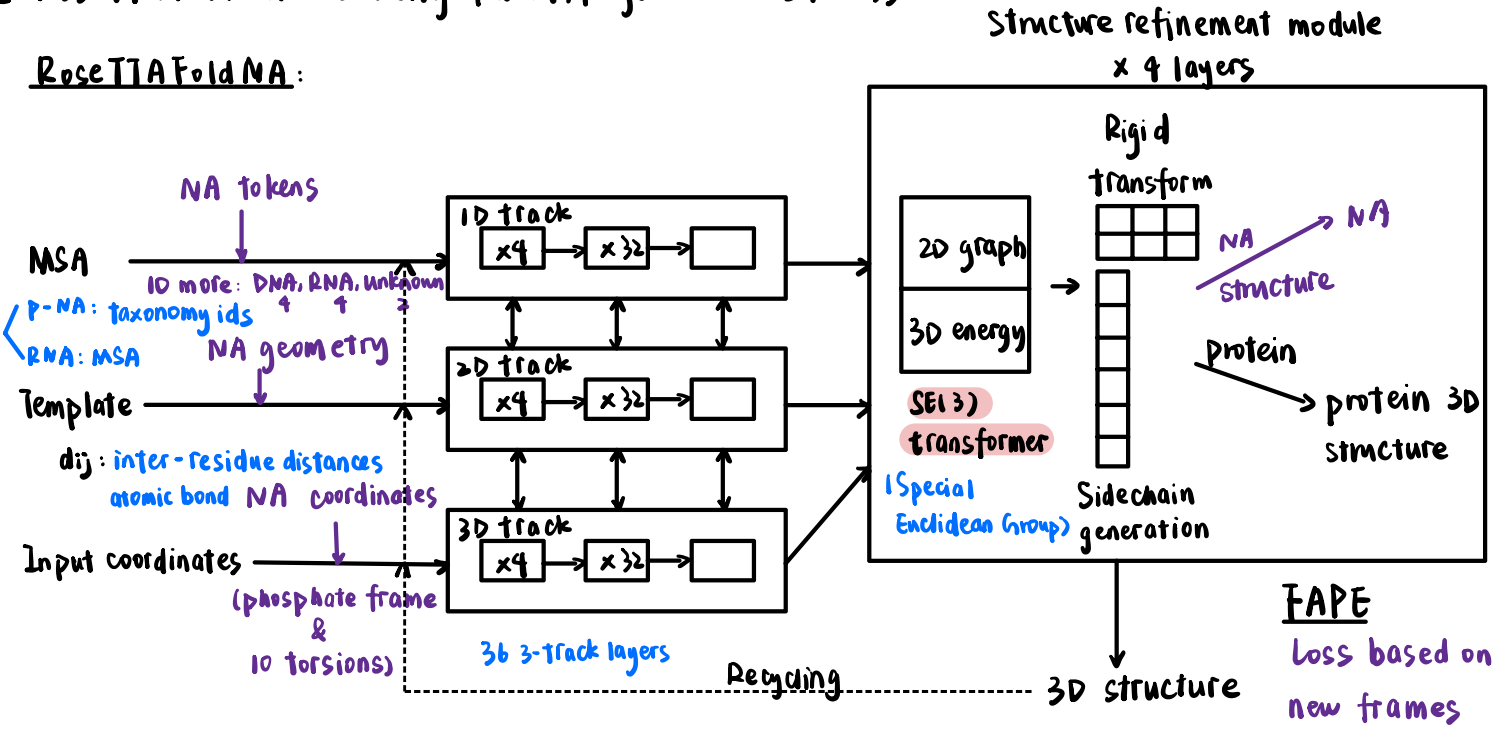
  - $\{s_i\}, \{z_{ij}\}$ += PairformerStack($\{s_i\}, \{z_{ij}\}$)
  - $\{\hat{s}_i\}, \{\hat{z}_{ij}\} \leftarrow \{s_i\}, \{z_{ij}\}$
- end for

