

OmegaFold and Protein Structure Prediction

CCATS Group



OmegaFold

bioRxiv preprint doi: <https://doi.org/10.1101/2022.07.21.500999>; this version posted July 22, 2022. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

July 20th, 2022

Title: High-resolution *de novo* structure prediction from primary sequence

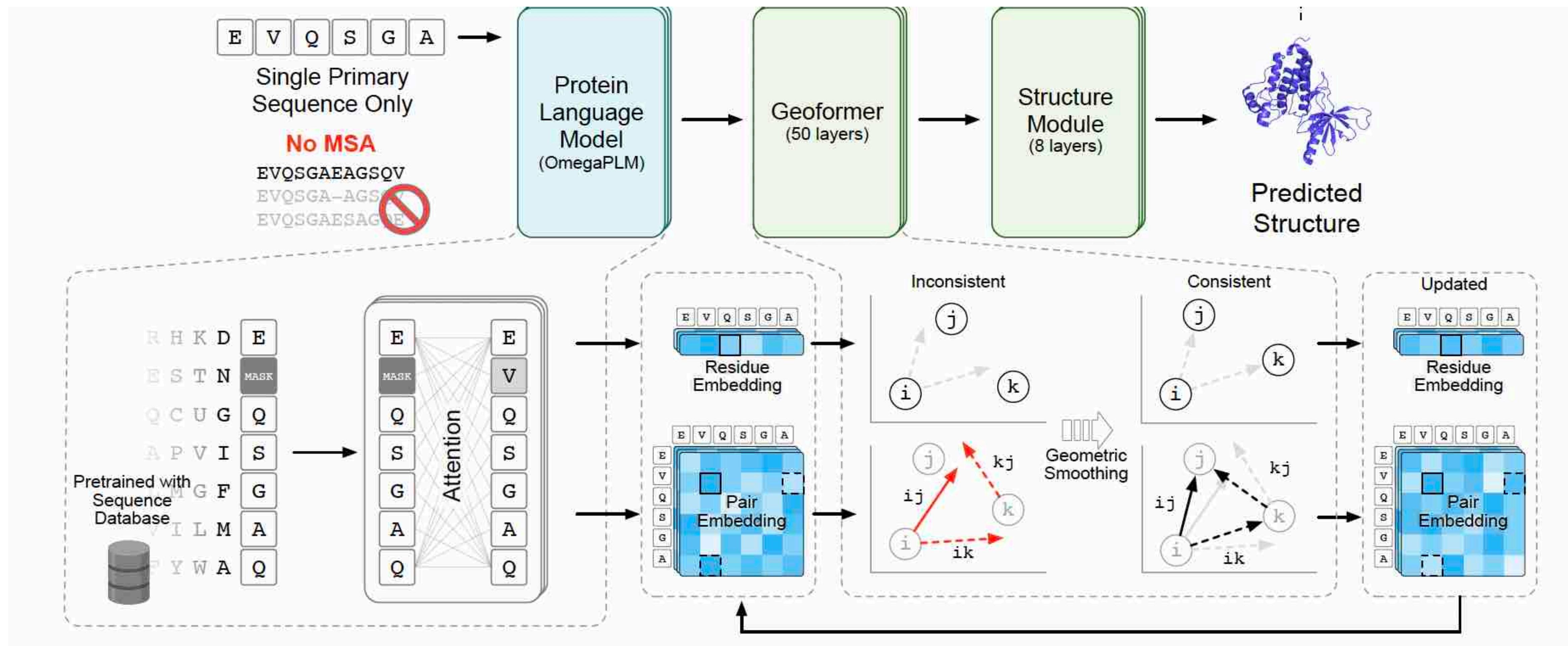
Authors: Ruidong Wu^{a,1}, Fan Ding^{a,1}, Rui Wang^{a,1}, Rui Shen^{a,1}, Xiwen Zhang^a, Shitong Luo^a, Chenpeng Su^a, Zuofan Wu^a, Qi Xie^b, Bonnie Berger^{c,2}, Jianzhu Ma^{a,2}, Jian Peng^{a,2}

Affiliations: ^aHelixon US Inc, USA; ^bWestlake Laboratory of Life Sciences and Biomedicine, Hangzhou, Zhejiang, China; ^cComputer Science & Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139

How is OmegaFold Different?

