

# Moving Beyond Memorisation

## Resources for Generalisable Protein Complex Prediction

Connected concepts in datasets,  
benchmarking, and training



**plinder**

The Protein-Ligand Interaction  
Dataset and Evaluation Resource

more data  
more diversity  
more flexible



**pinder**

The Protein Interaction  
Dataset and Evaluation Resource

higher quality  
less leakage  
better evaluation

▼ TRAINING WORKSHOP

# event-general  
# 01\_leak-vs-generalize  
# 02\_metrics-and-eval  
# 03-1\_pinder-dataloader  
# 03-2\_plinder-dataloader  
# 04\_mlsb-leaderboard  
# tech-support

# Welcome to #01\_leak-vs-generalize!

This is the start of the #01\_leak-vs-generalize channel.

 Edit Channel

September 24, 2024



Vladas Today at 1:15 PM

How would you characterise the current state-of-the-art (SOTA) in complex prediction?

Select one answer

It's a solved problem - need to redefine the frontier!

Current models generalise well (just need more data)

Models use memorization to exploit leaky benchmarks

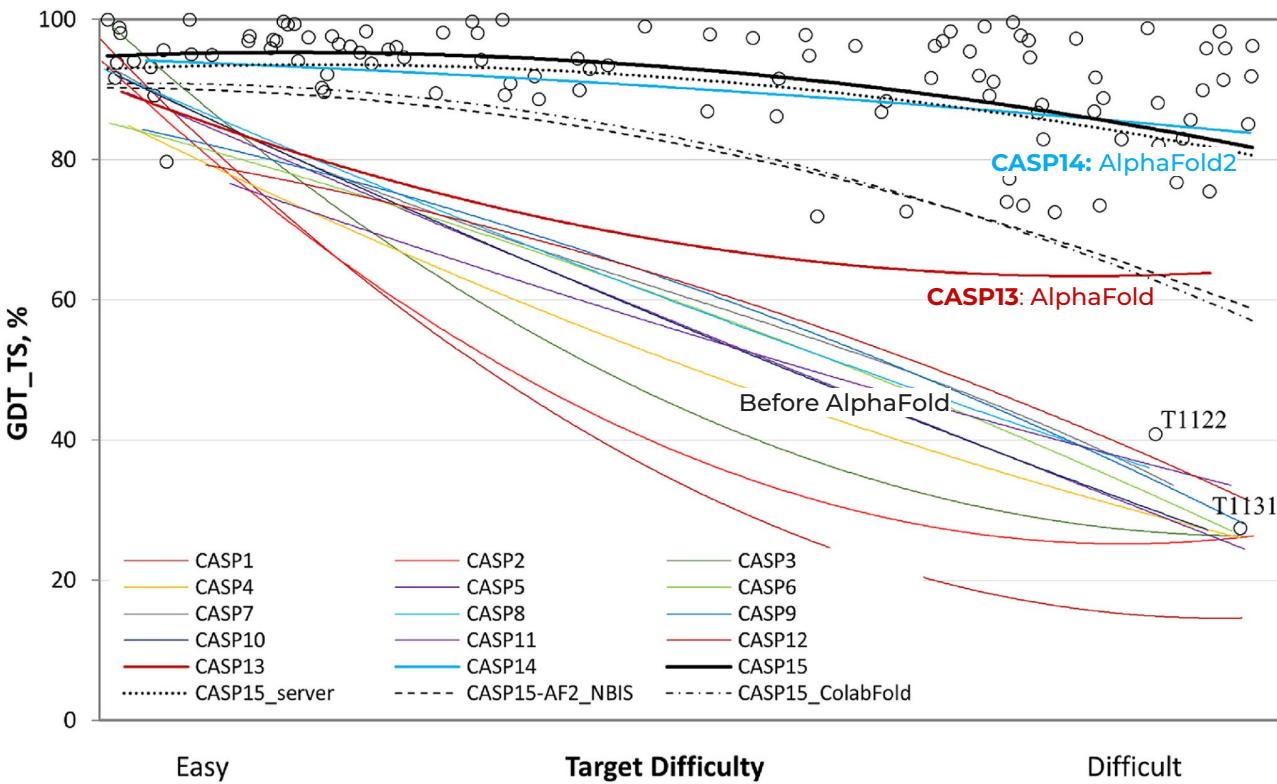
SOTA is impossible to judge without fair benchmarks

0 votes • 23h left

Show results

Vote

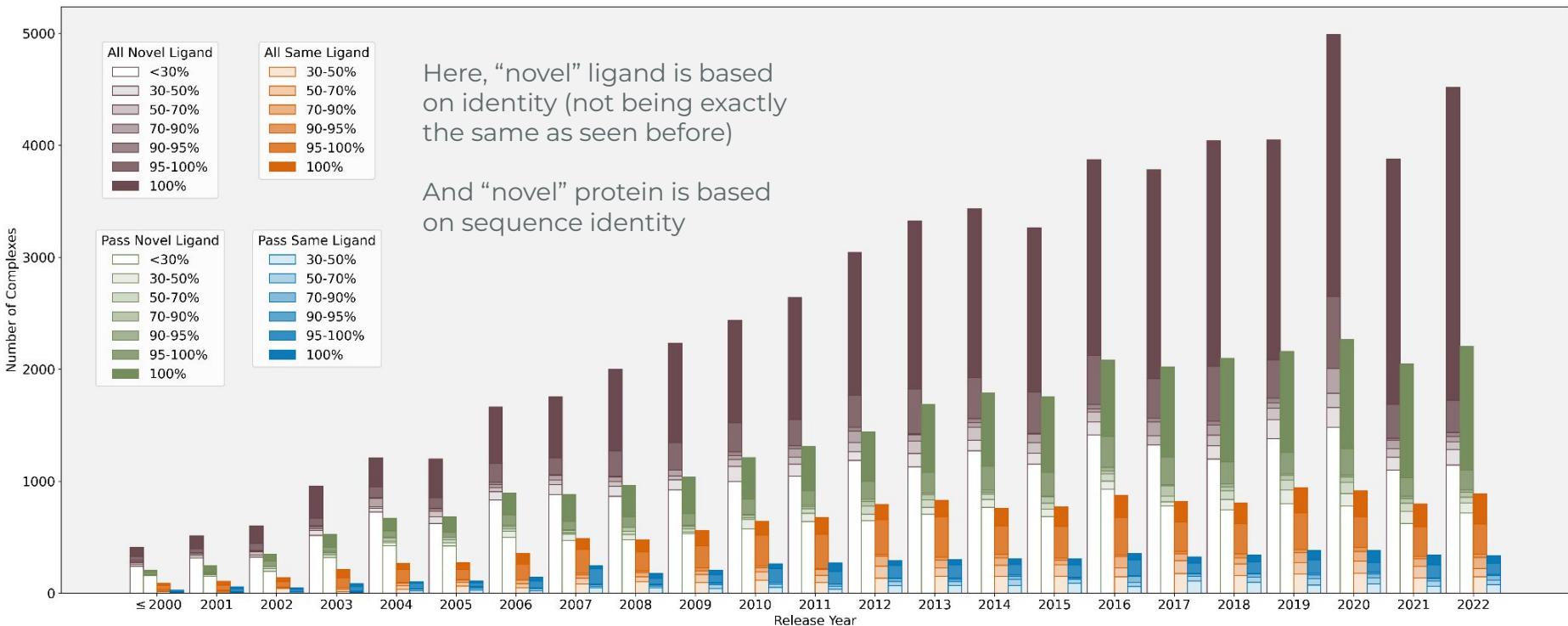
# Breakthroughs require generalisation



AlphaFold2 was a breakthrough due to “**Difficult**” not “**Easy**”

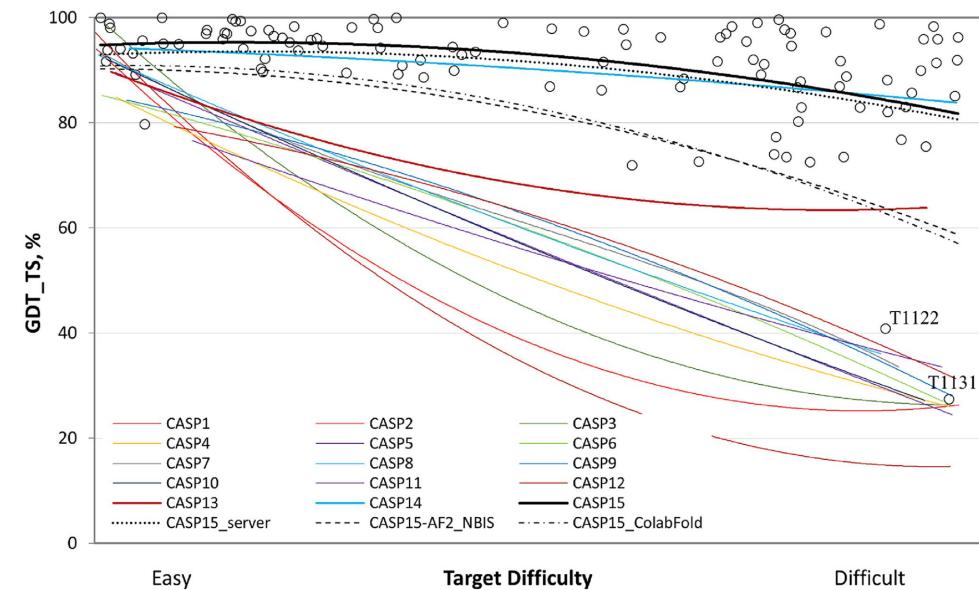
(in monomer prediction)

