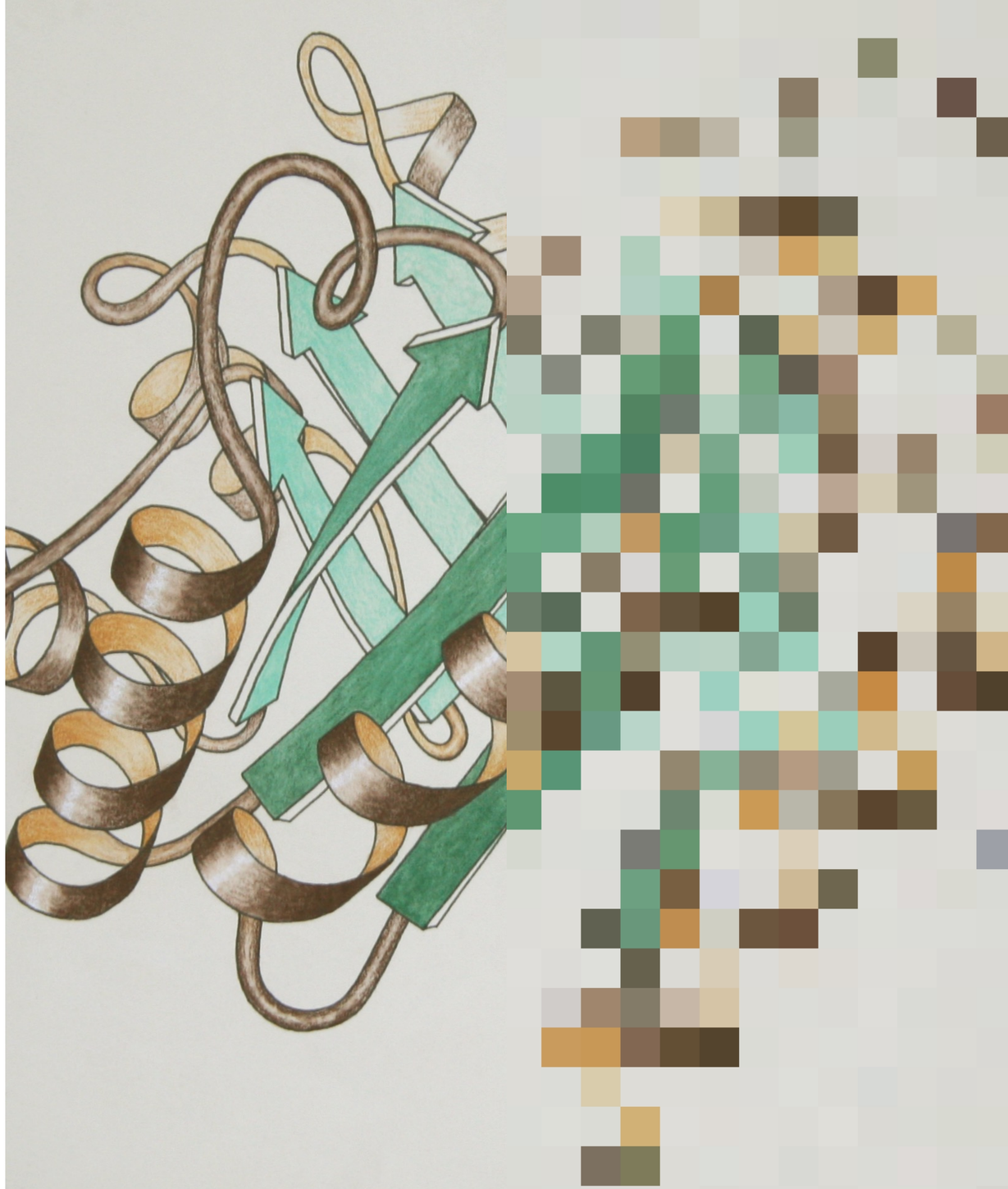
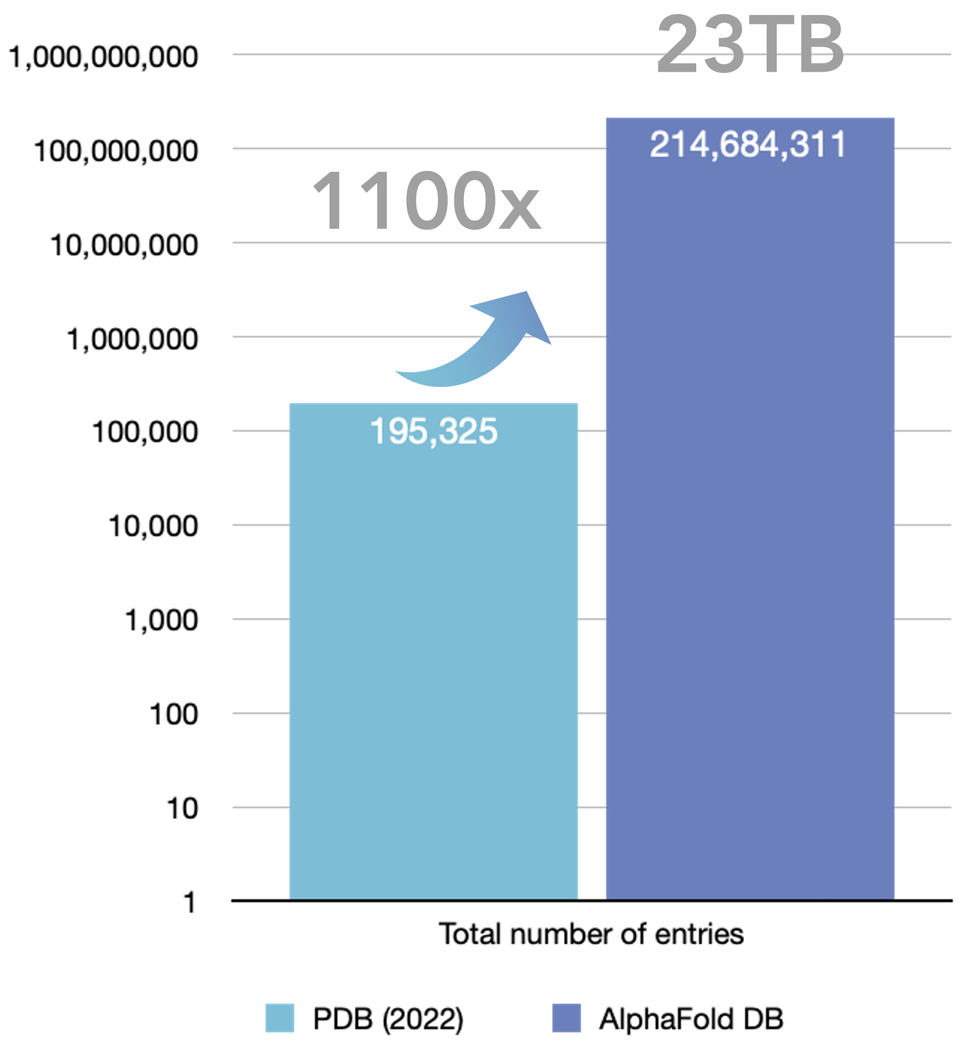
Foldcomp: scalable solution for compressing huge protein structure database

Hyunbin Kim
Seoul National University, Korea
SteineggerLab

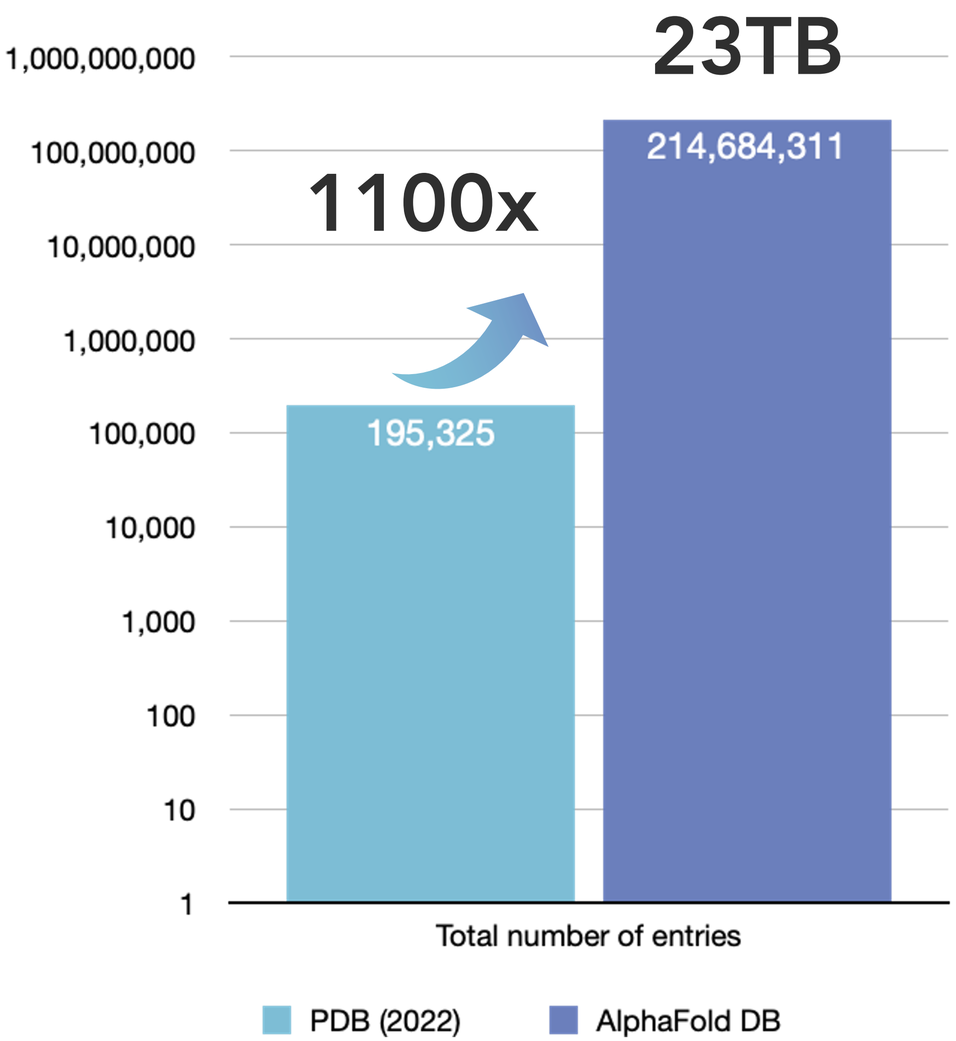


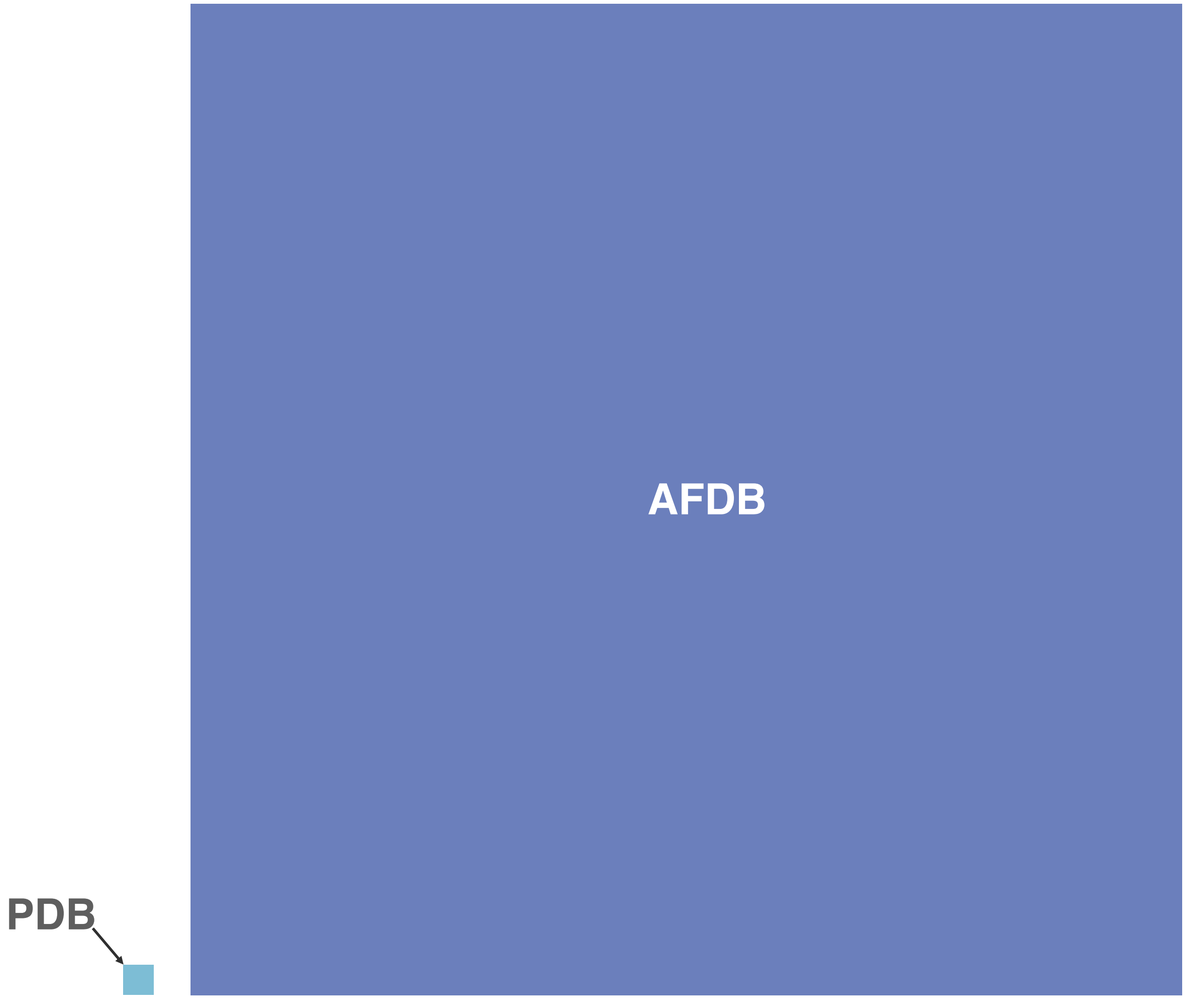
1TIM.pdb | hand-drawn by J. Richardson, 1981
modified from the original

