RAPID PROTEIN STABILITY PREDICTION USING DEEP LEARNING REPRESENTATIONS

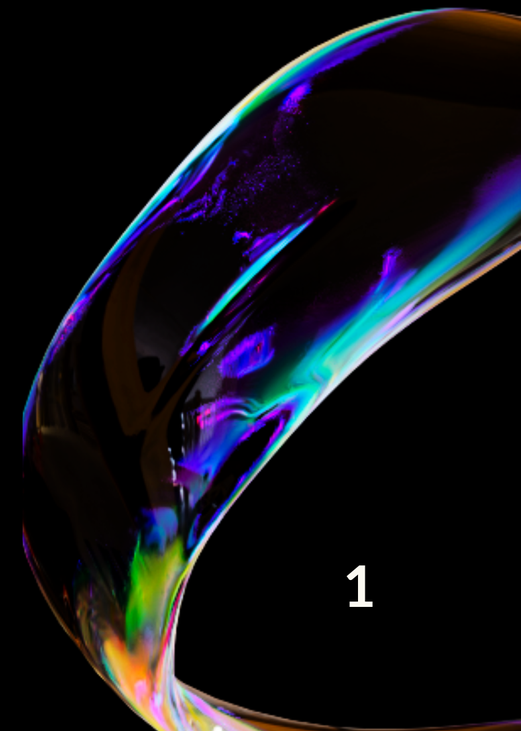
(RaSP)

Bio yapper

Beginning

and

Related tools



Calculate Protein Protein ΔΔG

$$\Delta\Delta G = \Delta G_{\text{mutant}} - \Delta G_{\text{wild type}}$$

where:

ΔG_mutant is the Gibbs free energy after the mutation.

ΔG_wild type is the Gibbs free energy before the mutation.

Many software tools can introduce mutations in silico, e.g., PyMOL, FoldX, or Rosetta.

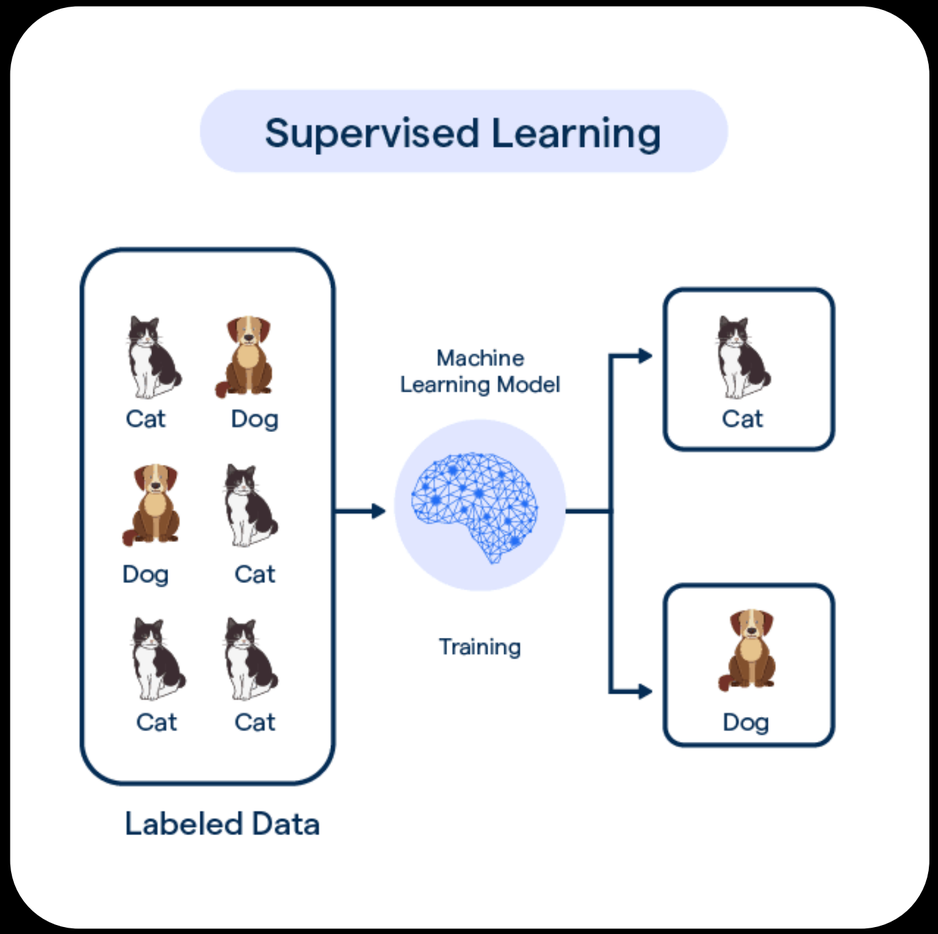
Types of Machine learning

Machine learning models have also been developed to predict changes in protein stability and can roughly be split into two types:

- supervised models
- self-supervised models.