# “New” PDBs != Novel complexes



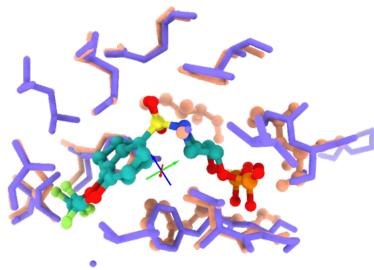
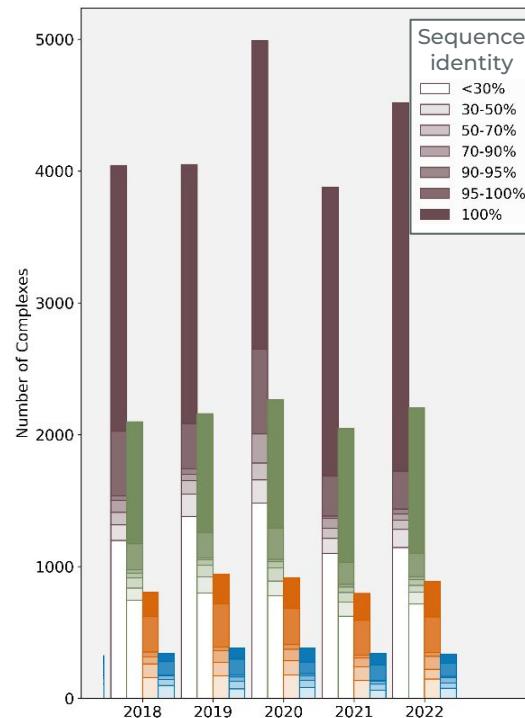
Protein-ligand complexes released per year compared to everything before

# Time splits overestimate performance

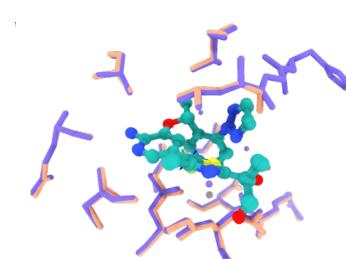


AF2 was a breakthrough  
due to “Difficult” not “Easy”  
(in monomer prediction)

“New” PDBs != Novel complexes



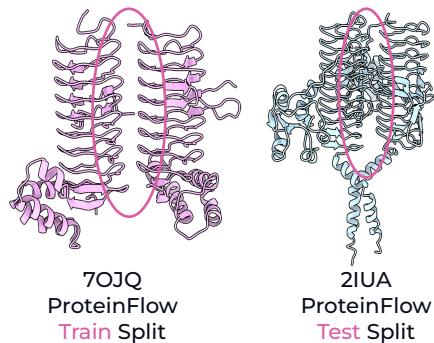
7L03 | PoseBusters  
2CLI | PDBBind



7R7R | PoseBusters  
4CD0 | PDBBind

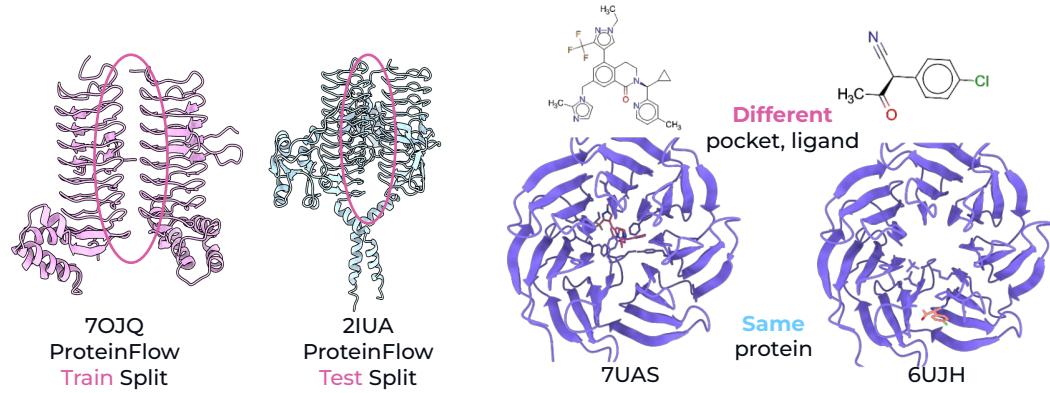
# “Novelty” definition depends on the task

- Generalisation difficulty  
= Interaction dissimilarity



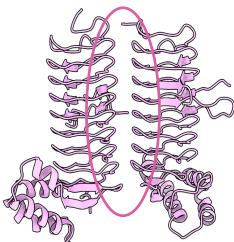
# “Novelty” definition depends on the task

- Generalisation difficulty  
= Interaction dissimilarity

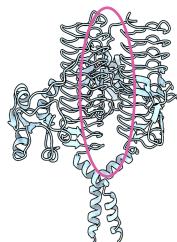


# “Novelty” definition depends on the task

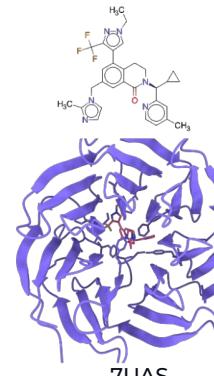
- Generalisation difficulty  
= Interaction dissimilarity



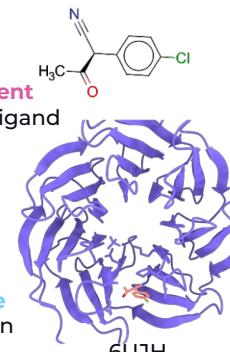
7OJQ  
ProteinFlow  
Train Split



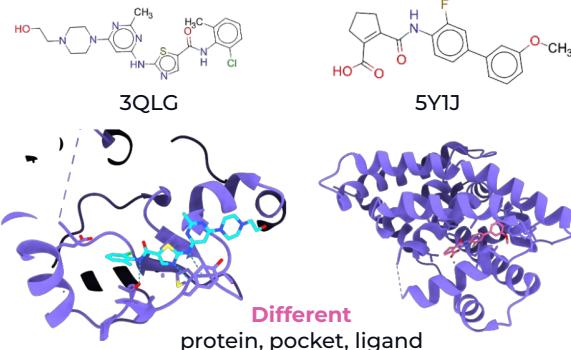
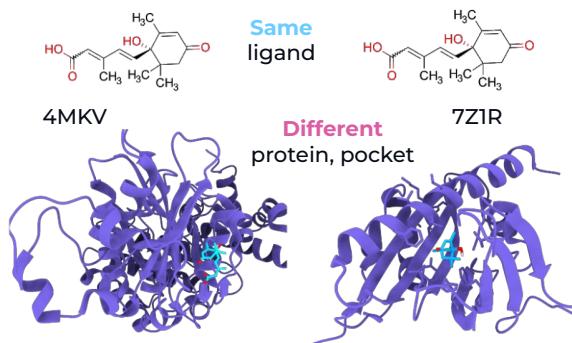
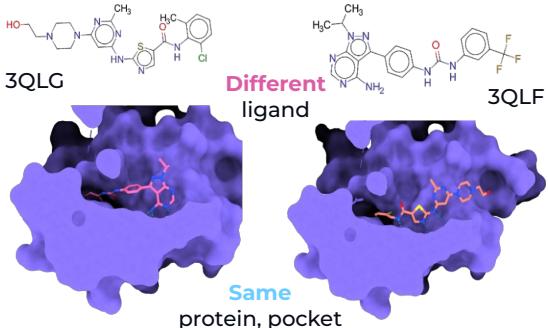
2IUA  
ProteinFlow  
Test Split