4. Decoder

- $\{\vec{x}^{pred}\}$ = SampleDiffusion({f*}, $\{s_i^{inputs}\}$, $\{s_i\}$, $\{z_{ij}\}$)
- $\{P_i^{plddt}\}, \{P_{ij}^{pae}\}, \{P_{ij}^{pde}\}, \{P_i^{resolved}\}$ = ConfidenceHead($\{s_i^{inputs}\}$, $\{s_i\}$, $\{z_{ij}\}$, $\{\vec{x}_i^{pred}\}$)
  (↳ atom, ↳ token)
- $P_{ij}^{distogram}$ = DistogramHead(z_{ij})
- return $\{\vec{x}_i^{pred}\}$, $\{P_i^{plddt}\}$, $\{P_{ij}^{pae}\}$, $\{P_{ij}^{pde}\}$, $\{P_i^{resolved}\}$, $P_{ij}^{distogram}$

2 RoseTTAFold NA scarcity of data, physical terms in loss

RoseTTAFold NA:



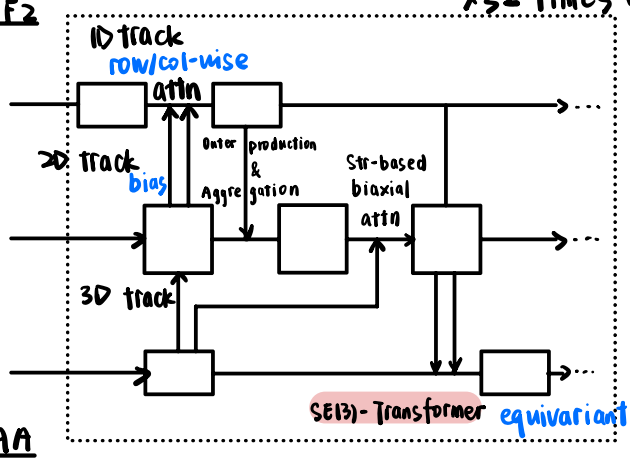
We refer to RF2 & RFAA for more details of the simulator stage and recycling.

One cycle of the simulator stage, RFAA forward pass:

```
def RFAA(msa_masked, msa_full, seq, idx, bond_feats, dist_matrix,
         templates, msa_prev, pair_prev, state_prev, xyz_prev, sc_torsions_prev,
         N_full_blocks=4, N_main_blocks=32, N_ref_blocks=4):

    msa_full = MSA_Full_Embed(msa_full)

    msa_cluster, pair, state = MSA_Cluster_Embed(msa_masked,
                                                 seq, idx, bond_feats, dist_matrix)

    pair_t = Bond_Embed(bond_feats)

    msa_cluster[0], pair, state_t = Recycle_Embed(recyclables)

    pair, state_t = Template_Embed(templates, pair, state)

    for i=1 to N_full_blocks:
        msa_full, pair, state, xyz, sc_torsions =
        Full_Block(msa_full, pair, state, xyz)

    for i=1 to N_main_blocks:
        msa_cluster, pair, state, xyz, sc_torsions =
        Main_Block(msa_cluster, pair, state, bond_feats, dist_matrix)

    for i=1 to N_ref_blocks:
        state, xyz, sc_torsions =
        Ref_Block(msa_cluster, pair, state, xyz,
                  idx, bond_feats, dist_matrix, ...)

    return msa_cluster, pair, state, xyz, sc_torsions
```

Main block architecture and pseudo code:

RF2



RFAA

```
def Main_Block(msa_cluster, pair, state, xyz, idx, bond_feats, dist_matrix):
    (iteratively update)

    str_bias = Str_Bias(xyz)

    rel_pos, bond_dist = Positional_Encoding(idx, bond_feats, dist_matrix)
                                                                  2D
    str_bias_t = Linear(rel_pos) + Linear(bond_dist)

    msa_cluster = 1D_Update(msa_cluster, pair, state, str_bias)
                  column-wise and row-wise attn
    pair = Aggregation(msa_cluster, pair)
    pair = 2D_Update(pair, state, str_bias)        use all 3 tracks

    xyz, state, sc_torsions = 3D_Update(msa_cluster, pair,
                                        state, xyz, idx, bond_feats, dist_matrix)
                                                        use all 3 tracks
    return msa_cluster, pair, state, xyz, sc_torsions
```

(ii) Strengths and Limitations

Since quantifiable and comparable data from these papers mainly focuses on model results, we will analyse prediction and confidence accuracy in this part, and leave qualitative assessment to question (e). We only have data for prediction accuracy, so the following part contains some figures from the papers.

