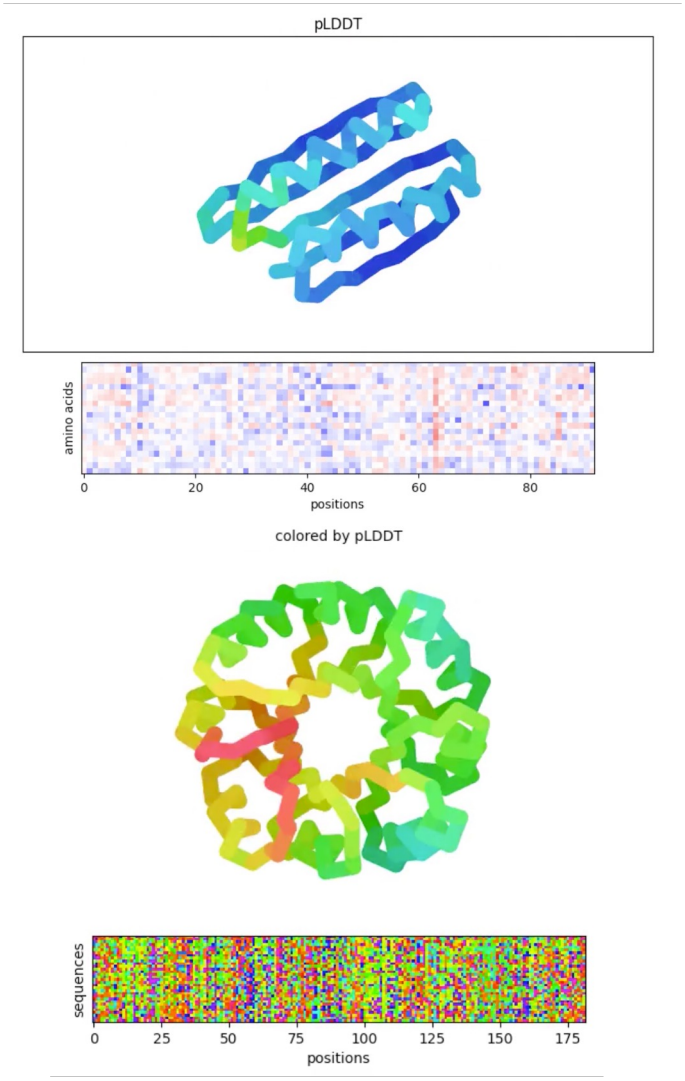
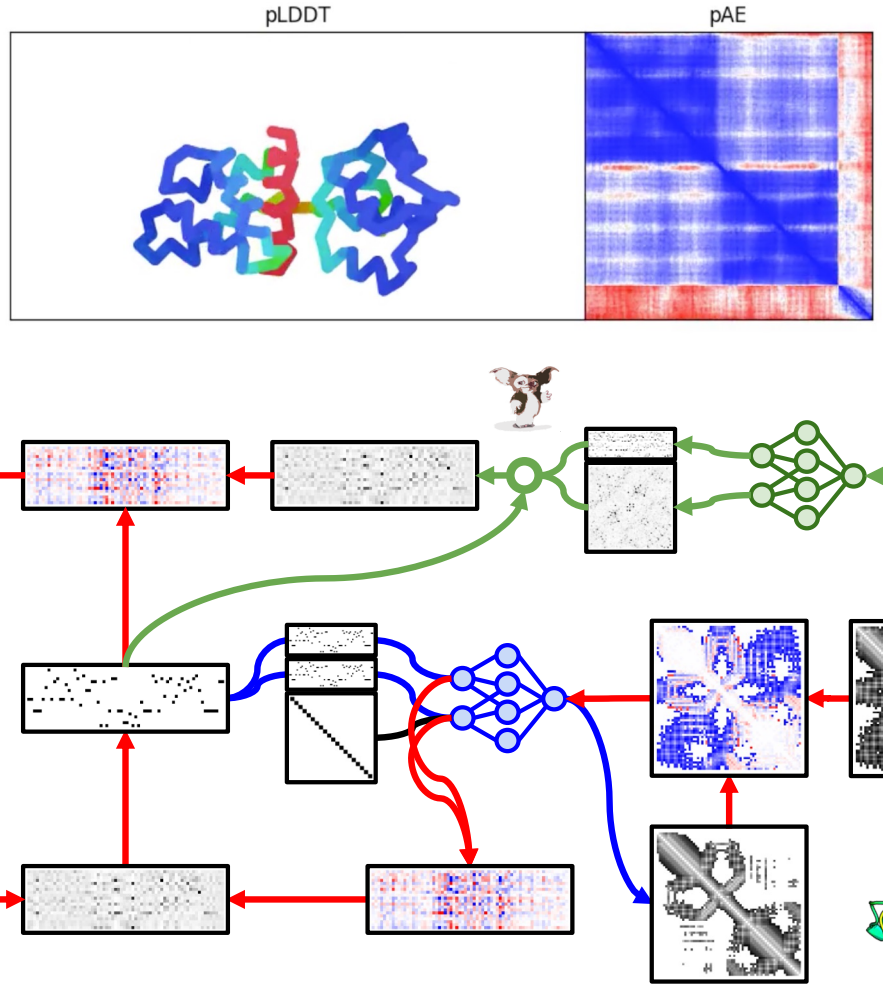
ColabDesign