Different  
pocket, ligand

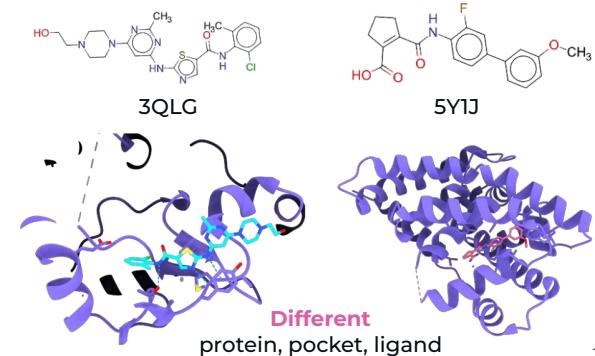
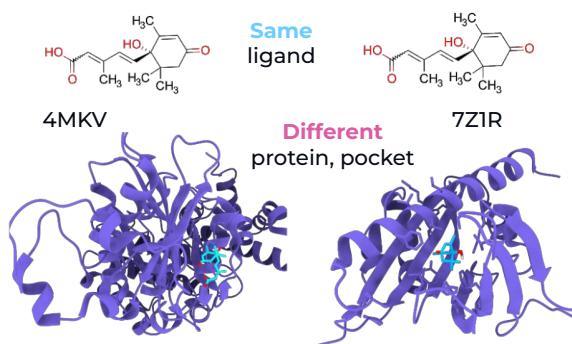
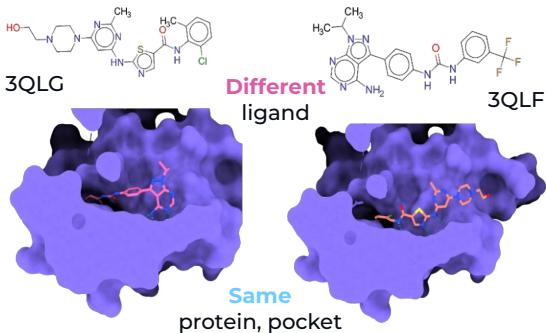
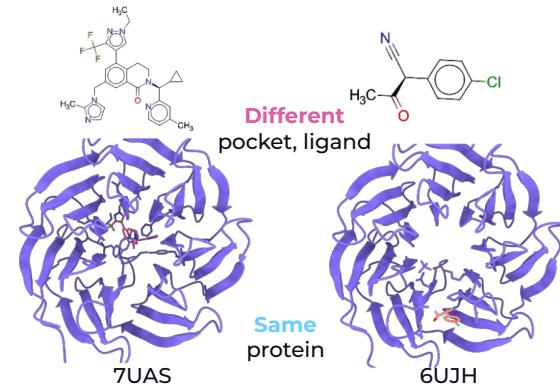
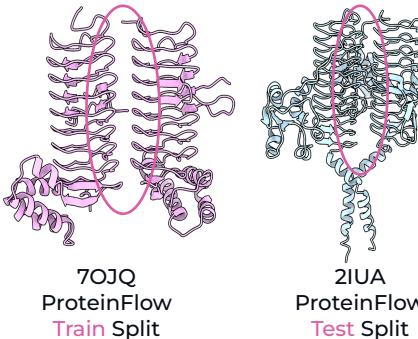


Same  
protein



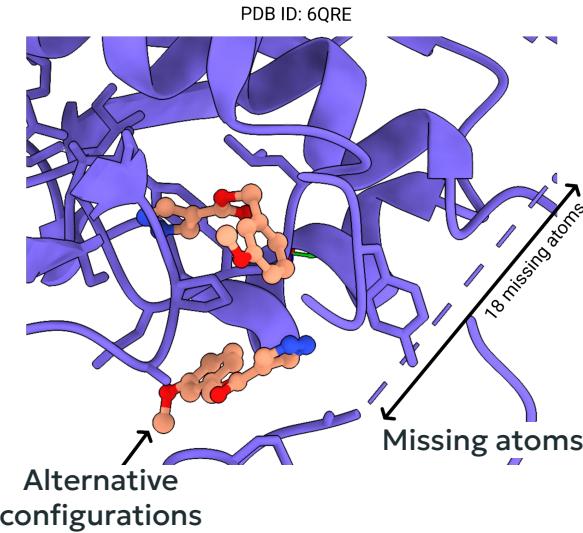
# “Novelty” definition depends on the task

- Generalisation difficulty  
= Interaction dissimilarity
- P(L)INDER measures,  
*clusters, and splits* on  
interaction similarities