With AlphaFold-DB, we now have 214M structures

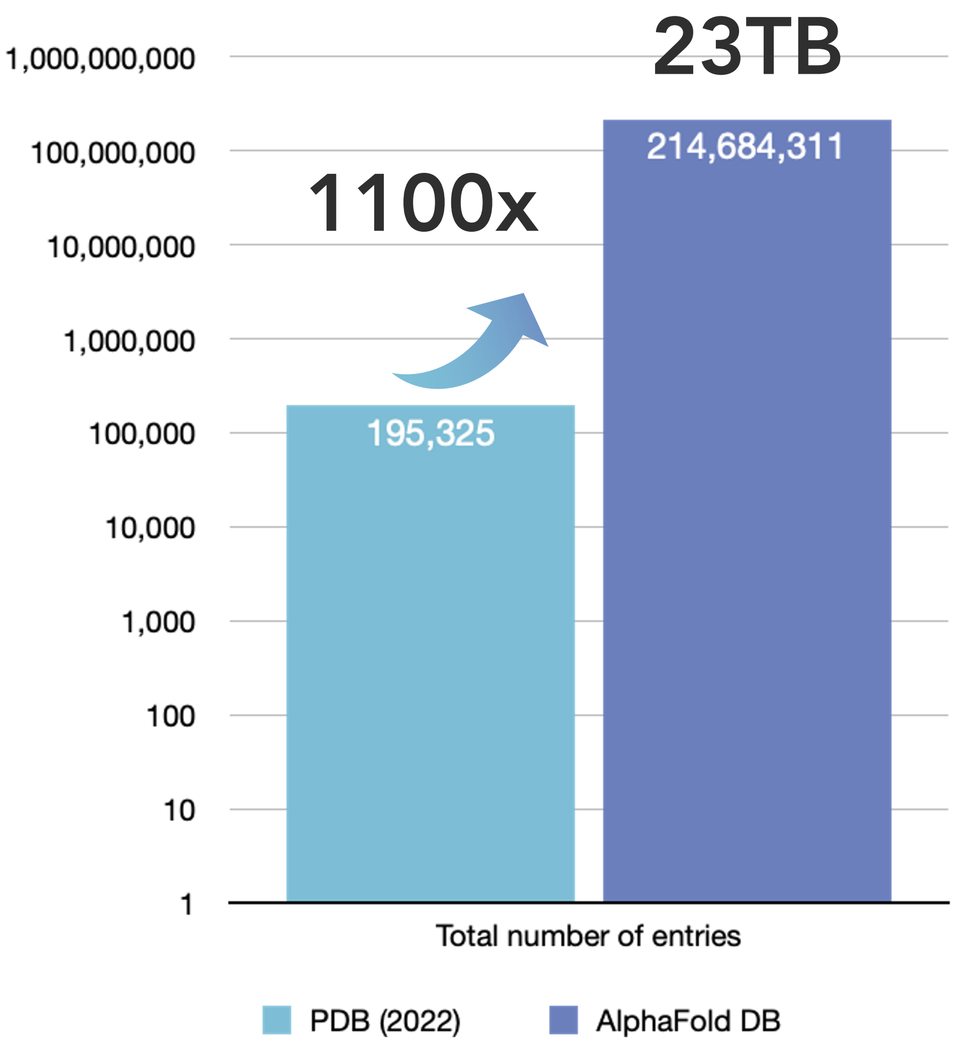


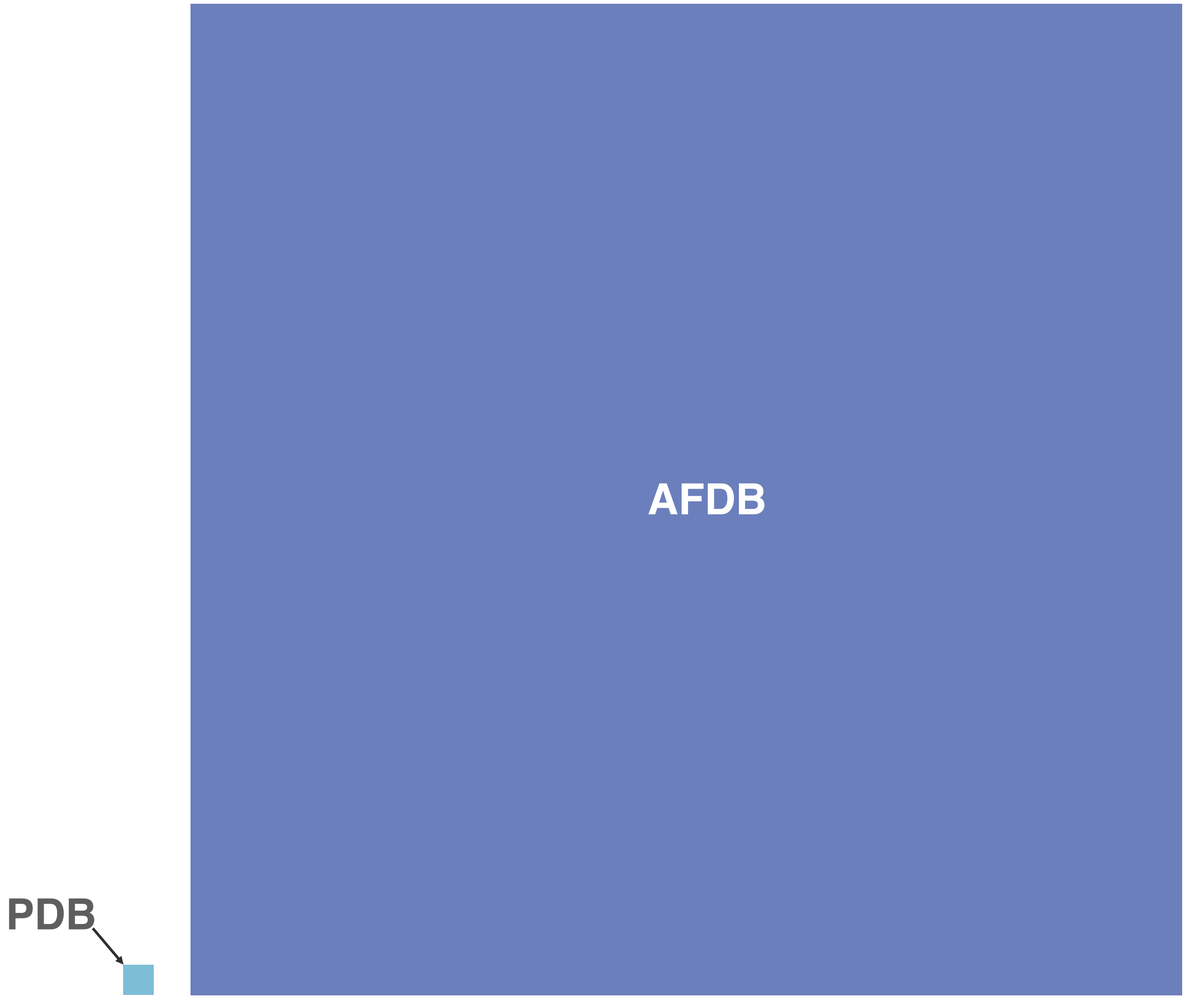
It actually looks like this





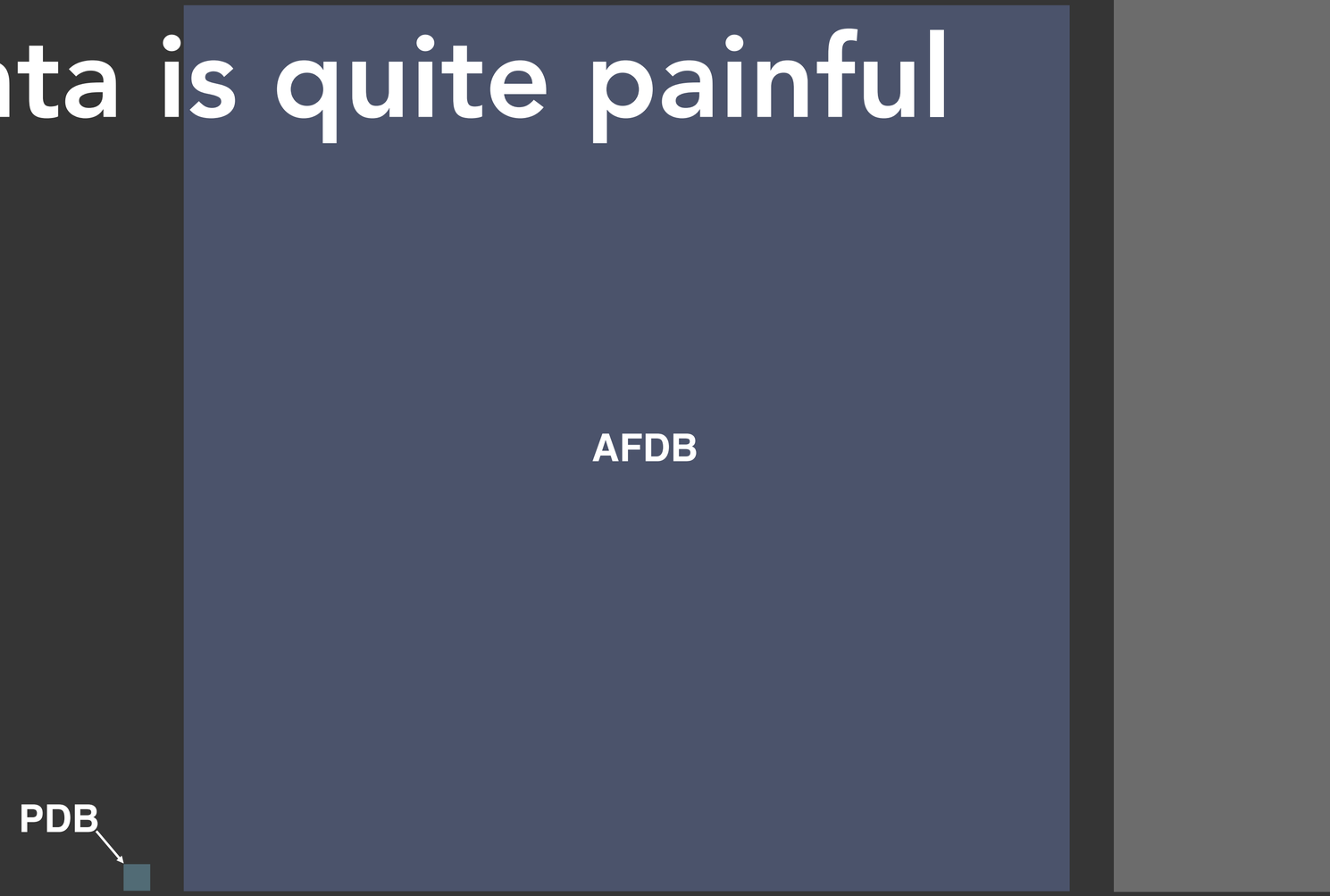
Not-too-distant future





>100TB

Dealing huge data is quite painful

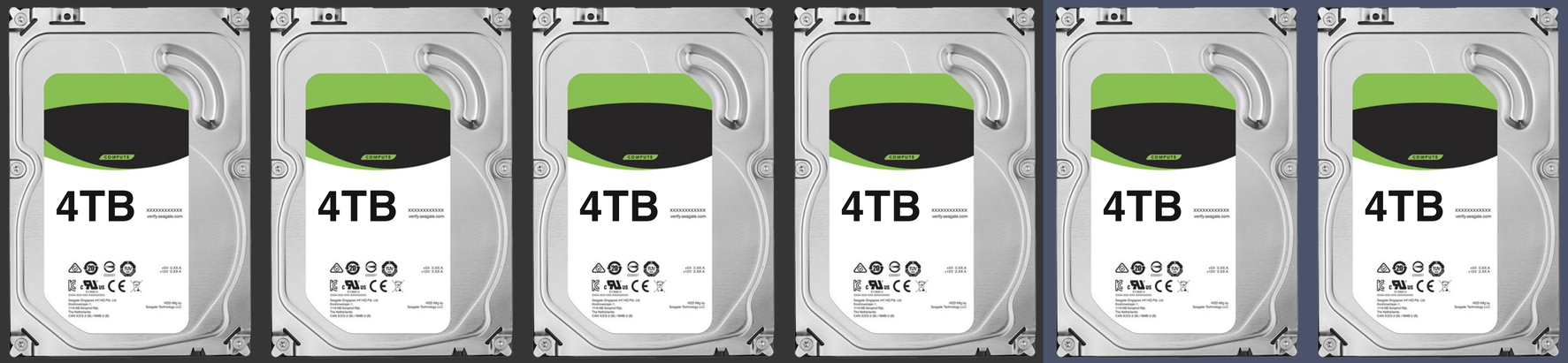


Dealing huge data is quite painful

It takes time,

23TB / 100Mbps = 21.3 days
/ 500Mbps = 4.26 days

Dealing huge data is quite painful
It takes time, storage,



https://www.pngfind.com/pngs/m/254-2540354_seagate-2tb-hard-disk-hd-png-download.png

Dealing huge data is quite painful
It takes time, storage, and money.

$80 * 6 = $480

AFDB

AWS S3

$0.023/GB *
23000GB = $529

https://aws.amazon.com/s3/pricing/

Initial idea on this problem

With the protein structure prediction problem solved, millions of high-resolution protein structures will become available.

The current PDB format needs to much memory to cope with the expected deluge of structural data.

We need a 10x compressed format with interfaces to common tools.

Our idea is to encode structures using the dihedral angles of the backbone and the side chains.
The atomic coordinates can then be regenerated from the dihedral angles using the NERF algorithm

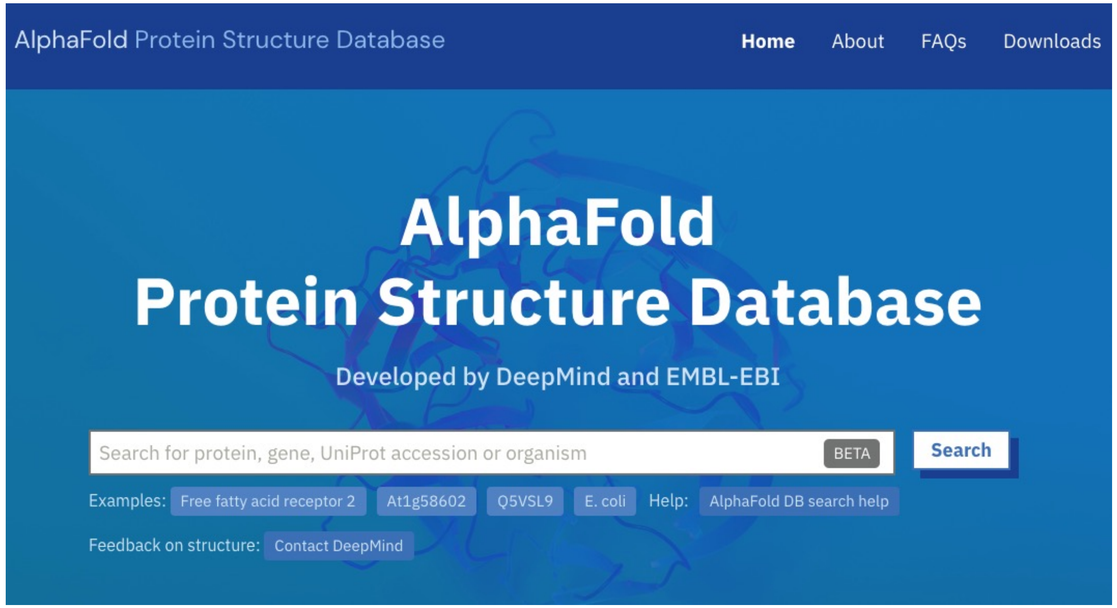
Huge structure databases

With the protein structure prediction problem solved, millions of high-resolution protein structures will become available.

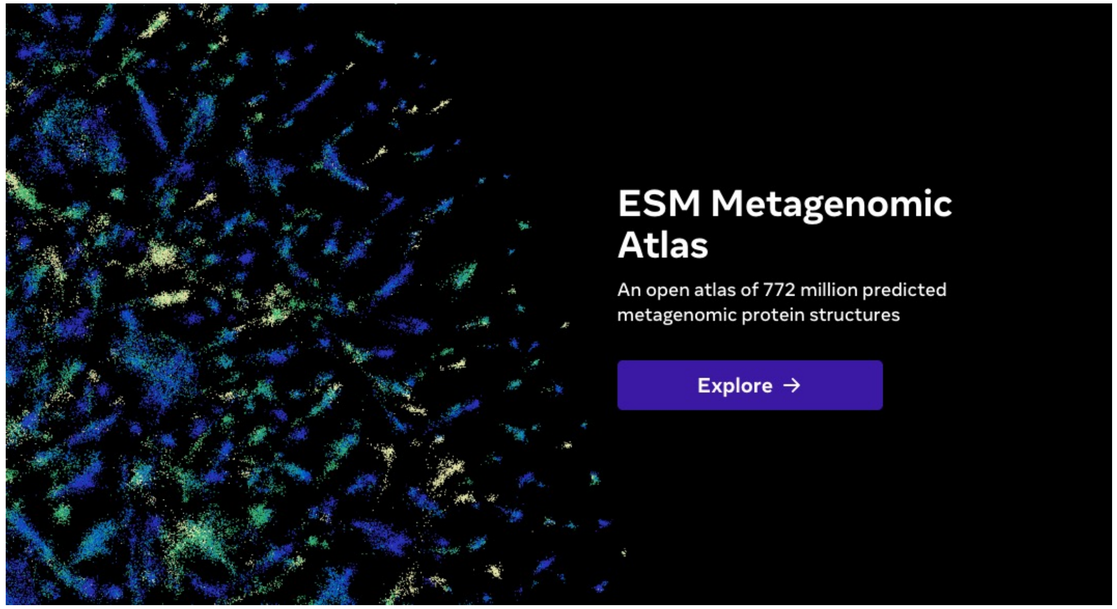
The current PDB format needs to much memory to cope with the expected deluge of structural data.

We need a 10x compressed format with interfaces to common tools.

Our idea is to encode structures using the dihedral angles of the backbone and the side chains.
The atomic coordinates can then be regenerated from the dihedral angles using the NERF algorithm



214M



>700M

Varadi, M et al. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. Nucleic Acids Research (2022).
Zeming Lin et al., Evolutionary-scale prediction of atomic-level protein structure with a language model.Science379, 1123-1130(2023).