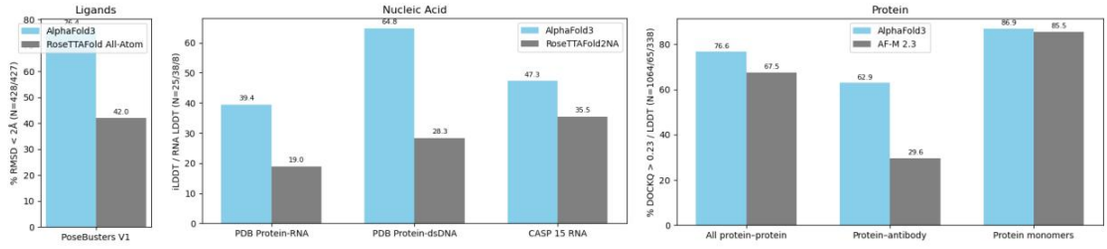


Figure 1 | Prediction accuracy Data extracted from the AF3 paper.[2]

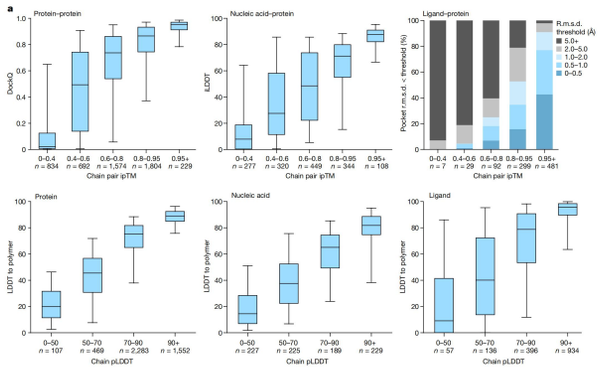


Figure 2 | AF3 confidences track accuracy[2]

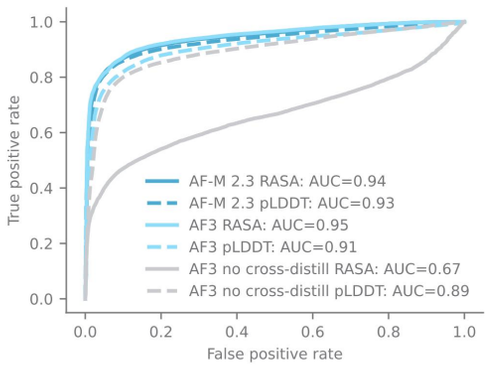


Figure 3 | Disorder prediction CAID 2[2]

As shown in Fig.1, AF3 excels in versatility and accuracy across molecular types, especially protein–ligand (76.4% with RMSD < 2Å) and protein–dsDNA (64.8% LDDT) interactions. It also has high performance on large and complex systems such as human coronavirus OC43 spike protein (4,665 residues), with LDDT=83.0.[2] Fig.2 shows the well-calibrated confidence metrics (ipTM and pLDDT) that correlate with model accuracy. Limitations include hallucinations (Fig.3) and stereochemical error (chirality violation rate of 4.4% in PoseBusters benchmark).