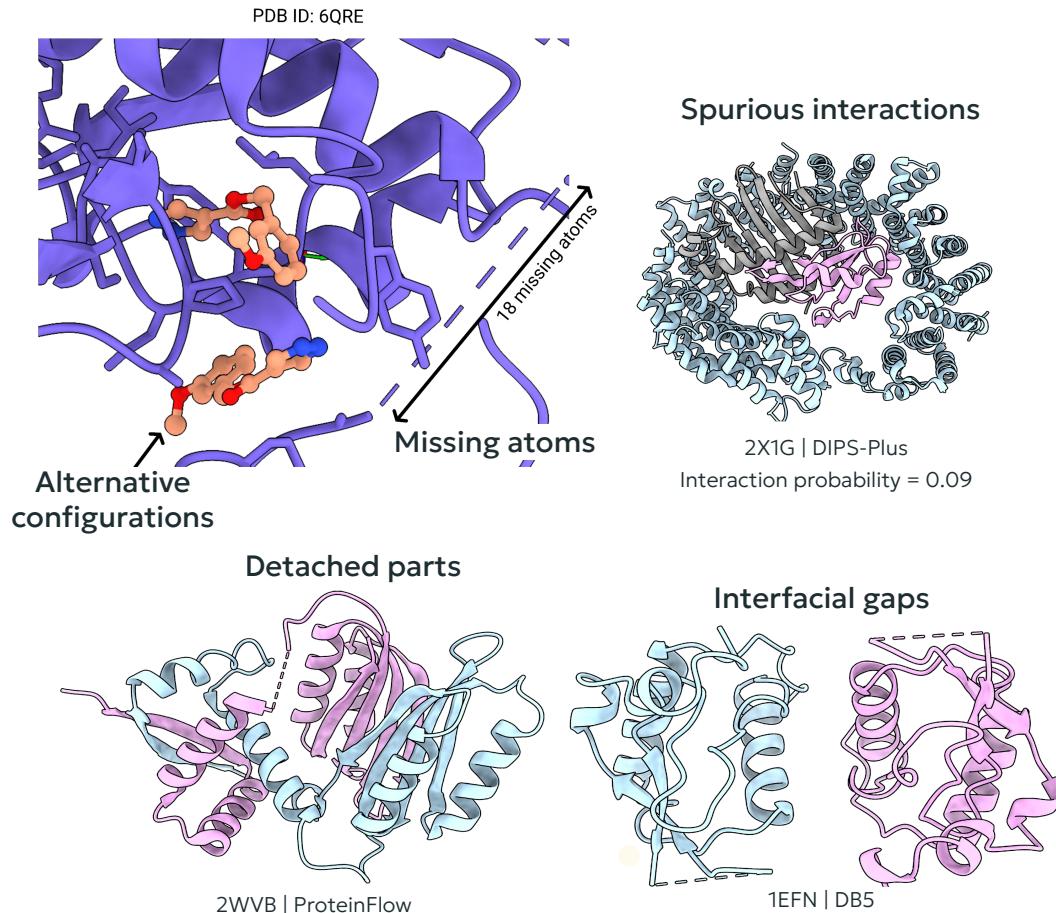
# Why quality matters

- Experimental structures vary in quality



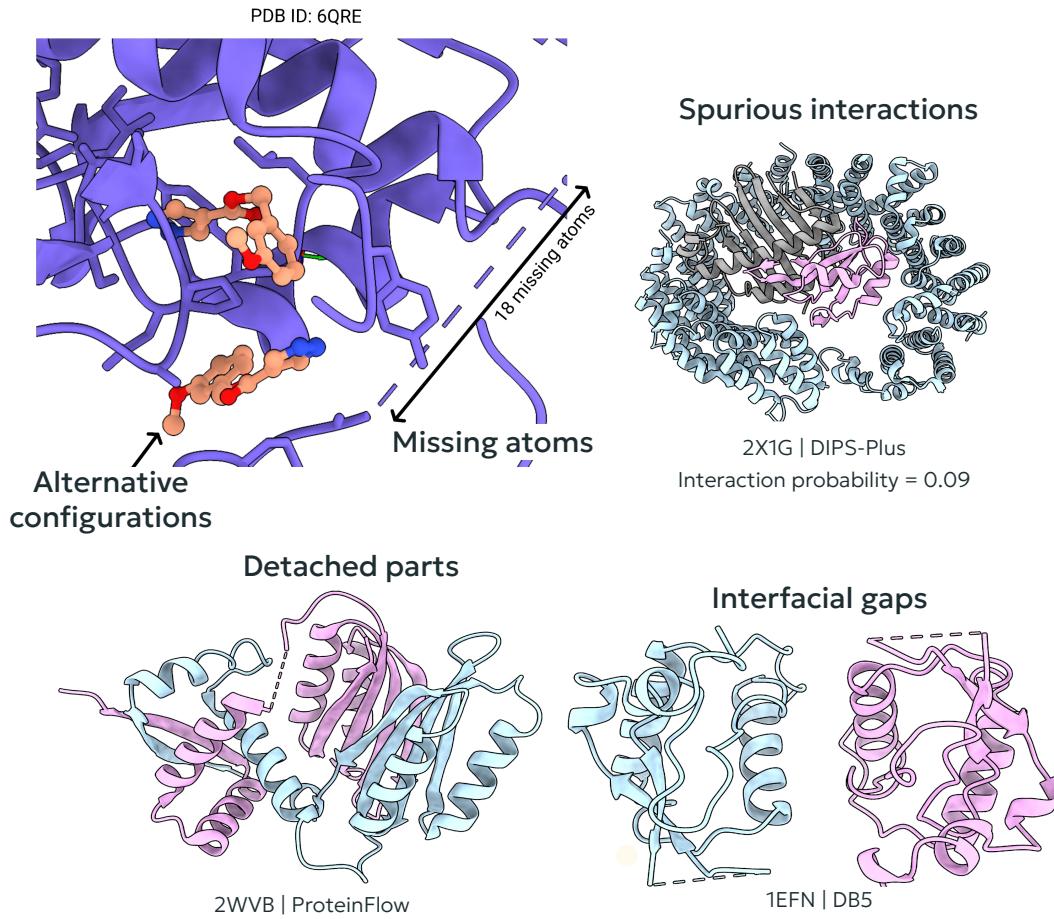
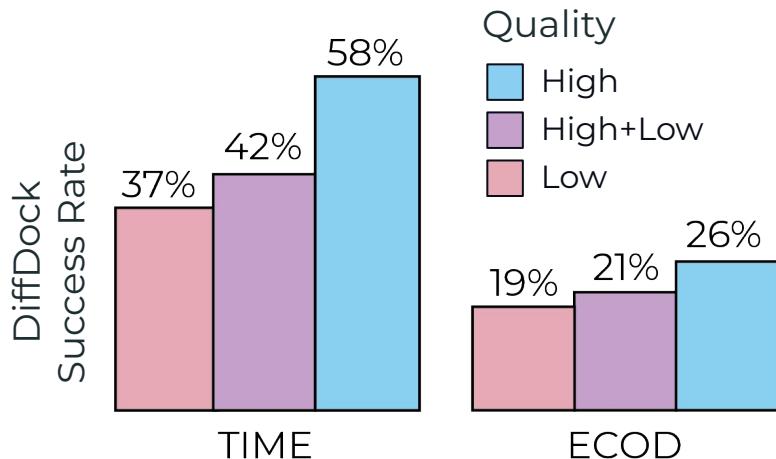
# Why quality matters

- Experimental structures vary in quality

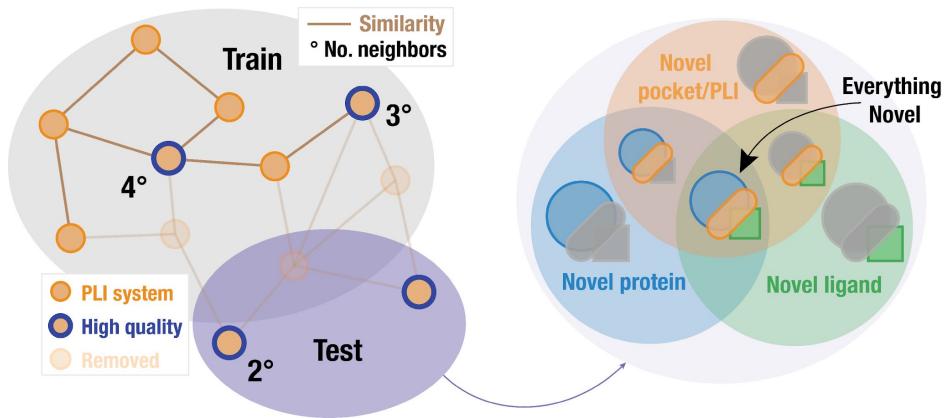


# Why quality matters

- Experimental structures vary in quality
- Need **reliable** ground truth for evaluation



# Split into relevant train/val/test sets



Optimize for

- ❖ Test set **quality**
- ❖ Test set **diversity**
- ❖ Minimal information **leakage**
- ❖ Training set **size** and **diversity**

Stratify into

- ❖ Novel **protein**
- ❖ Novel **ligand**
- ❖ Novel **pocket/Xns**
- ❖ **Everything** novel

**function** SPLITTING( $S, C, G, m, M$ )

```
proto_test  $\leftarrow \emptyset$ 
for  $s \in S$  do
    if pass_quality( $s$ ) then
         $N_s \leftarrow \emptyset$ 
        for  $g \in [1, |G|]$  do
            leaked  $\leftarrow$  neighbors( $s, G_g$ )
             $N_s \leftarrow N_s \cup$  leaked
        if  $m < |N_s| < M$  then
            proto_test.insert( $s$ )
     $\triangleright$  Sort  $s \in$  proto_test by  $|N_s|$ 
    test  $\leftarrow \emptyset$ 
    for  $c \in C$  do
        test  $\leftarrow$  test  $\cup$  upto 1 from proto_test for  $c$ 
        train  $\leftarrow S$ 
        for  $s \in$  test do
            train  $\leftarrow$  train  $\setminus N_s$ 
```

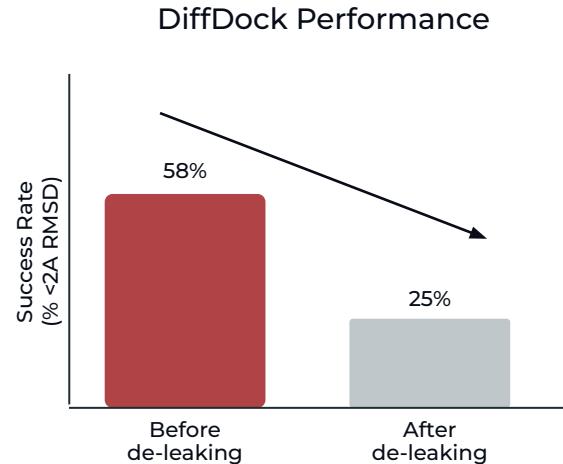
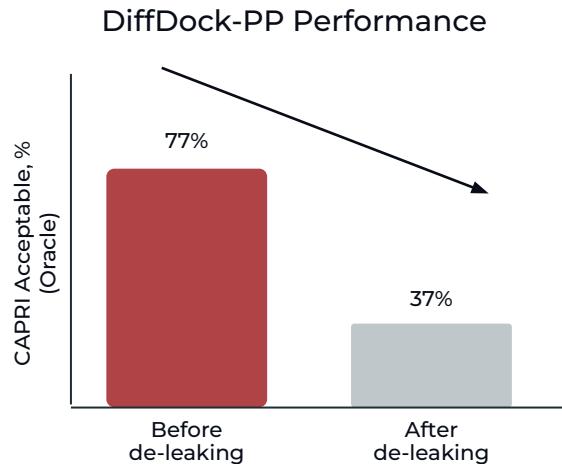
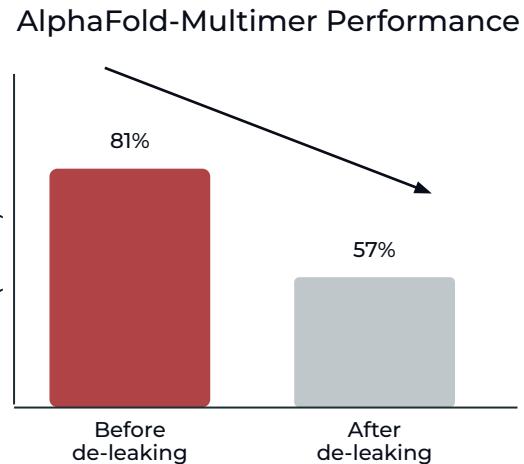
Systems

Clusters

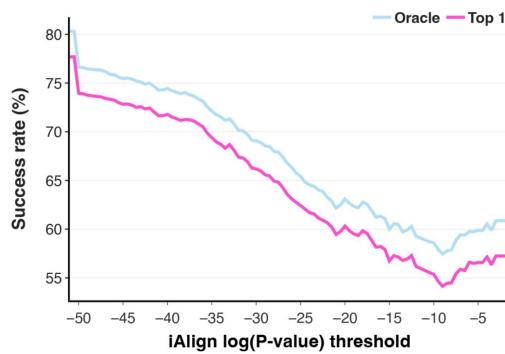
Graphs

min & Max. removal

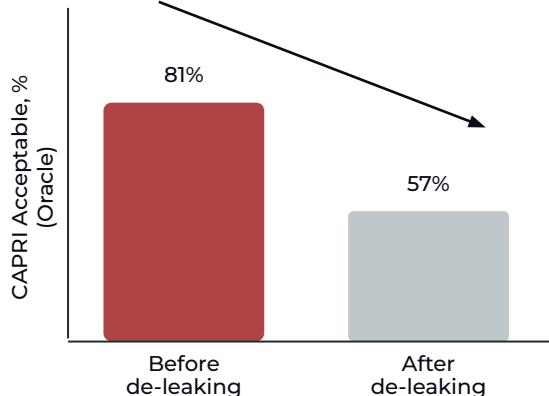
# Lack of generalisation $\approx$ memorisation