Making protein design accessible to all via Google Colab



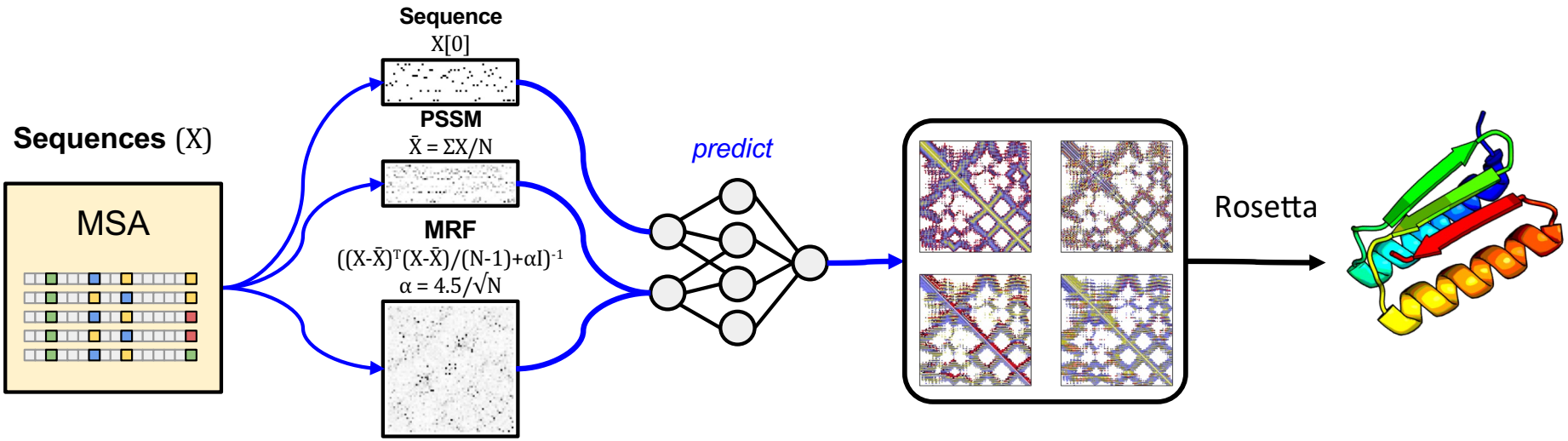


github.com/sokrypton/ColabDesign

Outline

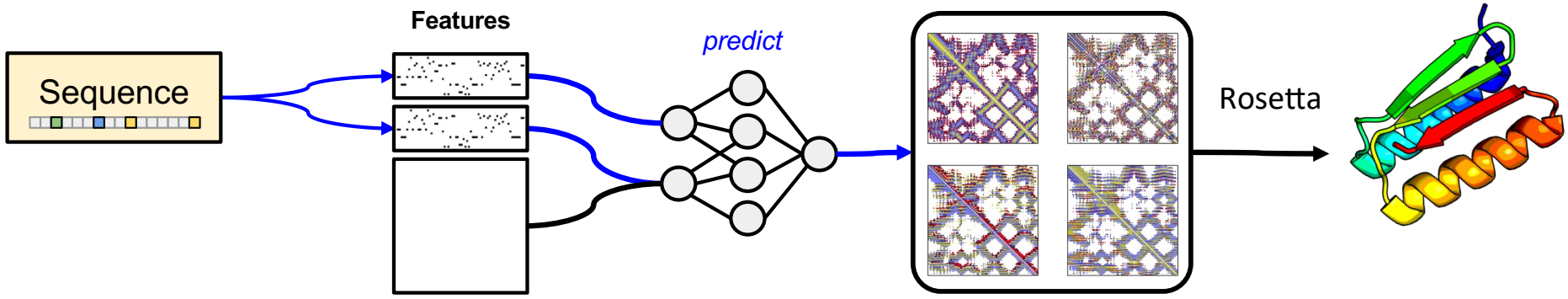
- Recap: TrRosetta
- TrDesign - Inverting TrRosetta for Protein Design
- TrMRF - Modeling P(seq|structure)
- Experimental Results
- Modern methods: RoseTTAFold & AlphaFold
- Jue Wang - Partial Hallucination
- AfDesign - Inverting AlphaFold for Protein Design

TrRosetta - Takes sequences and returns structure



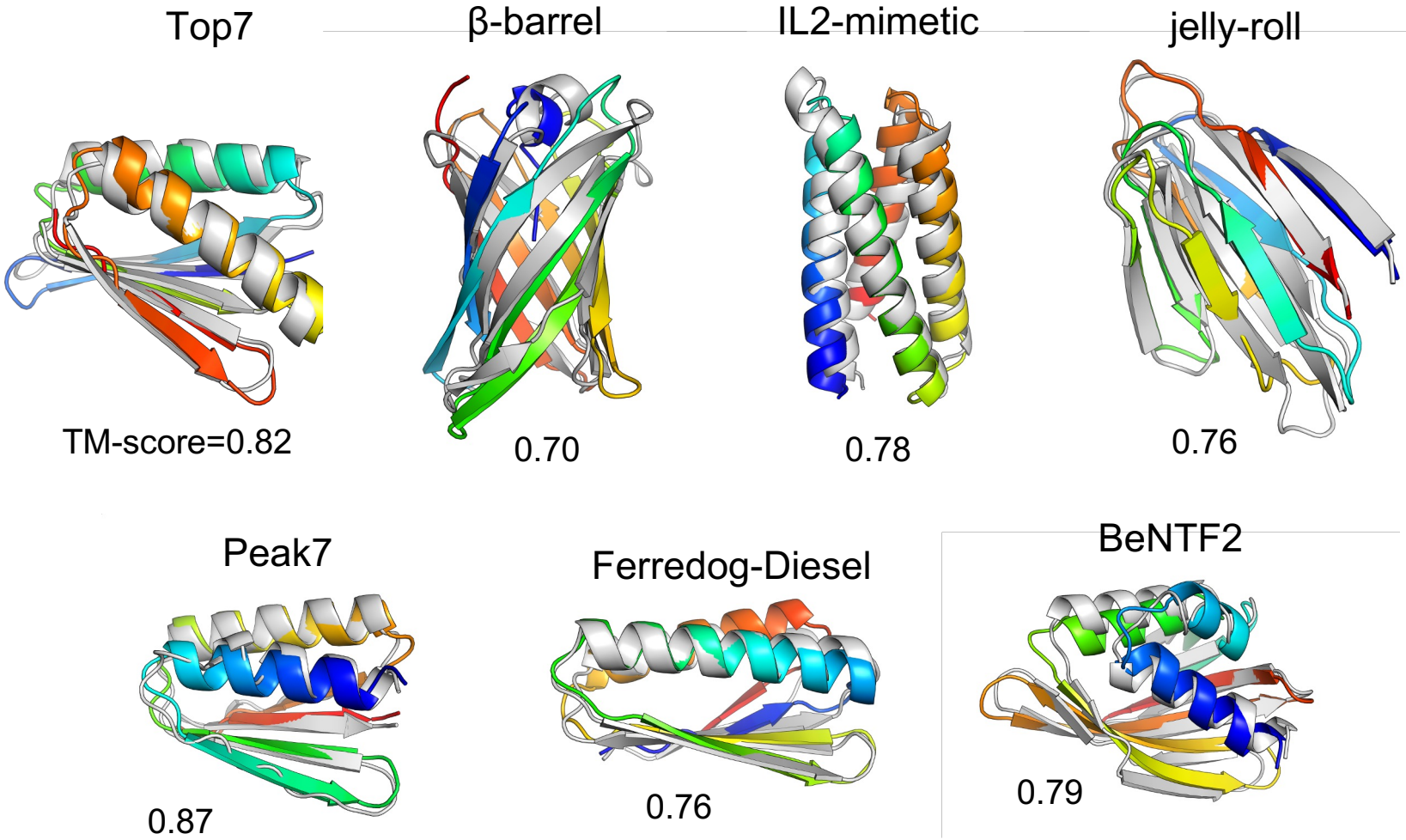
Yang J, Anishchenko I, Park H, Peng Z, Ovchinnikov S, Baker D. Improved protein structure prediction using predicted interresidue orientations. Proceedings of the National Academy of Sciences. 2020 Jan 2.