Huge structure databases take huge spaces

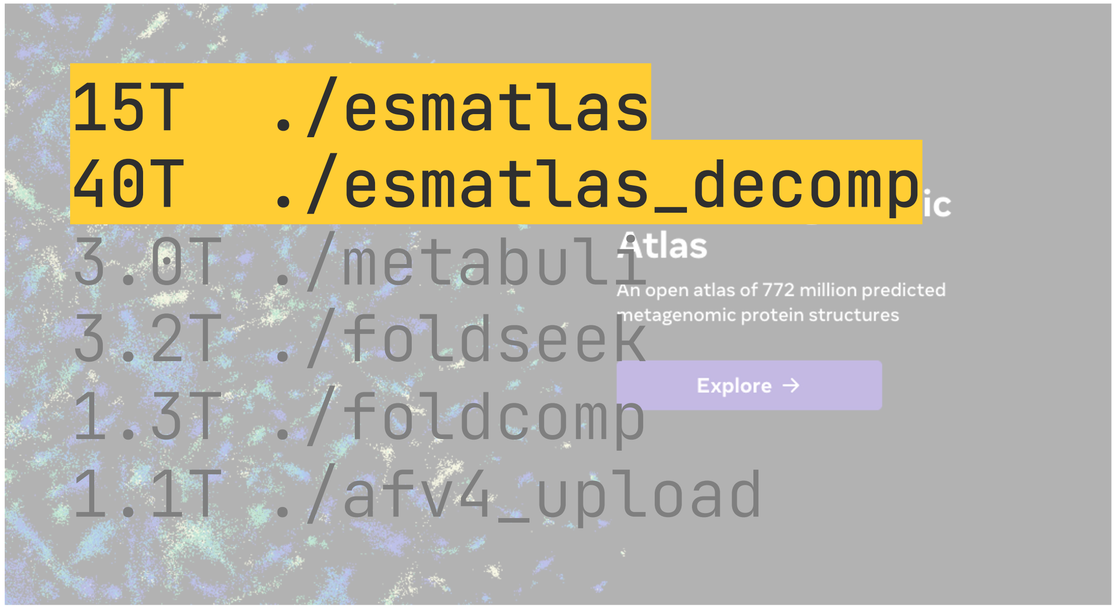
With the protein structure prediction problem solved, millions of high-resolution protein structures will become available.

The current PDB format needs to much memory to cope with the expected deluge of structural data.

We need a 10x compressed format with interfaces to common tools.

Our idea is to encode structures using the dihedral angles of the backbone and the side chains.
The atomic coordinates can then be regenerated from the dihedral angles using the NERF algorithm

214M



>700M

Varadi, M et al. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. Nucleic Acids Research (2022).
Zeming Lin et al., Evolutionary-scale prediction of atomic-level protein structure with a language model. Science (2023).

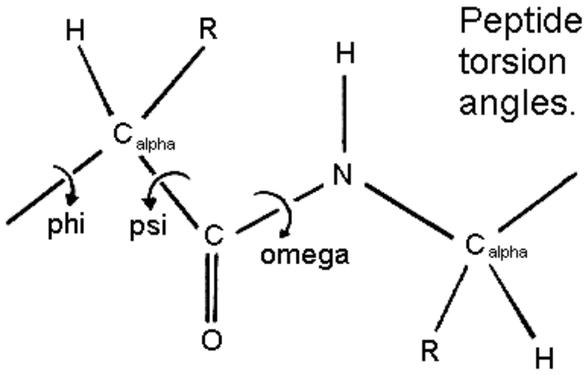
Initial idea embodied

With the protein structure prediction problem solved, millions of high-resolution protein structures will become available.

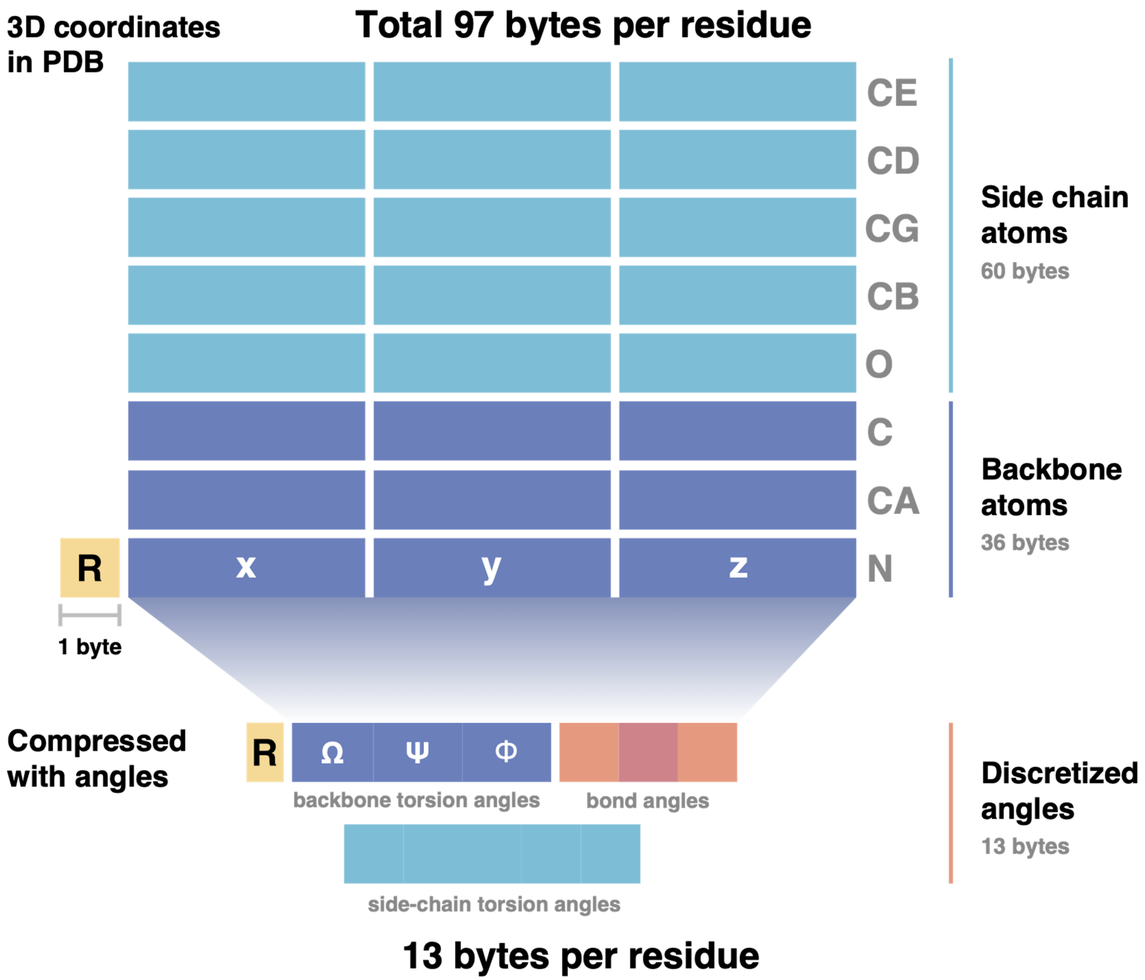
The current PDB format needs to much memory to cope with the expected deluge of structural data.

We need a 10x compressed format with interfaces to common tools.

Our idea is to encode structures using the dihedral angles of the backbone and the side chains.



Peptide torsion angles.



NeRF to convert from internal to cartesian coordinates

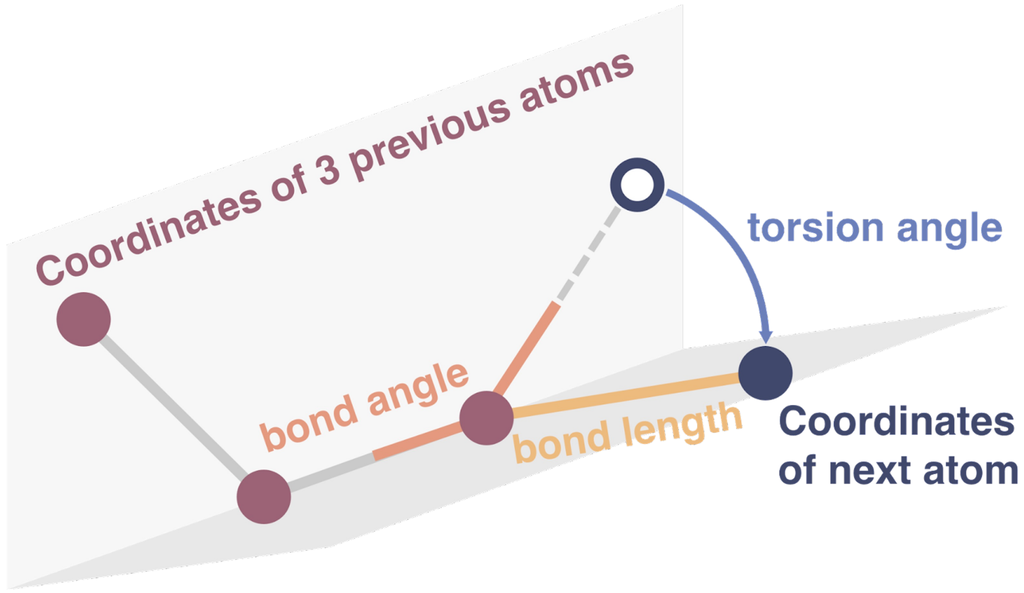
With the protein structure prediction problem solved, millions of high-resolution protein structures will become available.

The current PDB format needs to much memory to cope with the expected deluge of structural data.

We need a 10x compressed format with interfaces to common tools.

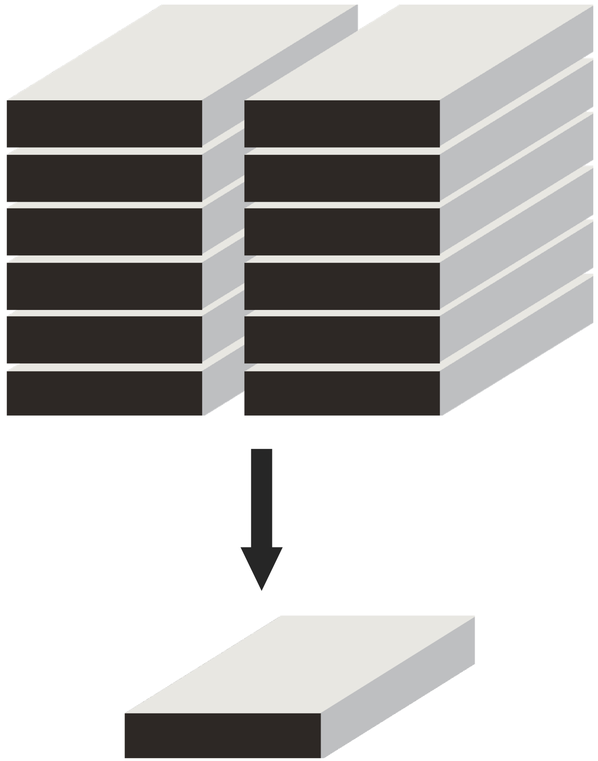
Our idea is to encode structures using the dihedral angles of the backbone and the side chains.
The atomic coordinates can then be regenerated from the dihedral angles using the NERF algorithm

Natural **E**xtension **R**eference **F**rame

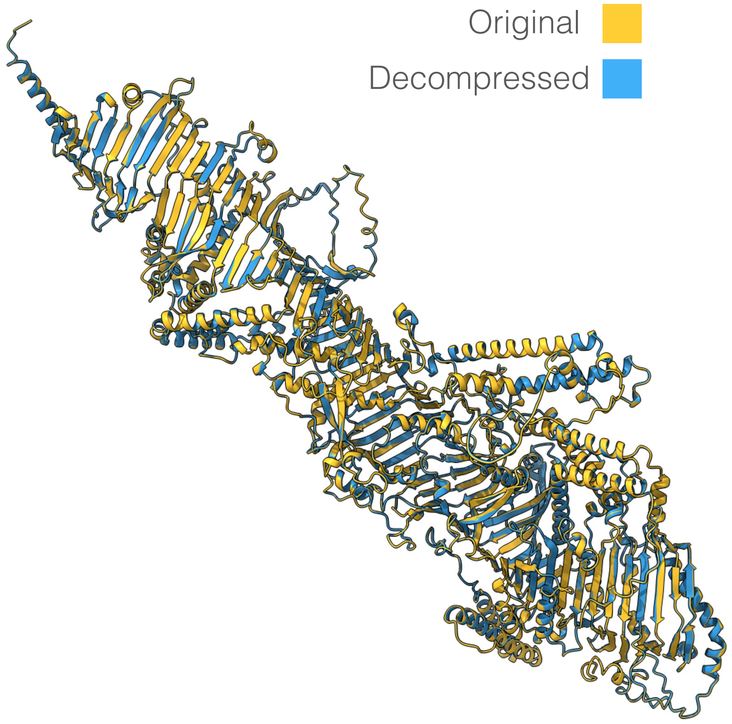


Goals to achieve

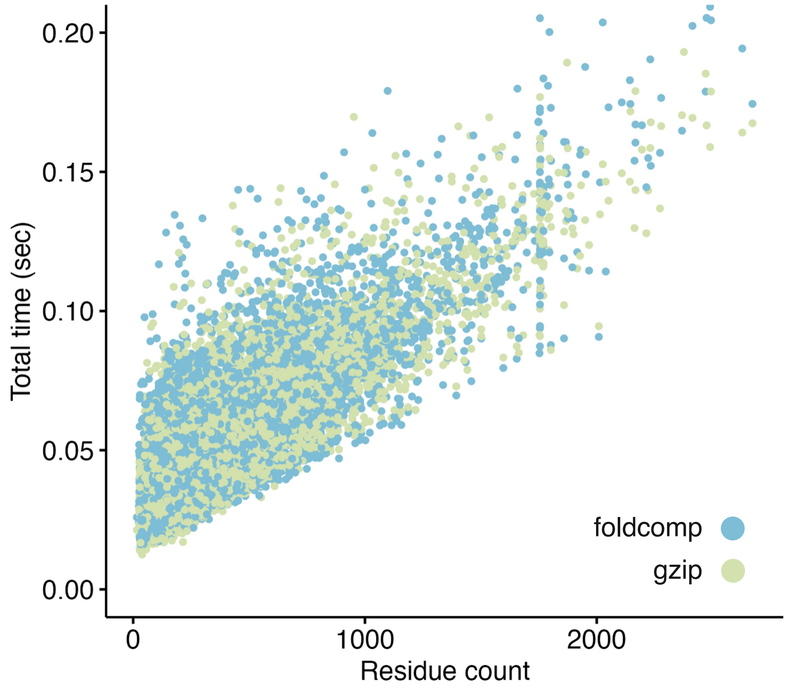
High compression ratio



Little loss (RMSD)