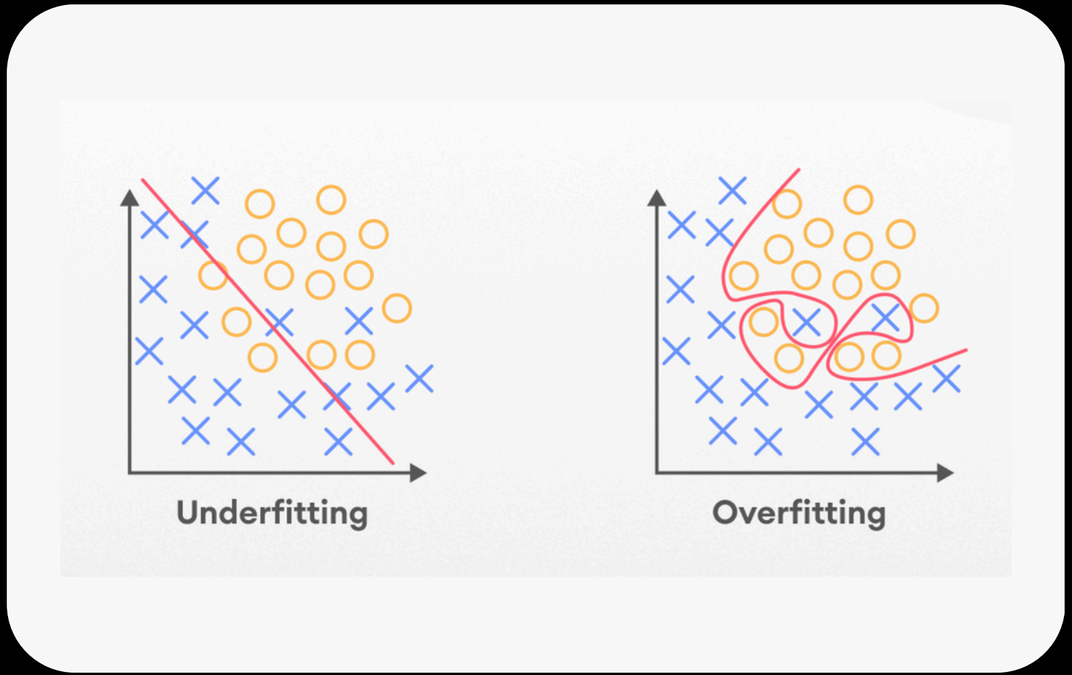
supervised models

are immediately appealing as they are trained directly on experimental data and are able to make predictions at the correct absolute scale.



a type of machine learning where the model is trained on labeled data

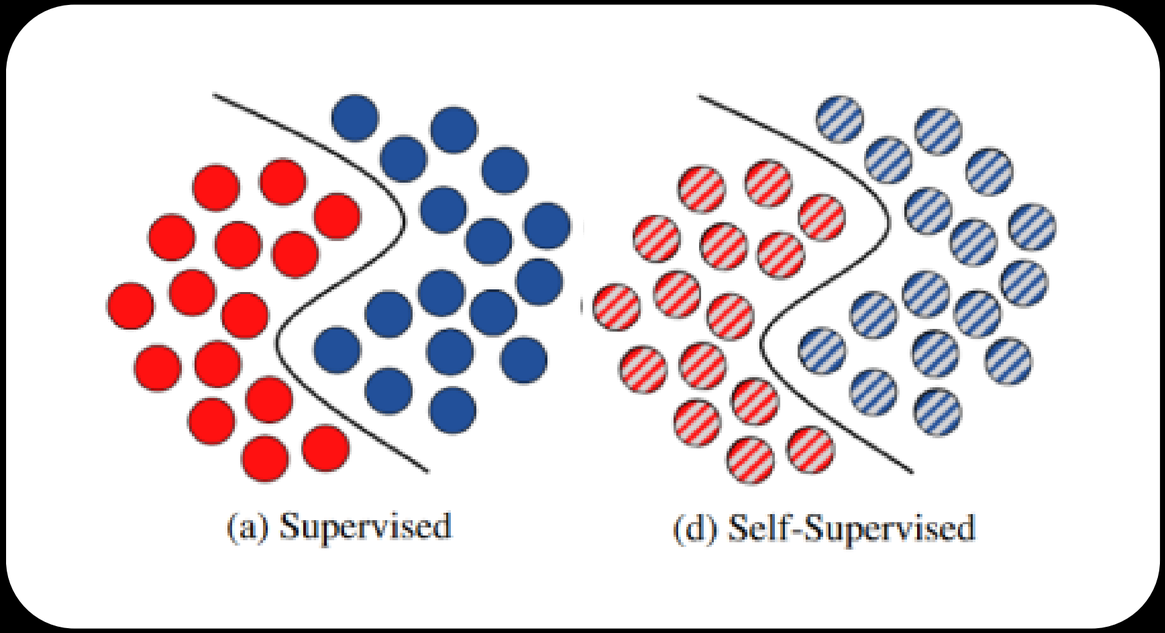
Overfitting of the training data



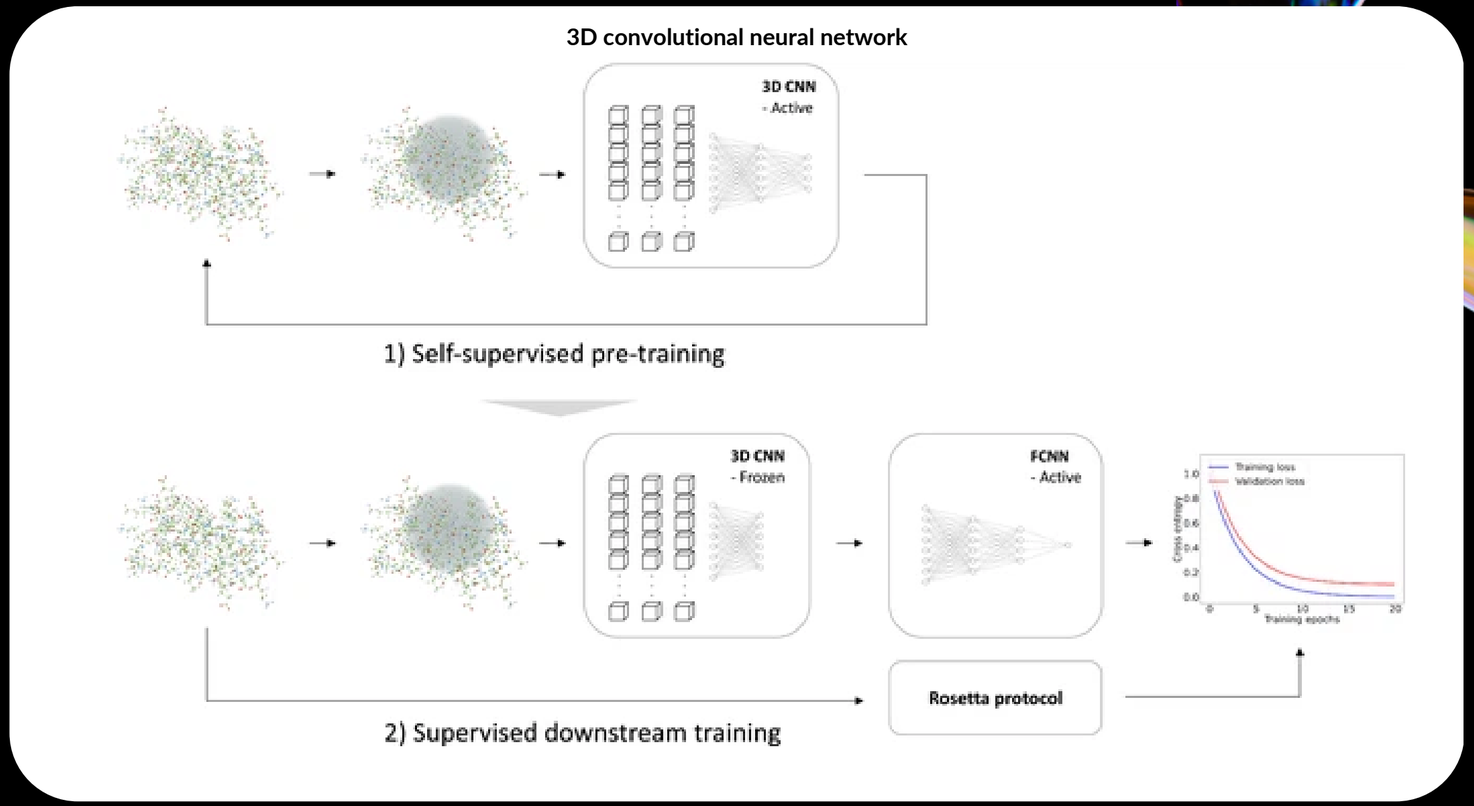
The training data size is too small and does not contain enough data samples to accurately represent all possible input data values.