Using a single-sequence the method can accuractely predict **denovo designed** proteins



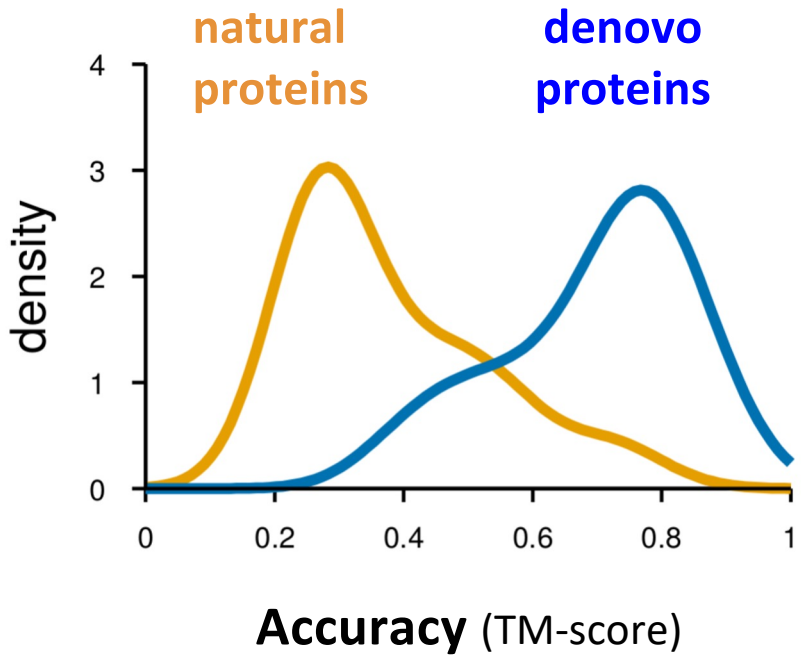
Yang J, Anishchenko I, Park H, Peng Z, Ovchinnikov S, Baker D. Improved protein structure prediction using predicted interresidue orientations. Proceedings of the National Academy of Sciences. 2020 Jan 2.

Using a single-sequence the method can accuractely predict **denovo designed** proteins