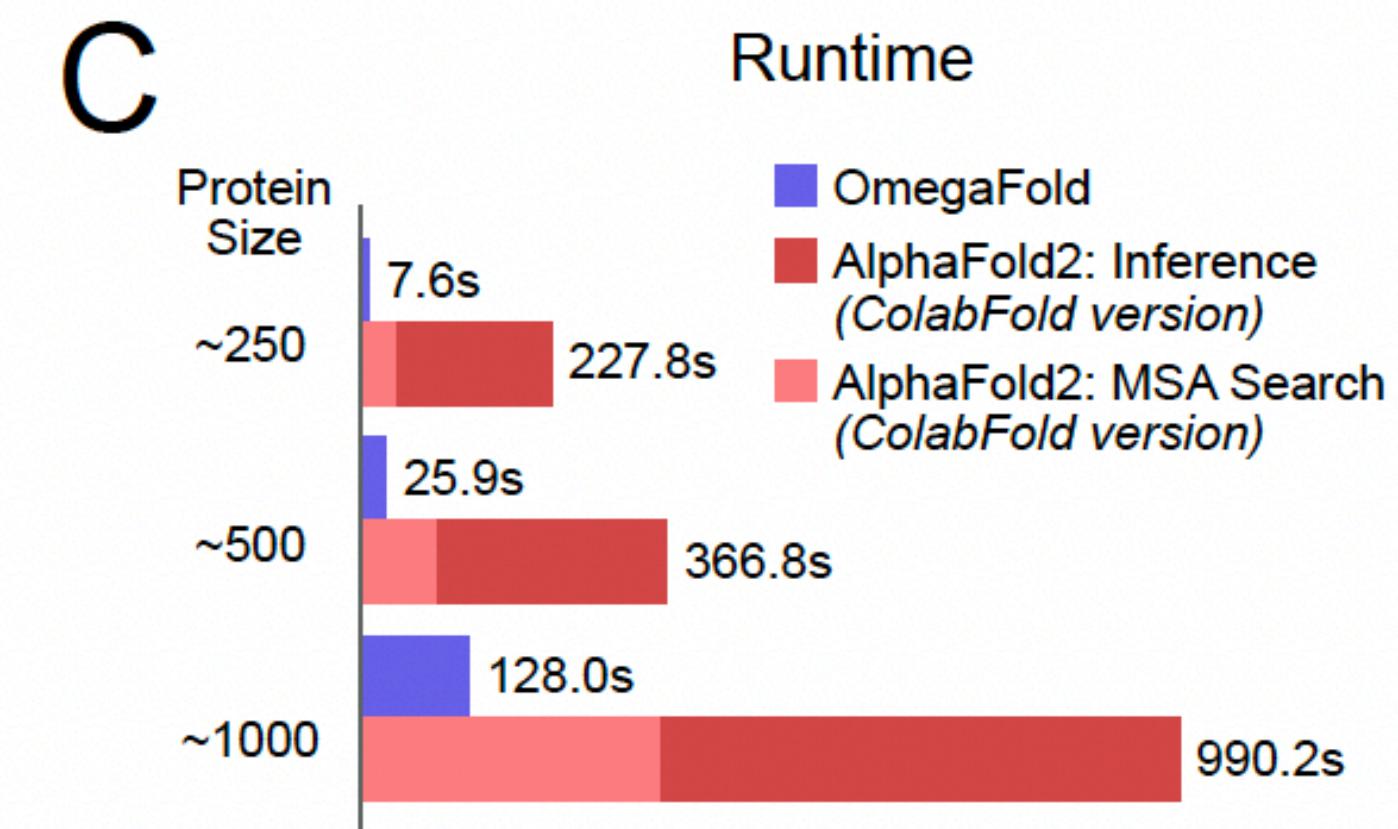
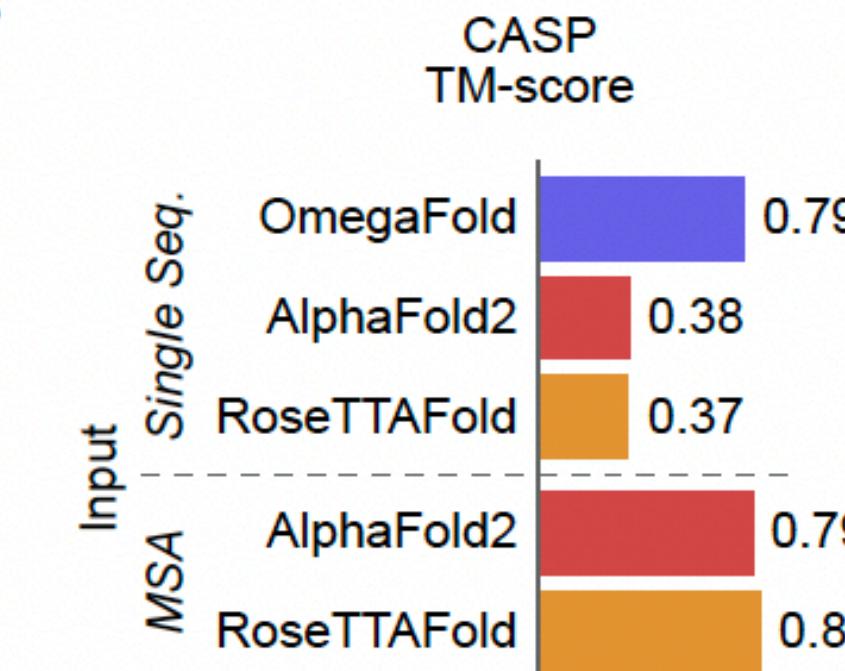
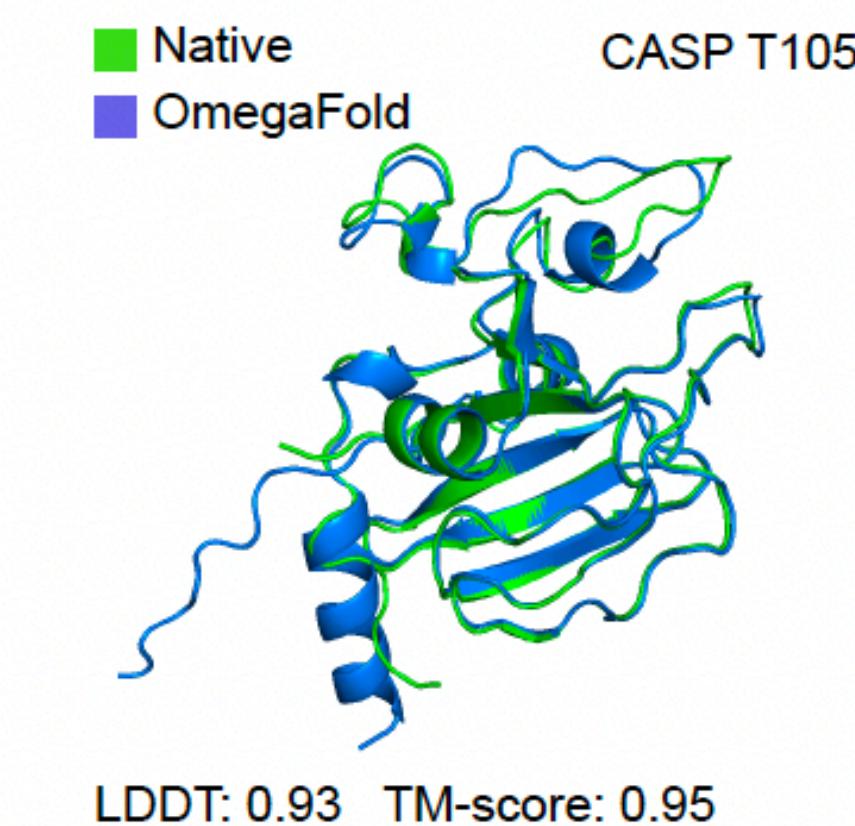
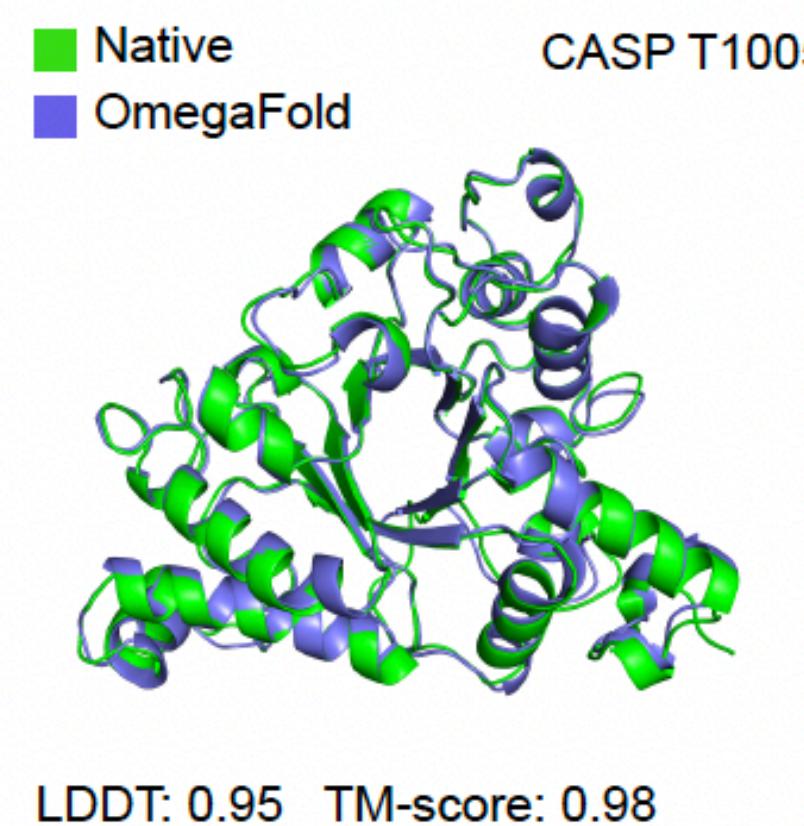
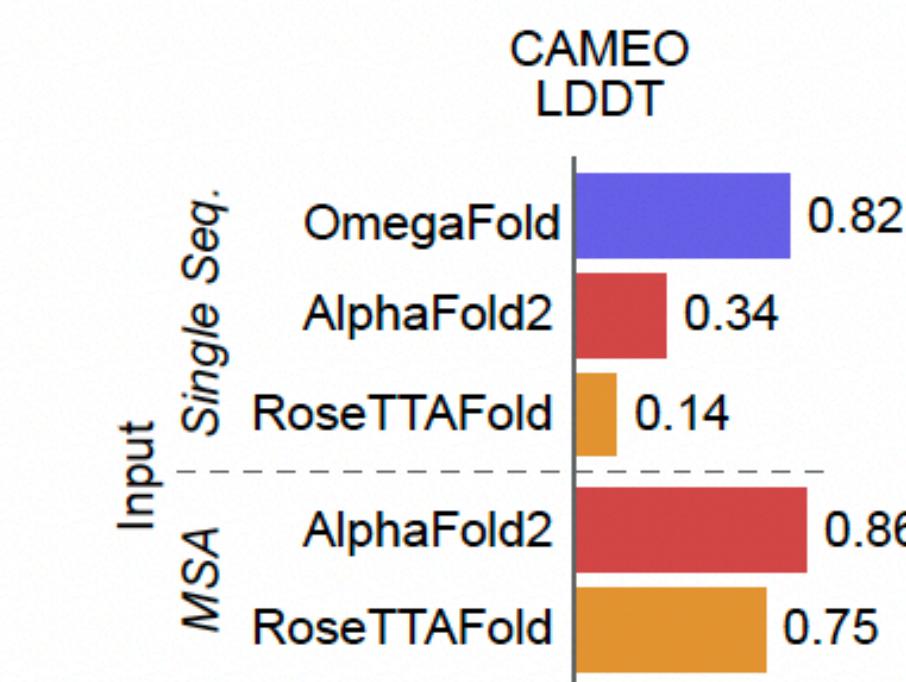
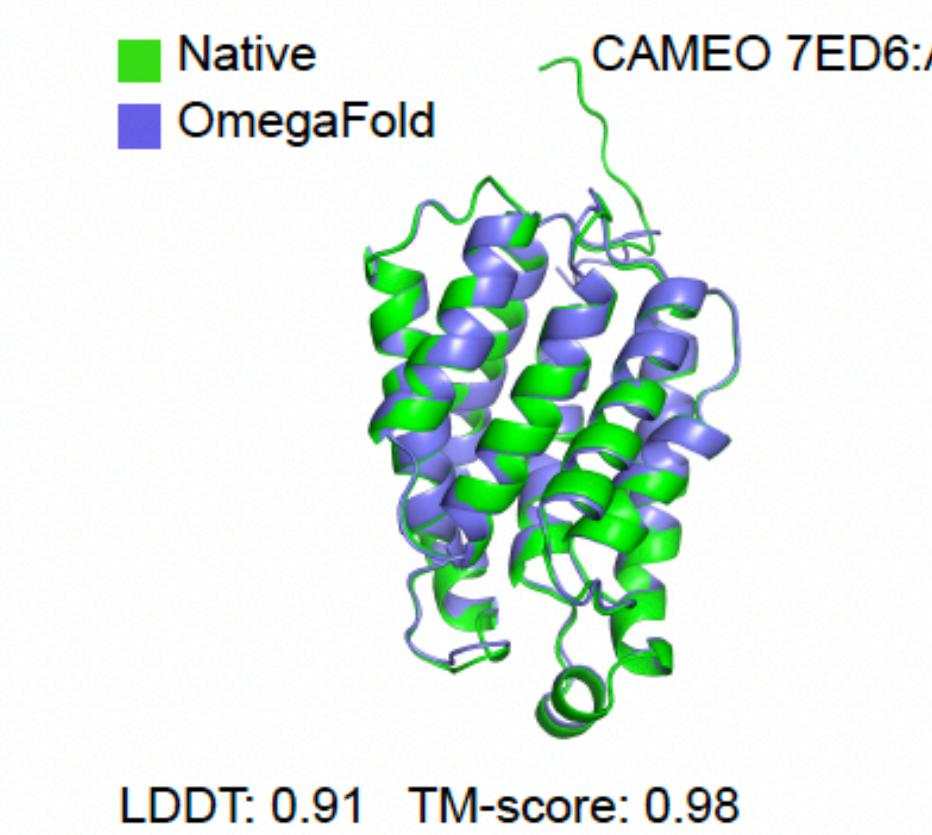
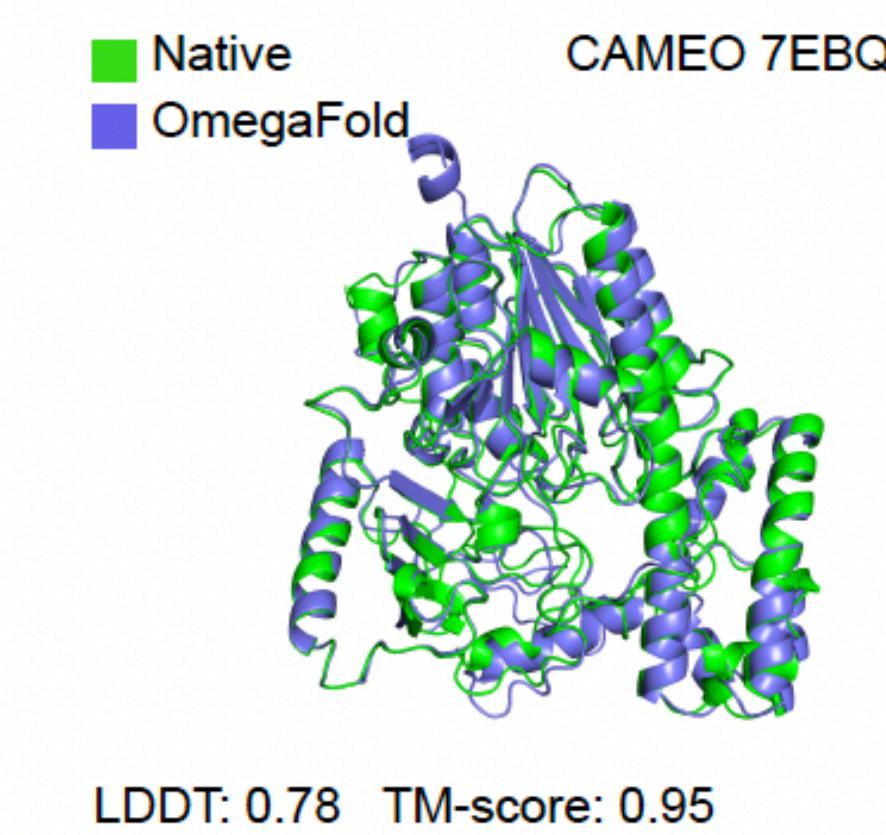
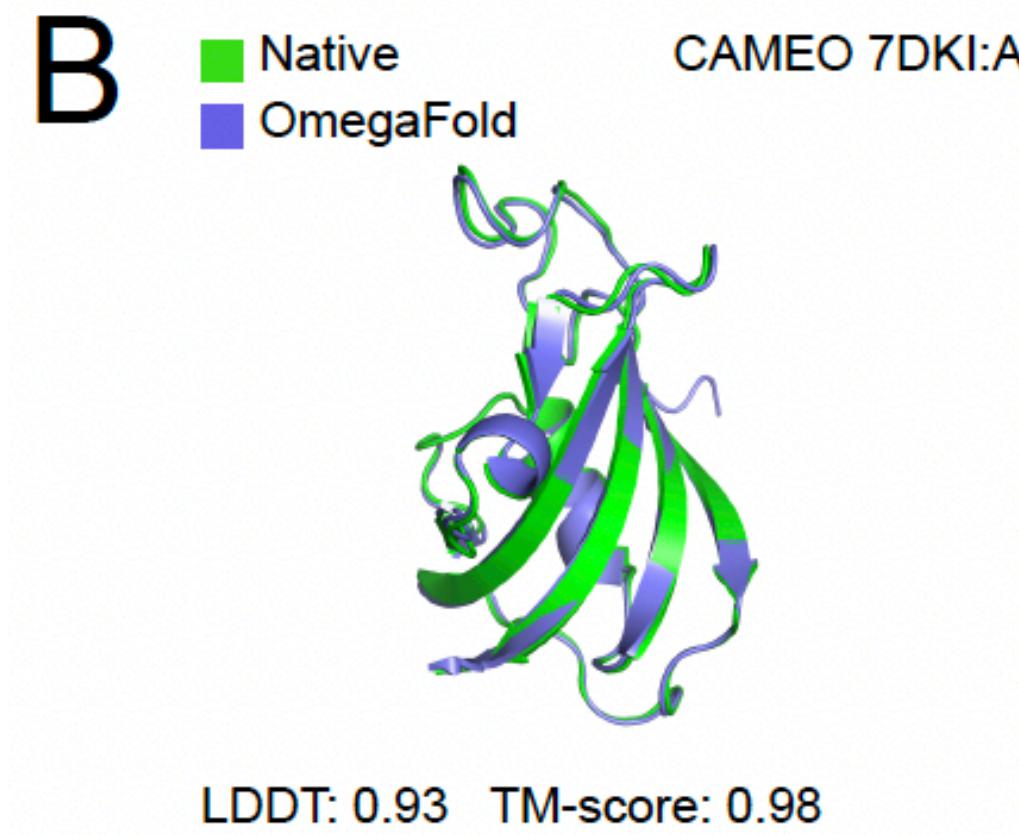
- Based on the understanding that
 1. MSA does not always work, especially for fast-evolving antibodies, and orphan proteins.
 2. Protein folds in a natural setting without exploiting evolutionary information.
- **OmegaFold** predicts protein structure from a single primary sequence alone, i.e. *alignment-free*
- It uses a **pre-trained** protein language model (PLM) to generate single- and pair-wise embeddings (i.e. representations)
 - training on a large collection of unaligned and unlabelled protein sequences
 - fed into Geoformer, which will further distill structural/physical pairwise relationships