Self-supervised models

In contrast, self-supervised models can be trained without the use of experimental protein stability measurements and are trained to predict masked amino acid labels from structure or sequence information, thereby learning a likelihood distribution over possible amino acid types

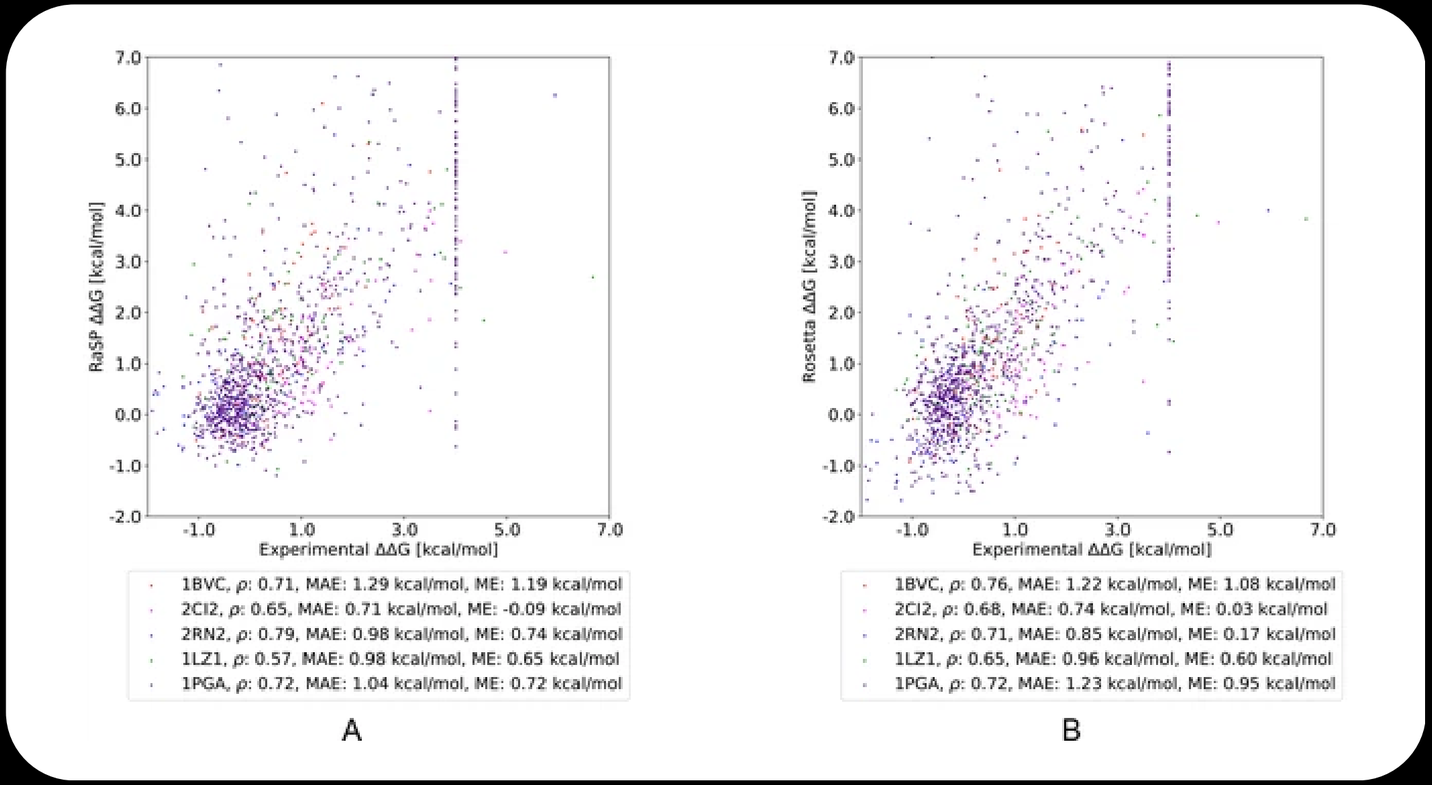


Trained the RaSP model



We trained the RaSP model in two steps. First, we trained a self-supervised representation model to learn an internal representation of protein structure. Second, we trained a supervised downstream model using the learned structure representation as input to predict protein stability changes on an absolute scale.

Comparing RaSP and Rosetta predictions to experimental stability measurements.



custom loss function to focus the model on ΔΔG values

Improvement Technique: Fine-tuning the RaSP model

Why

- The current model use the traning set purely from

Rosetta Protocol

How

- Use data set from other resources like AlphaFold2, FoldX, etc,.