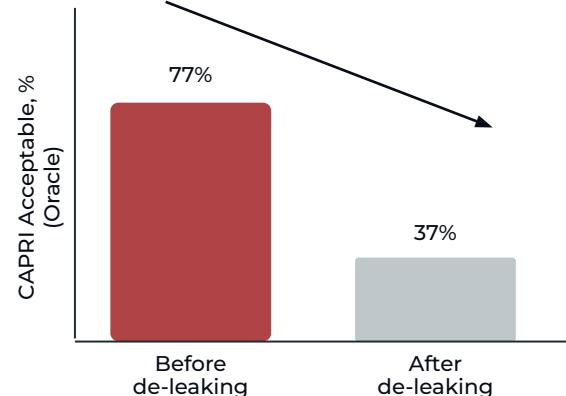
# Lack of generalisation $\approx$ memorisation



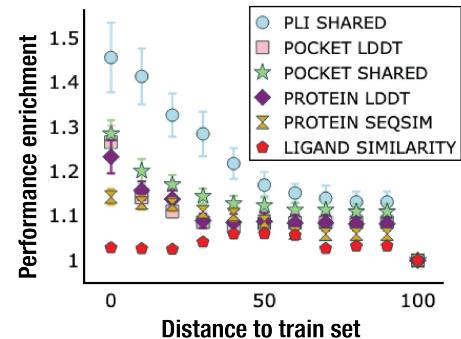
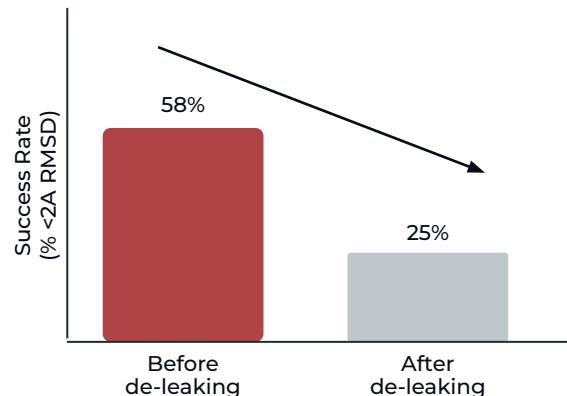
AlphaFold-Multimer Performance



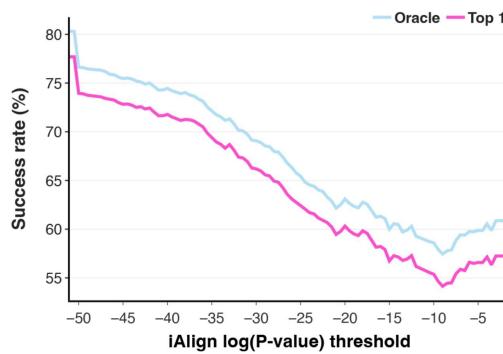
DiffDock-PP Performance



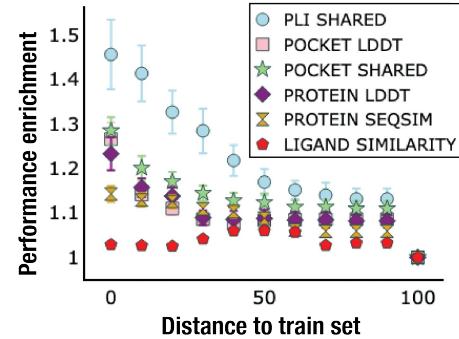
DiffDock Performance



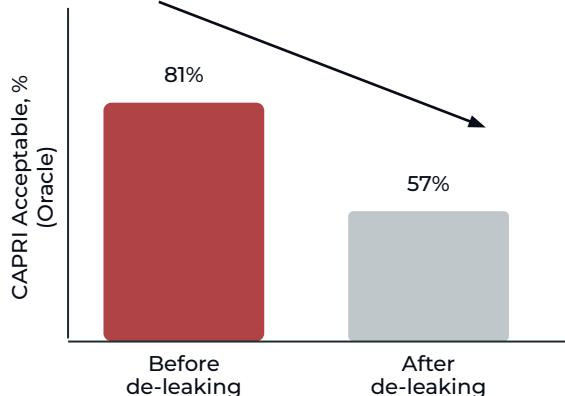
# Lack of generalisation $\approx$ memorisation



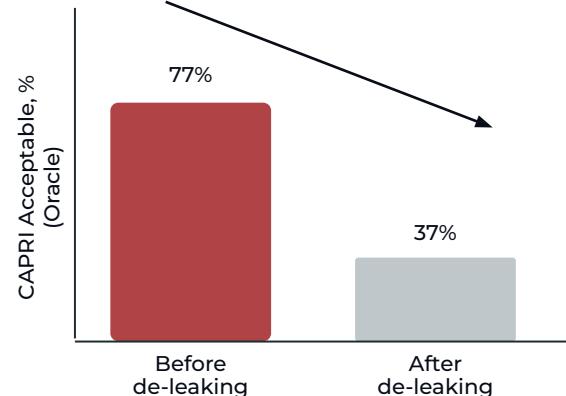
Complex prediction still  
needs a breakthrough



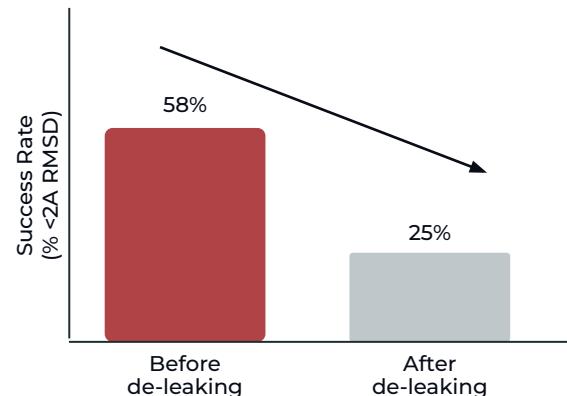
AlphaFold-Multimer Performance



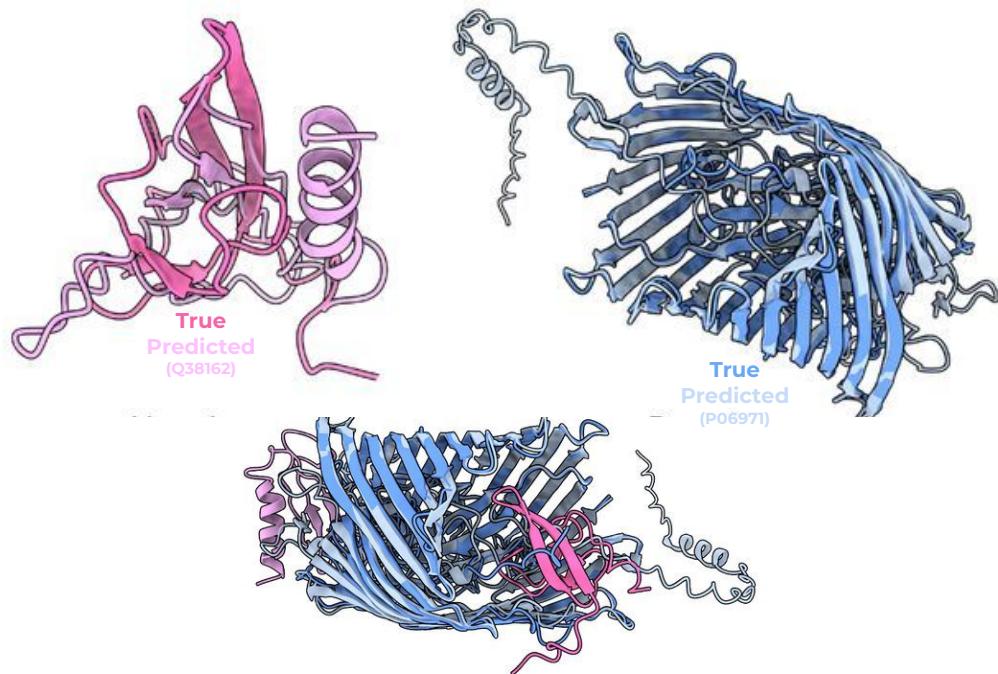
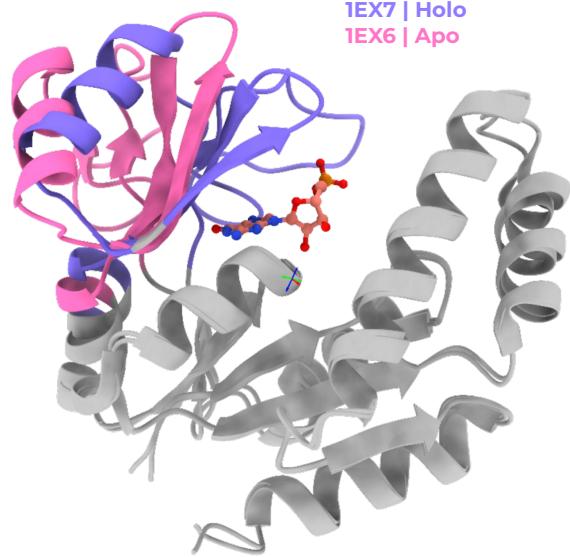
DiffDock-PP Performance



