HIGH PERFORMANCE RESEARCH COMPUTING

Introduction to AlphaFold for 3D Protein Structure Prediction on Grace

March 28, 2025

Michael Dickens

High Performance
Research Computing
DIVISION OF RESEARCH

AlphaFold for 3D Protein Structure Prediction on Grace

- AlphaFold History
- Running AlphaFold2
  - Grace ParaFold AlphaFold2 workflow
  - AlphaFold2 with reduced_dbs
  - AlphaFold2 confidence metrics
- Running AlphaFold3 on Grace
- HPRC Cluster Utilities
- Visualization of Results
  - view predictions in ChimeraX
- Job Resource Monitoring

AlphaFold History

- An Artificial Intelligence program developed by DeepMind
- 2018 AlphaFold 1 placed 1st at CASP 13
- 2020 AlphaFold 1 code released as open source
- 2020 AlphaFold2 placed 1st at CASP 14
- 2021 AlphaFold publication in Nature
  - Highly accurate protein structure prediction with AlphaFold
- 2021 AlphaFold2 code released as open source on GitHub
- 2024 AlphaFold3 available on the deepmind AlphaFold Server
  - 20 jobs per day allowed for academic researchers
- 2024 AlphaFold3 code available on github
- 2025 AlphaFold3 available on HPRC Grace cluster

Resource Limitations

- AlphaFold
  - Currently AlphaFold(2,3) can only utilize one GPU.
  - About 90% of processing is done on CPU when using DeepMind's workflow in a single job script.
- AlphaFold2 in Google Colab (web browser or ChimeraX app)
  - no guarantee of available resources in Colab
  - runs as a Jupyter notebook on Google Colab cloud servers
    - 12GB RAM max
    - a notebook can run for up to 12 hours per day
      - 24 hours per day with Colab Pro ($9.99/month)
    - not suitable for large predictions

Running AlphaFold 2 on Grace using ParaFold

AlphaFold2 Databases on Grace

`/scratch/data/bio/alphafold/2.3.2`

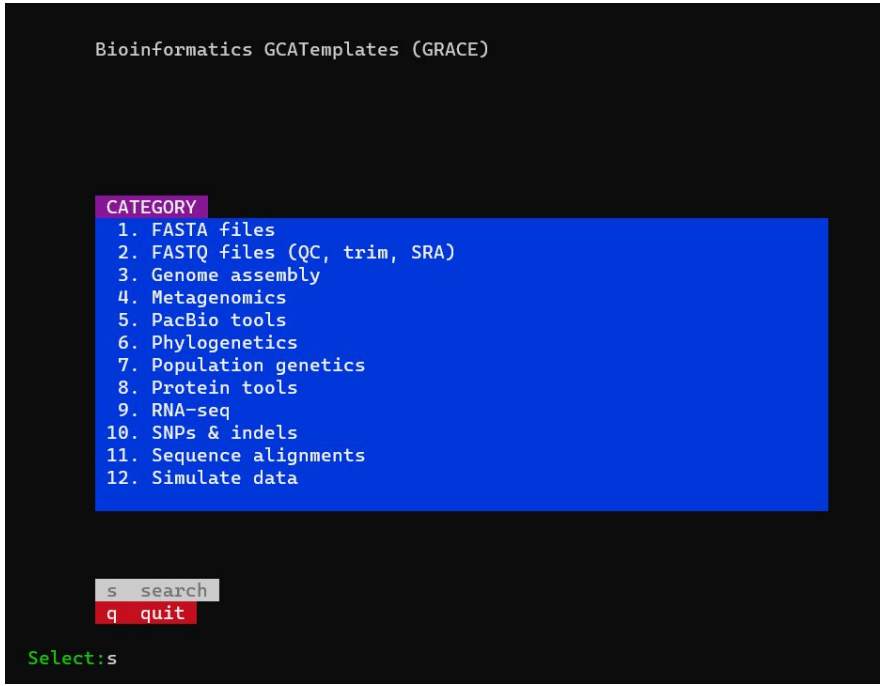
Database	Size	File Count	monomer	multimer
bfd	1.8T	7	✔	✔
mgnify	120G	2	✔	✔
params	5.3G	17	✔	✔
pdb70	56G	10	✔	-
pbd_mmcif	264G	211,106	✔	✔
pdb_seqres	257M	2	-	✔
uniprot	114G	2	-	✔
uniref30	467G	15	✔	✔
uniref90	77G	2	✔	✔
small_bfd	17G	2	✔	✔
example_data	6K	5	✔	✔
TOTAL	**2.9T**	**211,170**		

Finding AlphaFold2 template job scripts using GCATemplates on Grace

- Genomic Computational Analysis Templates have example input data so you can run the script for demo purposes.

```
mkdir $SCRATCH/af_demo
cd $SCRATCH/af_demo
gcatemplates
```

- Type **s** for search then enter **parafold** to search for the **parafold_2.0+alphafold 2.3.2** template script and select the **monomer_ptm** script option
- Review and submit the job script, which takes about 3 hours to complete.



Submit and Monitor the Job

- Run the cpuavail utility to see cluster usage status.

```
cpuavail
```

- Edit your job script to use 10 cores and 100 GB memory for CPU job.
- Submit the job script to the Slurm scheduler.
  - completes in about 7.5 hours so we will review a completed job

```
sbatch run_parafold_2.0_alphafold_2.3.2_a100_monomer_ptm_grace.sh

Submitted batch job 11760808
```

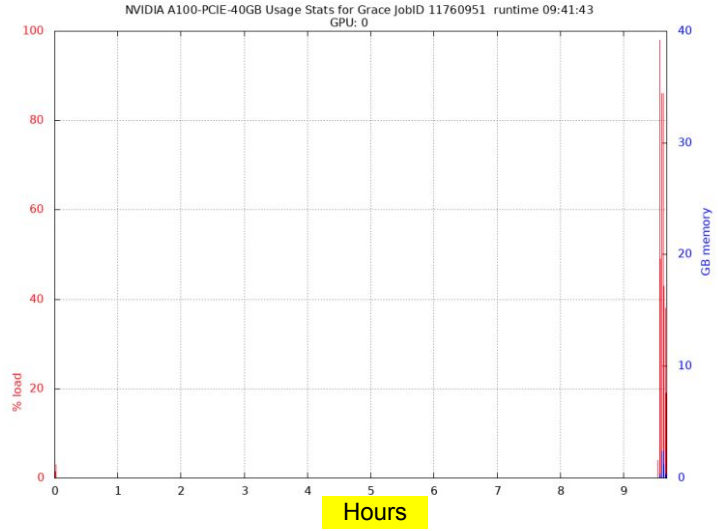
- Monitor the job status.

```
squeue --me

JOBID     NAME           USER      PARTITION  NODES  CPUS  STATE   TIME   TIME_LEFT   START_TIME          REASON  NODELIST
11760808  parafold-cpu   userid    cpu        1      24    RUNNING 6.59   6-23:53:01  2024-10-21T15:20    None    c398
```

Comparison of DeepMind vs ParaFold Workflows

- AlphaFold DeepMind's one job workflow (1 GPU job) vs ParaFold's two job workflow (1 CPU-only job + 1 GPU job) for the same monomer_ptm full_dbs analysis
- The ParaFold workflow significantly reduces GPU idle time and SUs charged.
  - DeepMind Workflow: **1639** SUs ParaFold Workflow: **226** SUs (cpu+gpu)

AlphaFold 2.3.1



AlphaFold 2.3.2

The first job of the ParaFold workflow (CPU-only) completed in 7.5 hours

ParaFold Workflow

- The ParaFold module uses the official AlphaFold2 code.
- ParaFold divides the AlphaFold2 workflow into two steps which can be run as two separate jobs:
  - CPU-only: processing the CPU steps to generate multiple sequence alignments
  - GPU: processing the GPU steps to generate predictions
- Test run of multimer (T1083_T1084_multimer.fasta) with full_dbs on ACES
- Runtimes for the same job script varied +- 1 hour; TM-scores also vary