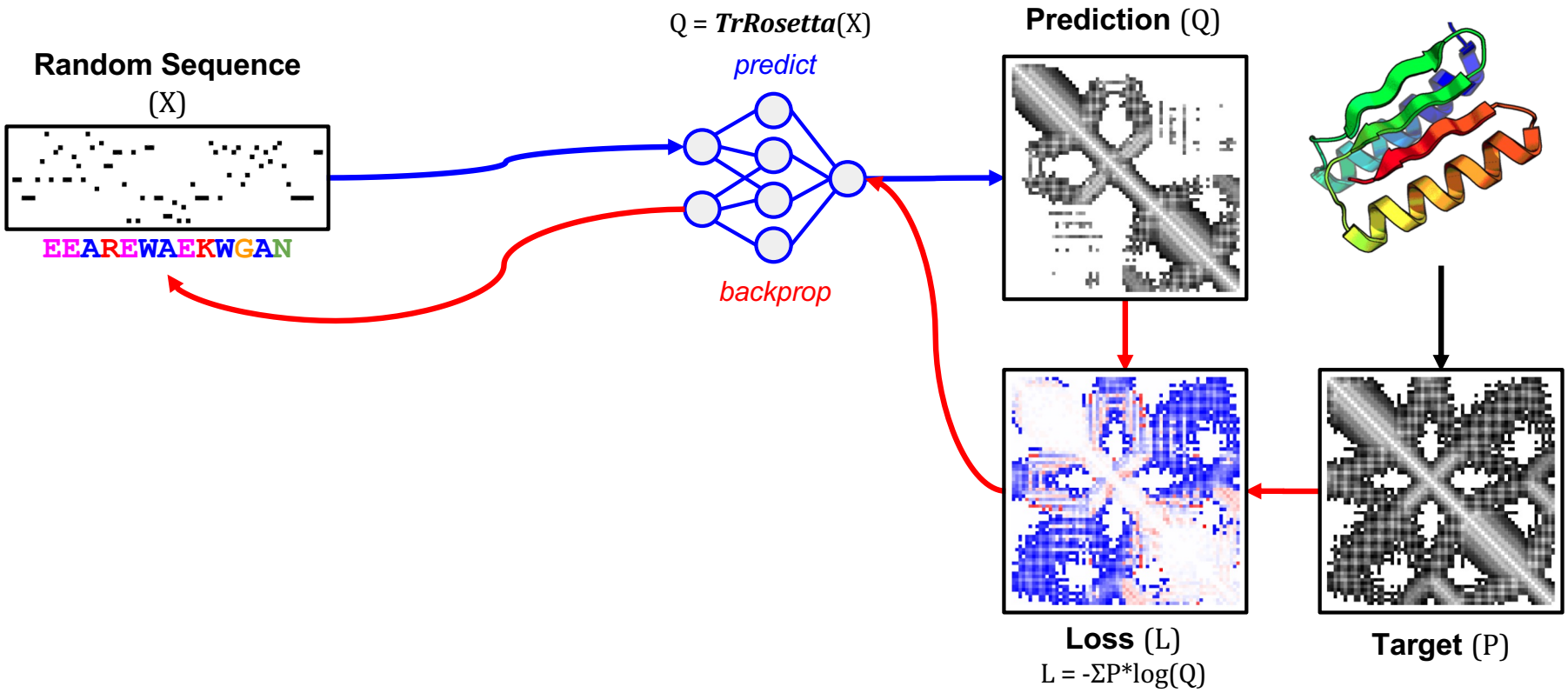


Slide modified from
Ivan Anishchenko

Using a single-sequence the method can accuractely predict **denovo designed** proteins but not **natural proteins**



Can we invert TrRosetta for Protein Design?

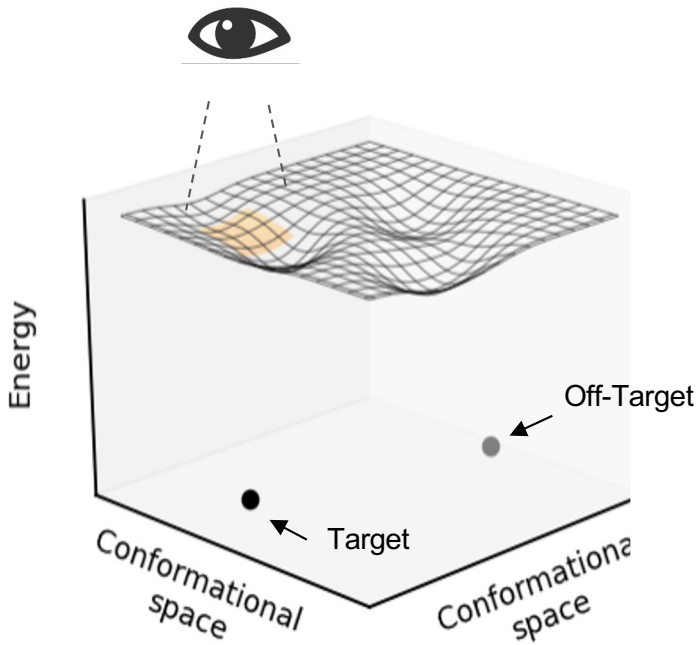


But how is this different from other models that return sequence given structure?

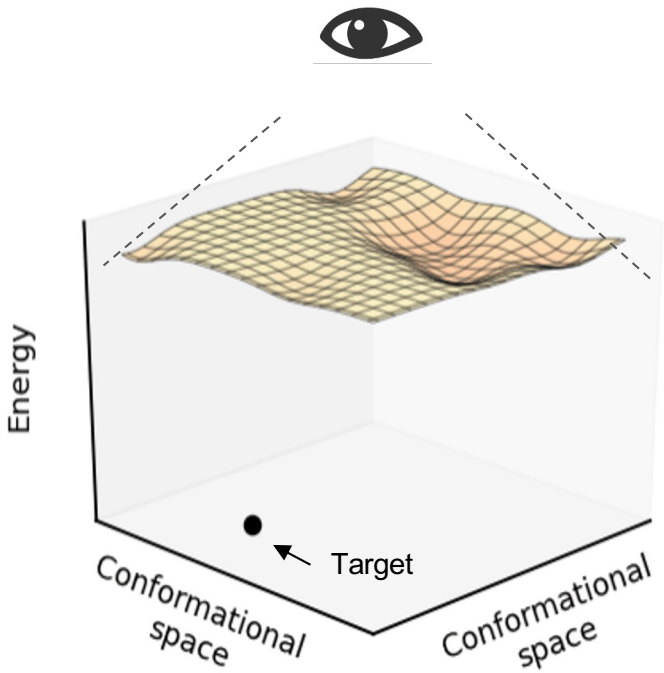
They do not **"see"** the alternative conformation the sequence can adopt!

Explicit conformational landscape optimization

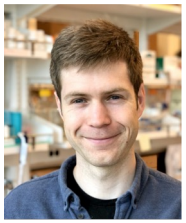
P(sequence | structure)



P(structure | sequence)



Protein sequence design by conformational landscape optimization.
Norn, C., Wicky, B.I., Juergens, D., Liu, S., Kim, D., Tischer, D., Koepnick, B., Anishchenko, I., Baker, D. and Ovchinnikov, S., 2021. *Proceedings of the National Academy of Sciences*, *118*(11).