Workflow

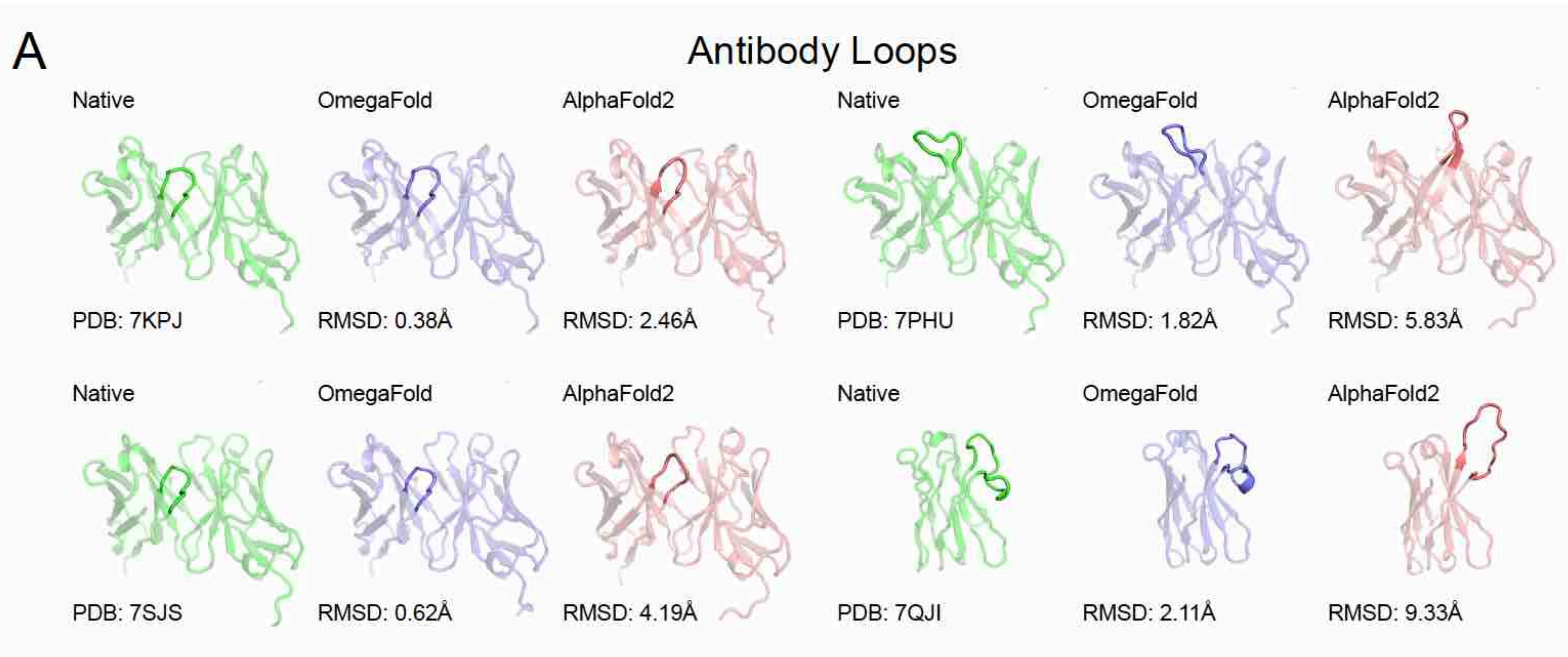


Prediction Results

- For CASP and CAMEO proteins, **OmegaFold** is as accurate as **AlphaFold2** and **RoseTTAFold**.
- **OmegaFold** is much faster.