AlphaFold 2.3.2	Runtime	Highest Scoring Model	TM-score**
ParaFold	3 hrs 10 min*	model_1_multimer_v3_pred_4	0.892
DeepMind	2 hrs 45 min	model_1_multimer_v3_pred_1	0.883

***** combined time for the separate CPU 3 hour job and GPU 10 min job
****** measure of similarity between two protein structures

https://github.com/Zuricho/ParallelFold

AlphaFold2 Runtimes on Grace

- Can be run as a job script requesting one GPU
- Shared databases are available: 2.6TB total size

AlphaFold 2.2.3	monomer_ptm 20 aa	multimer 98 & 73 aa
A100	36 minutes	4 hours 49 minutes
A40	35 minutes	-
RTX 6000	33 minutes	4 hours 44 minutes
T4	33 minutes	4 hours 45 minutes
CPU only	40 minutes	6 hours 11 minutes

AlphaFold2 reduced_dbs Job Script

Example AlphaFold2 Job Script Using reduced_dbs

- ParaFold does not support reduced_dbs. Use AlphaFold workflow for reduced_dbs/
- **monomer + reduced_dbs**
- dbs in **red** required for monomer + reduced_dbs
- Small_bfd_database is a subset of BFD and is generated by taking the first non-consensus sequence from every cluster in BFD.
- AlphaFold can only use one GPU, so reserve half the CPU and memory resources so another job can use the other GPU.

```
#!/bin/bash
#SBATCH --job-name=alphafold        # job name
#SBATCH --time=2-00:00:00           # max job run time dd-hh:mm:ss
#SBATCH --ntasks-per-node=1         # tasks (commands) per compute node
#SBATCH --cpus-per-task=24          # CPUs (threads) per command
#SBATCH --mem=180G                  # total memory per node
#SBATCH --gres=gpu:a100:1           # request 1 A100 GPU
#SBATCH --output=stdout.%x.%j       # save stdout to file
#SBATCH --error=stderr.%x.%j        # save stderr to file

module purge
module load GCC/11.3.0  OpenMPI/4.1.4 AlphaFold/2.3.1-CUDA-11.7.0

ALPHAFOLD_DATA_DIR=/scratch/data/bio/alphafold/2.3.2

# run jobstats in the background (&) to monitor cpu and gpu usage
jobstats &

run_alphafold.py \
 --use_gpu_relax \
 --data_dir=$ALPHAFOLD_DATA_DIR \
 --uniref90_database_path=$ALPHAFOLD_DATA_DIR/uniref90/uniref90.fasta \
 --mgnify_database_path=$ALPHAFOLD_DATA_DIR/mgnify/mgy_clusters_2022_05.fa \
 --small_bfd_database_path=$ALPHAFOLD_DATA_DIR/small_bfd/bfd-first_non_consensus_sequences.fasta \
 --model_preset=monomer \
 --pdb70_database_path=$ALPHAFOLD_DATA_DIR/pdb70/pdb70 \
 --template_mmcif_dir=$ALPHAFOLD_DATA_DIR/pdb_mmcif/mmcif_files \
 --obsolete_pdbs_path=$ALPHAFOLD_DATA_DIR/pdb_mmcif/obsolete.dat \
 --max_template_date=2024-1-1 \
 --db_preset=reduced_dbs \
 --output_dir=out_alphafold \
 --fasta_paths=/scratch/data/bio/alphafold/example_data/1L2Y.fasta

# run jobstats to create a graph of cpu and gpu usage for this job
jobstats
```

Unified Memory

```
#!/bin/bash
#SBATCH --job-name=alphafold        # job name
#SBATCH --time=2-00:00:00           # max job run time dd-hh:mm:ss
#SBATCH --ntasks-per-node=1         # tasks (commands) per compute node
#SBATCH --cpus-per-task=24          # CPUs (threads) per command
#SBATCH --mem=180G                  # total memory per node
#SBATCH --gres=gpu:a100:1           # request 1 A100 GPU
#SBATCH --output=stdout.%x.%j       # save stdout to file
#SBATCH --error=stderr.%x.%j        # save stderr to file

# version 2.3.1 has the unified memory variable already configured
module purge
module load GCC/11.3.0  OpenMPI/4.1.4 AlphaFold/2.3.1-CUDA-11.7.0
```

- Unified memory can be used to request more than just the total GPU memory for the JAX step in AlphaFold2.
  - A100 GPU has 40GB memory.
  - `XLA_PYTHON_CLIENT_MEM_FRACTION` (configured to 3.0 in the AlphaFold 2.3.1 module)
    - ParaFold is configured to 4.0
- this example script has 120 GB of unified preallocated memory:
  - 40 GB from A100 GPU + 80 GB DDR from motherboard.

  https://jax.readthedocs.io/en/latest/gpu_memory_allocation.html

AlphaFold2 Confidence Metrics

AlphaPickle for Visualization of Confidence Scores

AlphaPickle can be used to create graphs for pLDDT and PAE scores.

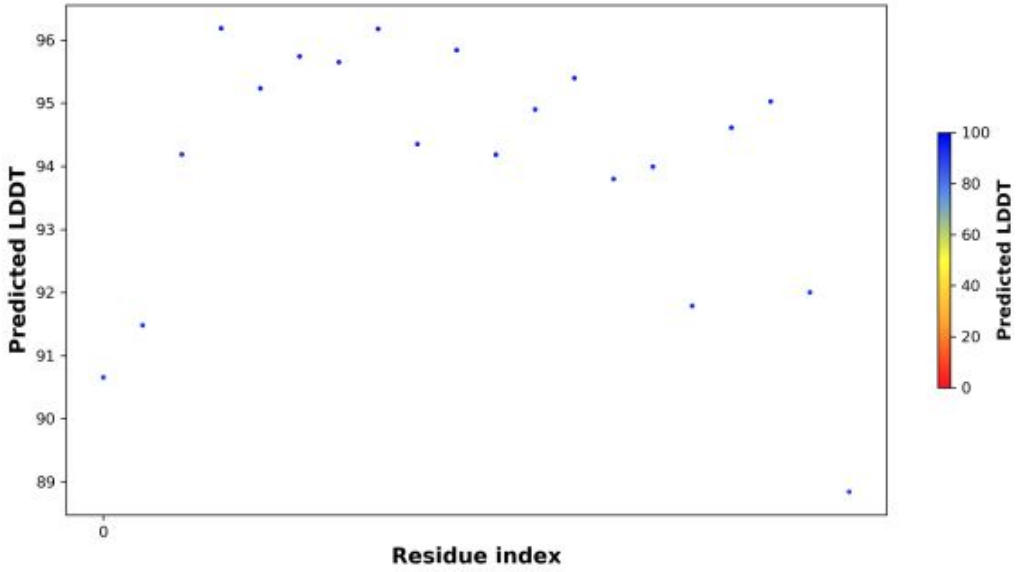
- Graphing PAE scores is only available for the **monomer_ptm** and **multimer** model presets.
- Load the AlphaPickle module at the beginning of the job script
- Run AlphaPickle at the end, specifying the output directory used in the run_alphafold.py command.