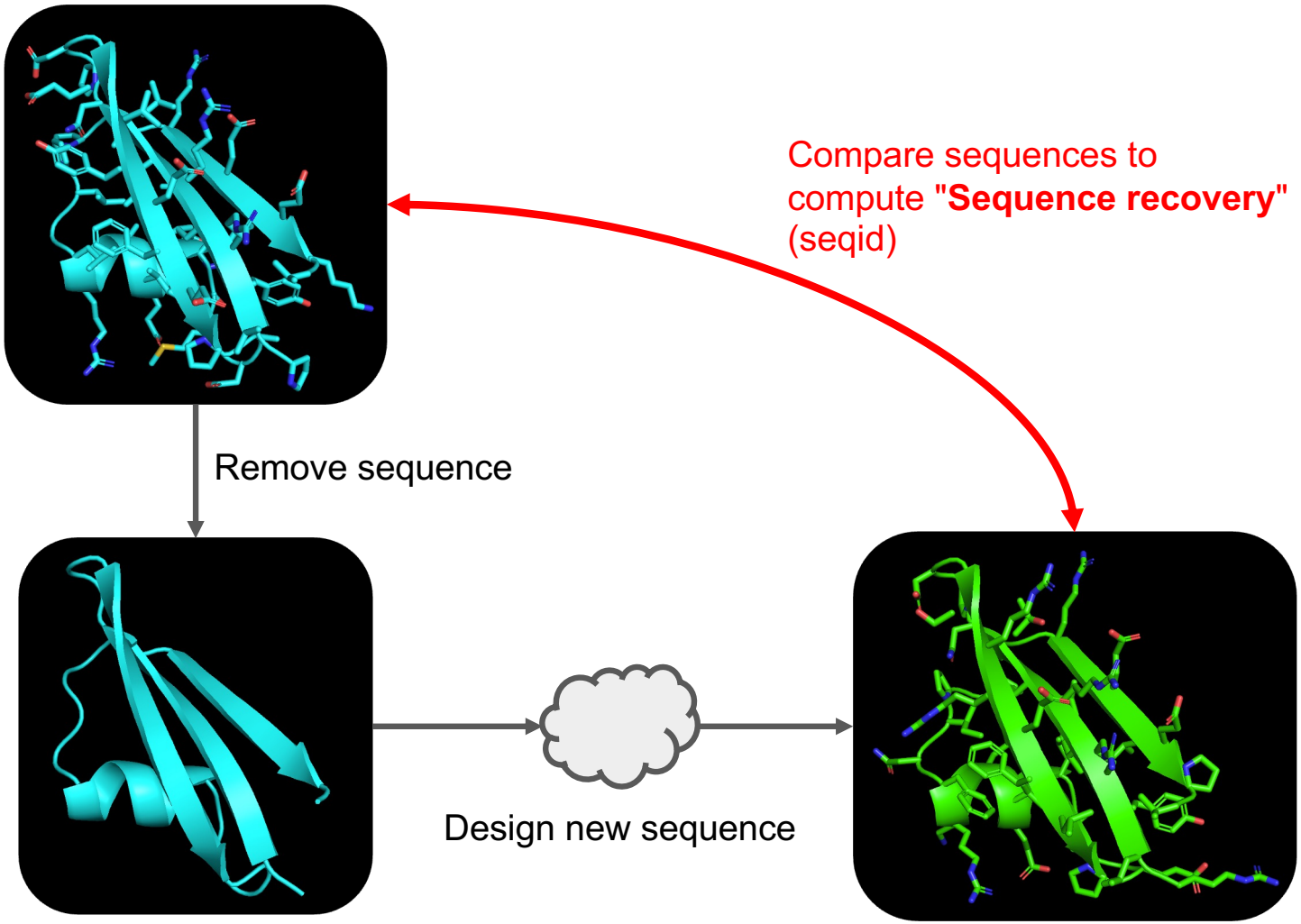


Chris Norn

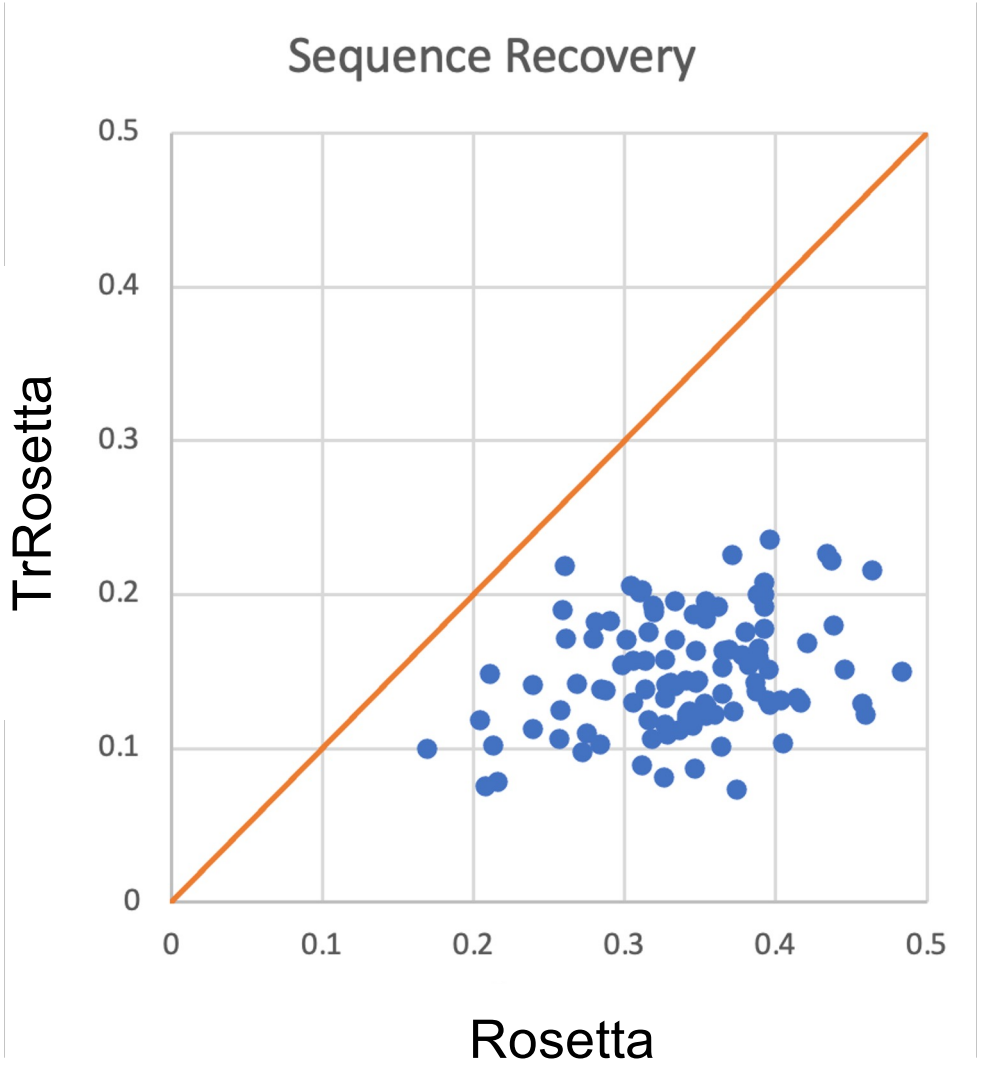
Basile Wicky

Experiment - for given backbone design new sequence

- For ~200 proteins from the PDB not in the TrRosetta training set, we design a new sequence using Rosetta (as is) and TrRosetta.