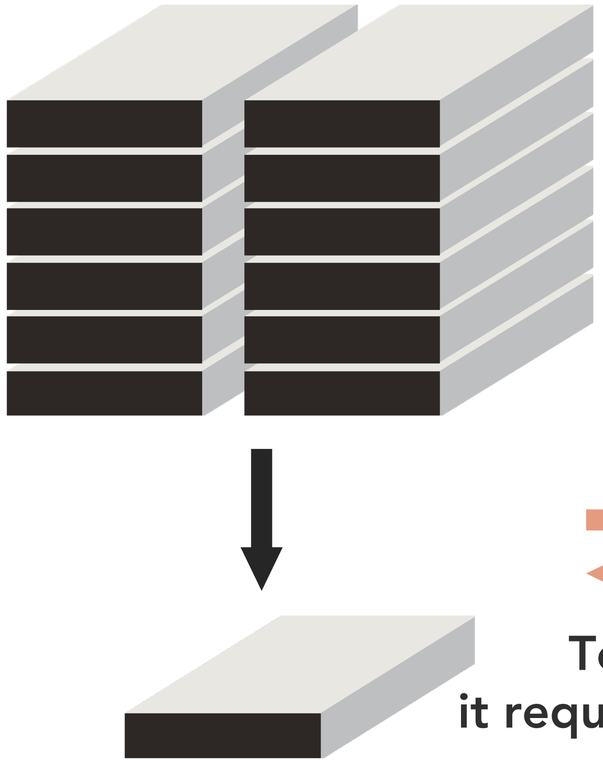


High speed

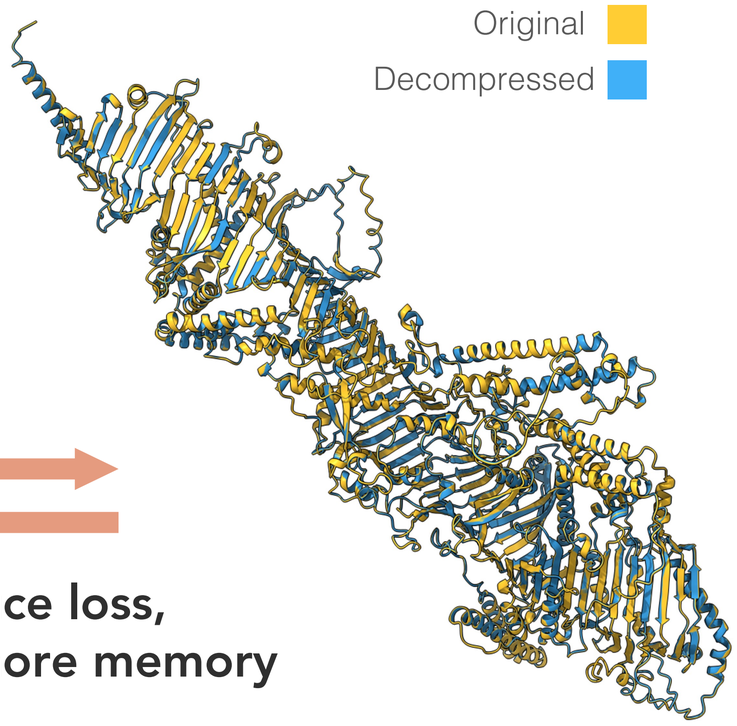


Goals to achieve

High compression ratio

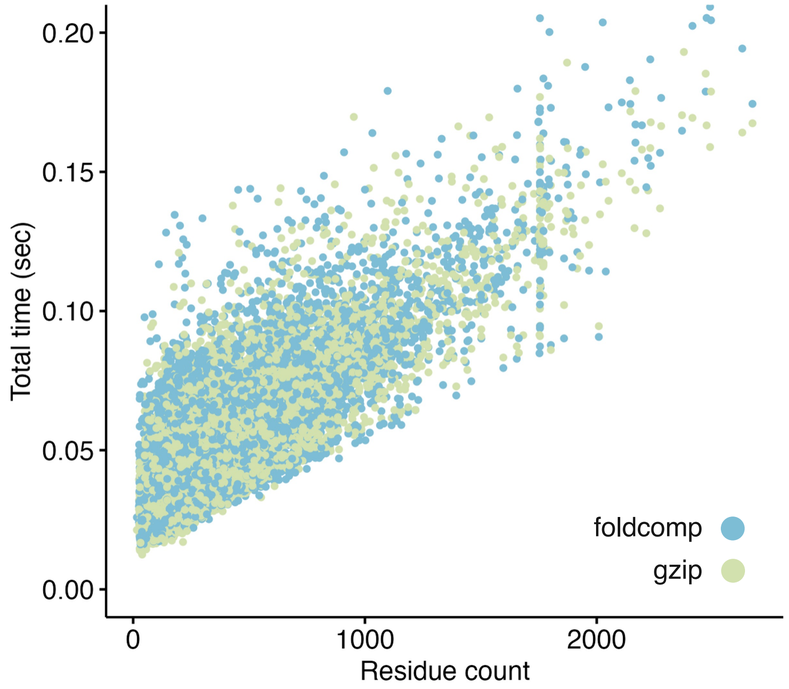


Little loss (RMSD)

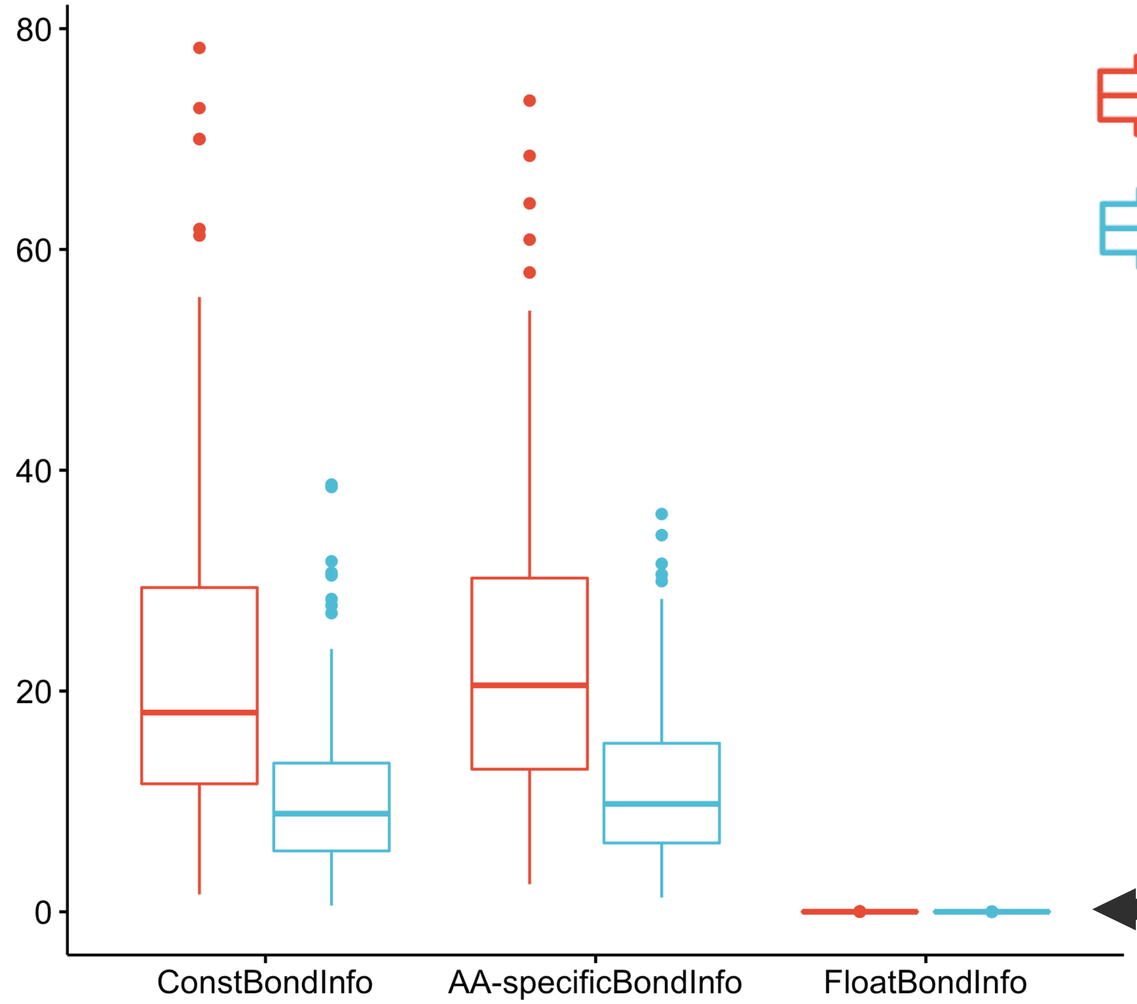


To reduce loss,
it requires more memory

High speed



Problem #1 for loss: Bond angles are not constants



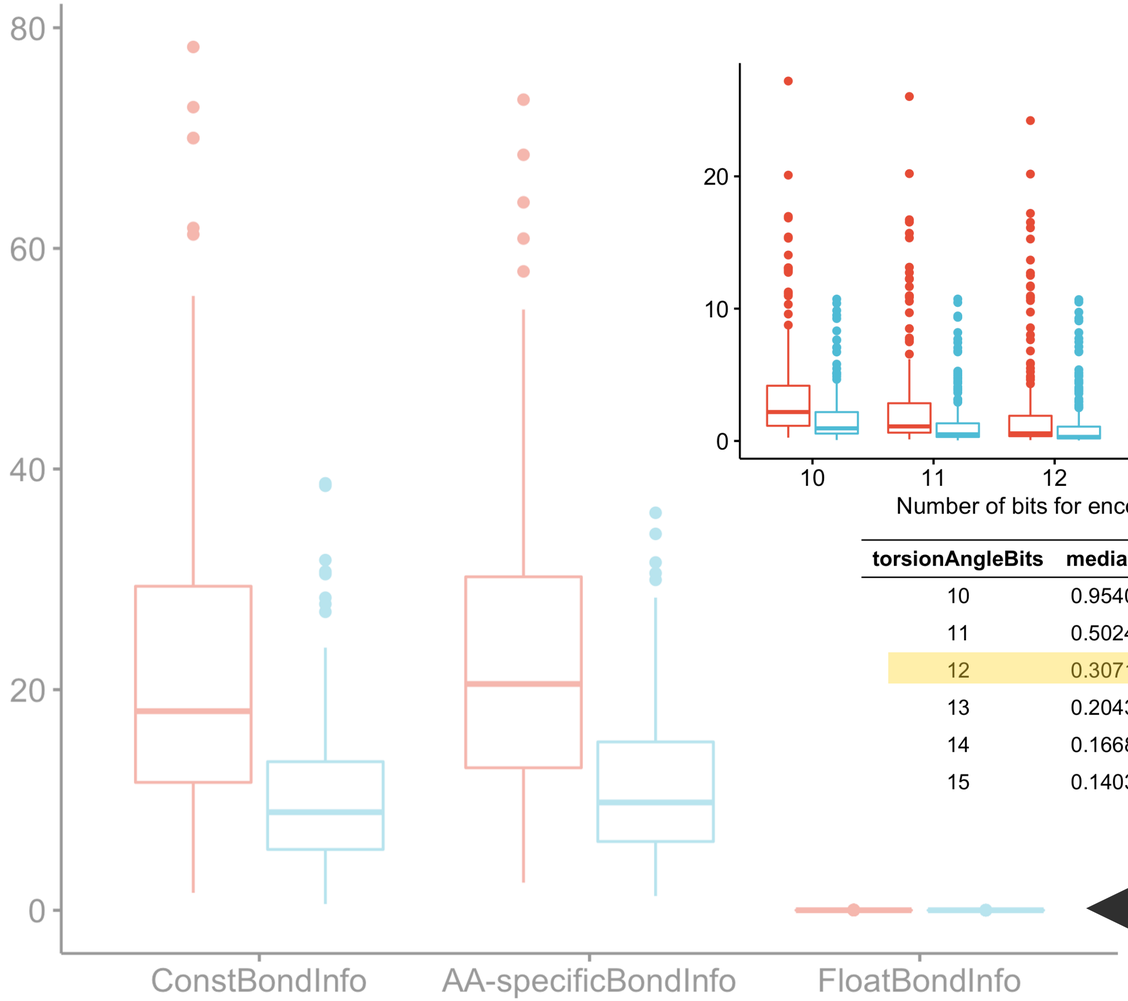
RMSD between original & decompressed structures

RMSD after superposition

The less, the better

Bond angle should be encoded
to reduce loss

Problem #1 for loss: Bond angles are not constants



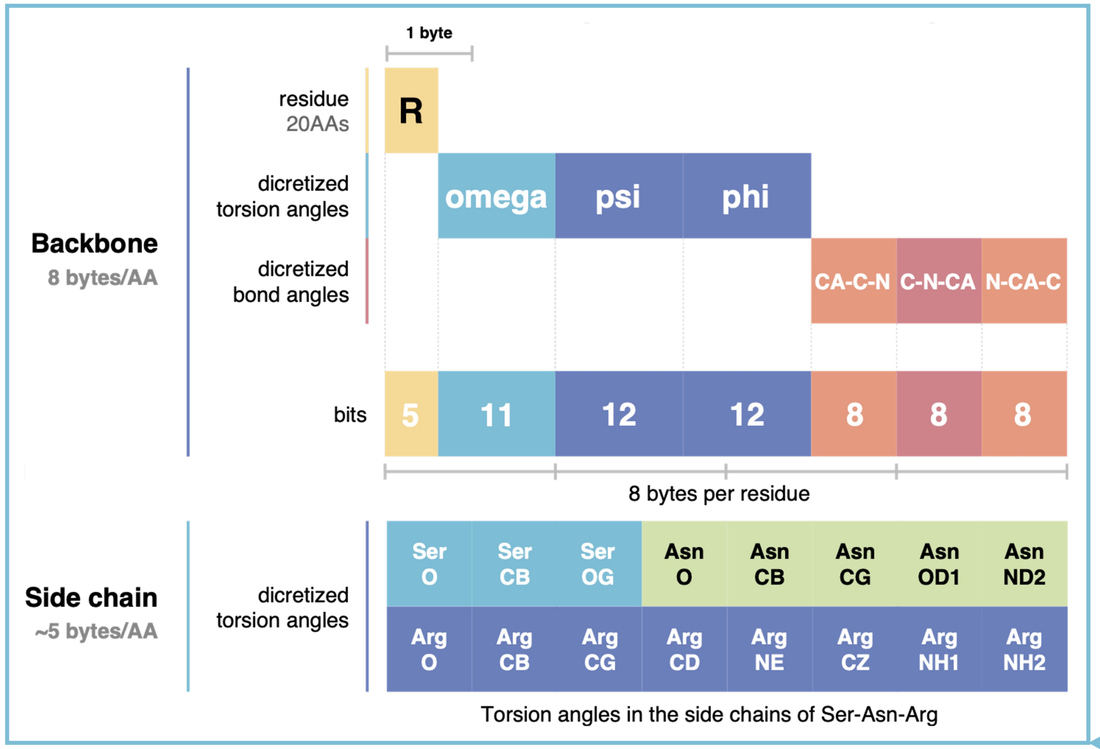
torsionAngleBits	median	mean	sd
10	0.9540	1.800	2.068
11	0.5024	1.394	2.078
12	0.3071	1.218	2.125
13	0.2043	1.143	2.154
14	0.1668	1.115	2.170
15	0.1403	1.105	2.176

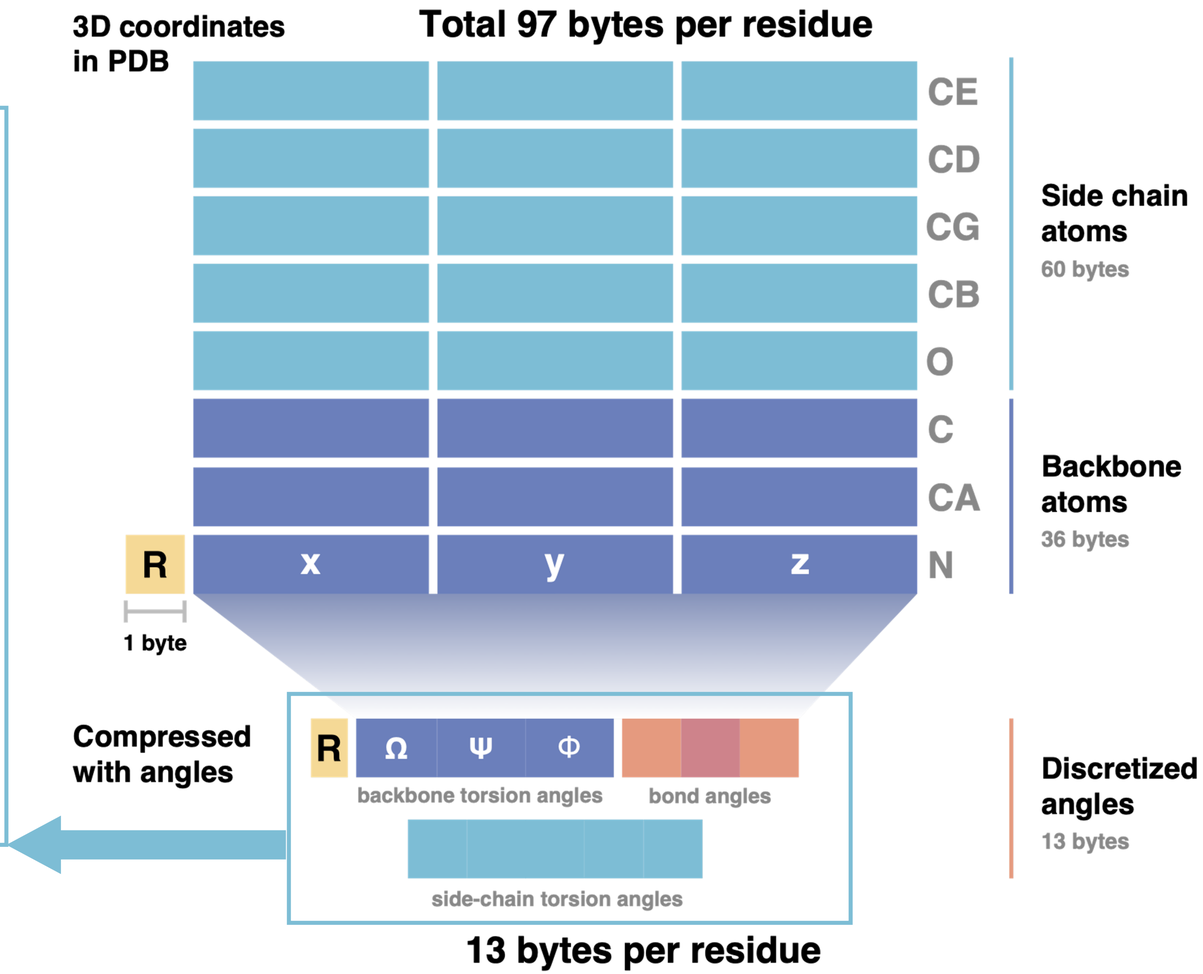
torsionAngleBits	bondAngleBits	totalBytes	median	mean	sd
10	11	8	0.9624	1.811	2.072
12	9	8	0.3319	1.258	2.117
14	5	8	1.3184	2.622	2.959
14	7	8	0.3839	1.362	2.171
16	4	8	2.5505	4.540	5.227
16	5	8	1.3100	2.598	2.944
16	6	9	0.6745	1.738	2.383
16	7	9	0.3717	1.341	2.173