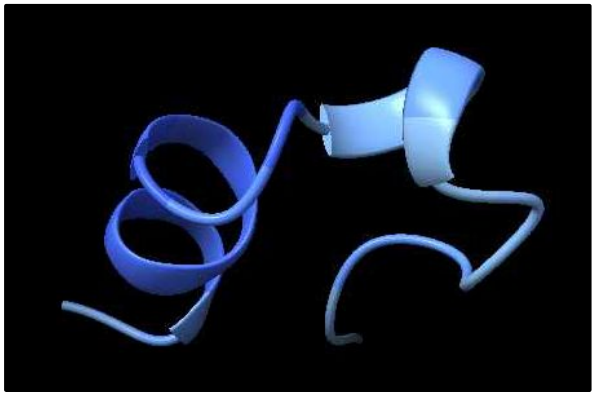
- pLDDT: scale from 0 - 100 of per-residue estimate of prediction confidence
- PAE: Predicted Alignment Error

https://github.com/mattarnoldbio/alphapickle

```
#!/bin/bash
#SBATCH --job-name=parafold-cpu     # job name
#SBATCH --time=2-00:00:00           # max job run time dd-hh:mm:ss
#SBATCH --ntasks-per-node=1         # tasks (commands) per compute node
#SBATCH --cpus-per-task=24          # CPUs (threads) per command
#SBATCH --mem=180G                  # total memory per node
#SBATCH --output=stdout.%x.%j       # save stdout to file
#SBATCH --error=stderr.%x.%j        # save stderr to file

module purge
module load GCC/11.3.0  OpenMPI/4.1.4 ParaFold/2.0-CUDA-11.7.0

ALPHAFOLD_DATA_DIR=/scratch/data/bio/alphafold/2.3.2
protein_fasta=/scratch/data/bio/alphafold/example_data/1L2Y.fasta
max_template_date=2025-1-1

# First, run CPU-only steps to get multiple sequence alignments:
run_alphafold.sh -d $ALPHAFOLD_DATA_DIR -o pf_output_dir -p monomer_ptm \
 -i $protein_fasta -t $max_template_date -f

# Second, run GPU step as a separate job after the first part completes successfully:
sbatch --job-name=parafold-gpu --time=2-00:00:00 --ntasks-per-node=1 \
--cpus-per-task=24 --mem=180G --gres=gpu:a100:1 --partition=gpu \
--output=stdout.%x.%j --error=stderr.%x.%j --dependency=afterok:$SLURM_JOBID<<EOF
#!/bin/bash
module purge
module load GCC/11.3.0  OpenMPI/4.1.4 ParaFold/2.0-CUDA-11.7.0
module load AlphaPickle/1.4.1
run_alphafold.sh -g -u 0 -d $ALPHAFOLD_DATA_DIR -o pf_output_dir -p monomer_ptm \
 -i $protein_fasta -t $max_template_date
# graph pLDDT and PAE .pkl files
run_AlphaPickle.py -od pf_output_dir/1L2Y
EOF
```

Visualize AlphaFold2 pLDDT Scores



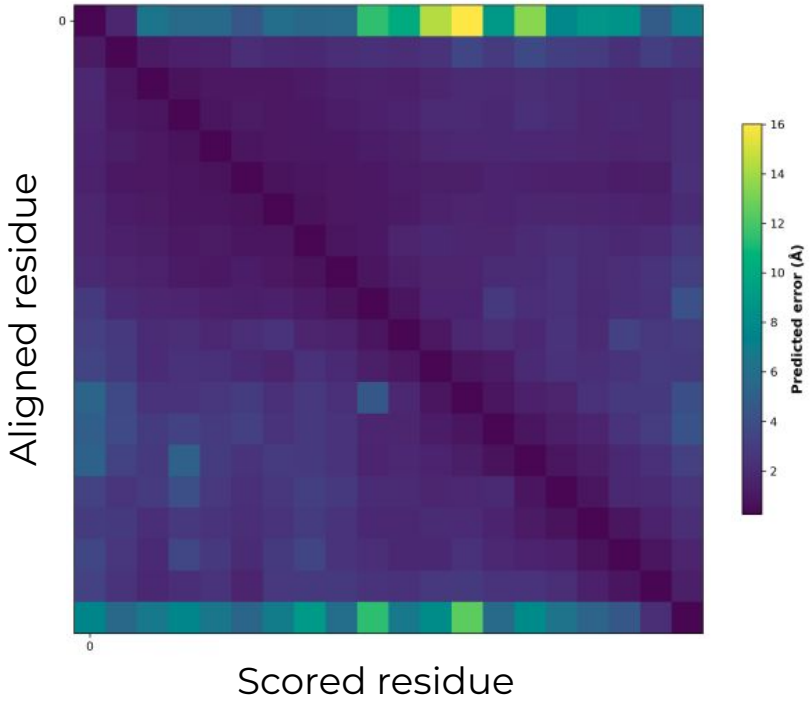


> 90 = Very high
70 - 90 = Confident
50 - 70 = Low
< 50 = Very low

`/scratch/training/parafold/monomer_ptm/out_parafold_1L2Y_monomer_ptm/1L2Y/ranked_0_pLDDT.png`

Use the portal Files and navigate to the output directory to view the image.
You may get different results compared to the image above when using reduced_dbs.

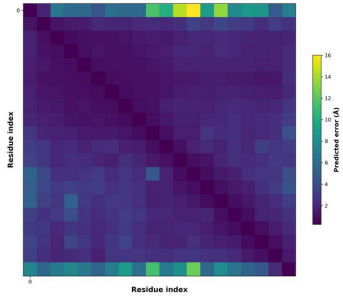
Visualize AlphaFold2 PAE Results (monomer_ptm)



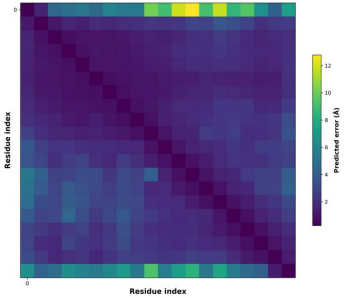
- Low Predicted Aligned Error (PAE) value has higher confidence of accuracy
- Must use monomer_ptm or multimer as model_preset to create PAE image
- The colour at position (x, y) indicates AlphaFold's expected position error at residue x, when the predicted and true structures are aligned on residue y.

```
/scratch/training/parafold/monomer_ptm/out_parafold_1L2Y_monomer_ptm/1L2Y/ranked_0_PAE.png
```