TrRosetta Sequence recovery is too low compared to Rosetta



Why is the sequence recovery so low?

- Hypothesis 1 - adversarial sequences
- Hypothesis 2 - the model is low resolution
- Hypothesis 3 - the model only sees a subset of key amino acids
- Hypothesis 4 - we are modeling P(structure | sequence) w/o P(sequence)
- Hypothesis 5 - natural backbones require MSA to design

H1 - adversarial sequences

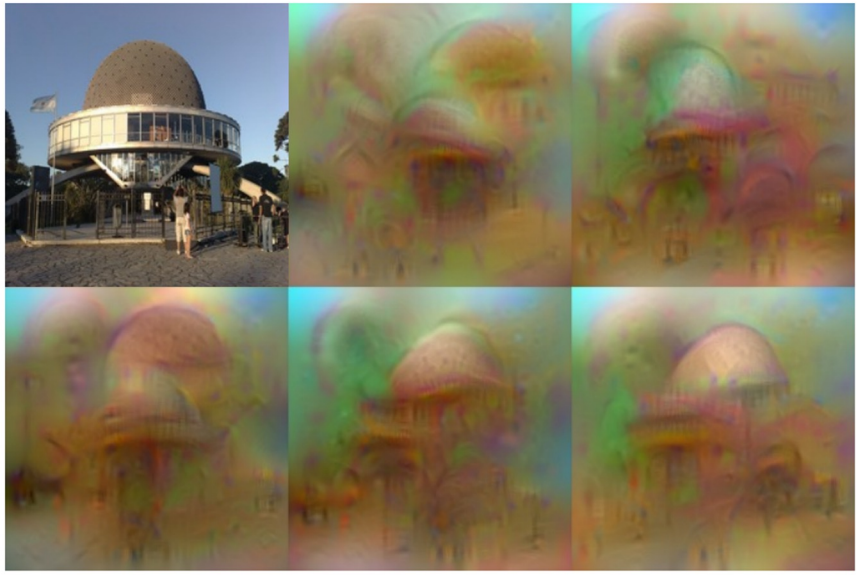
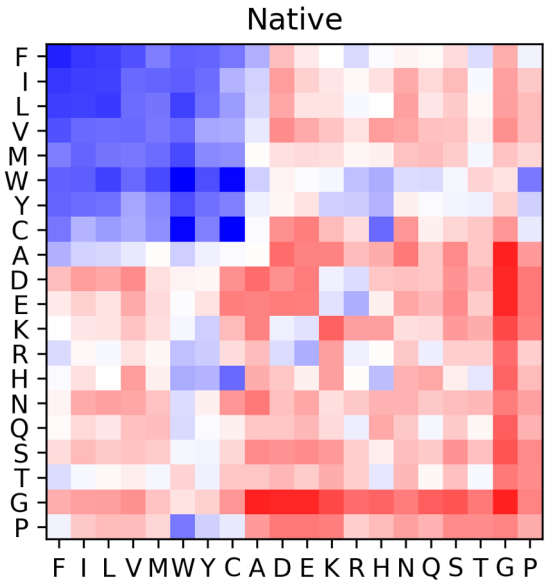
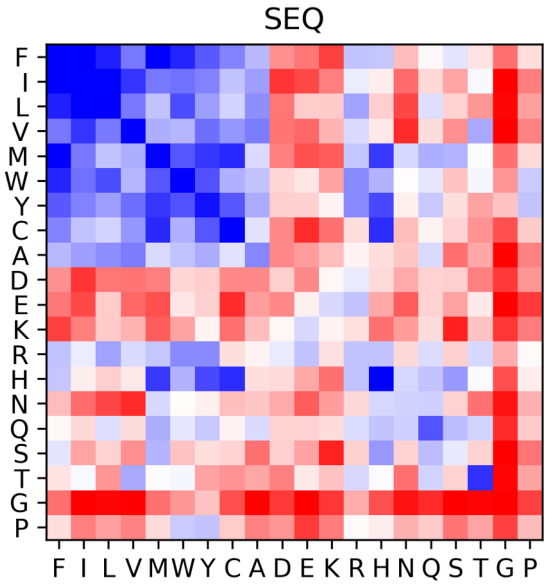


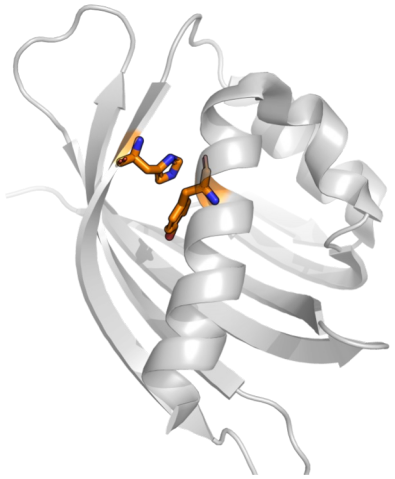
Figure 1. **What is encoded by a CNN?** The figure shows five possible reconstructions of the reference image obtained from the 1,000-dimensional code extracted at the penultimate layer of a reference CNN[13] (before the softmax is applied) trained on the ImageNet data. From the viewpoint of the model, all these images are practically equivalent. This image is best viewed in color/screen.