Picked optimal bit combinations to encode bond angles and torsion angles with least RMSD

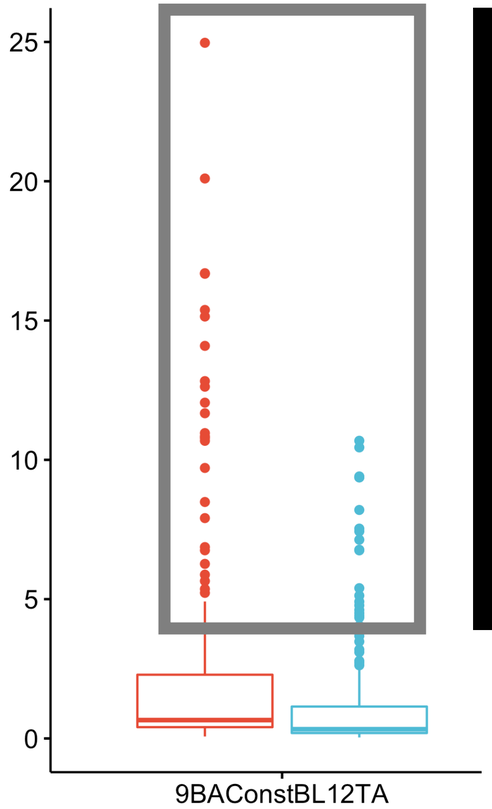
Bond angle should be encoded to reduce loss

Angle encoding – 8 bytes for backbone





Problem #2: Discontinuity in backbone

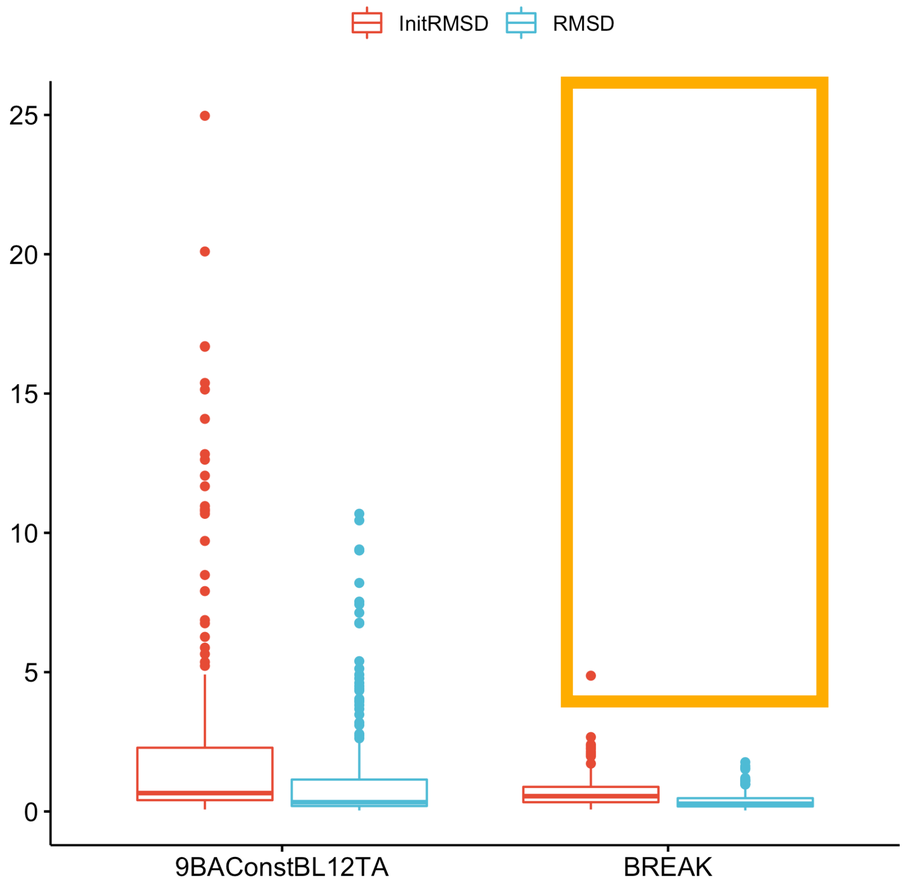


Visualization by Dongwook Kim

	Before
average	1.258
stdev	2.117
max	10.685
min	0.036

Reduced outliers by saving 3 previous atoms before unexpected breaks

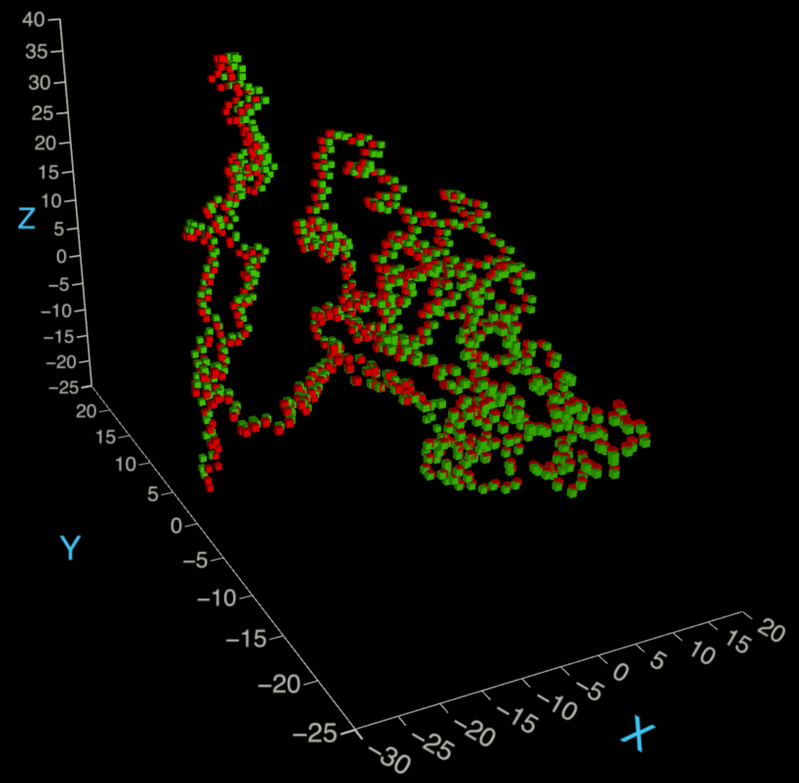
Solution for Problem #2

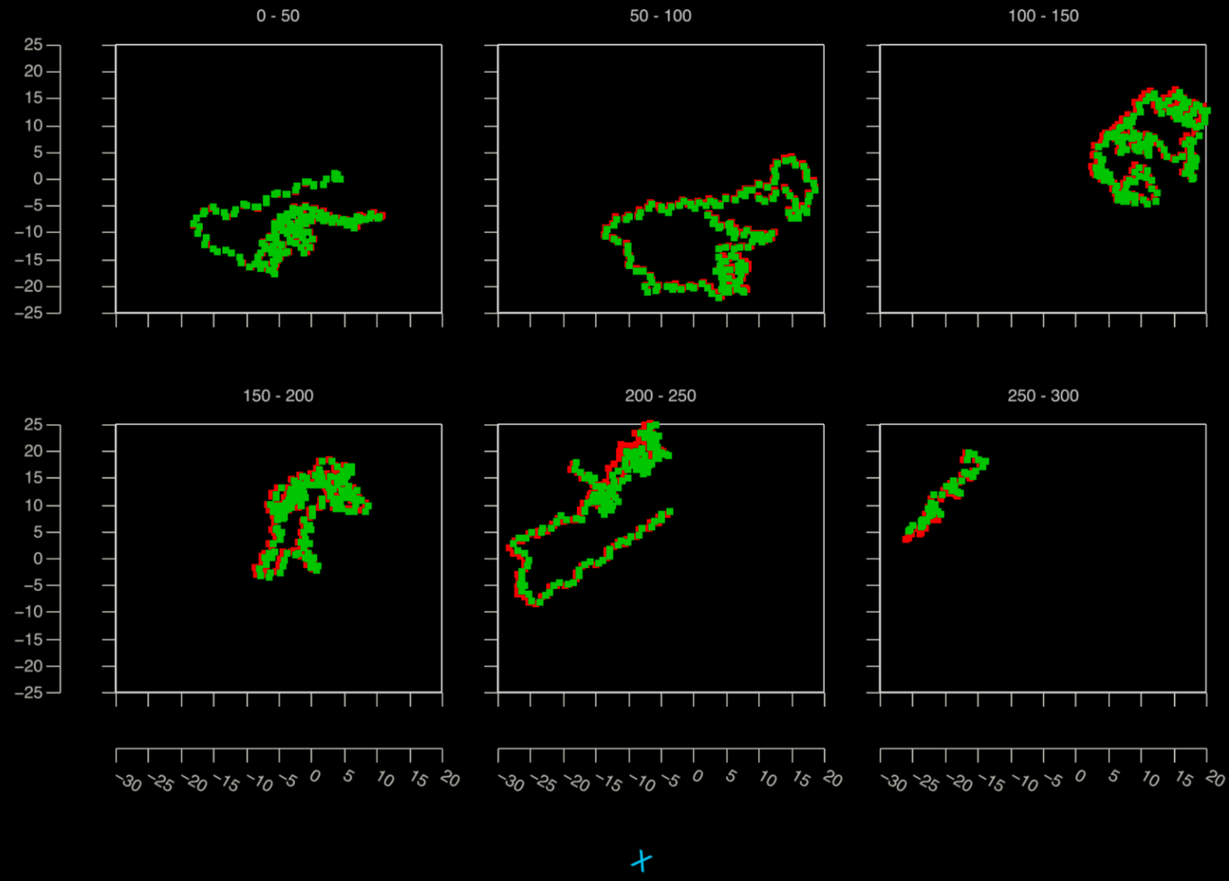


	Before	After
average	1.258	0.377
stdev	2.117	0.304
max	10.685	1.769
min	0.036	0.036

Reduced outliers by saving 3 previous atoms before unexpected breaks

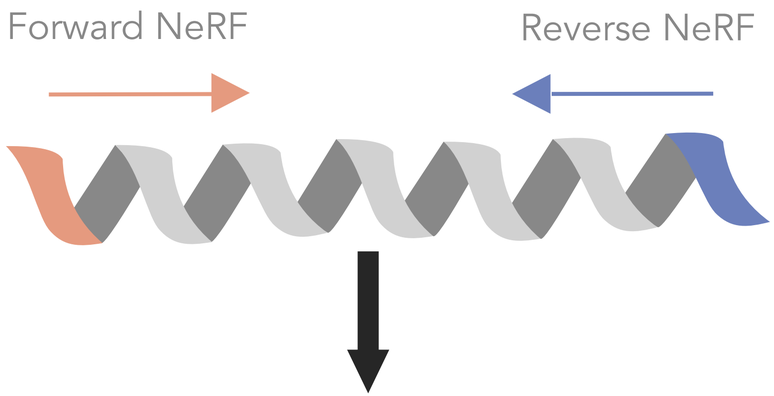
Problem #3: Loss accumulated as peptides get longer



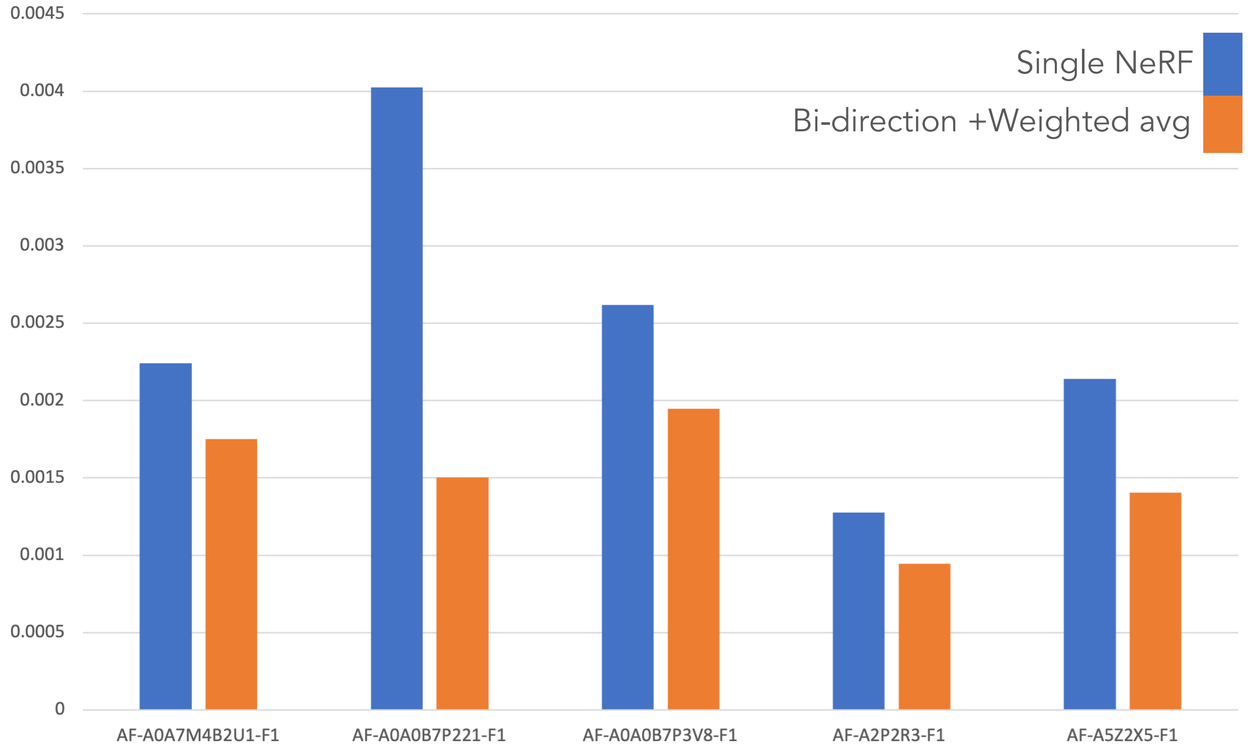


Higher deviation in latter position

Solution for Problem #3



Weighted average of coordinates
from bi-directional NeRF



RMSD per residue between original & decompressed structures

Solution for Problem #3

Internal anchor points to reset error accumulation

Save every 200 AA

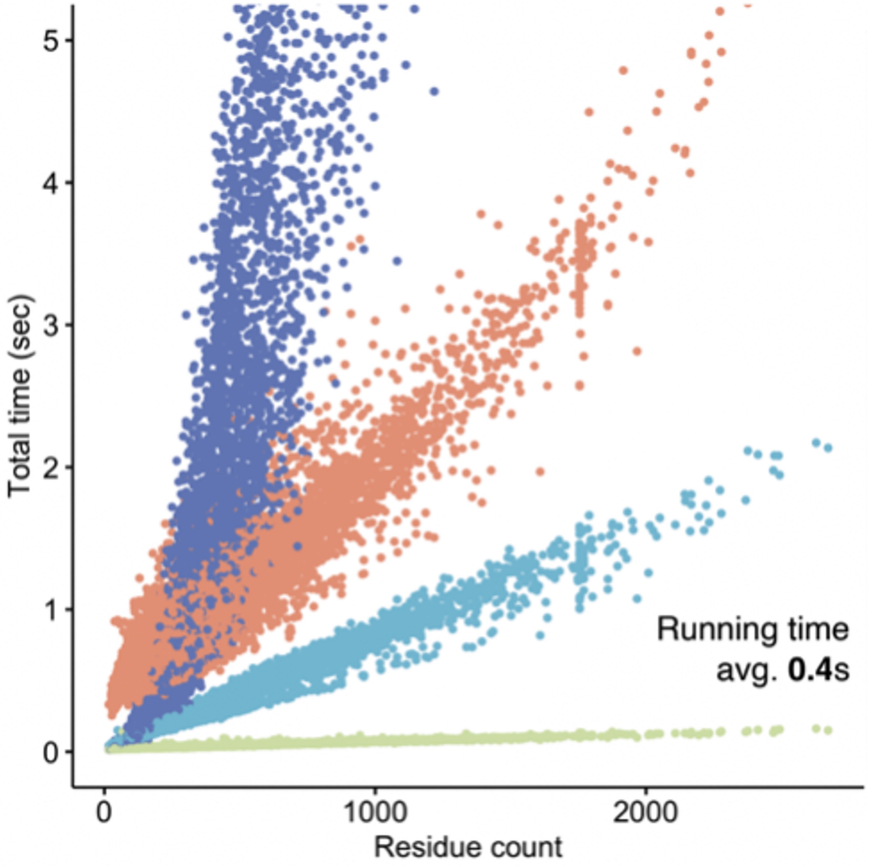
C-alpha RMSD: 0.995
Backbone RMSD: 0.995
All RMSD: 0.995