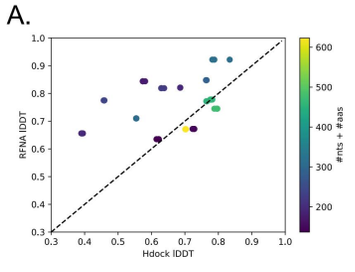


Figure 4 | Compared with docking[5]

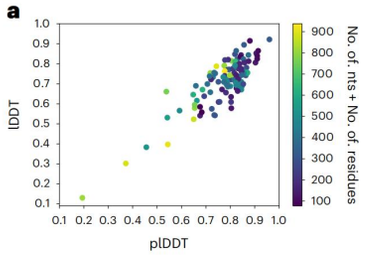


Figure 5 | lDDT vs plDDT[5]

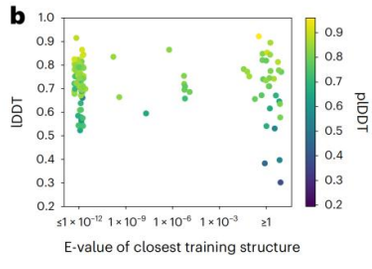


Figure 6 | lDDT of different structures[5]

In comparison, RFNA only targets at protein–nucleic acid complexes, achieves moderate accuracy for RNA predictions, but greatly lags behind AF3 overall. RFAA was not available for baselining at the time of writing AF3 paper, but the RFAA paper indicates their accuracy is below specialist predictors in almost all categories.[2]

RFNA has specialized frame-torsion modeling tailored for protein–nucleic acid interactions, outperforming traditional docking methods (Fig.4). Fig.5 shows the model

correctly identifies inaccurate predictions, with 81% of high-confidence models successfully capture protein-nucleic acid interfaces by CAPRI metrics. In addition, Fig.6 shows the model generalizes well, with no clear performance difference between structures with and without sequence homologs in the protein–NA training set. For the 33 clusters with no sequence homologs, we have average lDDT of 0.68 and 24% of models > 0.8 lDDT.

One limitation of RFNA is limited protein-NA data: there are only 1,632 RNA clusters, 1,556 protein-nucleic acid clusters compared to 26,128 all protein clusters. One other is that RFNA has poor prediction of individual subunits, typically large multidomain proteins, large RNAs (>100 nt) and small single-stranded nucleic acids.

(e) Assessments

AlphaFold3 (AF3) and RoseTTAFoldNA (RFNA) are state-of-the-art models for structural biology. While AF3 excels in versatility and accuracy, RFNA specializes in protein–nucleic acid complexes. Other solutions, such as AlphaFold-Multimer and AIchemy_RNA, also contribute significantly to specific niches within this domain. Here are some notable assessments:

Model	Targets	Performance Highlights	Strengths	Limitations
AF3	Unified model	Best in almost all categories	High versatility and accuracy, diffusion-based, robust confidence metrics	Computationally expensive, limited dynamic behavior modeling, hallucination and stereochemical error
RFNA	Protein-NA complexes	Moderate accuracy for RNA modeling	Tailored for protein–NA systems, efficient for RNA/DNA modeling	Limited training data, struggles with large RNAs and multi-domain proteins
AF-M	Protein interactions	Improved multimeric interface predictions	Accurate protein–protein interaction modeling	Less focus on nucleic acids or small molecules
Alchemy_RNA	RNA	Best performance for CASP 15 RNA dataset	Specialization in RNA modeling, superior for CASP RNA targets	Dependency on curated datasets, limited in broader applications

While each model provides a special approach to its respective problem, AF3 stands out as the SOTA model for almost all biological systems. Individually, people can dig deeper into alternatives for specialized tasks, for example fine-tune RFNA for protein–NA specific problems, use AF-M in protein engineering to aid synthetic biology, and better Alchemy_RNA for RNA-centric research.

AF3 is bound to be more significant in the history of bioinformatics. It demonstrates the possibility to accurately predict the structure of a wide range of biomolecular systems in a unified network with SOTA accuracy without homology and less dependence on MSAs. The success of AF3 shows that given the right network, we only need significantly less data and

can amplify existing data.

In the future, with the parallel developments of experimental and computational methods, more training data and improved deep learning models based on AF3 will propel us into an era of structurally informed biological understanding and therapeutic development.