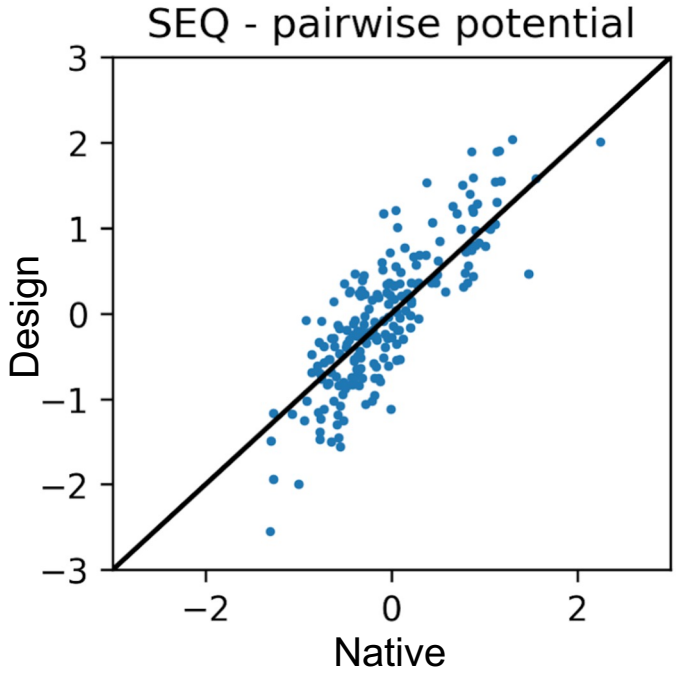
Understanding Deep Image Representations by Inverting Them
Aravindh Mahendran, Andrea Vedaldi
https://arxiv.org/abs/1412.0035

Comparing pairwise potential



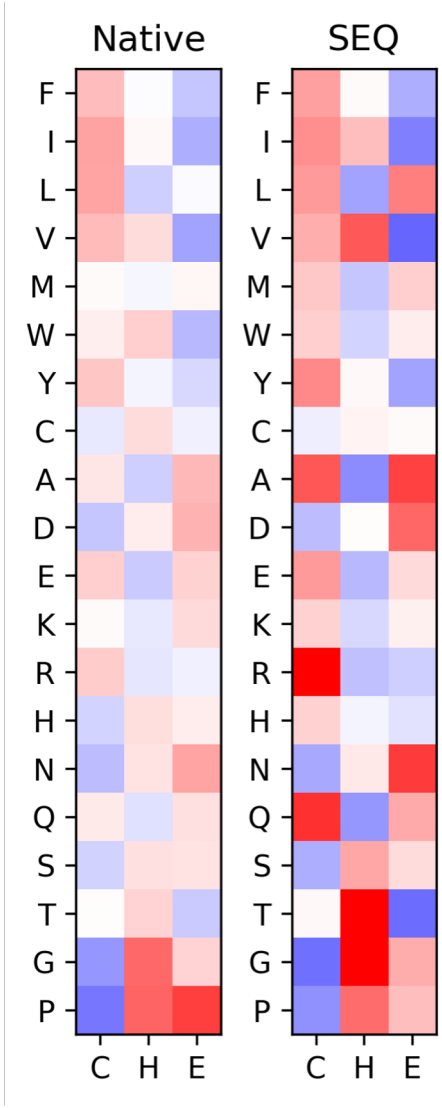


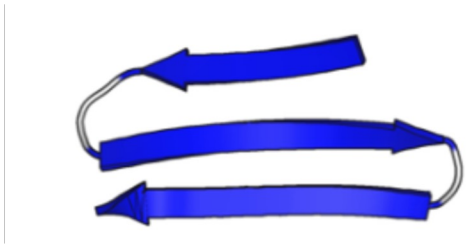




log(P(A,B|Contact) / P(A)*P(B))

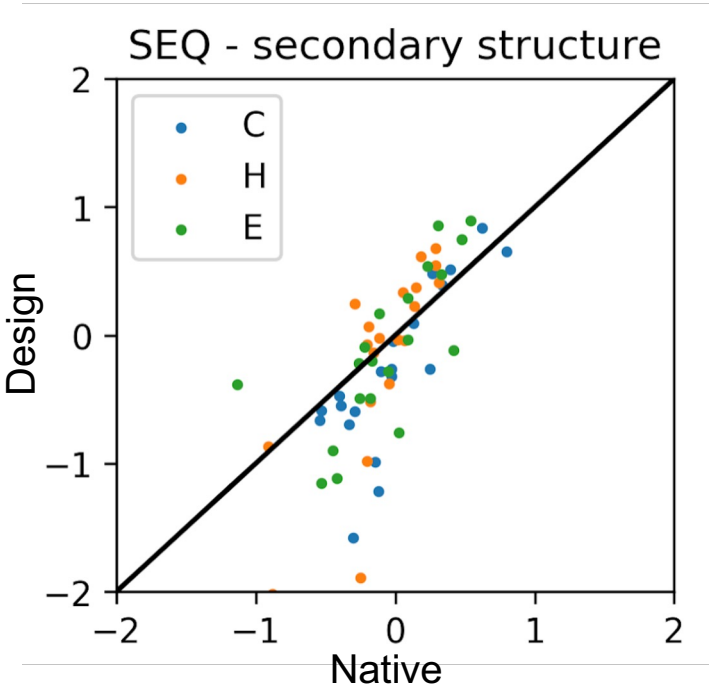
Comparing secondary structure propensity