FoldX5.1

Finding our dataset:

1. Create account for academic usuage at FoldX website.
2. Download the executable file.
3. Run the following command

./foldx –command=BuildModel –pdb=protein.pdb –mutant-file=./mutation

Chosen Protein: 5J5X

```
mkdir foldxtest
cd foldxtest
wget
https://files.rcsb.org/download/5J5X.pdb
ml FoldX
```

Mutation List:

```
LA224A;
LA224E;
LA224W;
IA339A;
IA339E;
IA339W;
```

Command:

```
./foldx –command=BuildModel –pdb=protein.pdb –mutant-file=./mutation
```

```
Starting BuildModel
Reading LA224A;
Reading LA224E;
Reading LA224W;
Reading IA339A;
Reading IA339E;
Reading IA339W;
Residue to Mutate LEUA224 has residue index 215
Residue to Mutate LEUA224 has residue index 215
Residue to Mutate LEUA224 has residue index 215
Residue to Mutate ILEA339 has residue index 330
Residue to Mutate ILEA339 has residue index 330
Residue to Mutate ILEA339 has residue index 330
Mutating residue = LEUA224 into ALA
Mutating residue = LEUA224 into GLU
Mutating residue = LEUA224 into TRP
Mutating residue = ILEA339 into ALA
Mutating residue = ILEA339 into GLU
Mutating residue = ILEA339 into TRP
Your file run OK
End time of FoldX: Thu Nov 14 13:20:41 2024
Total time spend: 46.24 seconds.
validated file "protein_1.pdb" => successfully finished
validated file "protein_2.pdb" => successfully finished
validated file "protein_3.pdb" => successfully finished
validated file "protein_4.pdb" => successfully finished
validated file "protein_5.pdb" => successfully finished
validated file "protein_6.pdb" => successfully finished
Cleaning BuildModel...DONE
PS C:\Users\tonwa\Downloads\foldx5Windows64>
```

Mutation	total energy	Solvation Polar	Solvation Hydrophobic	Van der Waals clashes
LA224A;	3.87343	0.0541713	3.35989	-0.0026094
LA224E;	5.63786	3.15016	1.57699	0.00600314
LA224W;	7.99308	1.42464	-0.522222	7.21608
IA339A;	-0.145273	-0.597786	0.436306	0
IA339E;	-0.176464	-0.54349	0.34954	1.37877E-07
IA339W;	0.235274	-0.282032	0.158089	0

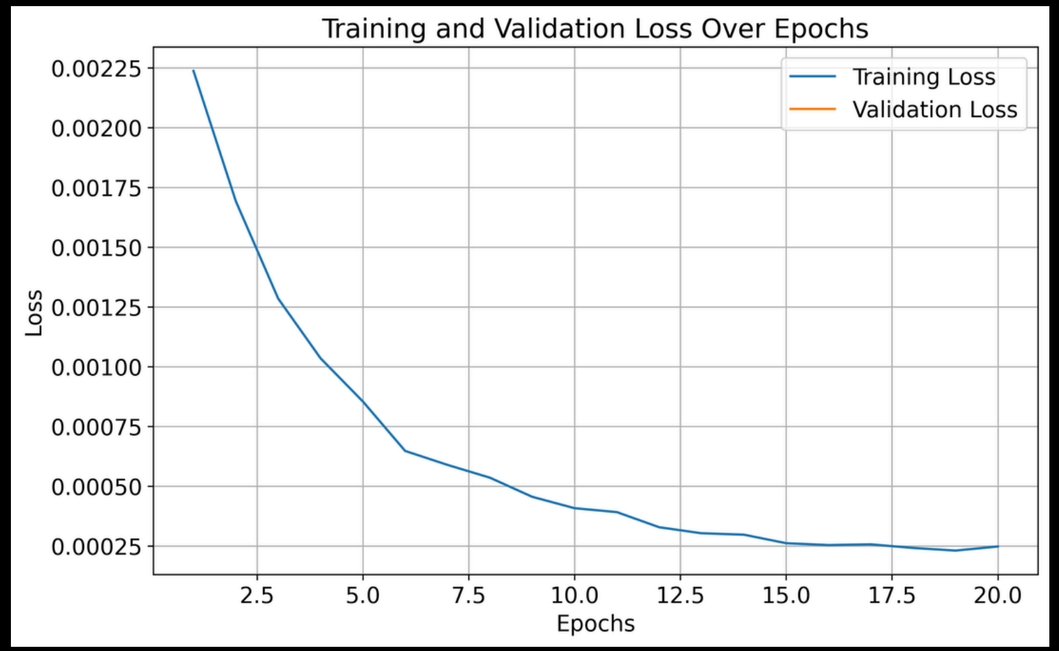
Our code:

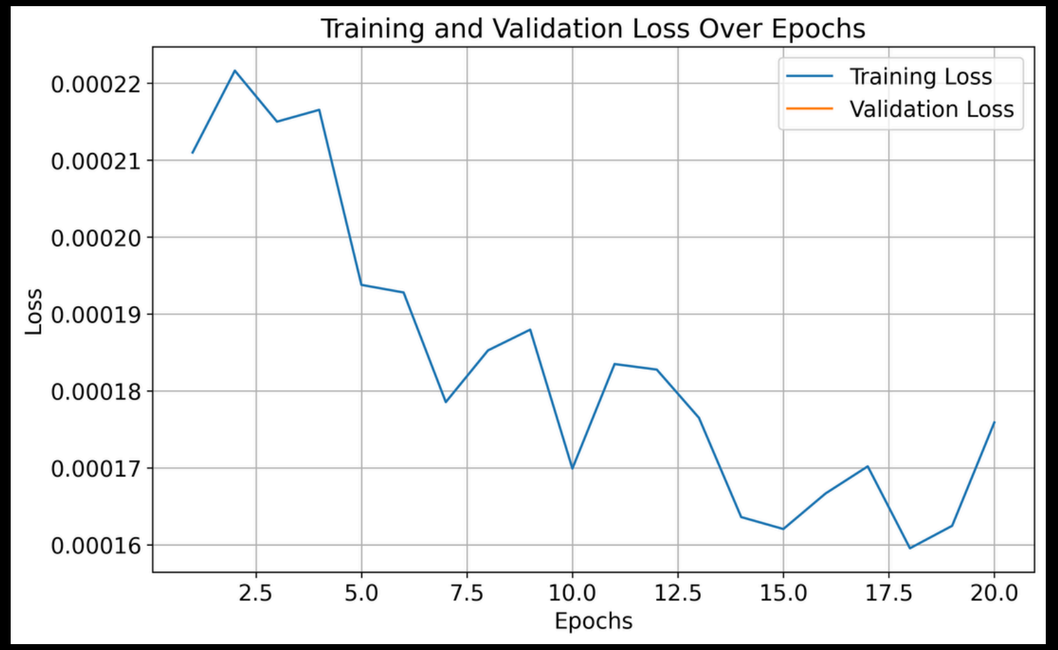
```python
for model_idx in range(NUM_ENSEMBLE):
  print(f"Training model: {model_idx+1}/{NUM_ENSEMBLE}")
  ds_train_val(
              df_structure,
              dataloader_train_ds,
              dataloader_val_ds,
              cavity_model_net,
              ds_model_net,
              loss_ds,
              optimizer_ds,
              model_idx,
              EPOCHS_DS,
              DEVICE,
          )
  print(f"Finished training model: {model_idx+1}/{NUM_ENSEMBLE}")
```

```
Training model: 1/3
Epoch: 1/20
Epoch: 2/20
Epoch: 3/20
Epoch: 4/20
Epoch: 5/20
Epoch: 6/20
Epoch: 7/20
```