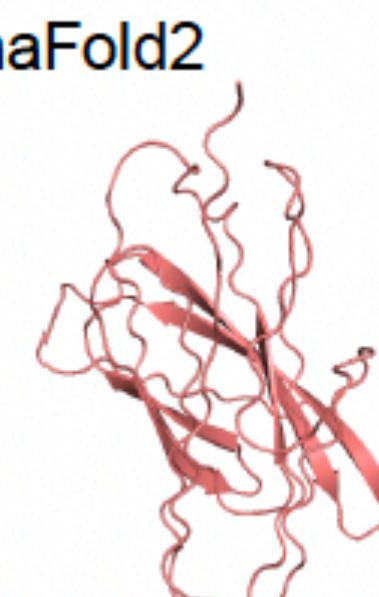
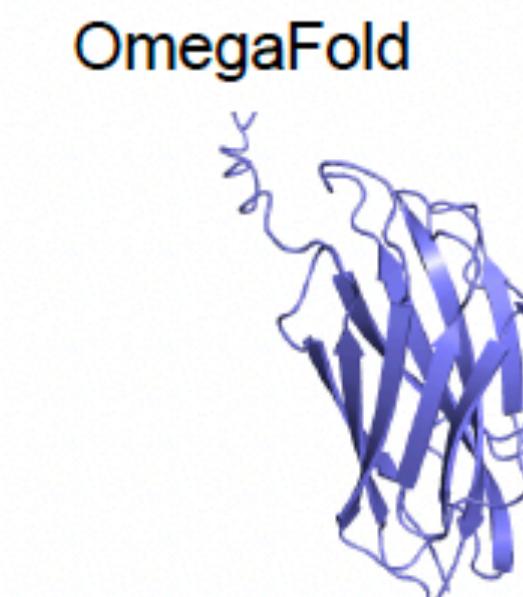
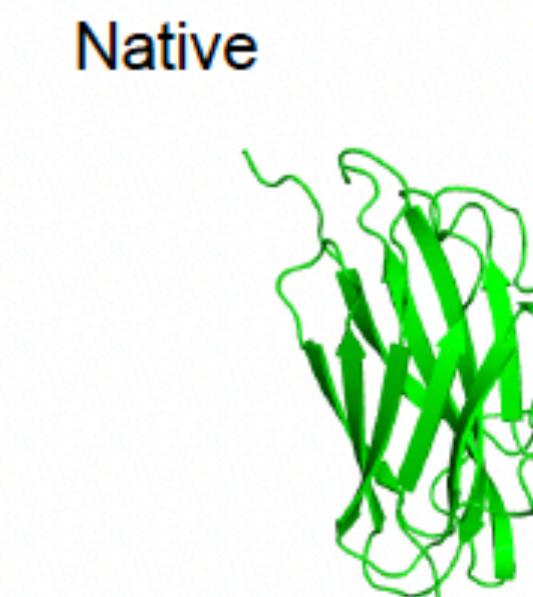
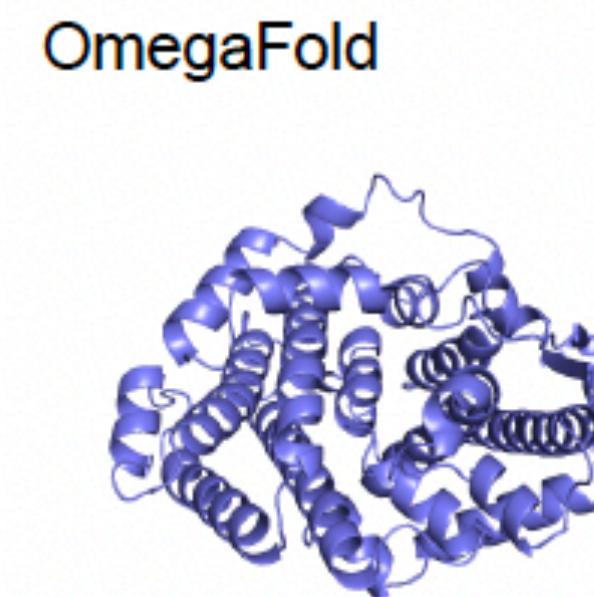
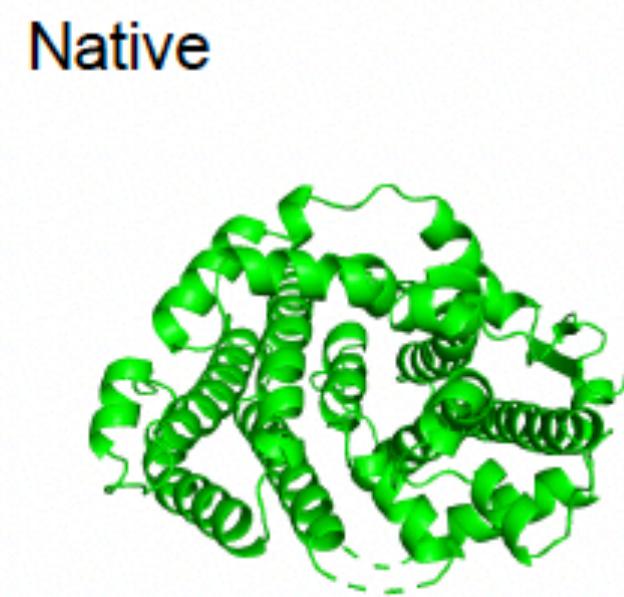
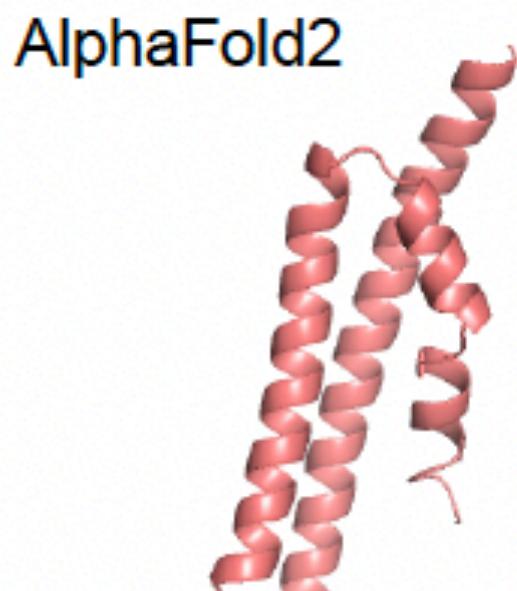
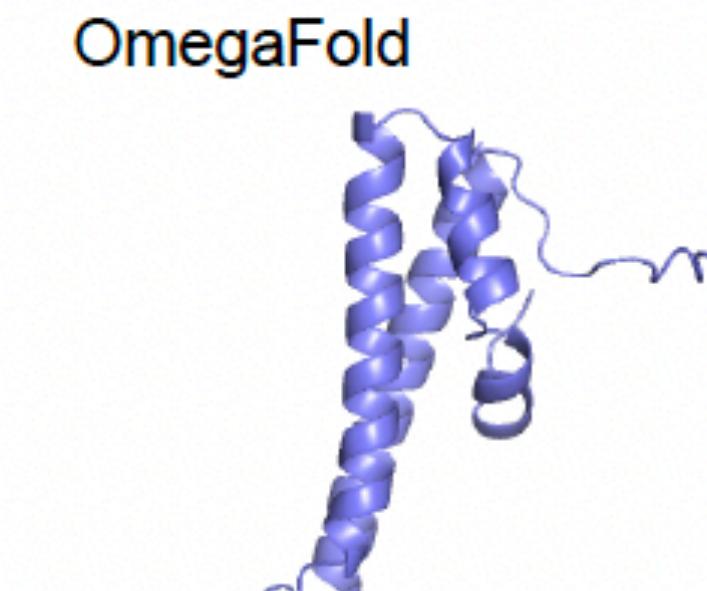
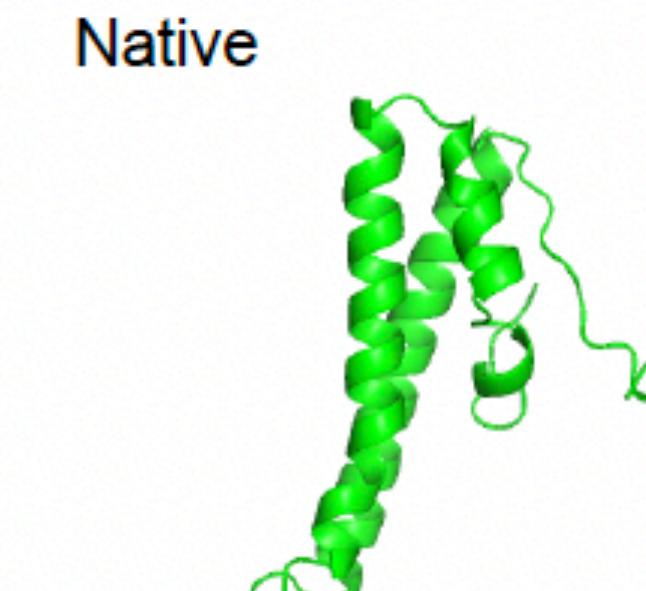
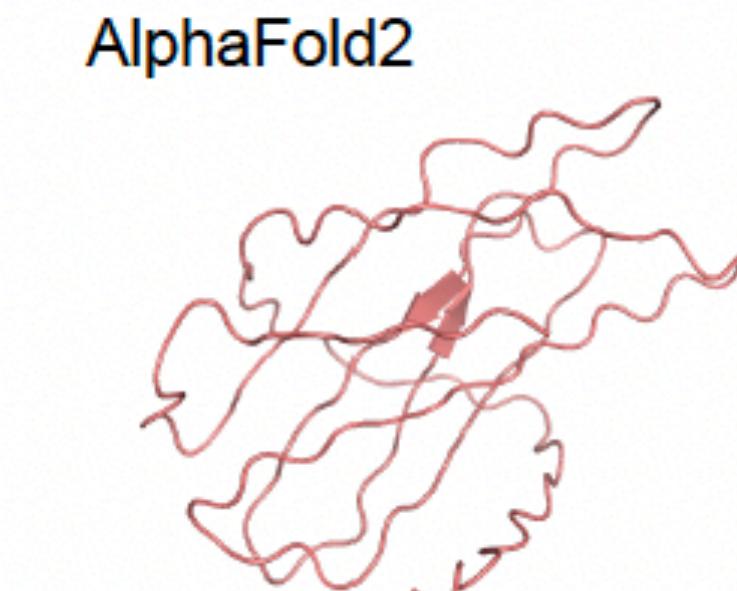
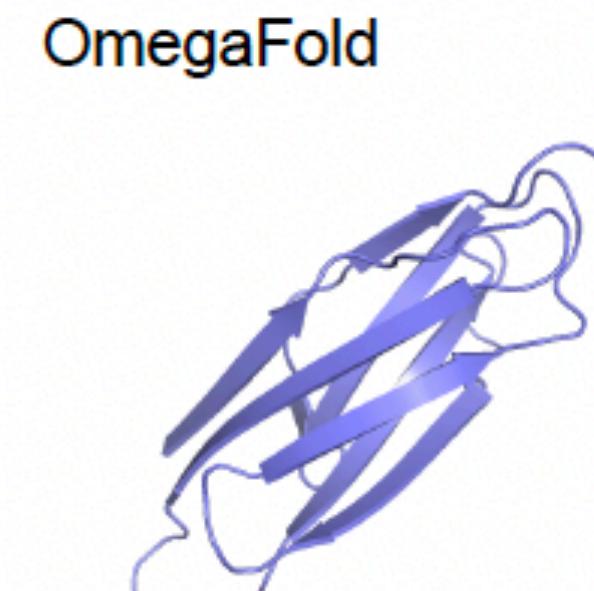
Prediction of Antibody Loops