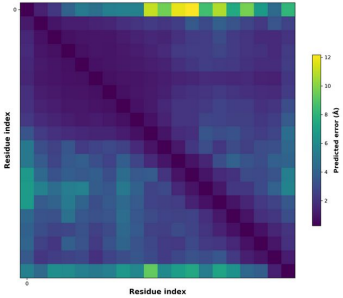
Evaluating Models



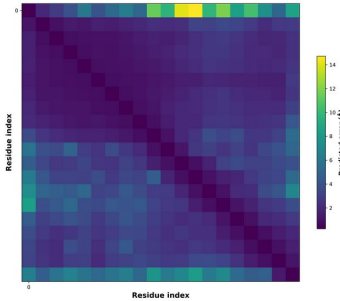
ranked_0



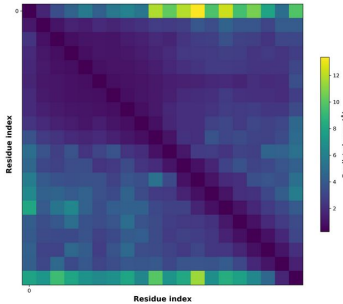
ranked_1



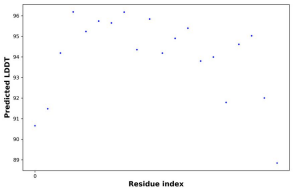
ranked_2

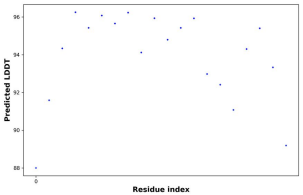


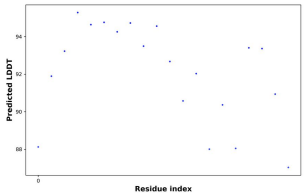
ranked_3

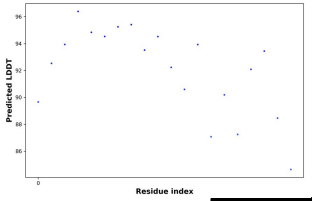


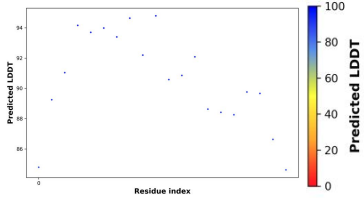
ranked_4











See which model has the top rank based on pLDDT score.

```
cat
/scratch/training/parafold/monomer_ptm/out_parafold_1L2Y_monomer_
ptm/1L2Y/ranking_debug.json
```


```
"plddts": {
  "model_1_ptm_pred_0": 94.00143084962215,
  "model_2_ptm_pred_0": 93.92314232455674,
  "model_3_ptm_pred_0": 90.57195481381758,
  "model_4_ptm_pred_0": 92.07012004352153,
  "model_5_ptm_pred_0": 92.0243899816539
},
"order": [
  "model_1_ptm_pred_0",
  "model_2_ptm_pred_0",
  "model_4_ptm_pred_0",
  "model_5_ptm_pred_0",
  "model_3_ptm_pred_0"
]
```


ranked_0

ranked_4

Running AlphaFold3 on Grace

AlphaFold2 vs AlphaFold3

The new AlphaFold model demonstrates substantially improved accuracy over many previous specialized tools: far greater accuracy for protein–ligand interactions compared with state-of-the-art docking tools, much higher accuracy for protein–nucleic acid interactions compared with nucleic-acid-specific predictors and substantially higher antibody–antigen prediction accuracy compared with AlphaFold-Multimer v.2.37,8.

(from AlphaFold3 Abstract)

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

AlphaFold3 requires non-commercial users to agree to the terms of use in their Google form in order to download the models parameters file

ALPHAFOLD 3 MODEL PARAMETERS TERMS OF USE

Last Modified: 2024-11-09

AlphaFold 3 is an AI model developed by Google DeepMind and Isomorphic Labs. It generates 3D structure predictions of biological molecules, providing model confidence for the structure predictions. We make the trained model parameters and output generated using those available free of charge for certain non-commercial uses, in accordance with these terms of use and the AlphaFold 3 Model Parameters Prohibited Use Policy.

Key things to know when using the AlphaFold 3 model parameters and output

1. The AlphaFold 3 model parameters and output are **only** available for non-commercial use by, or on behalf of, non-commercial organizations (*i.e.*, universities, non-profit organizations and research institutes, educational, journalism and government bodies). If you are a researcher affiliated with a non-commercial organization, provided **you are not a commercial organisation or acting on behalf of a commercial organisation,** this means you can use these for your non-commercial affiliated research.
2. You **must not** use nor allow others to use:
    i. AlphaFold 3 model parameters or output in connection with **any commercial activities, including research on behalf of commercial organizations;** or
    ii. AlphaFold 3 output to **train machine learning models** or related technology for **biomolecular structure prediction** similar to AlphaFold 3.
3. You ***must not*** **publish or share AlphaFold 3 model parameters**, except sharing these within your organization in accordance with these Terms.
4. You ***can*** **publish, share and adapt AlphaFold 3** ***output*** in accordance with these Terms, including the requirements to provide clear notice of any modifications you make and that ongoing use of AlphaFold 3 output and derivatives are subject to the AlphaFold 3 Output Terms of Use.

https://github.com/google-deepmind/alphafold3/blob/main/WEIGHTS_TERMS_OF_USE.md

Decompress your af3.bin.zstd file

The af3.bin.zstd file is the file downloaded after filling out the user agreement Google form for AlphaFold3 models

```
1.  module purge
2.  module load GCCcore/13.3.0 zstd/1.5.6
3.  unzstd af3.bin.zstd
4.  module purge
```

Checking GPU Configuration & Availability on Grace

- Use the command line (shell) to see the current GPU configuration and availability.
- If there are no GPU nodes in the AVAILABILITY output, it means that the GPUs are busy with other jobs and a GPU job that you submit may take a while to start.

```
[userid@grace ~]$ gpuavail

    CONFIGURATION
NODE           NODE
TYPE           COUNT
--------------------
gpu:a100:2     100
gpu:a40:3      15
gpu:rtx:2      9
gpu:t4:4       8


               AVAILABILITY
NODE   GPU     GPU    GPUs     CPUs   GB MEM
NAME   TYPE    COUNT  AVAIL    AVAIL  AVAIL
-----------------------------------------------
g003   a100      2      1       47    296
g004   a100      2      2       48    360
g005   a100      2      2       48    360
g006   a100      2      2       48    360
g008   a100      2      1       47    356
```

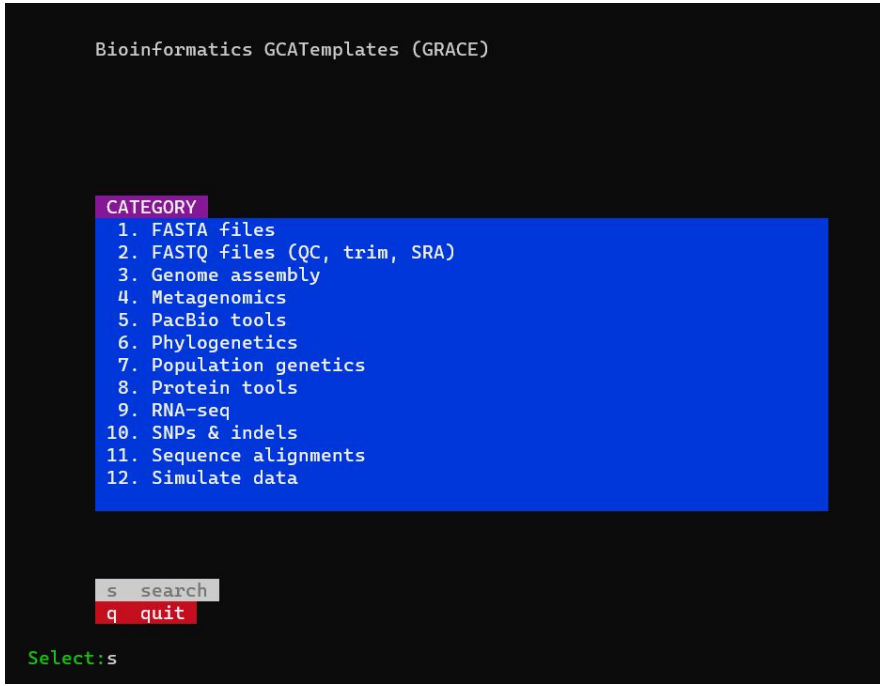
https://hprc.tamu.edu/kb/Software/useful-tools/gpuavail

Finding AlphaFold3 template job scripts using GCATemplates on Grace

- Genomic Computational Analysis Templates have example input data so you can run the script for demo purposes.

```
mkdir $SCRATCH/af3_demo
cd $SCRATCH/af3_demo
gcatemplates
```

- Type **s** for search then enter **alphafold** to search for the **alphafold_3.0.1** template scripts. Select the script based on GPU availability from gpuavail command.
- Follow the 10 steps in the job script to prepare the working directory.