(f) Suggestions

① Integrate dynamics

One limitation of protein structure prediction models is that they typically predict static structures as seen in the PDB, not the dynamical behaviour of biomolecular systems in solution,[2] where most biological activities and reactions take place. To address this problem, the development of hybrid models integrating molecular dynamics (MD) simulations or neural approximations of dynamics is a promising field. Such models would not only predict static structures but also capture conformational ensembles and dynamic states, essential for understanding biological processes.

A feasible idea is to integrate recent advancements in neural network-based simulation tools, such as Graph Neural Networks (GNNs), which can approximate dynamics. This would significantly enhance the biological realism of these models, enabling them to represent intermediate states. Such models can be applied to areas where dynamics and flexibility are paramount, such as drug discovery and RNA folding. Additionally, hybrid models could complement existing static prediction methods, refining or validating predictions with dynamic insights for greater accuracy and utility.

However, this approach is not without its challenges. Incorporating MD simulations would require substantial computational resources, particularly for large systems, posing a barrier to scalability. Furthermore, early implementations may face difficulties in balancing computational feasibility with prediction precision, potentially leading to accuracy trade-offs.

Despite these limitations, the integration of dynamic conformational states into predictive frameworks has the potential to revolutionize the field, bridging the gap between static and dynamic representations of biomolecular systems, and unlocking new possibilities for therapeutic and biotechnological applications.

② Mutual distillation

Given that structure prediction models are an application of the Transformer structure in Biology, we can draw inspiration for them from the more general models in computer science, such as Large Language Models (LLMs) and Large Multimodal Models (LMMs). In this case, we can improve model performance and reduce computational cost by bootstrapping and leveraging mutual distillation.

Unified models excel in generalization and specialized models achieve high accuracy in specific domains. Mutual distillation addresses these complementary strengths and weaknesses by iteratively refining each other's predictions through a synergistic process.

The workflow begins with the unified model generating broad-spectrum predictions, which the specialized model then refines in its domain of expertise. The improved results from the specialized model are distilled back into the unified model's training process, allowing it to learn domain-specific insights without requiring additional architecture modifications. This iterative bootstrapping ensures that both models benefit from each other's strengths.

This approach has several expected advantages. By combining the two models, mutual distillation enhances the overall accuracy of predictions without sacrificing versatility. It also improves the efficiency of training by reusing predictions as additional training data. Furthermore, this method is modular, enabling the flexibility to add new specialized models into the framework without retraining the entire unified model.

However, there are challenges to consider. Careful model design is essential to ensure convergence, avoiding issues like overfitting in specialized models or over-generalization in the unified model. Additionally, integrating predictions from models trained on different datasets and architectures may require sophisticated data harmonization techniques.

Despite these limitations, mutual distillation offers a powerful mechanism to improve accuracy, scalability, and utility across biomolecular modeling tasks, paving the way for more advanced predictive tools in structural biology and therapeutic design.

1. Jumper, J., Evans, R., Pritzel, A. et al. Highly accurate protein structure prediction with AlphaFold. Nature 596, 583–589 (2021). https://doi.org/10.1038/s41586-021-03819-2

2. Abramson, J., Adler, J., Dunger, J. et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. Nature 630, 493–500 (2024). https://doi.org/10.1038/s41586-024-07487-w

3. Minkyung Baek et al. Accurate prediction of protein structures and interactions using a three-track neural network. Science373,871-876(2021).DOI:10.1126/science.abj8754

4. Minkyung Baek et al. Efficient and accurate prediction of protein structure using RoseTTAFold2. bioRxiv 2023.05.24.542179; doi: https://doi.org/10.1101/2023.05.24.542179

5. Baek, M., McHugh, R., Anishchenko, I. et al. Accurate prediction of protein–nucleic acid complexes using RoseTTAFoldNA. Nat Methods 21, 117–121 (2024). https://doi.org/10.1038/s41592-023-02086-5

6. Rohith Krishna et al. Generalized biomolecular modeling and design with RoseTTAFold All-Atom. Science384,eadl2528(2024).DOI:10.1126/science.adl2528