Save every 25 AA

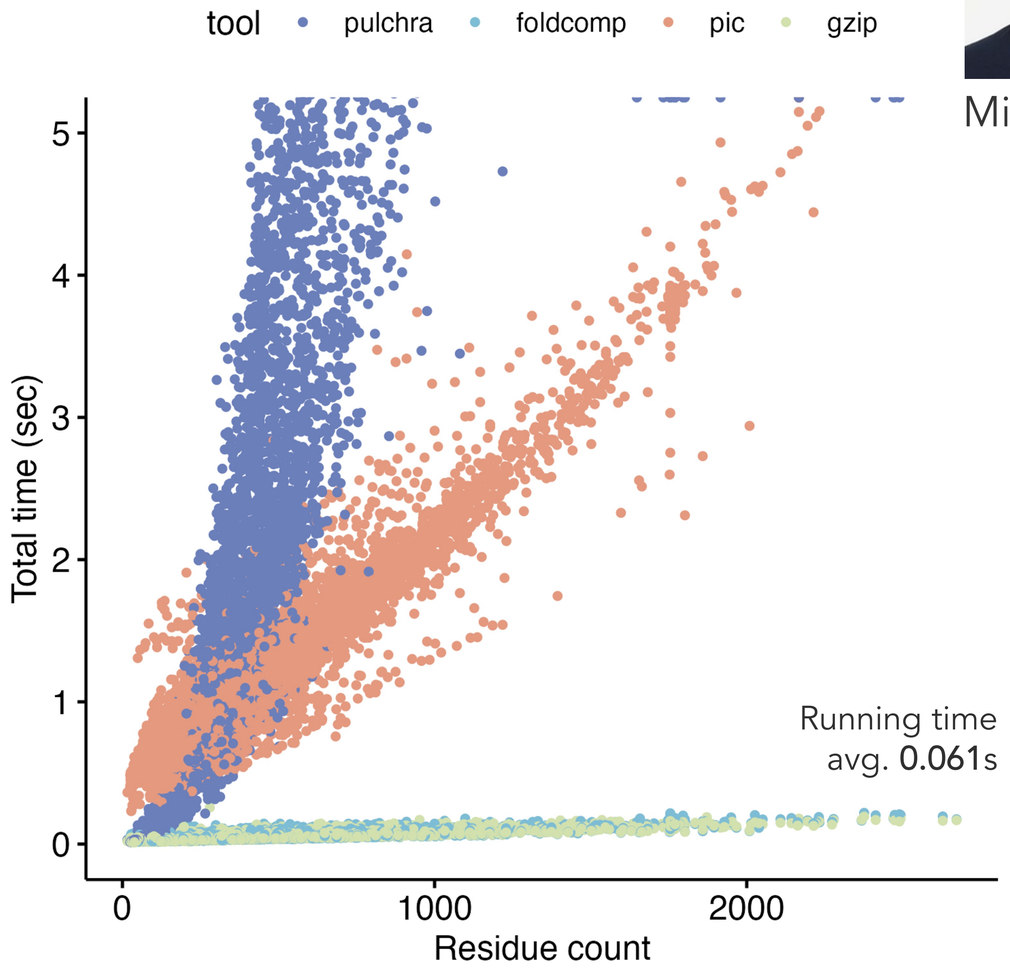
C-alpha RMSD: 0.134
Backbone RMSD: 0.134
All RMSD: 0.181
17511 bytes

Solution for Problem #5

Milot Mirdita



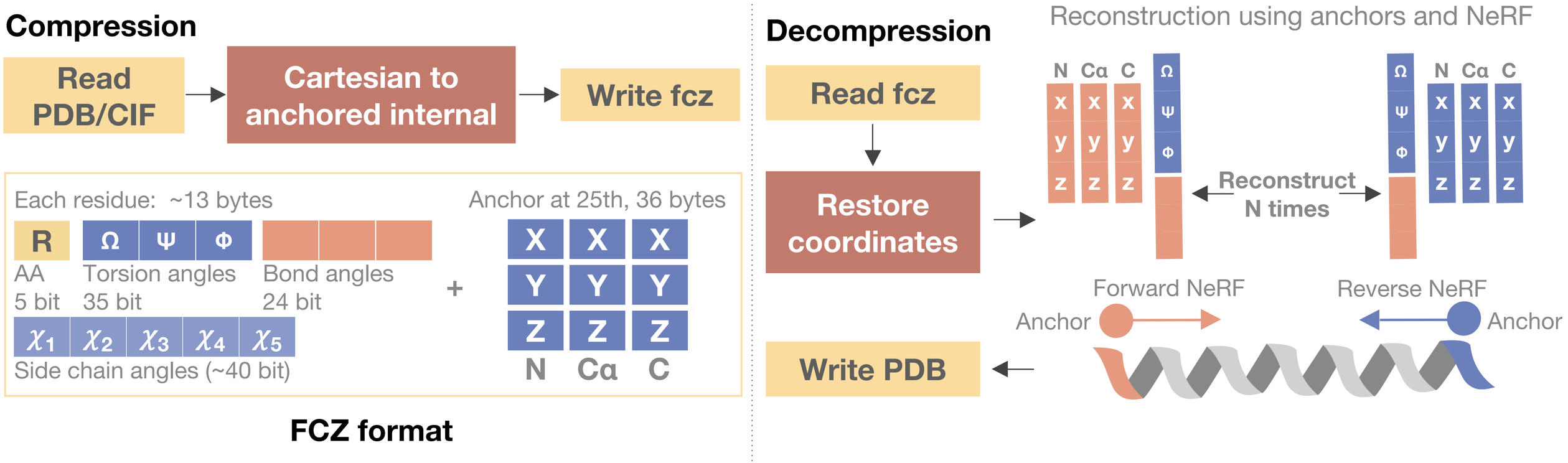
ISMB2022



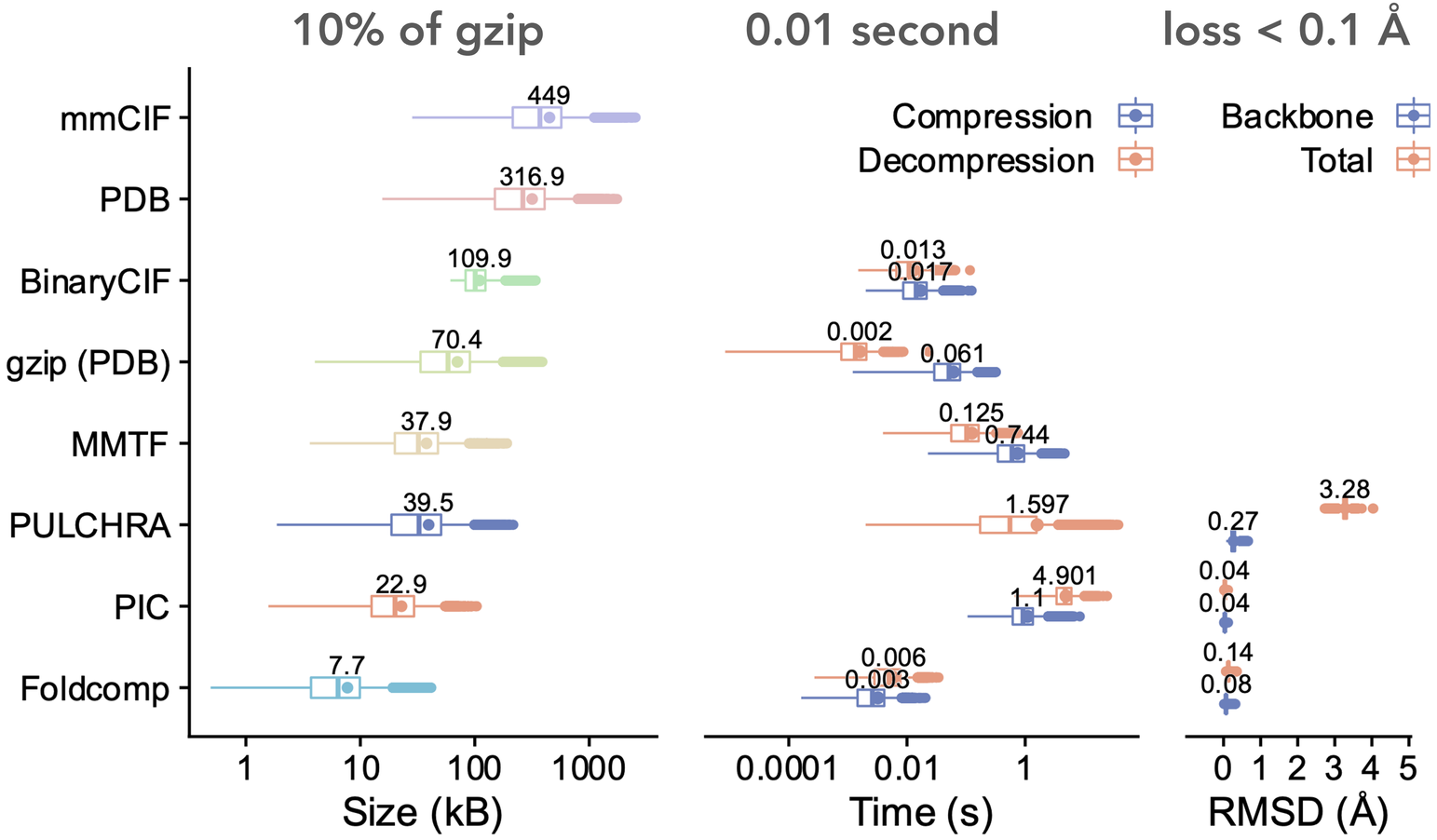
ISMB2023

- Optimizing memory usage
- Finding out bottlenecks in the process

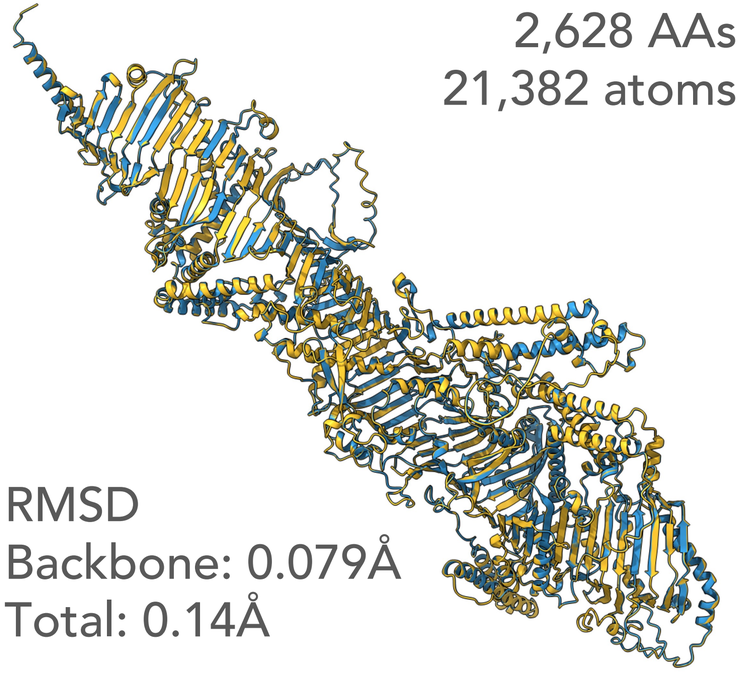
Foldcomp efficiently save internal coordinates and restore coordinates with bi-directional NeRF



Foldcomp has the best compression rate while being nearly as fast as gzip in decompression



AF-Q06179-F1 (FMP27-YEAST)