Prepare the Working Directory

1. mkdir code_3.0.1
2. cd code_3.0.1
3. git clone https://github.com/google-deepmind/alphafold3.git
4. cd alphafold3
5. git checkout v3.0.1
6. cd ../..
7. mkdir input output models
8. move your af3.bin file into the models directory (symlink doesn't work)
9. configure your sequence in the file: input/alphafold_input.json
10. Your starting directory structure will look like the following:

```
.
├── code_3.0.1
│   └── alphafold3
├── inputs
│   └── alphafold_input.json
├── outputs
├── run_alphafold_3.0.1_a100_1job_grace.sh
└── models
    └── af3.bin
```

Submit and Monitor the Job

- Run the gpuavail utility to see cluster usage status.

```
gpuavail
```

- Submit the job script to the Slurm scheduler.
  - completes in about 15 minutes

```
sbatch run_alphafold_3.0.1_a100_1job_grace.sh

Submitted batch job 12876476
```

- Monitor the job status.

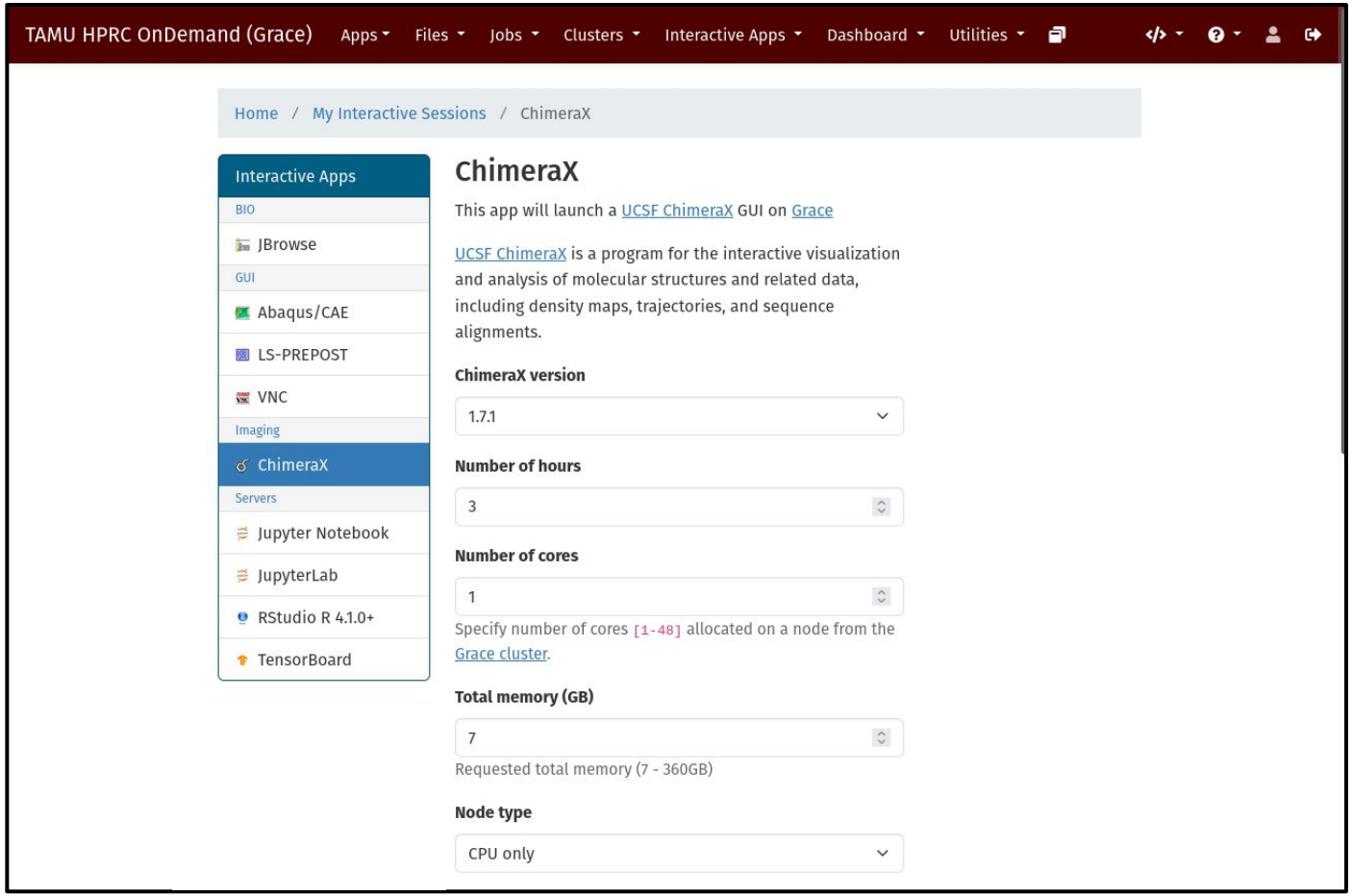
```
squeue --me

JOBID     NAME           USER      PARTITION  NODES  CPUS  STATE    TIME    TIME_LEFT    START_TIME          REASON  NODELIST
12876476  alphafold3     userid    gpu        1      16    RUNNING  6.59    6-23:53:01   2025-03-27T14:05    None    g063
```

AlphaFold3 Databases on Grace

`/scratch/data/bio/alphafold3/2025.03.13/`

Database	Size	File Count
bfd	117G	1
rnacentral	13G	1
rfam	218M	1
pdb_seqres	223M	1
nt_rna	76G	1
mgy_clusters	120G	1
uniref90	67G	1
mmcif_files	234G	195,860
uniprot_all	102G	1
TOTAL	**729G**	**195,868**

Launch ChimeraX for Visualizing Results



HPRC Cluster Utilities

HPRC Cluster Utilities

Many cluster utilities are available to help you query resources at the command line such as available nodes, GPUs, cores, memory, template job scripts and shared conda and Python environments.

myjob	**gpuavail**
maxconfig	**cpuavail**
gcatemplates	**envsavail**
jobstats	**venvavail**
toolchains	**maintenance**

use the **-h** or **--help** flag with any utility to see available options

Show Your Job Details Using myjob

- The **`myjob`** command can be used to see detailed information related to your job.
  - Status (PENDING, RUNNING, COMPLETED, FAILED, …)
  - Node List
  - Submit time, Start time, End time, Total runtime
  - CPU Efficiency
  - Memory Utilized, Memory Efficiency
- **`myjob`** will advise you if your job is PENDING due to a scheduled maintenance.
- **`myjob`** will advise you if your job FAILED due to CRLF characters in the job script and provide a link to the HPRC documentation on how to resolve this issue.
- **`myjob`** will advise you if your job FAILED due to file or disk quota being reached.
  - will show you the directory in your $HOME directory that has the most files when $HOME file quota is reached.

https://hprc.tamu.edu/kb/Software/useful-tools/myjob

Show Your Job Details Using myjob

```
[userid@grace ~]$ myjob 12876476

              Job ID: 12876476
             Cluster: grace
          User/Group: userid/userid
             Account: 1234567891011
         SUs charged: 110.76
               State: COMPLETED (exit code 0)
           Partition: gpu
          Node Count: 1
            NodeList: g063
      Cores per node: 16
        CPU Utilized: 01:31:32
      CPU Efficiency: 8.09% of 18:51:12 core-walltime
         Submit time: 2025-03-27 14:05:27
          Start time: 2025-03-27 14:05:33
            End time: 2025-03-27 15:16:15
 Job Wall-clock time: 01:10:42
     Memory Utilized: 7.24 GB
   Memory Efficiency: 4.52% of 160.00 GB
            Job Name: alphafold3
Job Submit Directory: /scratch/user/netid/af3_demo
         Submit Line: sbatch run_alphafold_3.0.1_a100_1job_grace.sh
```

use the -h flag to view usage

```
myjob -h
```

PENDING Job due to a Scheduled Maintenance

```
[userid@grace ~]$ myjob 1320633

               Job ID: 1320633
              Cluster: faster
           User/Group: userid/userid
              Account: 123456789101
                State: PENDING
               Reason: ReqNodeNotAvail, Reserved for maintenance
          Submit time: 2024-10-14 10:09:59
            Partition: cpu
           Node Count: 1
             NodeList: None assigned
                Cores: 1
                 Note: Efficiency not available for jobs in the PENDING state.
             Job Name: picard
Job Submit Directory: /scratch/user/userid/myproject/picard
          Submit Line: sbatch run_bwa_samtools_pilon_faster.sh
                 Note: job is PENDING due to runtime overlapping with maintenance window.
                 Note: maintenance will begin in  22 hours, and 49 minutes.
```