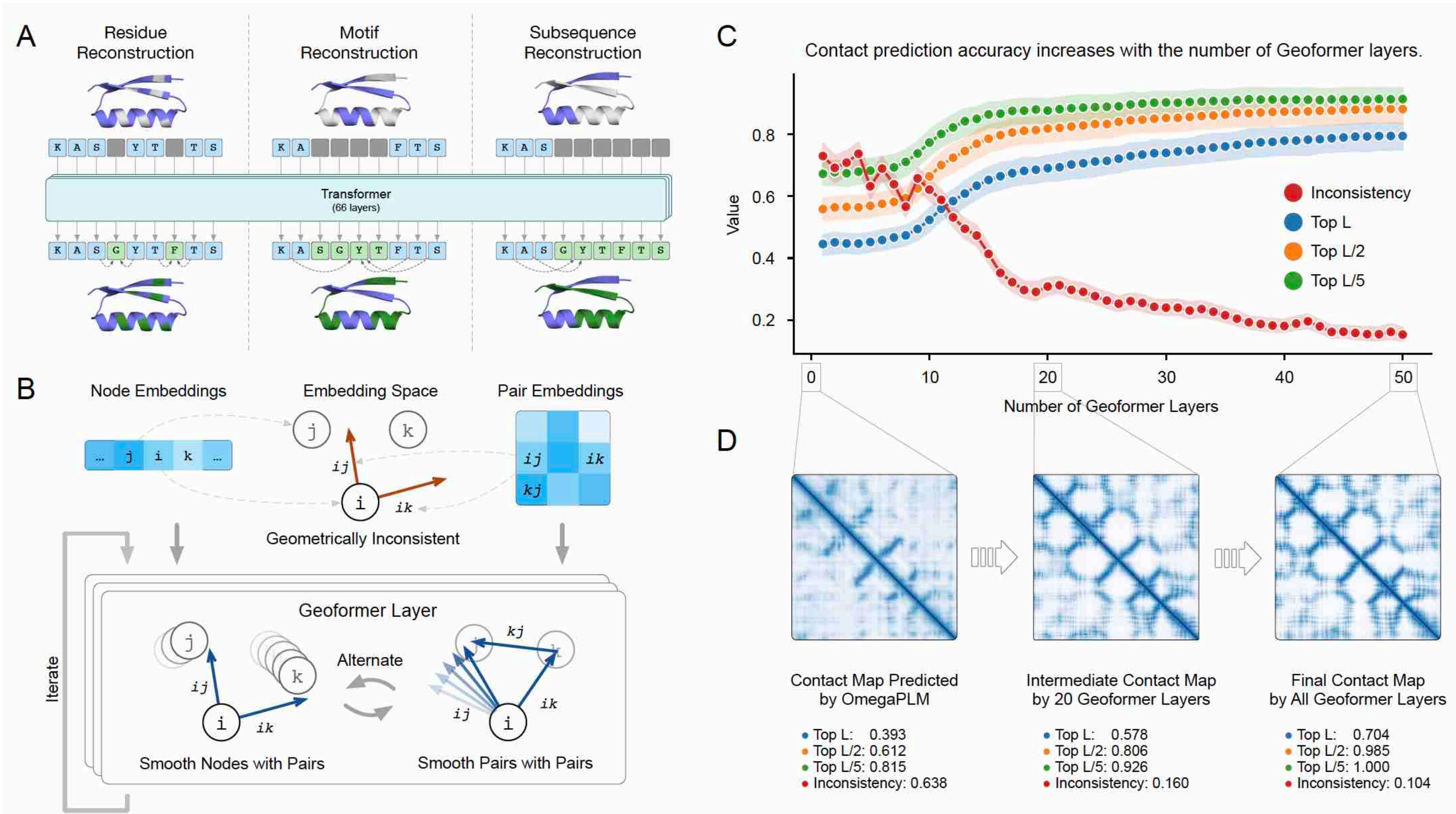
Prediction of Orphan Proteins

B

Orphan Proteins



Geoformer



Installing OmegaFold

- On OSCER, clone:

```
git clone https://github.com/HeliXonProtein/OmegaFold
```

- Make new Conda environment:

```
conda create -n omegafold python=3.9
```

```
conda activate omegafold
```

- The two types of GPUs on OSCER are NVIDIA K80m and A100

- 1 K80m on 1 node (c302, c307), 2 K80m on 1 node (c301), and 2 A100 on 1 node (c736)

- All are not supported by > PyTorch 1.10 (Omegafold was written with Pytorch 1.12)

- Thank you Nelson Liu for developing back support for old GPUs

- <https://github.com/nelson-liu/pytorch-manylinux-binaries/releases>

Installing OmegaFold

- Thank you Nelson Liu for developing back support for old GPUs

<https://github.com/nelson-liu/pytorch-manylinux-binaries/releases>

- Remake requirements.txt > req_backsupport.txt (PyTorch 1.9.0)
- In the omegafold conda environment

pip install -r req_backsupport.txt

- Finally,

python ./setup.py install

- To run: **omegafold [input.fasta] [working directory]**

Tutorial on OmegaFold

- Copy the following directory to your home:

```
cp -r /home/van/MacroMol_Modelling_Tutorial/04_omegafold/ .
```

- The folder is organized as follow:
 - 04_omegafold
 - Results
 - Tutorial
 - bax
 - parDE
 - paparde.fasta
 - of_monomer.slurm

04_omegafold/Tutorial/bax

- Bax.fasta
 - Amino acid sequence
- of_monomer.slurm

```
1 >bax
2 MDGSQEPRGGGPTSSEQIMKTGALLQGFIQDRAGRMGEAPEL
ALDPVPQDASTKKLSECLKRIGDELDNMELQRMIAAVTDSPRE
VFFRVAADMFSDGNFNWGRVVALFYFASKLVLKALCTKVPELIRT
IMGWTLDFLRERLLGWIQDQGGWDGLLSYFGPTWQTIVAGV
LTASLTIWKKMG
```

- SLURM script to run OmegaFold
- To run:

sbatch of_monomer.slurm

K80m (c307)

Wall Time: ~7 min

Prediction Time: ~172 sec

A100

Wall Time: ~5 min

Prediction Time: ~16 sec

```
1 #!/bin/bash
2 #SBATCH --partition=gpu
3 #SBATCH --ntasks=10
4 #SBATCH --output=%j.out
5 #SBATCH --error=%j.err
6 #SBATCH --time=00:30:00
7 #SBATCH --job-name=bax
8 #SBATCH --exclude=c736
9
10 date
11 hostname
12
13 module load CUDA/11.3.1
14
15 eval "$(ourdisk/hpc/ccats/dont_archive/van/.Programs/miniconda3/bin/conda shell.bash hook)"
16 conda activate omegafold
17
18 # To Run:
19 # omegafold [input.fasta] [working directory - output here]
20 omegafold bax.fasta .
21
22 date
23
```