https://github.com/KULL-Centre/_2022_ML-ddG-Blaabjerg/

Clear sign of overfitting

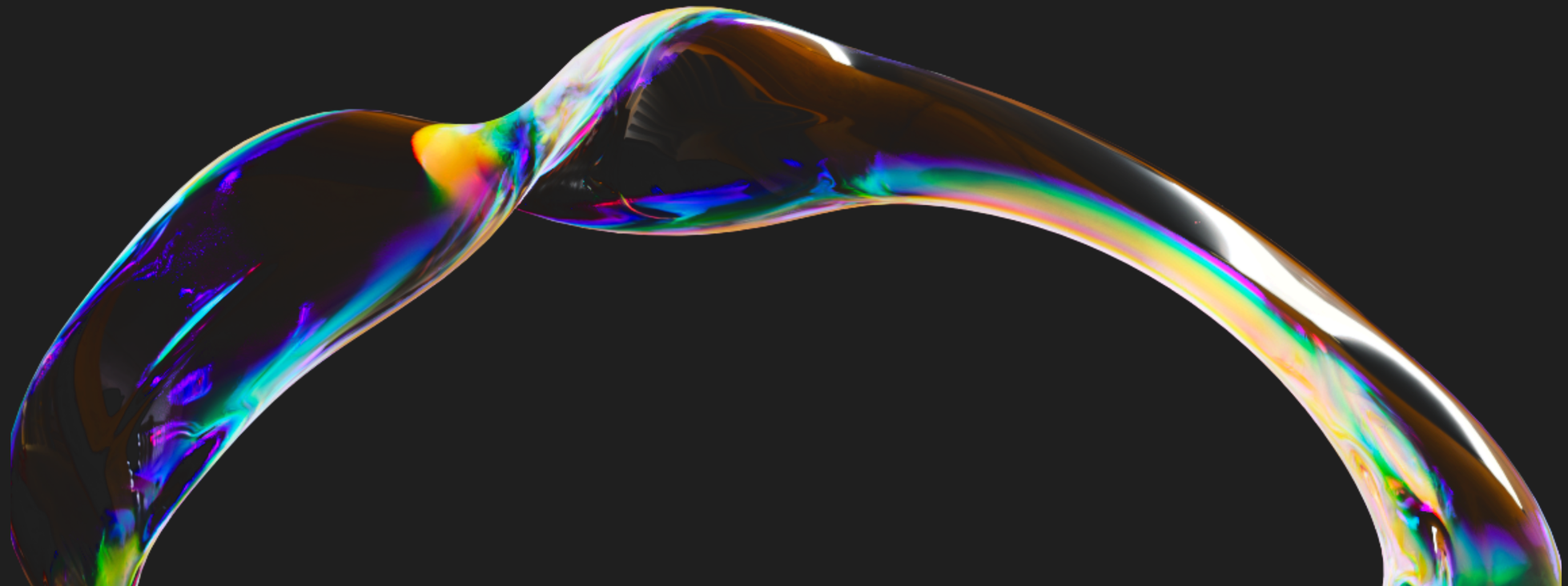




More overfit

EVALUATION

BIO YAPPER 2024



OUTPUT

```
# --------------------
# version: 1
# protein:
#     name: Unknown
#     organism: Unknown
#     uniprot: Unknown
#     sequence:
ATRRYYLGAVELSWDYMQSDLGELPVDARFPPRVPKSFPFNTSVVYKKTLFVEFTDHLFNIAKPRPPWMGLLGPTIQAEVYDTVVITLKNMASHPVSLHAVGVSYWKASEGAEYDDQTSQREKEDDKVFPGGSHTYVWQVLKENGPMASDPLCLTYS
YLSHVDLVKDLNSGLIGALLVCREGSLAKEKTQTLHKFILLFAVFDEGKSWHSETKNXXXXXXAASARAWPKMHTVNGYVNRSLPGLIGCHRKSVYWHVIGMGTTPEVHSIFLEGHTFLVRNHRQASLEISPITFLTAQTLLMDLGQFLLFCHISS
HQHDGMEAYVKVDSCPEEPQXXXXXXXXXXXXXXXXXXXXXXXXFDDDNSPSFIQIRSVAKKHPKTWVHYIAAEEEDWDYAPLVLAPDDRSYKSQYLNNGPQRIGRKYKKVRFMAYTDETFKTREAIQHESGILGPLLYGEVGDTLLIIFKNQASR
PYNIYPHGITDVRPLYSRRLPKGVKHLKDFPILPGEIFKYKWTVTVEDGPTKSDPRCLTRYYSSFVNMERDLASGLIGPLLICYKESVDQRGNQIMSDKRNVILFSVFDENRSWYLTENIQRFLPNPAGVQLEDPEFQASNIMHSINGYVFDSLQLS
VCLHEVAYWYILSIGAQTDFLSVFFSGYTFKHKMVYEDTLTLFPFSGETVFMSMENPGLWILGCHNSDFRNRGMTALLKVSSCDKNTGDYYEDSYED
#     pdb: 2R7E
#     chain: A
# cavity:
#     version: 1
# created: {'2024-11-13 04:35 (CPH time) - lasse.blaabjerg@bio.ku.dk'}
# variants:
#     number: 13860
#     coverage: 0.956
#     depth: 20.000
#     width: single mutants
# columns:
#     score_ml_fermi: Normalized cavity model ddG predictions (range [0;1])
#     score_ml: Cavity model ddG predictions
# --------------------
#
#
variant score_ml_fermi score_ml
A1= 0.2458501011133194 0.1978271299992318
A1C 0.2708051800727844 0.5236468265687355
A1D 0.2242480367422104 -0.10270021996113654
A1E 0.25118592381477356 0.26925668485862253
A1F 0.24029609560966492 0.1223578418666138
A1G 0.24298489093780518 0.15904012956790392
A1H 0.21854448318481445 -0.18542162448673294
A1I 0.27649804949760437 0.59525131171275
A1K 0.24689644575119019 0.21191565040101445
```