Viewing Maximum Available Resources

The **maxconfig** command will show the recommended Slurm parameters for the maximum available resources (cores, memory, time) per node for a specified accelerator or partition (default Grace partition: long) and show SUs charged.

```
[userid@grace ~]$ maxconfig

  Grace partitions:   short  medium  long  xlong  vnc  gpu  bigmem  special  gpu-a40
  Grace GPUs in gpu partition:  a100:2  a40:3  rtx:2  t4:4

  Showing max parameters (cores, mem, time) for partition long

  CPU-billing * hours * nodes =   SUs
           48 *   168 *     1 = 8,064

#!/bin/bash
#SBATCH --job-name=my_job
#SBATCH --time=7-00:00:00
#SBATCH --nodes=1           # max 64 nodes for partition long
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=48
#SBATCH --mem=360G
#SBATCH --output=stdout.%x.%j
#SBATCH --error=stderr.%x.%j
```

Viewing Maximum Available Resources

See the recommended Slurm parameters for requesting a 1 x A100 GPU with ½ the total CPUs and memory when there are 2 x A100s per node.

```
[userid@grace ~]$ maxconfig -g a100 -G 1

  Grace partitions:   short  medium  long  xlong  vnc  gpu  bigmem  special  gpu-a40
  Grace GPUs in gpu partition:  a100:2  a40:3  rtx:2  t4:4

  Showing 1/2 of total cores and memory for using 1 x a100 GPU

  (CPU-billing + (GPU-billing * GPU-count)) * hours * nodes = SUs
  (         25 + (         72 *         1)) *    96 *     1 = 9,312

#!/bin/bash
#SBATCH --job-name=my_job
#SBATCH --time=4-00:00:00
#SBATCH --nodes=1              # max 32 nodes for partition gpu
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=24
#SBATCH --mem=180G
#SBATCH --gres=gpu:a100:1
#SBATCH --output=stdout.%x.%j
#SBATCH --error=stderr.%x.%j
```

Grace SUs Charged: GPU vs CPU-only Jobs

show SU rate for 1 day CPU-only

```
maxconfig -d 1
```

```
CPU-billing * hours * nodes =  SUs
         48 *    24 *     1 =  1,152
```

show SU rate for 1 x A100 GPU for 1 day

```
maxconfig -g a100 -G 1 -d 1
```

```
(CPU-billing + (GPU-billing * GPU-count)) * hours * nodes = SUs
(         25 + (         72 *         1)) *    24 *     1 = 2,328
```

show SU charge rate

```
maxconfig -h
```

```
SU rate per GPU:
     GPU   SUs per hour
     ----  ------------
     a100        72
     a40         48
     rtx         48
     t4          24
```

Check non-GPU node Availability

Use the **`cpuavail`** command to see non-GPU nodes readily available for jobs. Non-GPU nodes not in the AVAILABILITY table are busy with other jobs and will be available after jobs have completed.

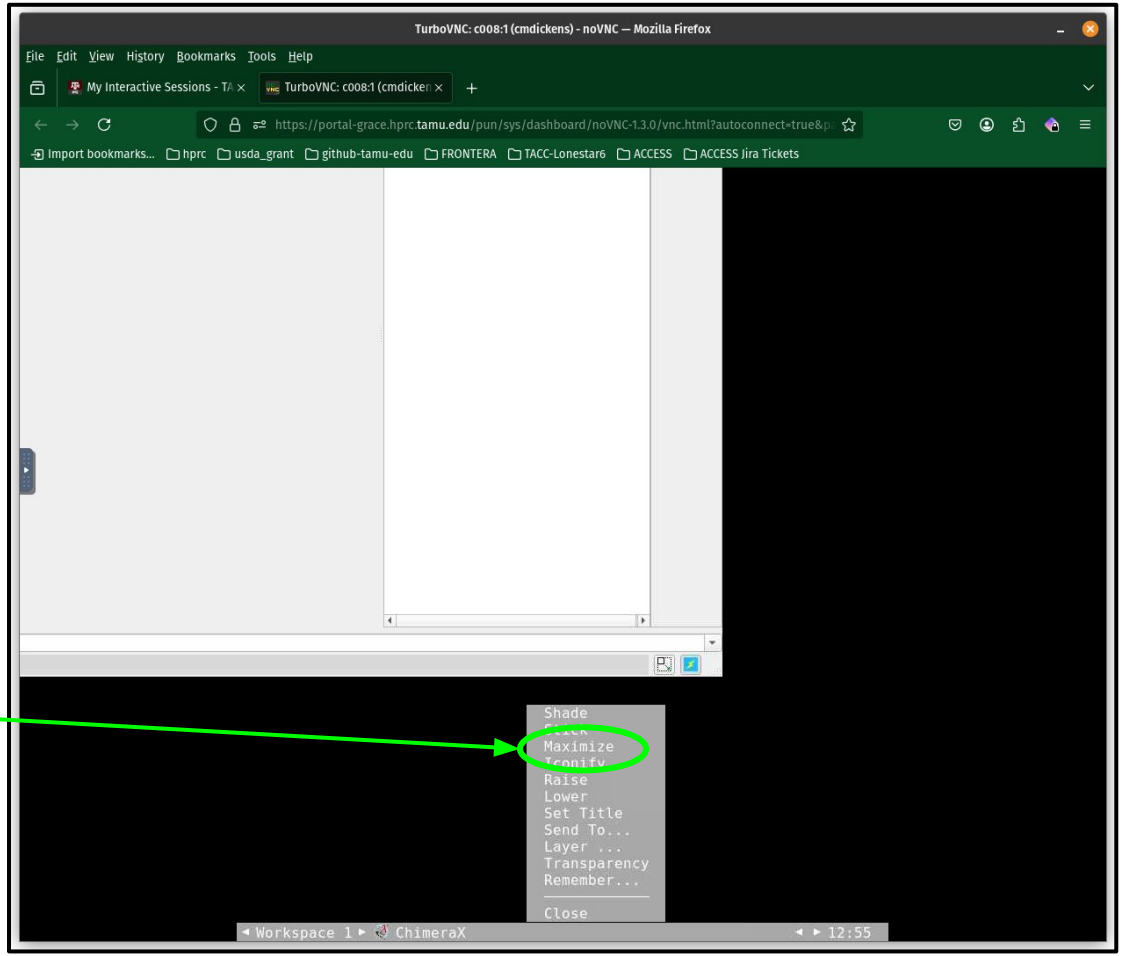
```
[userid@grace ~]$ cpuavail

    CONFIGURATION                          AVAILABILITY
  NODE          NODE                NODE        CPUs        GB MEM
  TYPE          COUNT               NAME        AVAIL       AVAIL
  ---------------------             ----------------------------
  CPU-only       808                c003          2          20
  GPU            132                c028         12          84
                                    c044         48         360
                                    c045         32         352
                                    c047         36         352
                                    c048         48         360
                                    c050          2         197
                                    c052         48         360
                                    c053         48         360
                                    c056         43         252
```