SLURM Allocation requests

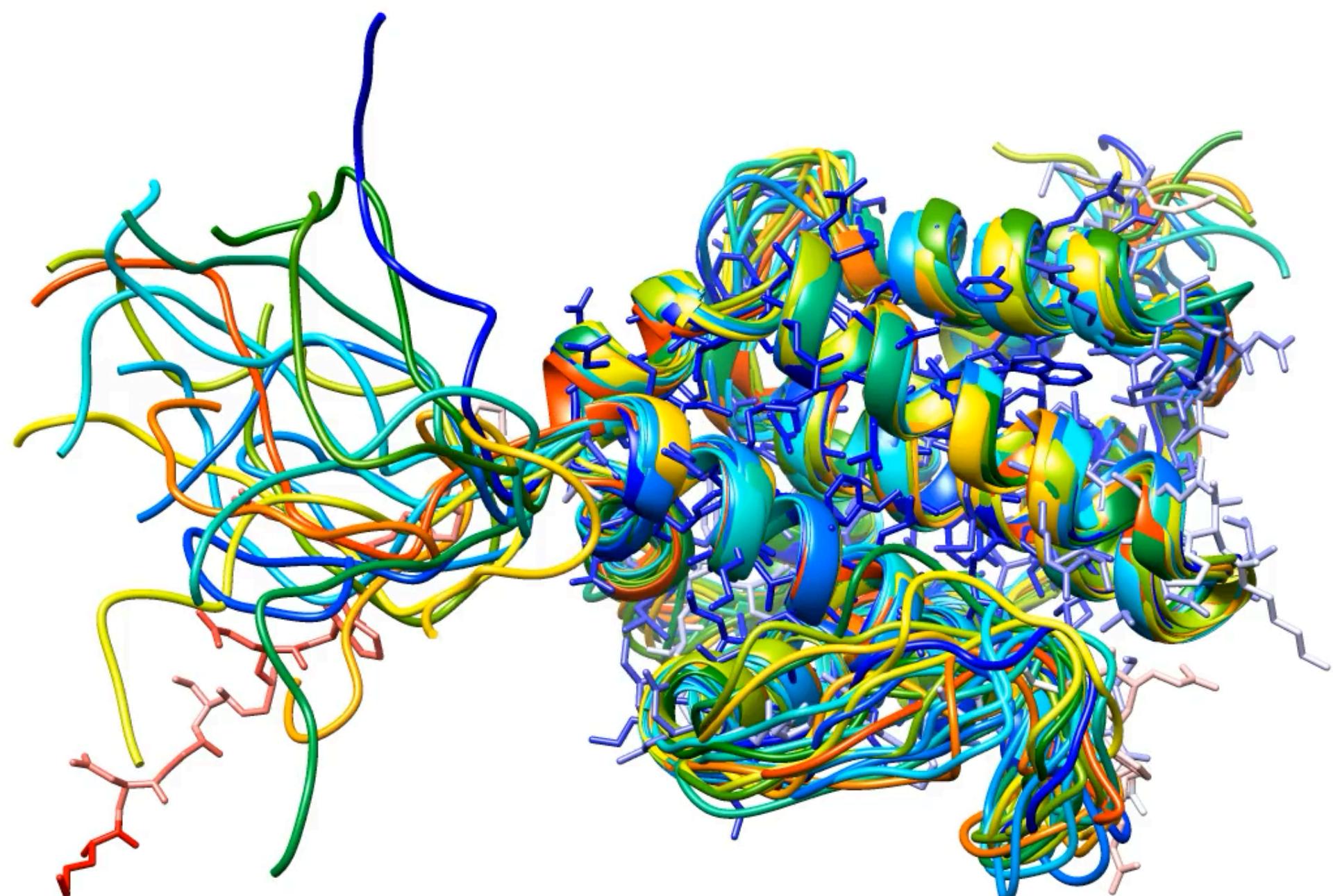
hostname = get name of node

Load CUDA

Omegafold conda environment

Running OmegaFold

bax.pdb



PDB 1F16
OmegaFold

04_omegafold/Tutorial/parDE1

- Currently, OmegaFold can only predict monomers
- However, input can take in multiple sequences and predict them individually

- paparde.fasta
 - Both ParD and ParE sequence

- of_monomer.slurm

- SLURM script to run OmegaFold

- To run:

```
sbatch of_monomer.slurm
```

```
1 >ParD
2 GSHMMMSLKWTRKAAADLDIAIYDHYVVLIGPEKALKAVQDIVEQVKP
   LQQVANQGAGRSEVPGVRTLTLERWPFSAPFRVKKGKEIQILRIDR
   VEITP
3 >ParE
4 GSHMMSTVVSFRADDALVAALDELARATHRDRPYHLRQALAQYLER
   QQWQVAAIDEGLADANAGRLLEHIEIEKRWGLQ
```

K80m (c307)

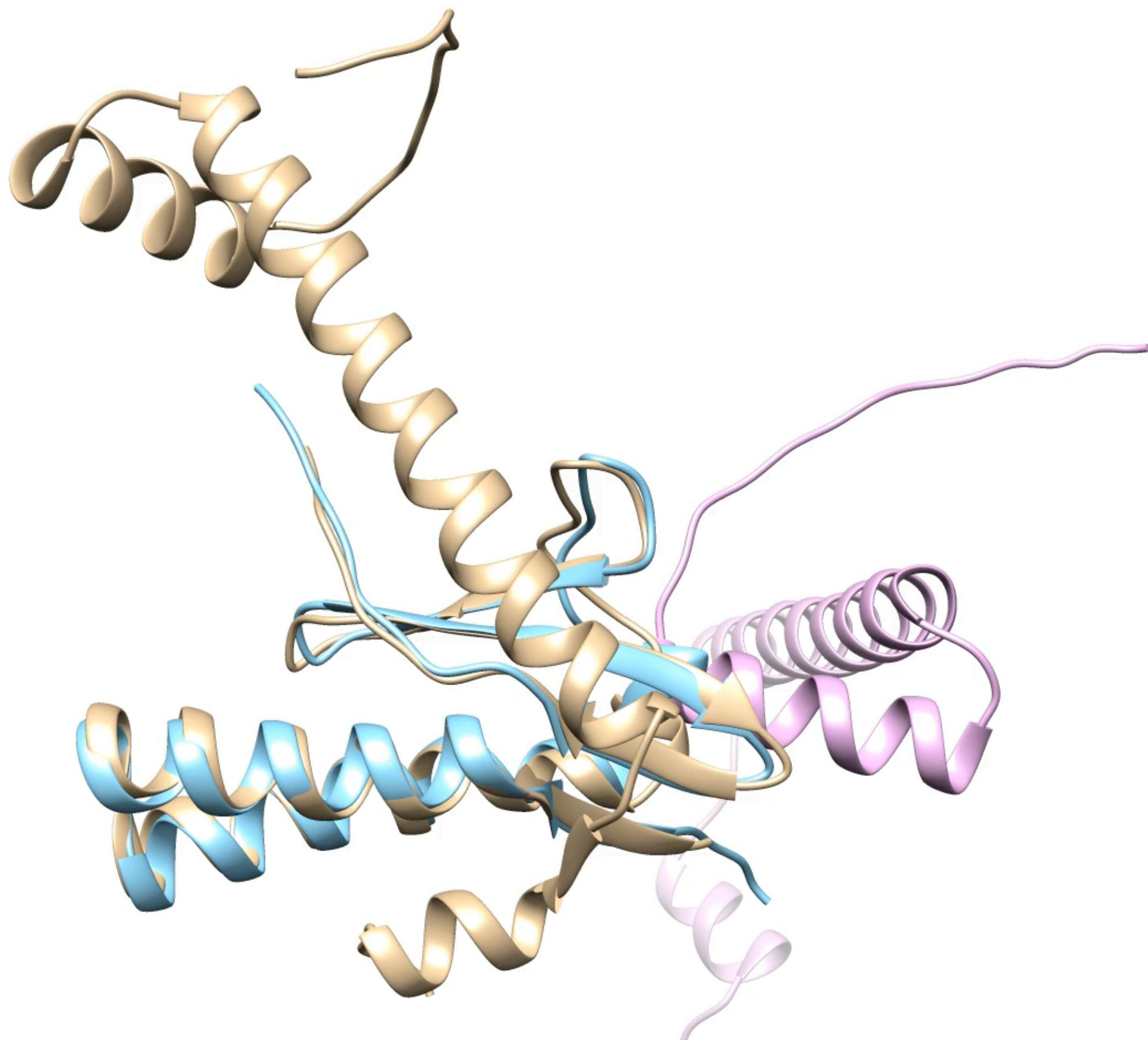
Wall Time: ~6 min

Prediction Time:

ParD ~39 sec

ParE ~50 sec

ParDE



PDB 6XRW
OmegaFold

04_omegafold/Tutorial/parDE2

- Lets modify the FASTA input so that we can predict the complex.