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	pdbid	chainid	variant	score	wt_idx	mt_idx	wt_nlf	mt_nlf	score_ml_	score_ml	pos	wt_AA	mt_AA
2	1D5R	A	R14A	0.056322	14	0	3.035567	2.483982	0.284613	0.695761	14	R	A
3	1D5R	A	R14C	1.177701	14	1	3.035567	4.33514	0.318802	1.10179	14	R	C
4	1D5R	A	R14D	0.803333	14	2	3.035567	2.82468	0.276293	0.592693	14	R	D
5	1D5R	A	R14E	0.743103	14	3	3.035567	2.712832	0.276301	0.592792	14	R	E
6	1D5R	A	R14F	1.582414	14	4	3.035567	3.207937	0.224495	-0.09915	14	R	F
7	1D5R	A	R14G	0.408161	14	5	3.035567	2.560213	0.319013	1.104225	14	R	G
8	1D5R	A	R14H	0.096667	14	6	3.035567	3.772712	0.234347	0.040187	14	R	H
9	1D5R	A	R14I	0.575517	14	7	3.035567	2.854018	0.262529	0.417834	14	R	I
10	1D5R	A	R14K	0.885517	14	8	3.035567	2.828313	0.27497	0.576127	14	R	K
11	1D5R	A	R14L	-0.64	14	9	3.035567	2.433114	0.220401	-0.15833	14	R	L
12	1D5R	A	R14M	0.095747	14	10	3.035567	3.785957	0.245574	0.194108	14	R	M
13	1D5R	A	R14N	0.437931	14	11	3.035567	3.127102	0.268959	0.50022	14	R	N
14	1D5R	A	R14P	-0.08506	14	12	3.035567	3.080555	0.453649	2.535159	14	R	P
15	1D5R	A	R14Q	0.313103	14	13	3.035567	3.338515	0.2766	0.596525	14	R	Q
16	1D5R	A	R14S	0.405517	14	15	3.035567	2.844502	0.286328	0.716787	14	R	S
17	1D5R	A	R14T	1.328621	14	16	3.035567	2.896883	0.280678	0.647241	14	R	T
18	1D5R	A	R14V	0.903103	14	17	3.035567	2.623741	0.274369	0.568584	14	R	V
19	1D5R	A	R14W	1.424828	14	18	3.035567	4.250501	0.211391	-0.29141	14	R	W
20	1D5R	A	R14Y	1.481609	14	19	3.035567	3.31158	0.225831	-0.08002	14	R	Y
21	1D5R	A	R15A	0.318966	14	0	3.035567	2.483982	0.255817	0.330437	15	R	A
22	1D5R	A	R15C	0.853103	14	1	3.035567	4.33514	0.279655	0.634571	15	R	C
23	1D5R	A	R15D	-0.34368	14	2	3.035567	2.82468	0.225668	-0.08235	15	R	D
24	1D5R	A	R15E	0.744828	14	3	3.035567	2.712832	0.227009	-0.0632	15	R	E
25	1D5R	A	R15F	-0.0223	14	4	3.035567	3.207937	0.23423	0.038551	15	R	F
26	1D5R	A	R15G	1.190345	14	5	3.035567	2.560213	0.254646	0.31504	15	R	G

Methods of Evaluation

BIO YAPPER 2024

Mean Absolute Error (MAE)

More is Better

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_{\text{actual},i} - y_{\text{predicted},i}|$$

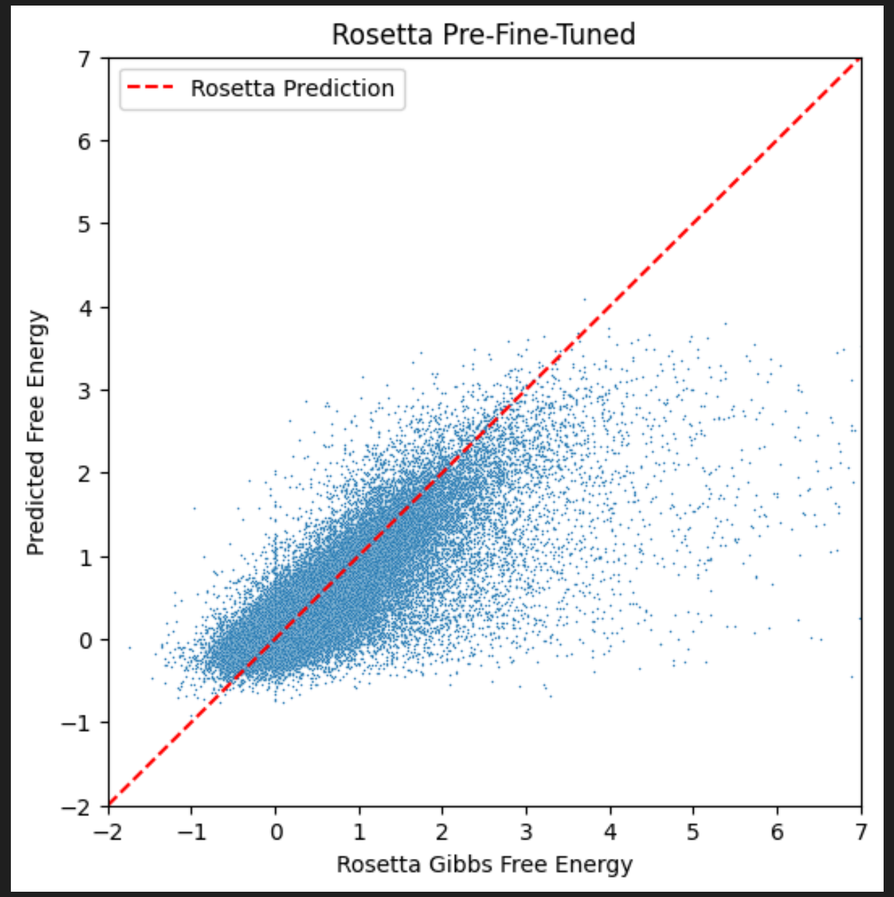
Pearson Correlation Coefficient (r)