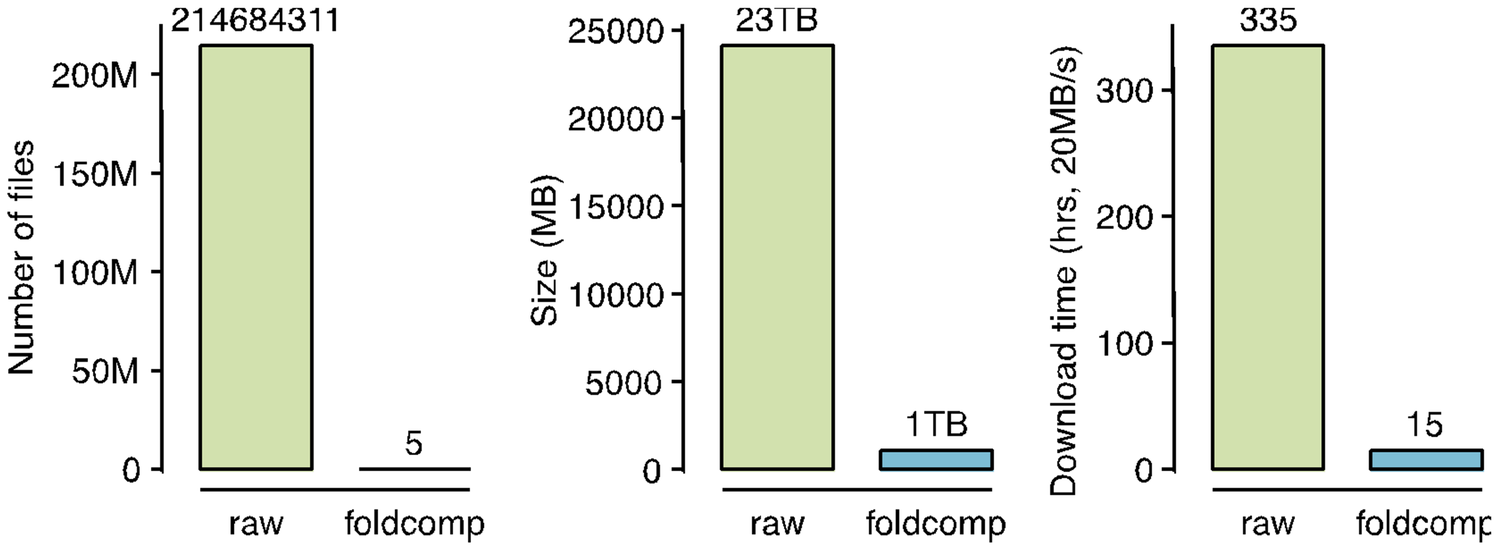


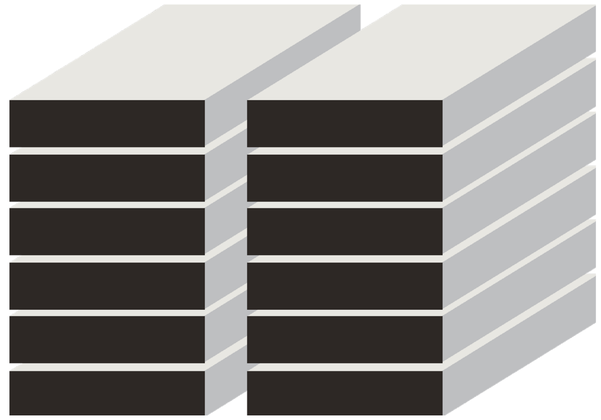
2,628 AAs
21,382 atoms

RMSD
Backbone: 0.079Å
Total: 0.14Å

Saving this structure requires
41.6 KB instead of 1.75 MB PDB

Foldcomp compressed AlphaFold-DB into 1 disk





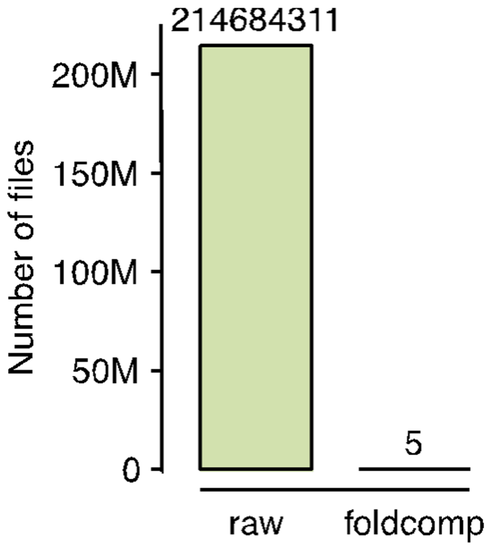
23TB

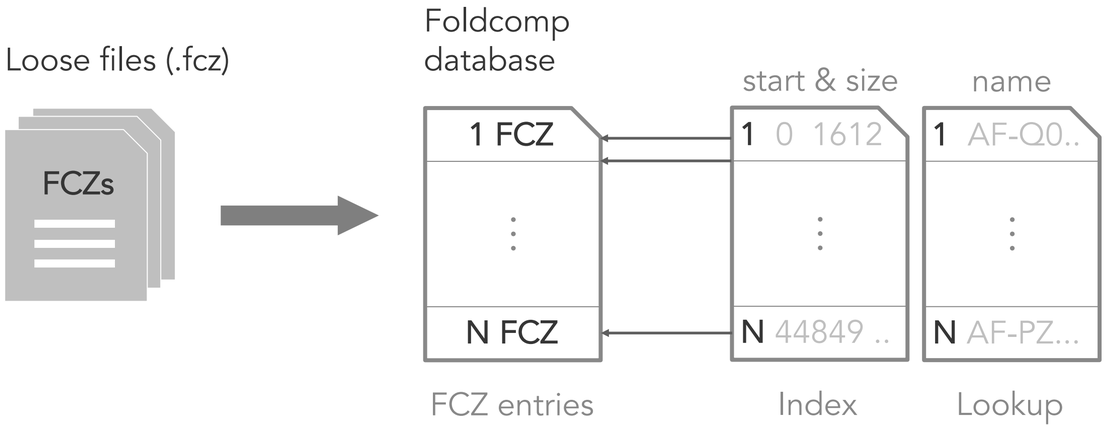
Need HPC to work on it

1.1TB

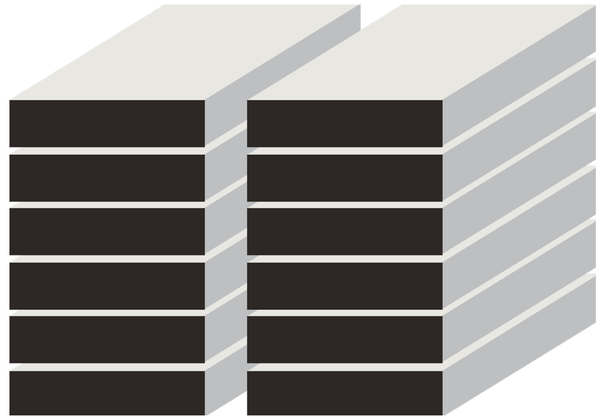
Work on your workstation

Foldcomp compressed AlphaFold-DB into 1 disk





Applied MMseqs2 database format to reduce unnecessary padding bytes used in TAR, which also reduced overhead from file numbers



214,684,311

File system overhead

5

Iterable & searchable

Foldcomp

https://doi.org/10.1093/bioinformatics/btad153

```
foldcomp compress some.pdb
foldcomp decompress other.fcz
```

Executable

```
# Compression
foldcomp compress <pdb_file|cif_file> [<fcz_file>]
foldcomp compress [-t number] <pdb_dir|cif_dir> [<fcz_dir>]

# Decompression
foldcomp decompress <fcz_file> [<pdb_file>]
foldcomp decompress [-t number] <fcz_dir> [<pdb_dir>]

# Extraction of sequence or pLDDT
foldcomp extract [--plddt|--fasta] <fcz_file> [<txt_file|fasta_file>]
foldcomp extract [--plddt|--fasta] [-t number] <fcz_dir|tar> [<output_dir>]

# Check
foldcomp check <fcz_file>
foldcomp check [-t number] <fcz_dir|tar>

# RMSD
foldcomp rmsd <pdb1|cif1> <pdb2|cif2>

# Options
 -h, --help            print this help message
 -t, --threads         threads for (de)compression of folders/tar files [default=1]
 -a, --alt             use alternative atom order [default=false]
 -b, --break           interval size to save absolute atom coordinates [default=25]
 -z, --tar             save as tar file [default=false]
 --plddt               extract pLDDT score (only for extraction mode)
 --fasta               extract amino acid sequence (only for extraction mode)
 --no-merge            do not merge output files (only for extraction mode)
```

Supports pdb, cif, tar, tar.gz, directory, file list

https://github.com/steineggerlab/foldcomp

Publicly available: Python API & Foldcomp DBs

Python API
- fast access
- DB downloads

```python
import foldcomp
# 01. Handling a FCZ file
with open("test/compressed.fcz", "rb") as fcz:
  fcz_binary = fcz.read()
  # Decompress
  (name, pdb) = foldcomp.decompress(fcz_binary)
  # Save to a pdb file
  with open(name, "w") as pdb_file:
    pdb_file.write(pdb)

  # Get data as dictionary
  data_dict = foldcomp.get_data(fcz_binary)
  # Keys: phi, psi, omega, torsion_angles, residues, bond_angles, coordinates
  data_dict["torsion_angles"]
  data_dict["coordinates"]  # coordinates of the backbone as list

# 02. Iterate over a database of FCZ files
ids = ["d1asha_", "d1it2a_"]
with foldcomp.open("test/example_db", ids=ids) as db:
  # Iterate through database
  for (name, pdb) in db:
      # save entries as seperate pdb files
      with open(name + ".pdb", "w") as pdb_file:
        pdb_file.write(pdb)
```

```
pip install foldcomp
```

Publicly available databases
- AlphaFold DB
- ESMatlas

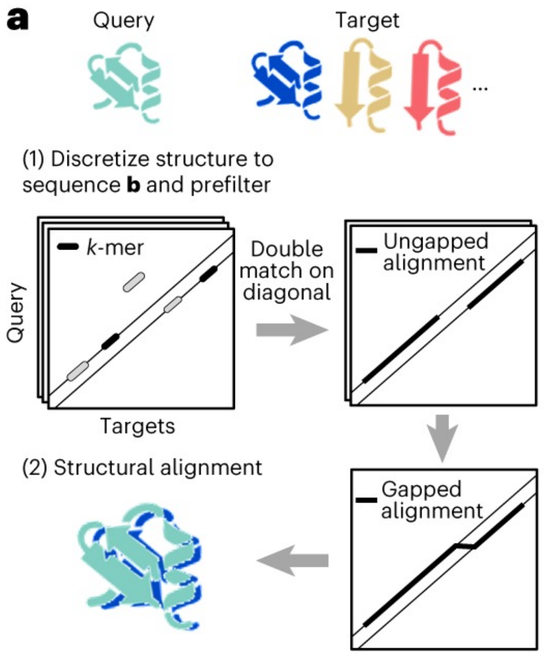
afdb_swissprot_v4.dbtype	Tue, 13 Dec 2022 07:44:23 GMT	4 B
afdb_swissprot_v4.index	Tue, 13 Dec 2022 07:44:24 GMT	11.6 MB
afdb_swissprot_v4.lookup	Tue, 13 Dec 2022 08:34:37 GMT	15.9 MB
afdb_swissprot_v4	Tue, 13 Dec 2022 07:45:25 GMT	2.8 GB
afdb_uniprot_v4.dbtype	Wed, 14 Dec 2022 03:52:06 GMT	4 B
afdb_uniprot_v4.index	Wed, 14 Dec 2022 03:51:45 GMT	5.6 GB
afdb_uniprot_v4.lookup	Wed, 14 Dec 2022 04:01:29 GMT	8.4 GB
afdb_uniprot_v4.source	Wed, 14 Dec 2022 04:03:17 GMT	38.4 MB
afdb_uniprot_v4	Tue, 13 Dec 2022 22:15:20 GMT	1.0 TB
esmatlas.dbtype	Thu, 18 May 2023 03:45:21 GMT	4 B
esmatlas.err.log	Fri, 19 May 2023 10:09:14 GMT	1.0 GB
esmatlas.index	Thu, 18 May 2023 03:40:22 GMT	15.9 GB
esmatlas.lookup	Thu, 18 May 2023 03:45:01 GMT	16.4 GB
esmatlas	Wed, 17 May 2023 10:25:41 GMT	1.7 TB

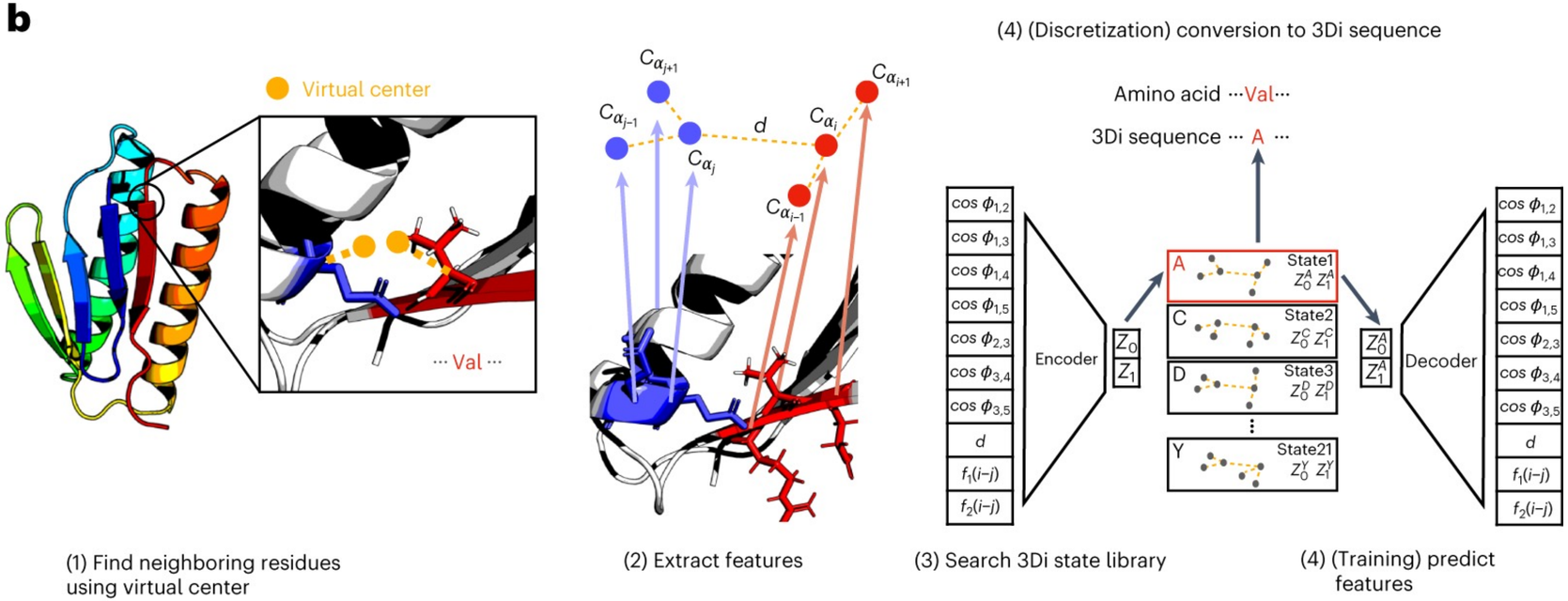
https://foldcomp.steineggerlab.workers.dev/

```
Alphafold db: 1.1TB
ESM atlas   : 1.8TB
Swissprot   : 2.9GB
```

Foldseek

Fast and accurate protein structure search with Foldseek

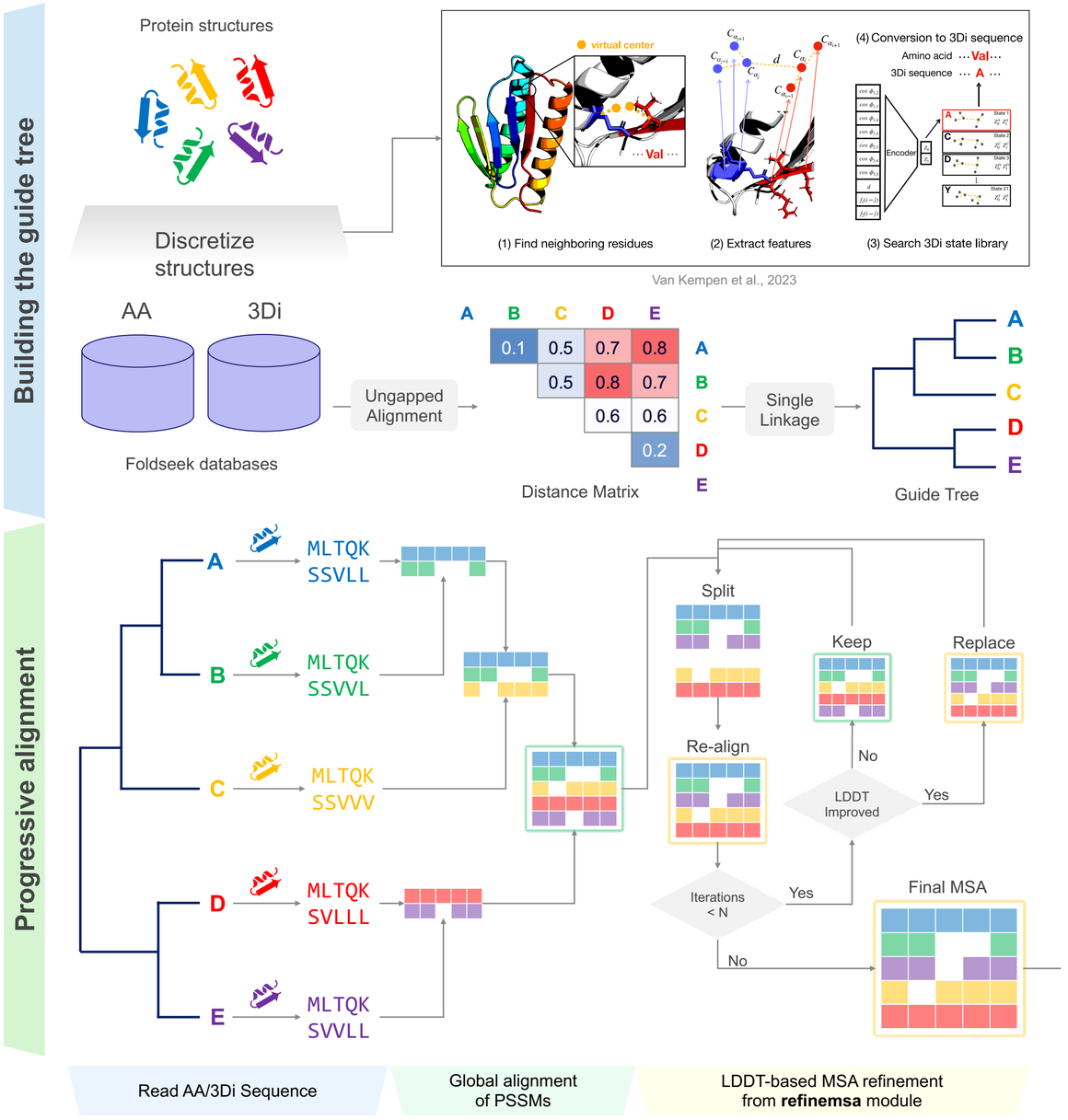




https://www.nature.com/articles/s41587-023-01773-0

Foldseek

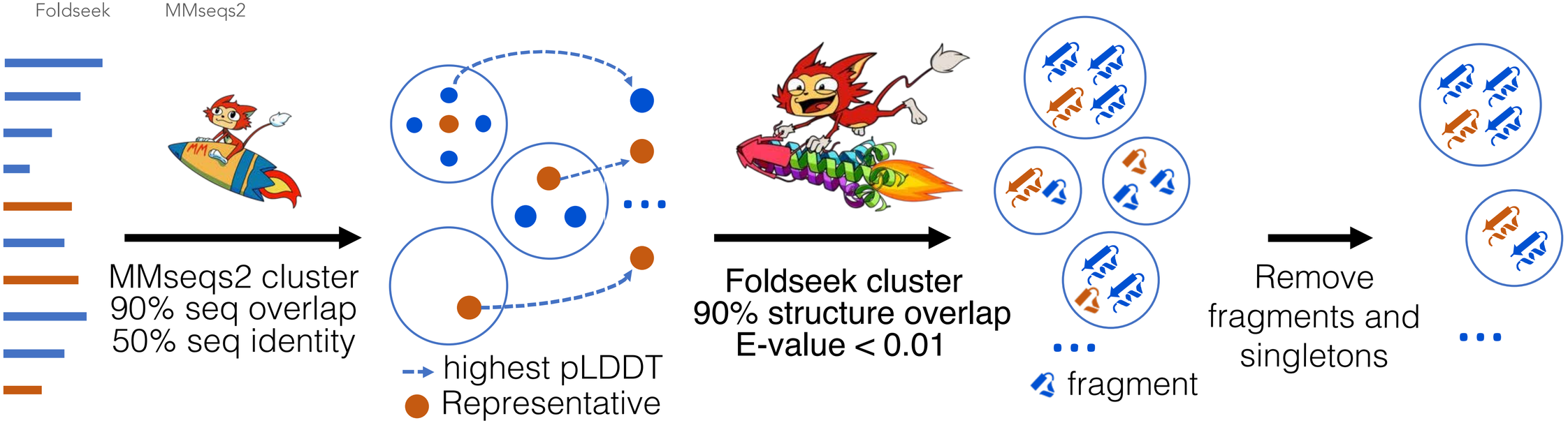
FoldMason: Comparative protein structure analysis in the era of next generation structure predictions