- paparde.fasta

- 4 G are added to connect ParD -> ParE

- of_monomer.slurm

- SLURM script to run OmegaFold

- To run:

```
sbatch of_monomer.slurm
```

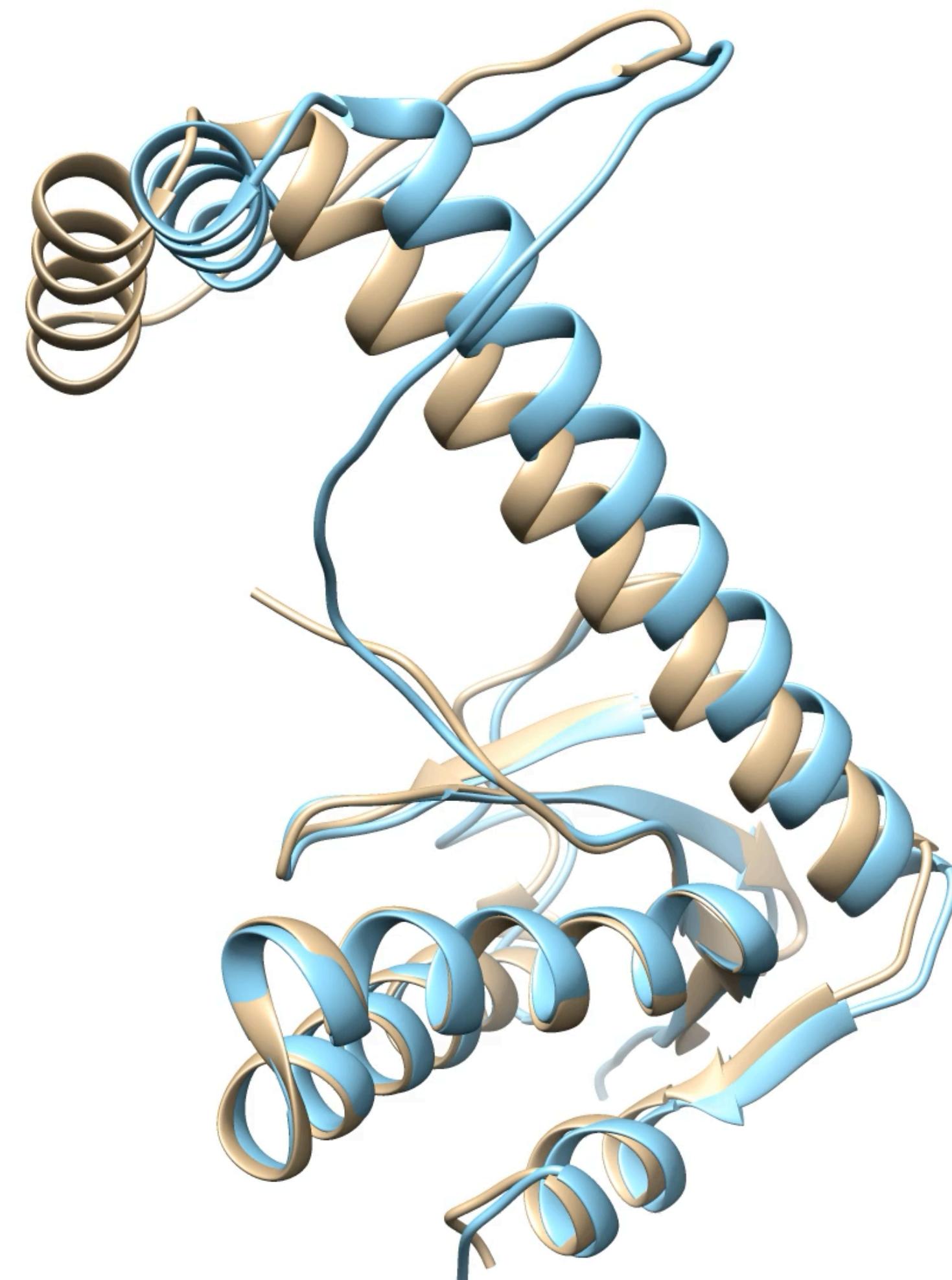
K80m (c302)

Wall Time: ~10 min

Prediction Time: ~ 156 sec

```
1 >ParDE.glinker
2 GSHMMSLKWTRKAAADLDIAIYDHVVVLIGPEKALKAVQDI
   VEQVKPLQQVANQGAGRSEVPGVRTLTLERWPFSAPFRV
   KGKEIQILRIDRVEITPGGGGGGSHMMSTVVSFRADDALV
   AALDELARATHRDRPYHLRQALAQYLERQQWQVAAIDEGL
   ADANAGRLLEHIEIEKRWGLQ
```

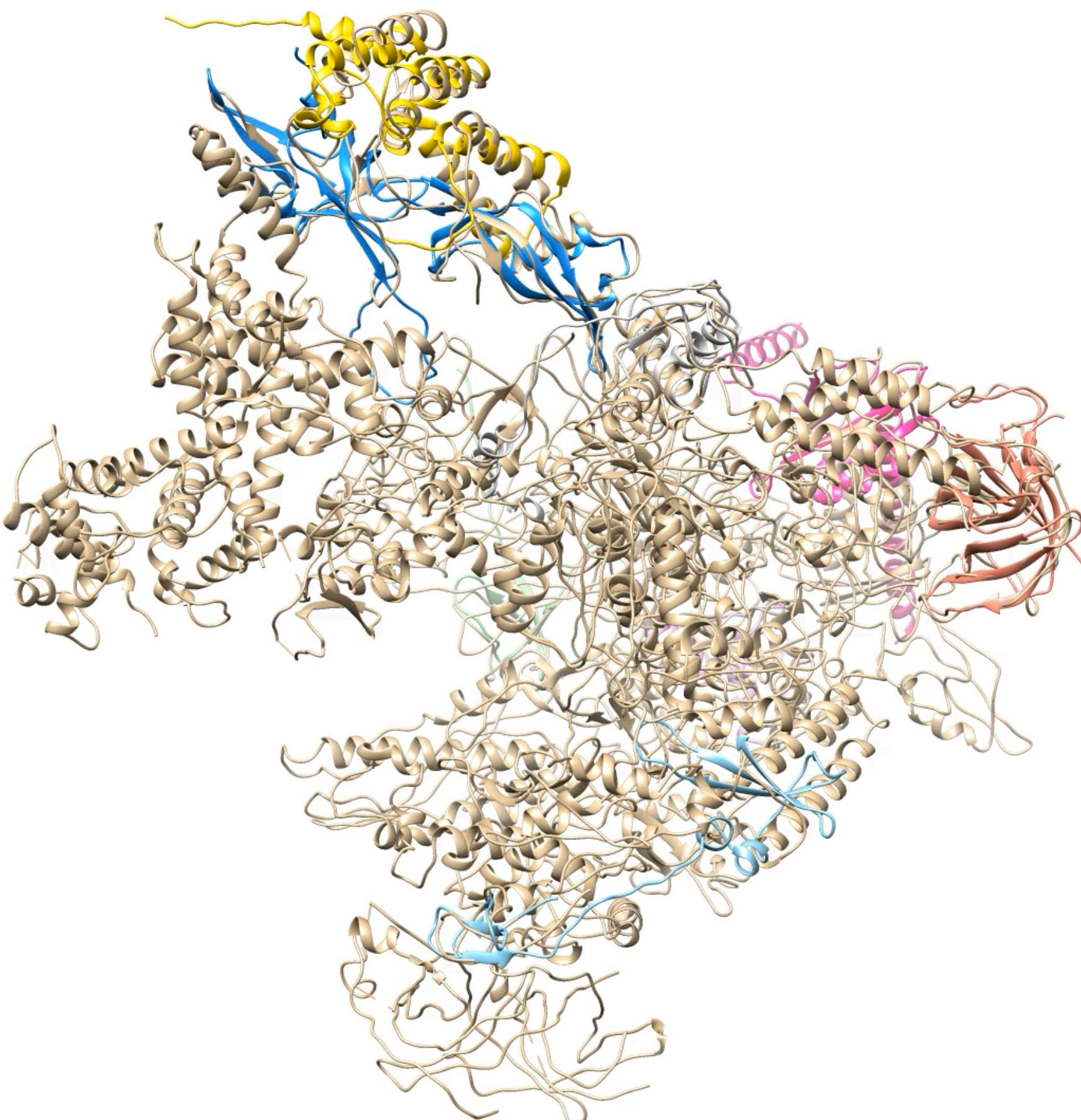
ParDE Complex



PDB 6XRW
OmegaFold

04_omegafold/Tutorial/pol3

- Lets try human RNA Pol3 (~1300 residues)
- PDB 7AST has Chain A-J (11)
- pol3.fasta
 - Ran this on K80m and out of memory (5 Gb total)



PDB 7AST
OmegaFold