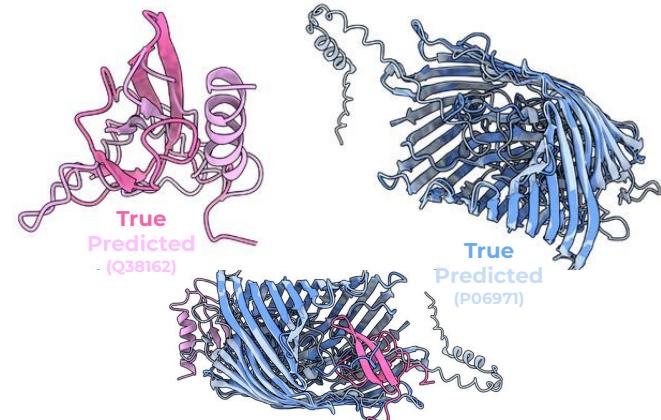
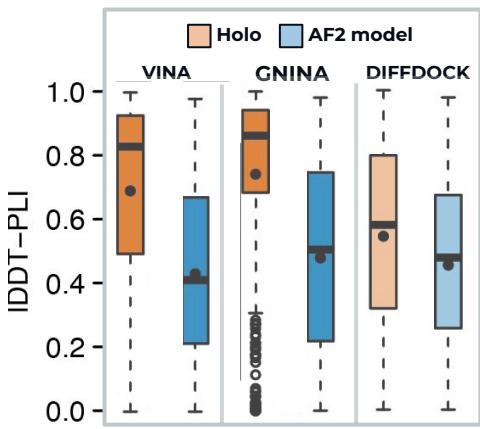
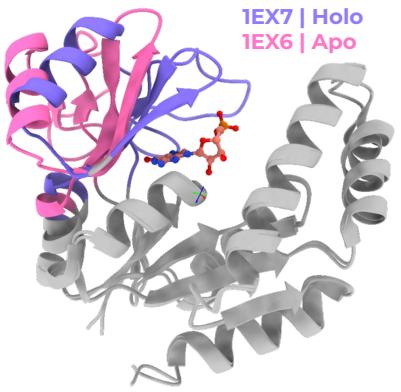
DiffDock Performance



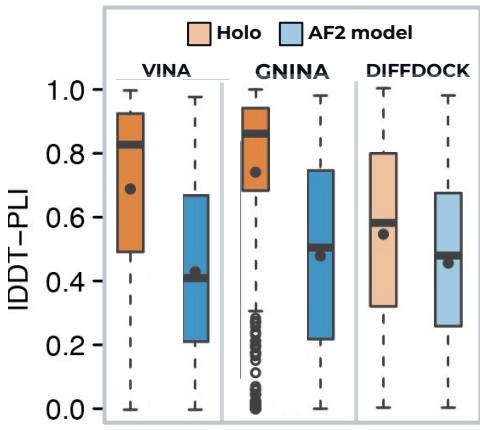
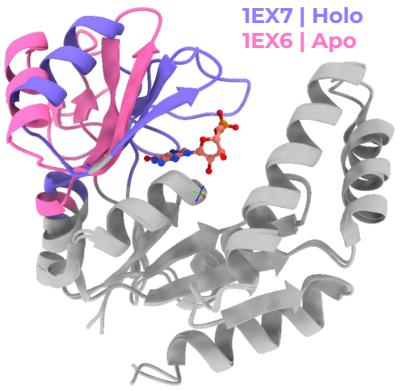
# Realistic inference != Rigid re-docking



# Realistic inference != Rigid re-docking

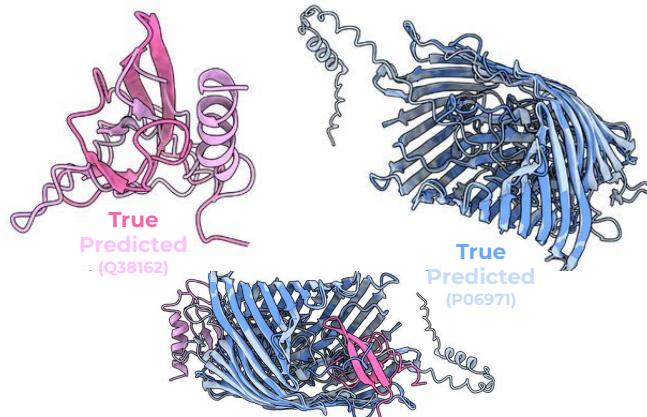
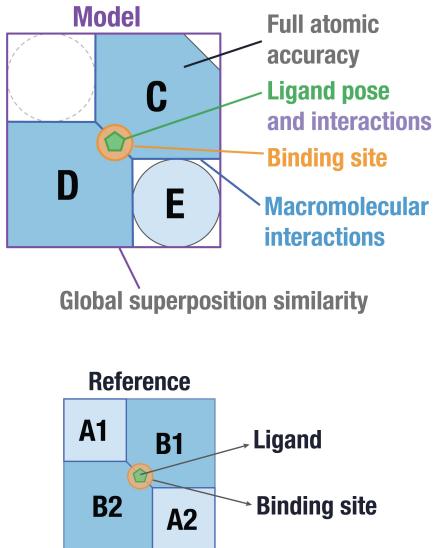


# Realistic inference != Rigid re-docking



**Need diverse and flexible metrics**

- ❖ Beyond **rigid placement** RMSD
- ❖ Evaluate **ligand poses**, **pocket residues** and **interactions**

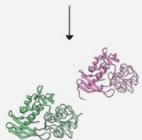


# Automated and reproducible end-to-end workflow

**pinder**

## Ingest

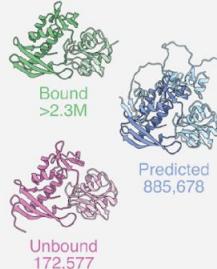
RCSB PDB & AFDB



- Fully automated and reproducible pipeline
- 2,319,564 systems with 9,430 unique ECOD domain pairs across 6,529 families
- 100+ annotations
- Interface quality assessment with 10+ metrics

## Expand

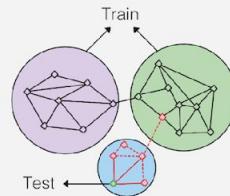
Comprehensive set of protein-protein complex structures



- Paired unbound and AlphaFold2 predicted monomers
- PPI interfaces stratified by flexibility

## Split

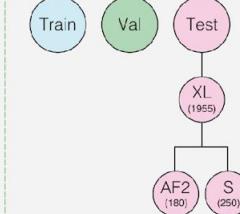
Interface based clustering and deleaking



- FoldSeek and MMSeqs based interface similarity comparison followed by transitive graph-clustering and additional deleaking via iAlign
- Extensive orthogonal leakage validation via ECOD-, PFAM-overlap & other metrics
- Maximum val and test set quality with minimal leakage

## Evaluate

ML-ready dataset splits with high quality benchmark structures

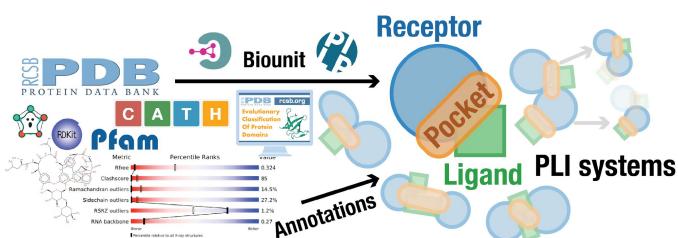


- Large split (XL) with smaller subset (S)
- AF2-training cutoff & interface structural deleaked XL subset (AF2)
- Automated evaluation harness with 38 CASP-CAPRI compatible metrics and leaderboard included
- Holo/Apo/Predicted input leaderboards across protein flexibility levels (easy, medium, hard)