https://github.com/gamcil/foldseek

Foldseek MMseqs2

Clustering predicted structures at the scale of the known protein universe



214M proteins
AFDB

52.3M clusters
AFDB50

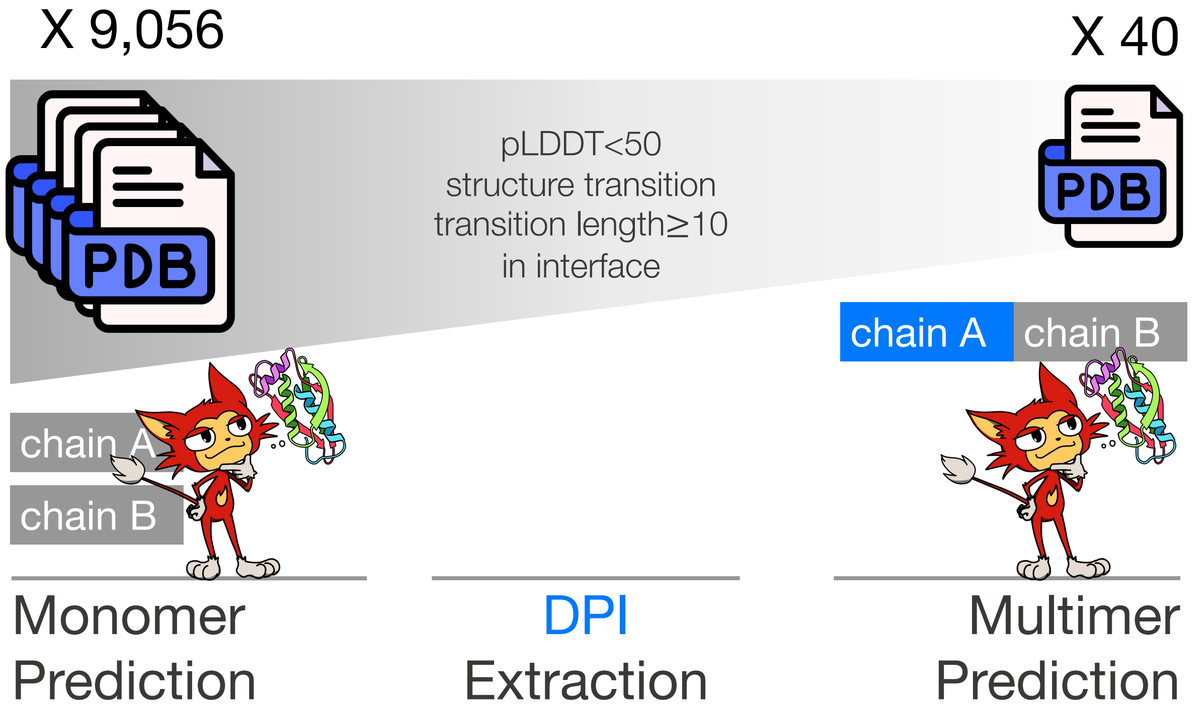
18.8M cluster
Foldseek clusters

2.30M cluster
AFDB clusters

https://www.biorxiv.org/content/10.1101/2023.03.09.531927v1

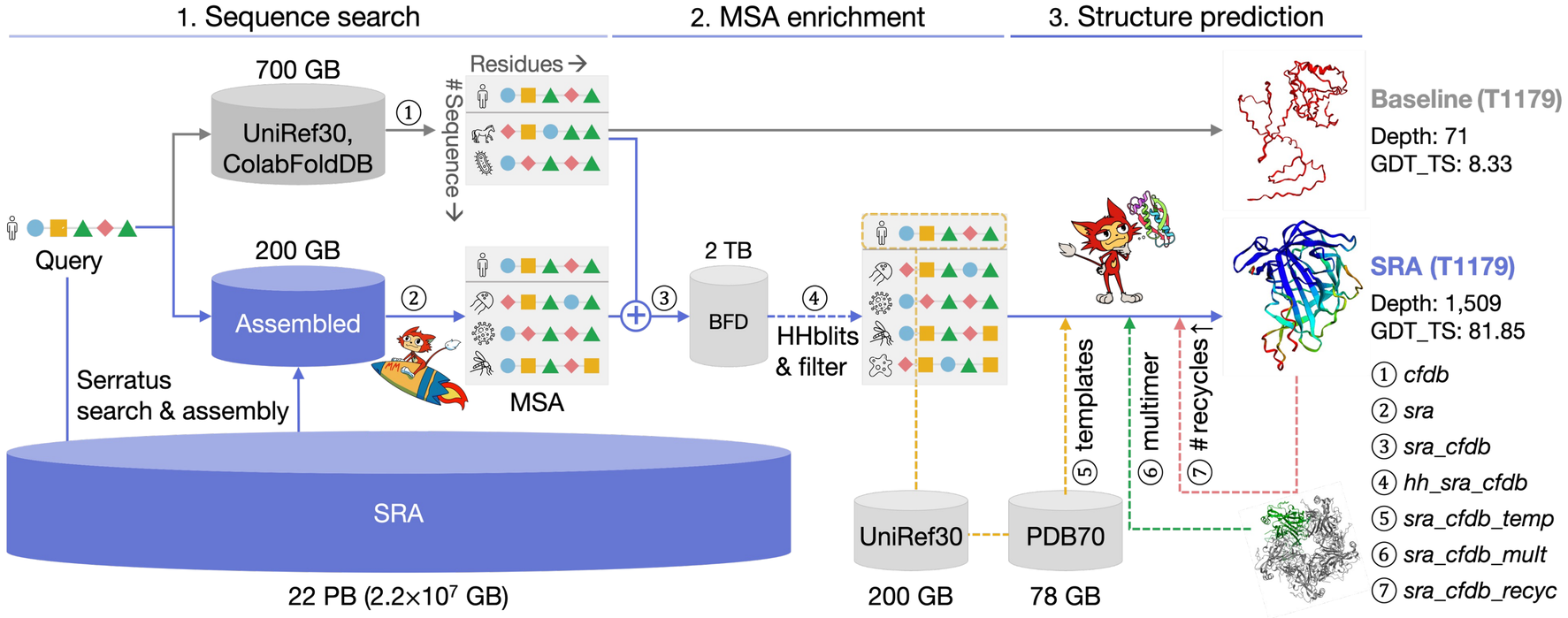
Colabfold

Exploring AlphaFold2's Capability in Predicting Intrinsically Disordered Protein Interactions



MMseqs2 Colabfold

Petascale Search for Protein Structure Prediction



cfdb: ColabFoldDB, **hh**: HHblits, **temp**: templates, **mult**: multimer, **recyc**: #recycle

https://www.biorxiv.org/content/10.1101/2023.07.10.548308v1

Acknowledgement

https://steineggerlab.com/en/

AFDB

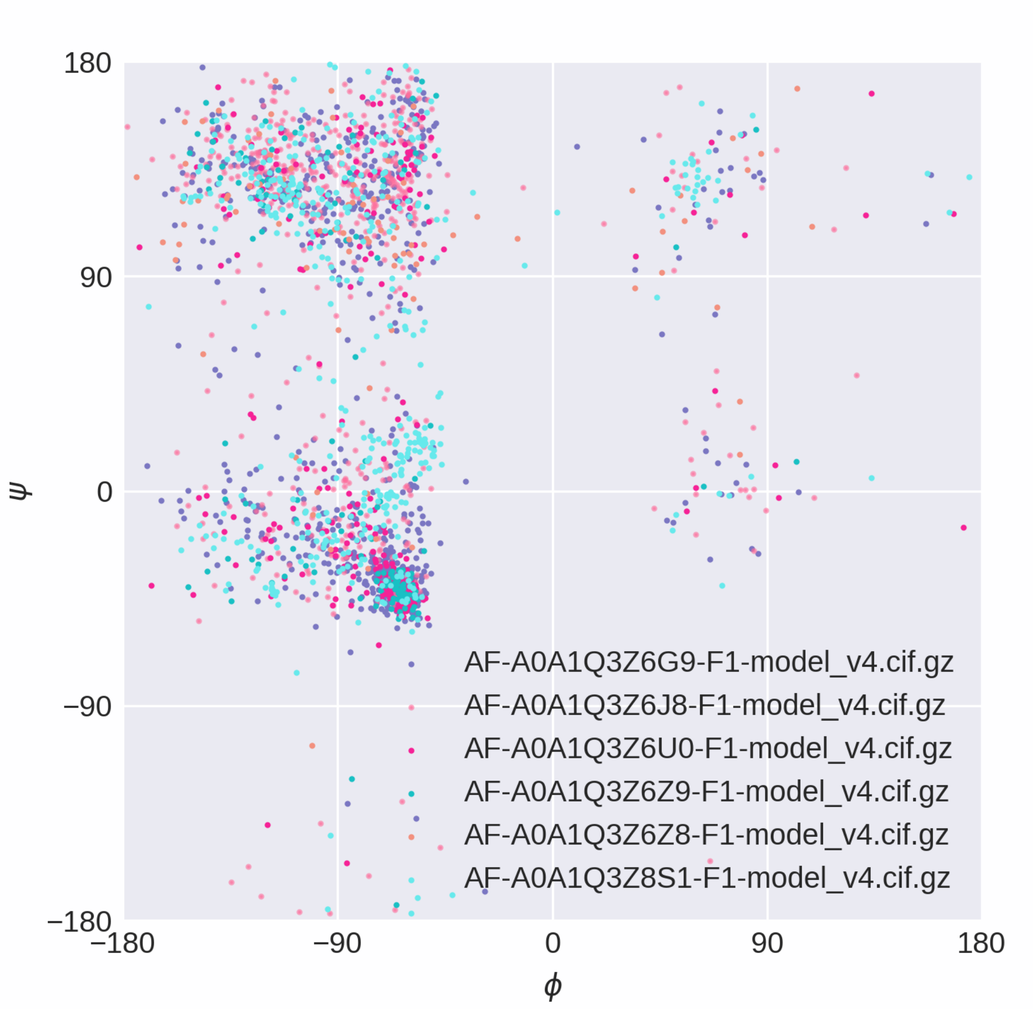
Thank you for listening

foldcomp

Foldcomp – python API

```
>>> import foldcomp
>>> fcz = open("./foldcomp/compressed.fcz", "rb").read()
>>> pdb = open("./foldcomp/decompressed.pdb", "r").read()
>>> data = foldcomp.get_data(pdb) # foldcomp.get_data(fcz) also works
>>> data.keys()
dict_keys(['phi', 'psi', 'omega', 'torsion_angles', 'bond_angles',
'residues', 'b_factors', 'coordinates'])
>>> data["residues"]
'SDDWEIPDGQITVGQRIGSGSFGTVYKGKWHGDVAVKMLNVTAPTPQQLQAFKNEVGVLRKTRHVNILLFMGY
STKPQLAIVTQWCEGSSLYHHLHIIETKFEMIKLIDIARQTAQGMDYLHAKSIIHRDLKSNNIFLHEDLTVKIG
DFGLATVKSRWSGSHQFEQLSGSILWMAPEVIRMQDKNPYSFQSDVYAFGIVLYELMTGQLPYSNINNRDQIIF
MVGRGYLSPDLSKVRSNCPKAMKRLMAECLKKKRDERPLFPQILASIELLARSLP'
>>> data["b_factors"][0:3]
[72.58999633789062, 71.2300033569336, 58.7099990844726 56]
>>> data["torsion_angles"][0:3]
[-178.83718872070312, -171.744049072265 62, -86.36867523193 36]
>>> data["coordinates"][10]
(23.875, -44.77299880981445, 3.26699995994567 87)
```

Foldcomp – python API




```python
import foldcomp
import matplotlib.pyplot as plt
import matplotlib.tri as tri
import numpy as np

list_of_data_dicts = []

# load the data
db_all = foldcomp.open("afdb_rep_v4")
N_PROTEINS = 6
db = [db_all[i] for i in range(N_PROTEINS)]

for (name, pdb) in db:
    list_of_data_dicts.append((name, foldcomp.get_data(pdb)))

def set_axis_for_ramachandran(ax):
    ax.set_xlim(-180, 180)
    ax.set_ylim(-180, 180)
    ax.set_aspect("equal")
    ax.set_xticks([-180, -90, 0, 90, 180])
    ax.set_yticks([-180, -90, 0, 90, 180])
    ax.set_xlabel(r"$\phi$")
    ax.set_ylabel(r"$\psi$")

fig, ax = plt.subplots(figsize=(5, 5))
set_axis_for_ramachandran(ax)

for i, (name, data) in enumerate(list_of_data_dicts[:N_PROTEINS]):
    ax.scatter(data["phi"], data["psi"], s=1, label=name)

ax.legend()

plt.tight_layout()
plt.savefig("ramachandran.png", dpi=300)
```

Foldcomp – additional features

```
(base) hyunbin@super003:/mnt/scratch/hyunbin/af_uniprot/AF2_Unipr
ot_FCMP_plddt$ grep -c ">" ./plddt.txt
214684311

(base) hyunbin@super003:/mnt/scratch/hyunbin/af_uniprot/AF2_Unipr
ot_FCMP_plddt$ grep -E "^000" -B 1 ./plddt.txt
>AF-A0A401S7I3-F1-model_v3.cif
0000233233324333434555544334554554544444455555555444445555555554444
3455555555444345545454444443444555554444345555444334444555543433344
5554443333344555554444334555555433333455555554433345555544333333334
5555544333345545555433333344455444333334444444333333444444433332233
3343333333233333333222222222322222222232222222222222222222222222232
3233333222333333333333433333444444333444444443434333344444444444444
44544444444343334444444444444444444443333223232233333015
--
>AF-A0A210PZP2-F1-model_v3.cif
0004324554356677888877655556667788888776555666778888876655556677 8
8888876655566677888888776666667888888877666666788888887666656688888
8888766666666888888877666566668888888877655556677887777765556667787 7
7777666566677777666665555656676666655555555666665656555556566666 5
6666565556566767666666655666677767666655455451645551152524556
```

Direct feature extraction