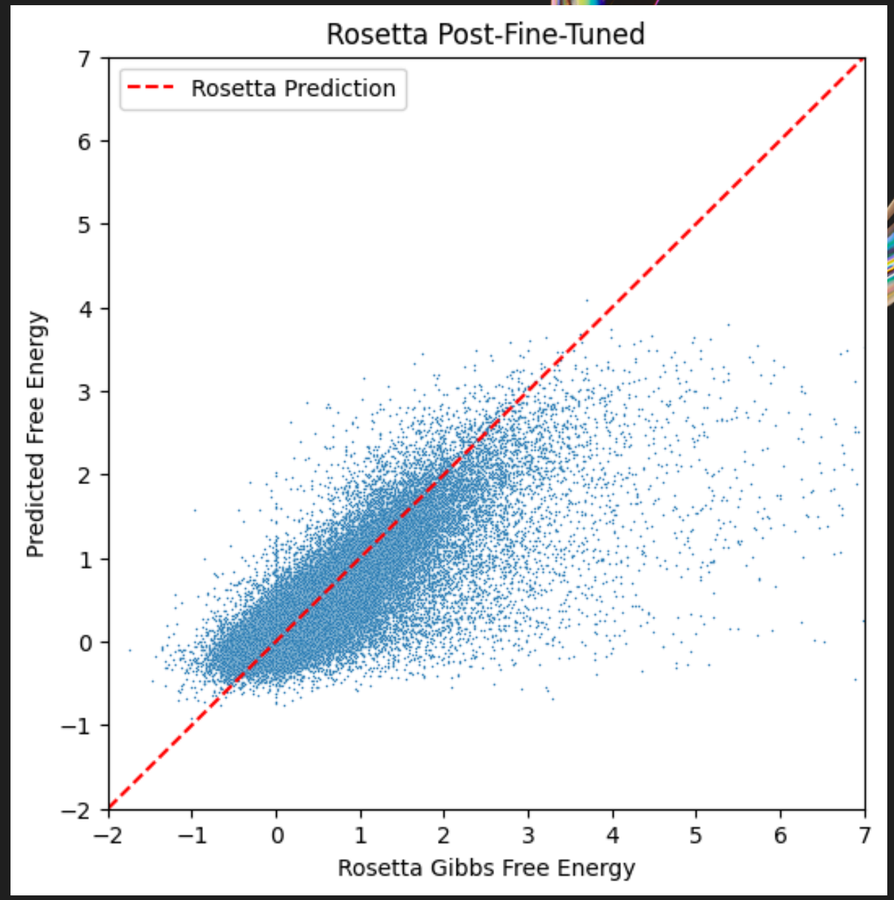
Closer to 1 is better

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Results on Rosetta

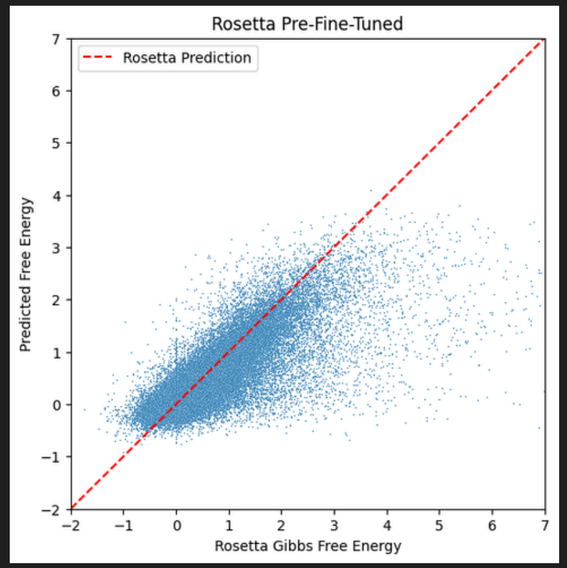
BIO YAPPER 2024

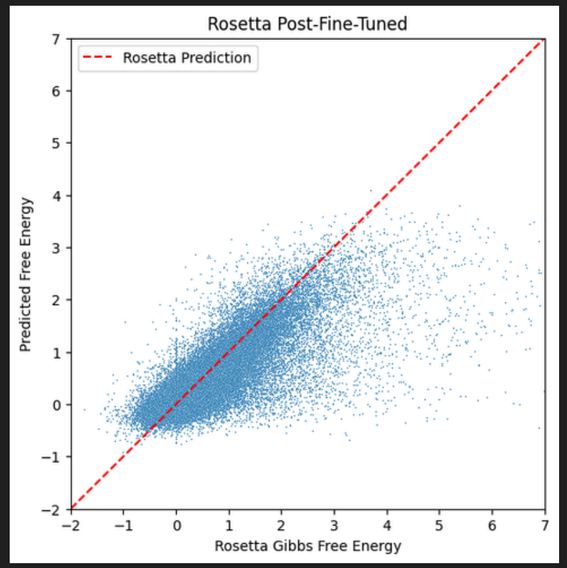




Results on Rosetta

BIO YAPPER 2024

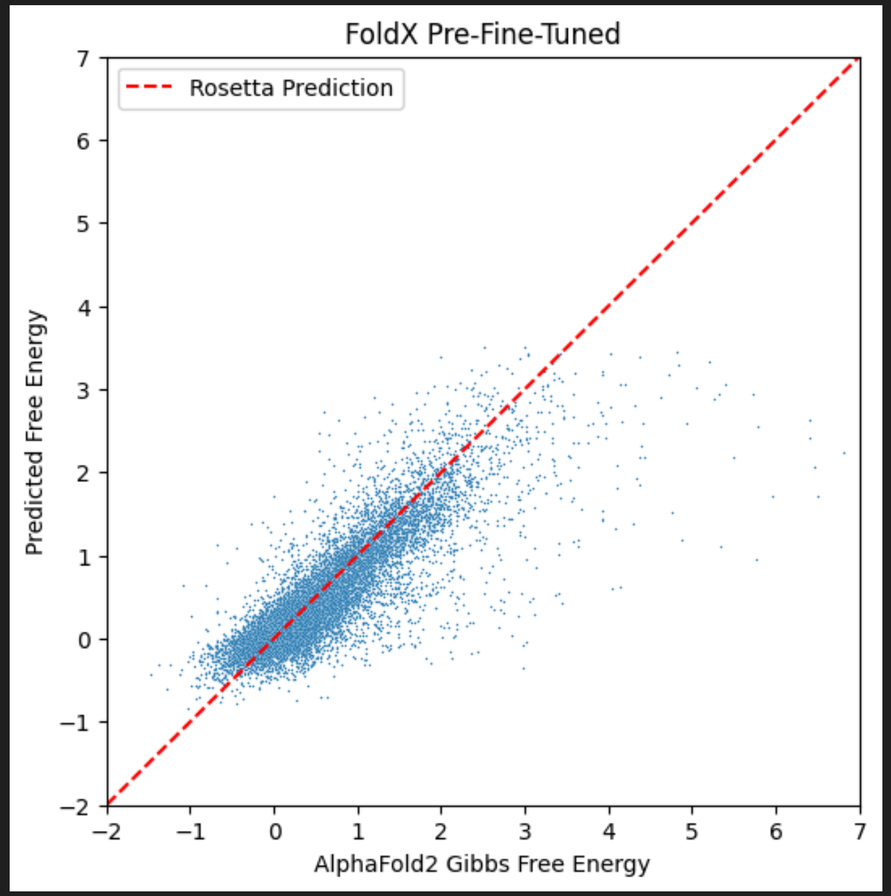


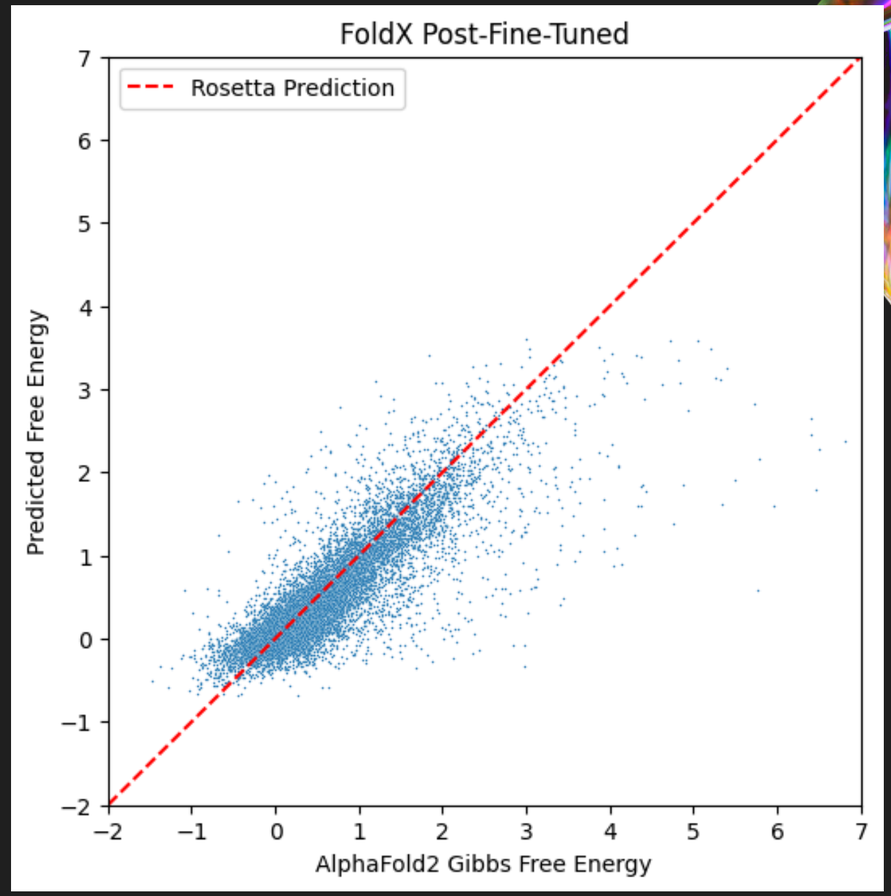


Mean Absolute Error (MAE) **Less** is Better	0.995	0.907
Pearson correlation coefficient (r) **C**loser to 1 is better	0.767	0.804

Results on FoldX

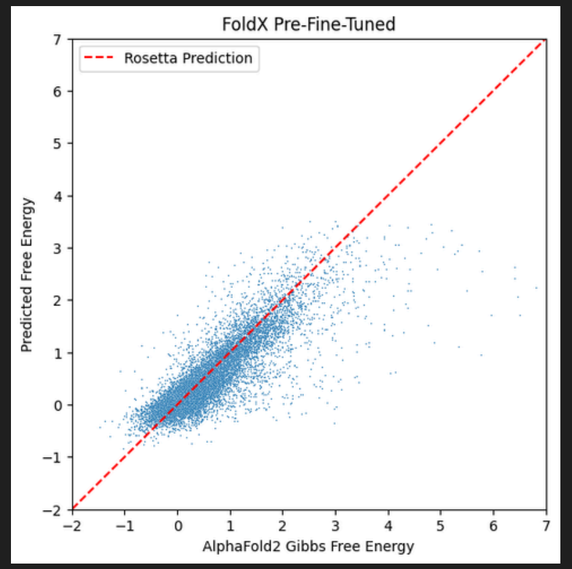