https://hprc.tamu.edu/kb/Software/useful-tools/cpuavail

AlphaFold 3 Results Visualization

Maximize ChimeraX Window

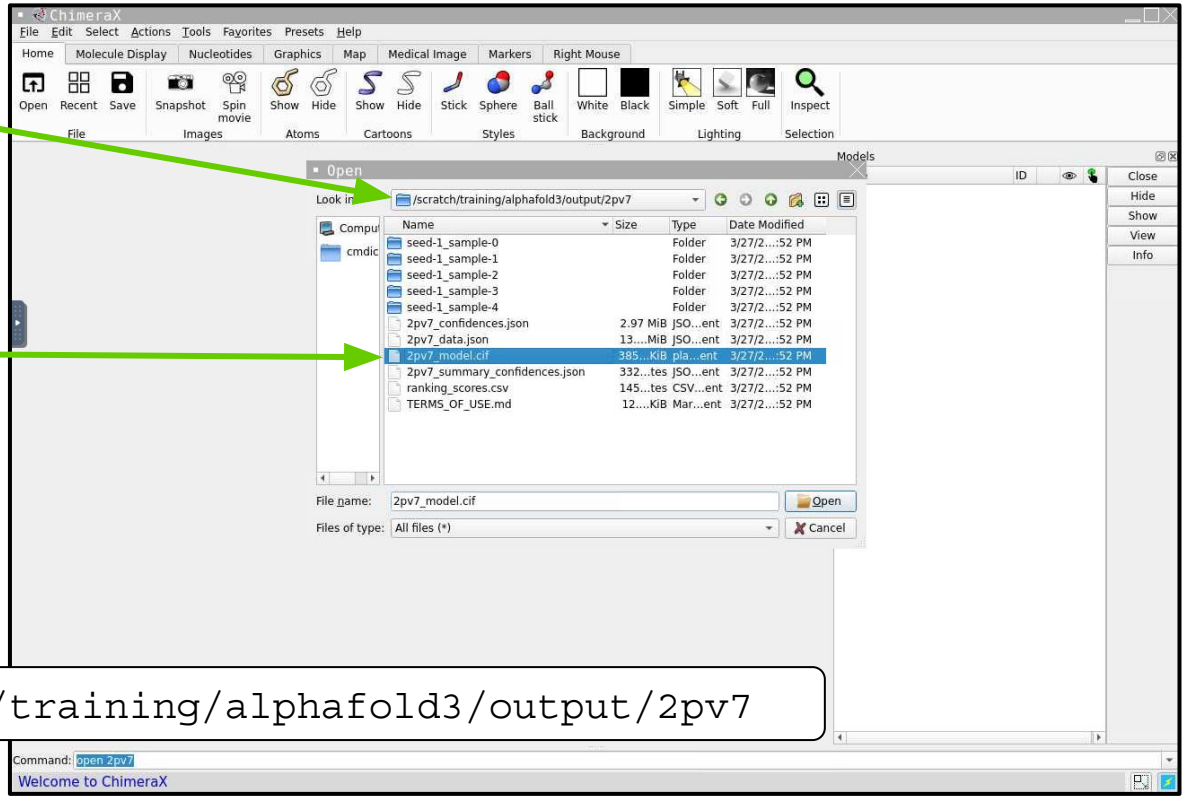
Right click the ChimeraX menu bar at the bottom and select Maximize if the ChimeraX window is off the screen.