# Automated and reproducible end-to-end workflow

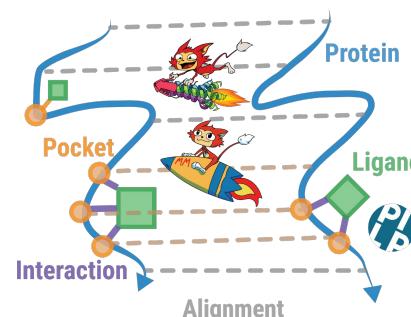
plinder

## Ingest



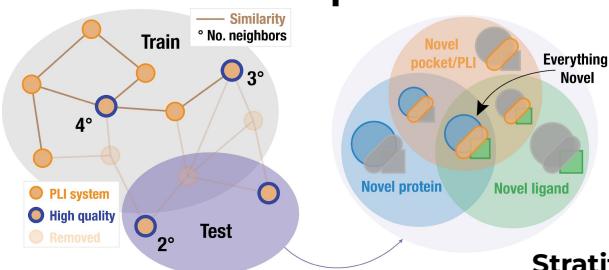
- ❖ Collect **all** PLIs in the PDB
- ❖ **Annotate everything**
- ❖ **Pair apo and AF2 structures**
- ❖ **Keep up with PDB updates**

## Compare



Sequence & structure similarity at **protein**, **pocket**, **ligand**, and **interaction** levels

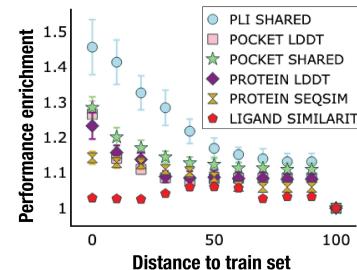
## Split



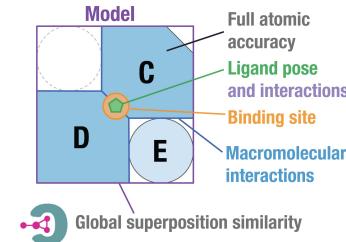
Maximizing test set **quality and diversity** while **minimizing** information leakage

Stratification for different tasks

## Evaluate



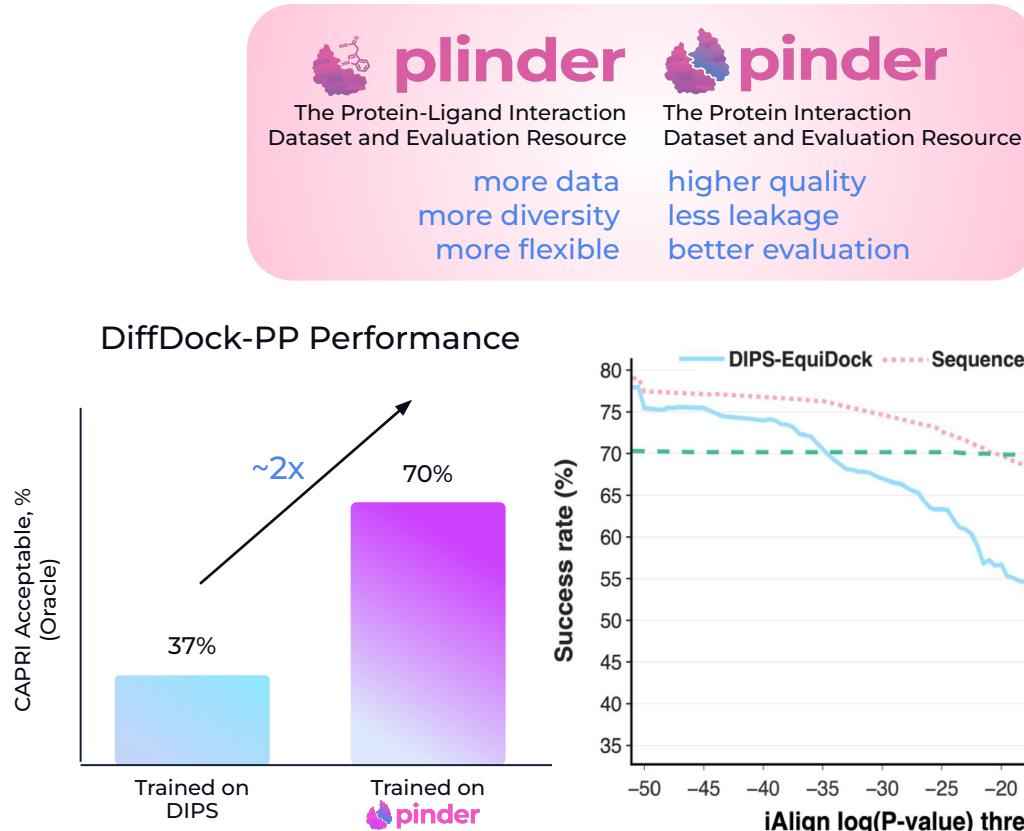
Resources to **inspect biases** in your models



Comprehensive evaluation harness

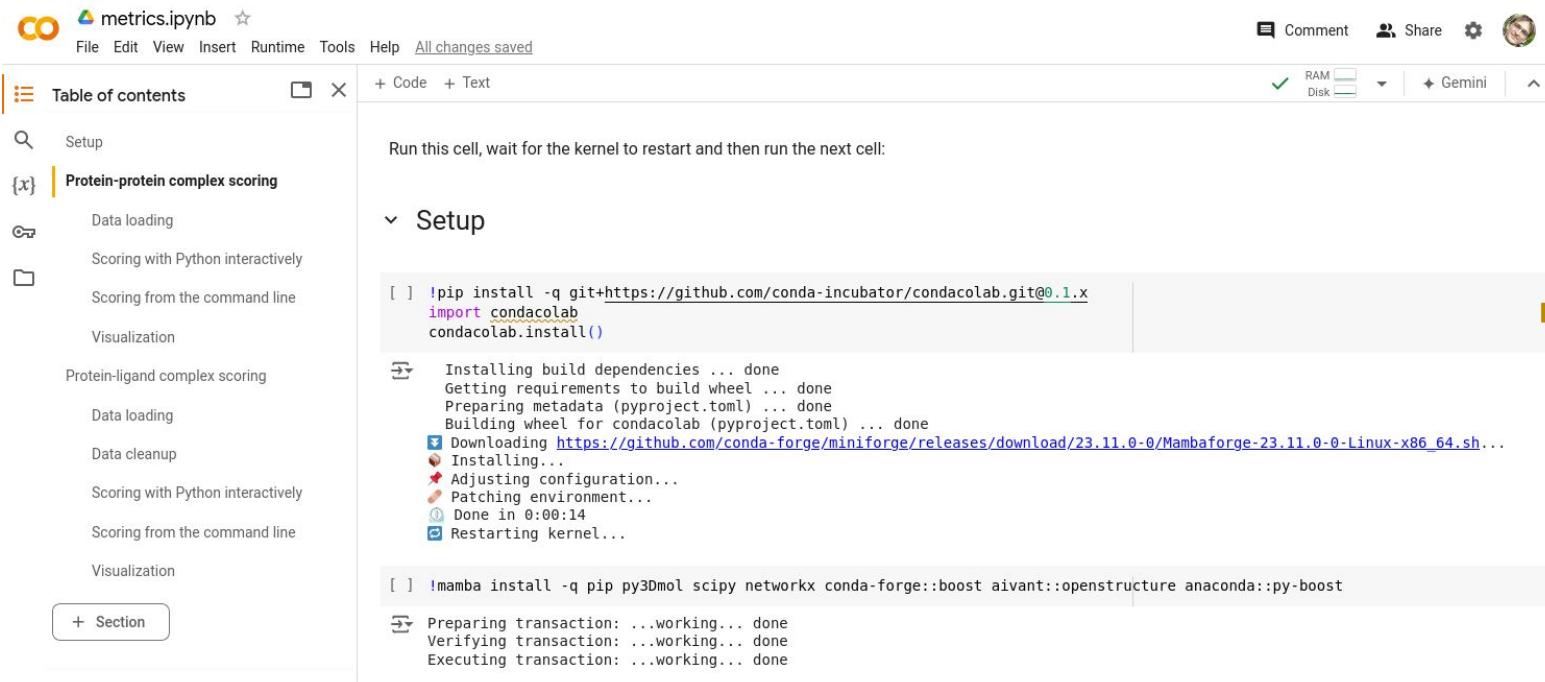
# Better datasets & splits = Better models

- ❖ Bigger more diverse training sets
- ❖ Assess val to stop training at the right time
- ❖ Augment training with conformational change
- ❖ Higher quality and diverse test sets
- ❖ Realistic inference with meaningful evaluation



# Hands on

Links available in the Discord chat and from GitHub



The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved.
- Toolbar:** Comment, Share, Settings, User profile.
- Table of Contents:** Shows sections like Setup, Protein-protein complex scoring, Data loading, Scoring with Python interactively, etc.
- Cell Content:** A cell with the instruction: "Run this cell, wait for the kernel to restart and then run the next cell:" followed by a collapsed section titled "Setup".
- Code Execution:** The cell contains the following command:

```
[ ] !pip install -q git+https://github.com/conda-incubator/condacolab.git@0.1.x
import condacolab
condacolab.install()
```