BIO YAPPER 2024

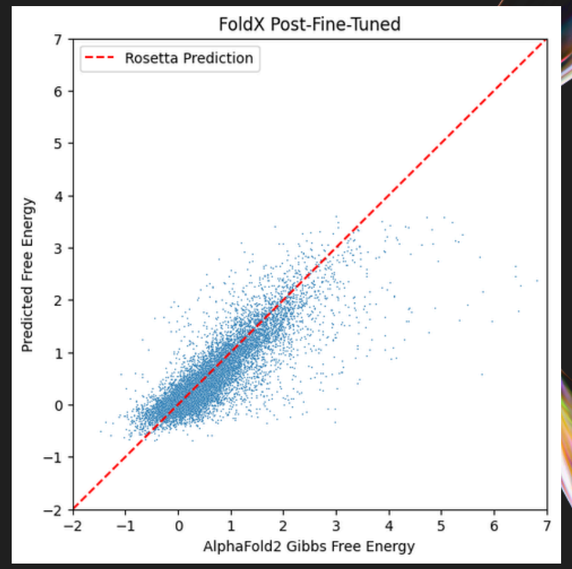




Results on FoldX

BIO YAPPER 2024





Mean Absolute Error (MAE) More is Better	0.815	0.667
Pearson correlation coefficient (r) **C**loser to 1 is better	0.639	0.840

Conclusion

BIO YAPPER 2024

POSSIBLE REASONS

- Potential Overfitting to FoldX Data
- Distribution Differences
- Inconsistent Data Between Protocols

THE NEXT STEPS

- Check Dataset Characteristics
- Domain Adaptation
- Undersampling/Oversampling

THANK YOU

for your time and attention

Present by Bio Yapper