Visualize AlphaFold3 Results with ChimeraX

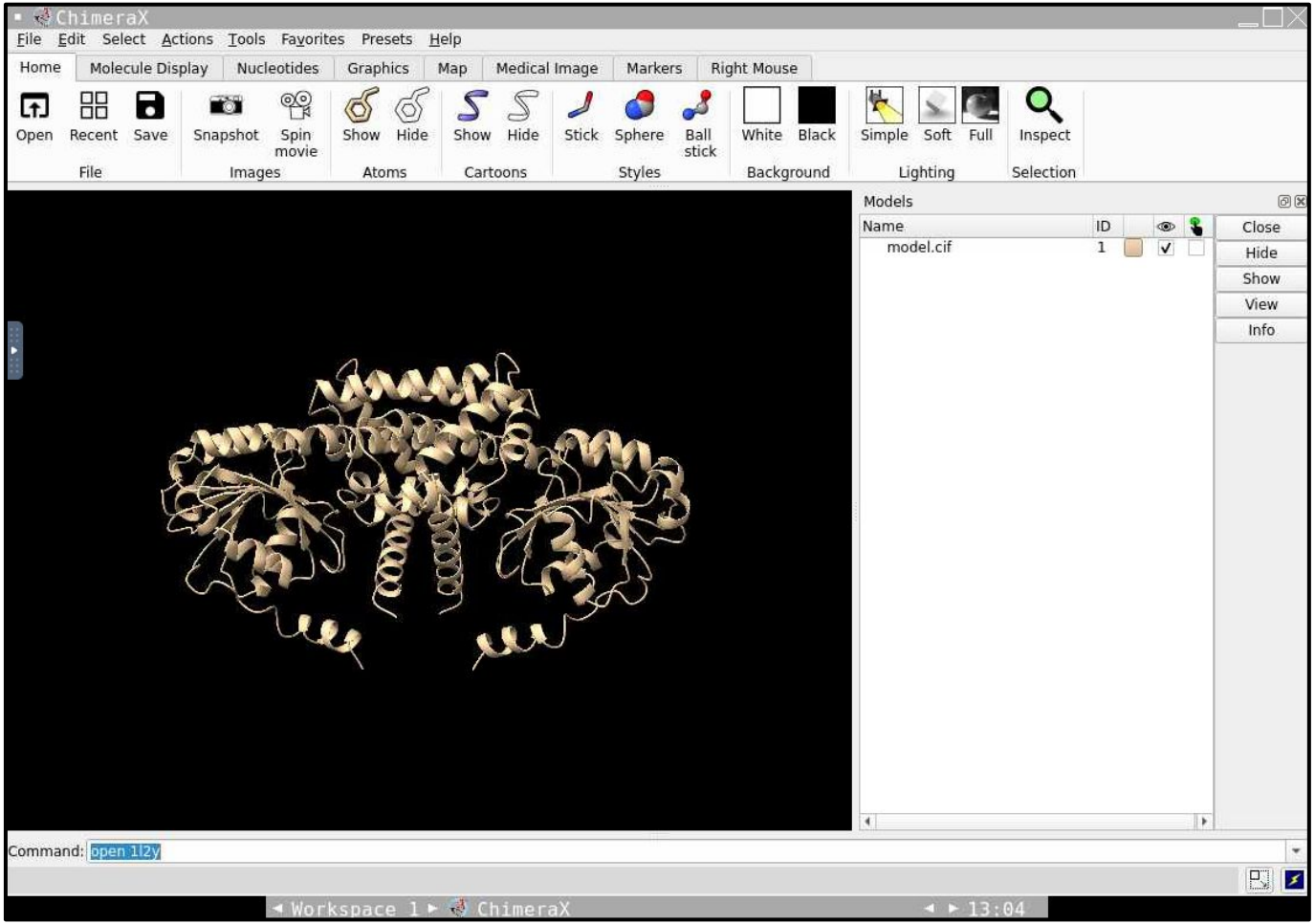
navigate to the example directory

select 2pv7_model.cif

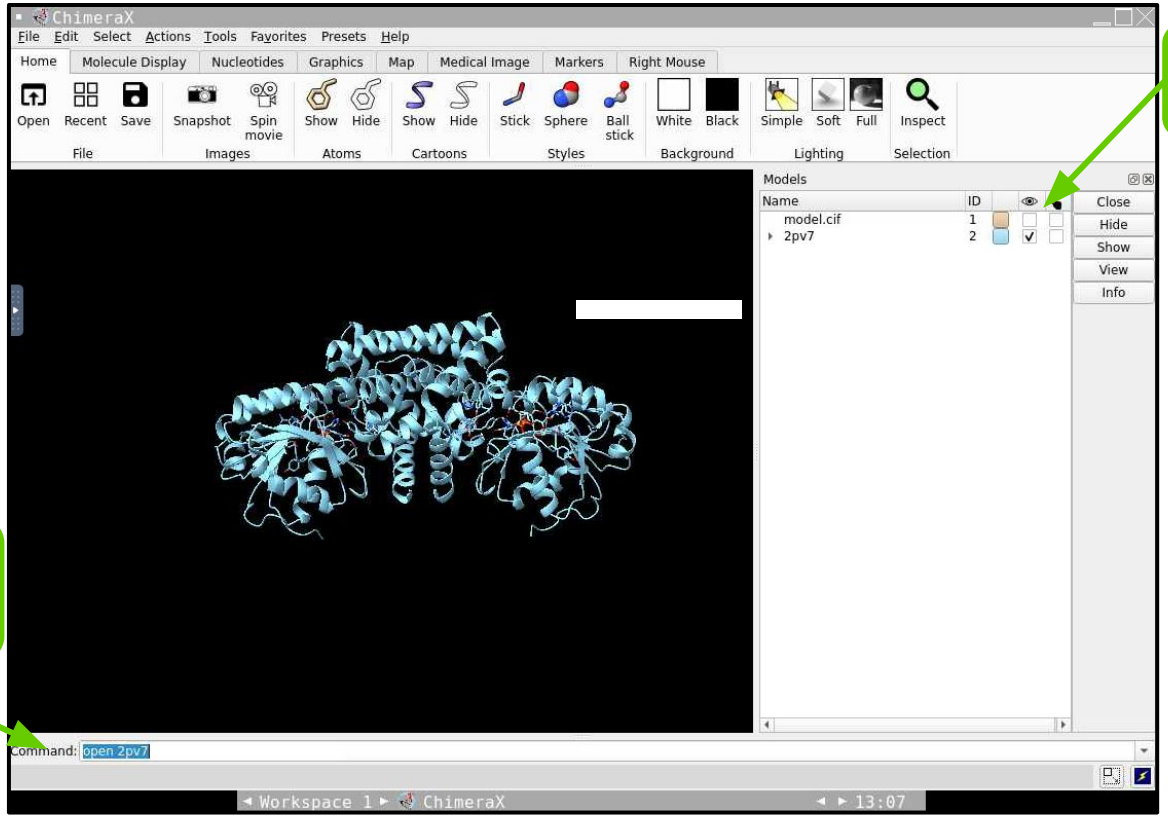


`/scratch/training/alphafold3/output/2pv7`

Visualize AlphaFold3 Results with ChimeraX



View PDB Structure if Available



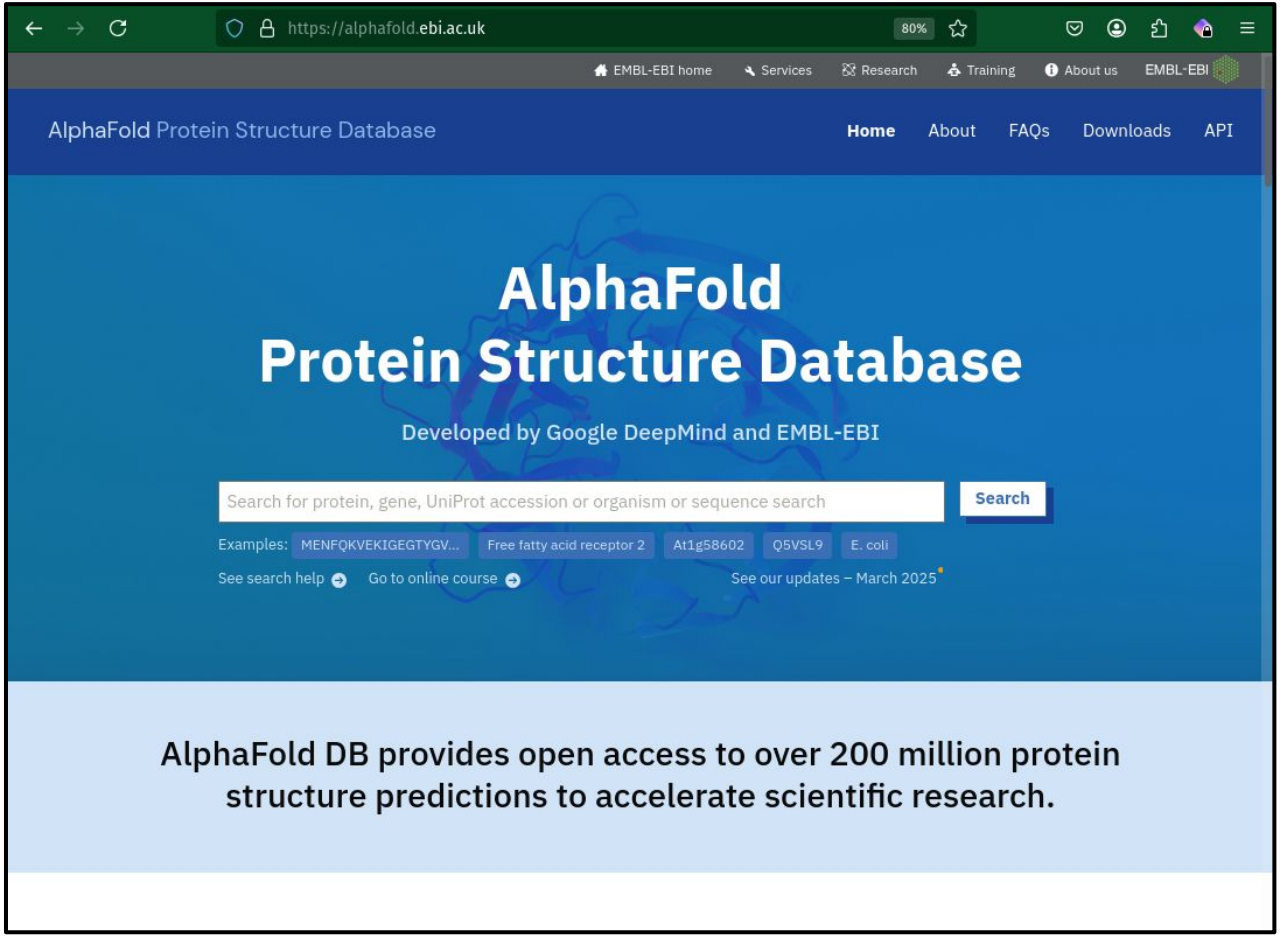
toggle visualization of your prediction

type ***open*** and the protein name ***2pv7***

Structure Database and References

DeepMind and EMBL's European Bioinformatics Institute (EMBL-EBI) have partnered to create AlphaFold DB to make these predictions freely available to the scientific community.

Search for your protein to see if it the structure has already been predicted using AlphaFold.



References

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

Tunyasuvunakool, K., Adler, J., Wu, Z. et al. Highly accurate protein structure prediction for the human proteome. Nature 596, 590–596 (2021). https://doi.org/10.1038/s41586-021-03828-1

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

Zhong, B, et al. (2021) ParaFold doi.org/10.48550/arXiv.2111.06340

Arnold, M. J. (2021) AlphaPickle doi.org/10.5281/zenodo.5708709

HIGH PERFORMANCE RESEARCH COMPUTING
TEXAS A&M UNIVERSITY

https://hprc.tamu.edu

HPRC Helpdesk:

help@hprc.tamu.edu
Phone: 979-845-0219

Take our short course survey!

https://u.tamu.edu/hprc_shortcourse_survey

Help us help you. Please include details in your request for support, such as, **Cluster** (ACES, FASTER, Grace, Launch), NetID (UserID), Job information (**JobID**(s), Location of your jobfile, input/output files, Application, Module(s) loaded, Error messages, etc), and Steps you have taken, so we can reproduce the problem.