A detailed log of the command's execution follows:
  - Installing build dependencies ... done
  - Getting requirements to build wheel ... done
  - Preparing metadata (pyproject.toml) ... done
  - Building wheel for condacolab (pyproject.toml) ... done
  - Downloading [https://github.com/conda-forge/miniforge/releases/download/23.11.0-0/Mambaforge-23.11.0-0-Linux-x86\\_64.sh](https://github.com/conda-forge/miniforge/releases/download/23.11.0-0/Mambaforge-23.11.0-0-Linux-x86_64.sh)...
  - Installing...
  - Adjusting configuration...
  - Patching environment...
  - Done in 0:00:14
  - Restarting kernel...
- Another Cell:** Contains the command:

```
[ ] !mamba install -q pip py3Dmol scipy networkx conda-forge::boost aivant::openstructure anaconda::py-boost
```

A log of its execution:
  - Preparing transaction: ...working... done
  - Verifying transaction: ...working... done
  - Executing transaction: ...working... done

# Hands on

Links available in the Discord chat and from GitHub



The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved.
- Toolbar:** Comment, Share, Settings, User icon.
- Table of Contents:** Shows sections like Protein-protein complex scoring, Data loading, Scoring with Python interactively, etc.
- Current Section:** Protein-protein complex scoring > Setup.
- Code Cell (Setup):** Contains the following code:

```
!git clone https://github.com/conda-incubator/condacolab.git@0.1.x
cd condacolab
!lab install()
```
- Output of the Code Cell:** Shows the execution process:

```
Cloning into 'condacolab'...
remote: Counting objects: 10, done.
remote: Total 10 (delta 0), reused 0 (delta 0), pack-reused 10
Unpacking objects: 100% (10/10), done.
Building wheel for condacolab (pyproject.toml) ... done
Building wheel for condacolab (pyproject.toml) ... done
  Downloading https://github.com/conda-forge/miniforge/releases/download/23.11.0-0/Mambaforge-23.11.0-0-Linux-x86_64.sh...
  Installing...
  Adjusting configuration...
  Patching environment...
  Done in 0:00:14
  Restarting kernel...
```
- Another Code Cell:** Contains the command:

```
[ ] !mamba install -q pip py3Dmol scipy networkx conda-forge::boost aivant::openstructure anaconda::py-boost
```
- Output of the Second Code Cell:** Shows the transaction process:

```
Preparing transaction: ...working... done
Verifying transaction: ...working... done
Executing transaction: ...working... done
```

# Hands on

Links available in the Discord chat and from GitHub

The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved.
- Table of contents:** Protein-protein complex scoring, Data loading, Scoring with Python interactively, Scoring from the command line, Visualization.
- Protein-protein complex scoring section:** Sub-sections include Data loading, Scoring with Python interactively, Scoring from the command line, Visualization.
- Setup section:** Contains the following code cell output:

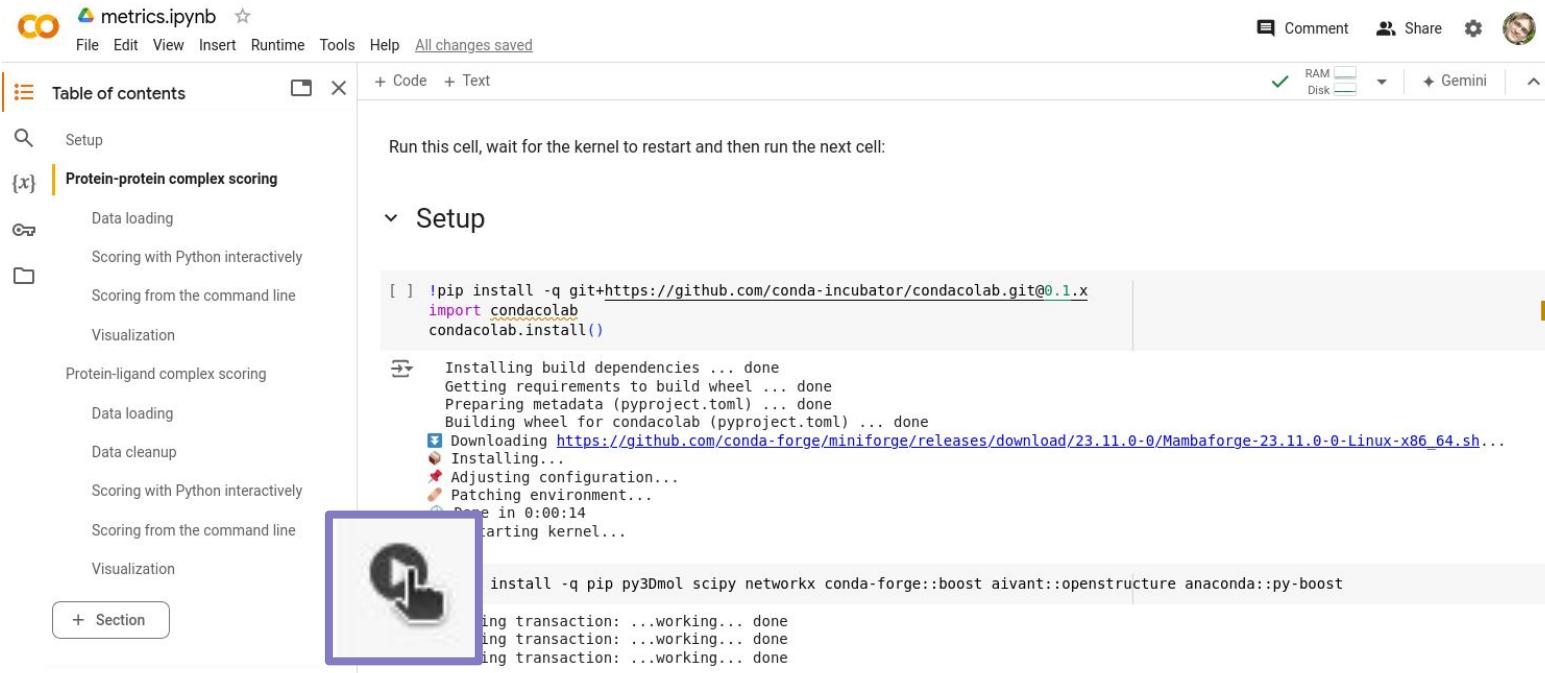
```
[ ] !pip install -q git+https://github.com/conda-incubator/condacolab.git@0.1.x
import condacolab
condacolab.install()
```

Output of the code cell:

  - Installing build dependencies ... done
  - Getting requirements to build wheel ... done
  - Preparing metadata (pyproject.toml) ... done
  - Building wheel for condacolab (pyproject.toml) ... done
  - Downloading [https://github.com/conda-forge/miniforge/releases/download/23.11.0-0/Mambaforge-23.11.0-0-Linux-x86\\_64.sh](https://github.com/conda-forge/miniforge/releases/download/23.11.0-0/Mambaforge-23.11.0-0-Linux-x86_64.sh)...
  - Installing...
  - Adjusting configuration...
  - Patching environment...
  - Done in 0:00:14
  - Restarting kernel...
- Runtime Logs:** Your session crashed for an unknown reason. [View runtime logs](#)
- Status Indicator:** A large green checkmark icon is displayed at the bottom right.

# Hands on

Links available in the Discord chat and from GitHub



The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** metrics.ipynb, All changes saved.
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help.
- Sidebar:** Table of contents, Setup, Protein-protein complex scoring (selected), Data loading, Scoring with Python interactively, Scoring from the command line, Visualization, Protein-ligand complex scoring, Data loading, Data cleanup, Scoring with Python interactively, Scoring from the command line, Visualization, + Section.
- Code Cell:** Run this cell, wait for the kernel to restart and then run the next cell:

```
[ ] !pip install -q git+https://github.com/conda-incubator/condacolab.git@0.1.x
import condacolab
condacolab.install()
```

Installing build dependencies ... done  
Getting requirements to build wheel ... done  
Preparing metadata (pyproject.toml) ... done  
Building wheel for condacolab (pyproject.toml) ... done  
Downloading [https://github.com/conda-forge/miniforge/releases/download/23.11.0-0/Mambaforge-23.11.0-0-Linux-x86\\_64.sh](https://github.com/conda-forge/miniforge/releases/download/23.11.0-0/Mambaforge-23.11.0-0-Linux-x86_64.sh)...  
Installing...  
Adjusting configuration...  
Patching environment...  
Done in 0:00:14  
Starting kernel...

```
install -q pip py3Dmol scipy networkx conda-forge::boost aivant::openstructure anaconda::py-boost

ing transaction: ...working... done
ing transaction: ...working... done
ing transaction: ...working... done
```
- Right Panel:** Comment, Share, Gemini, RAM/Disk status.