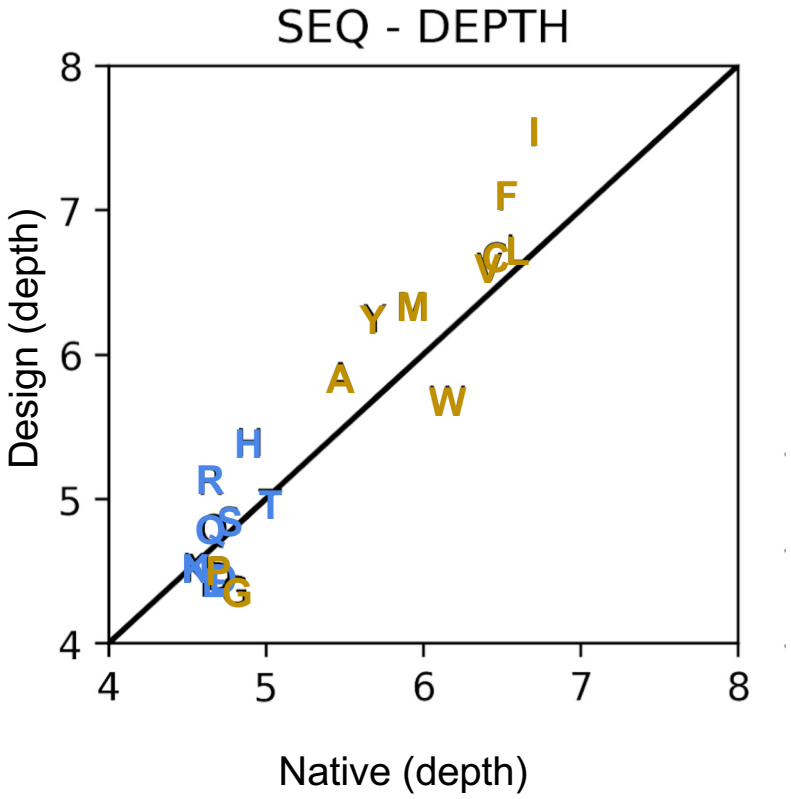
(E) Sheets

(H) Helix

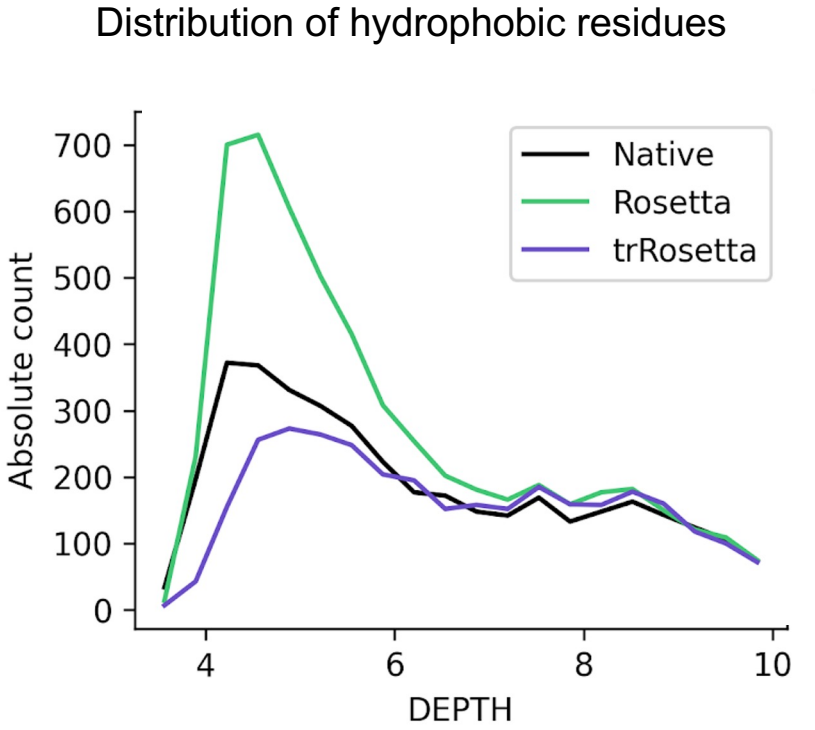


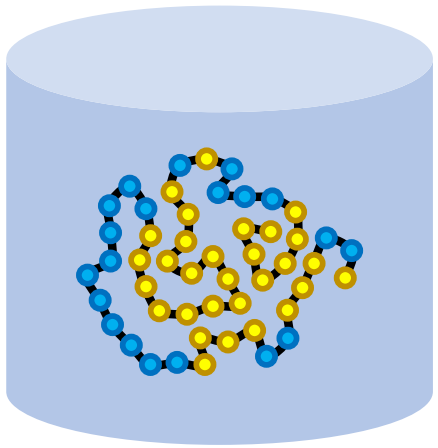
log(P(AA|SS) / P(AA))

Comparing average depth per amino acid type



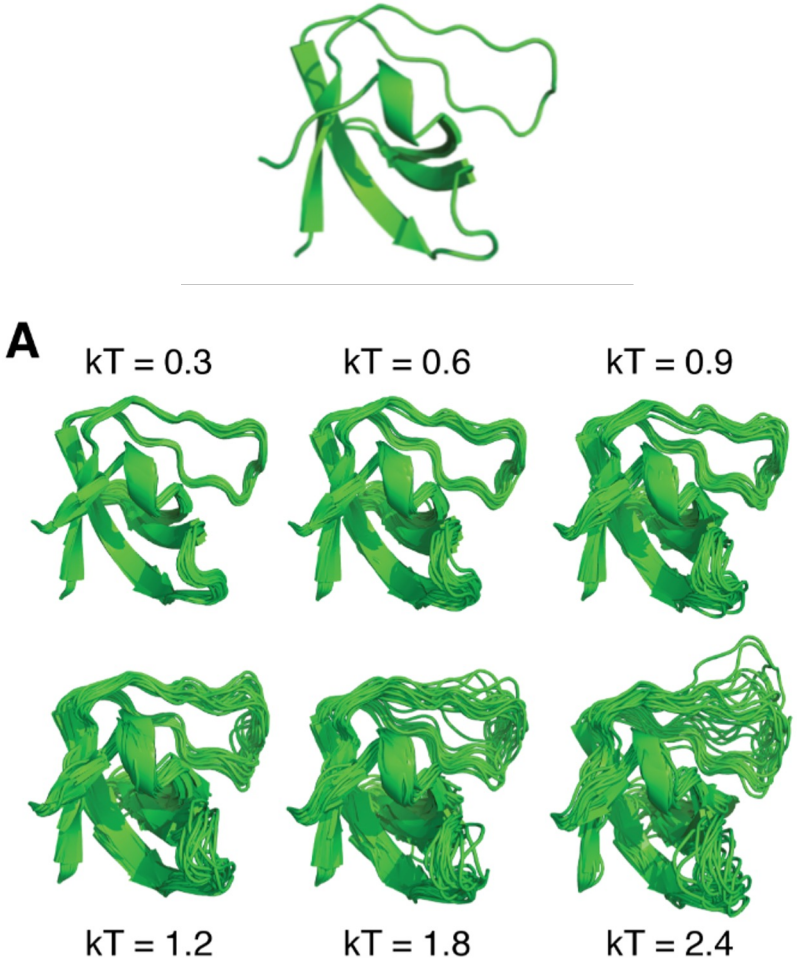
Depth = Distance from surface
aka. how buried is each amino acid

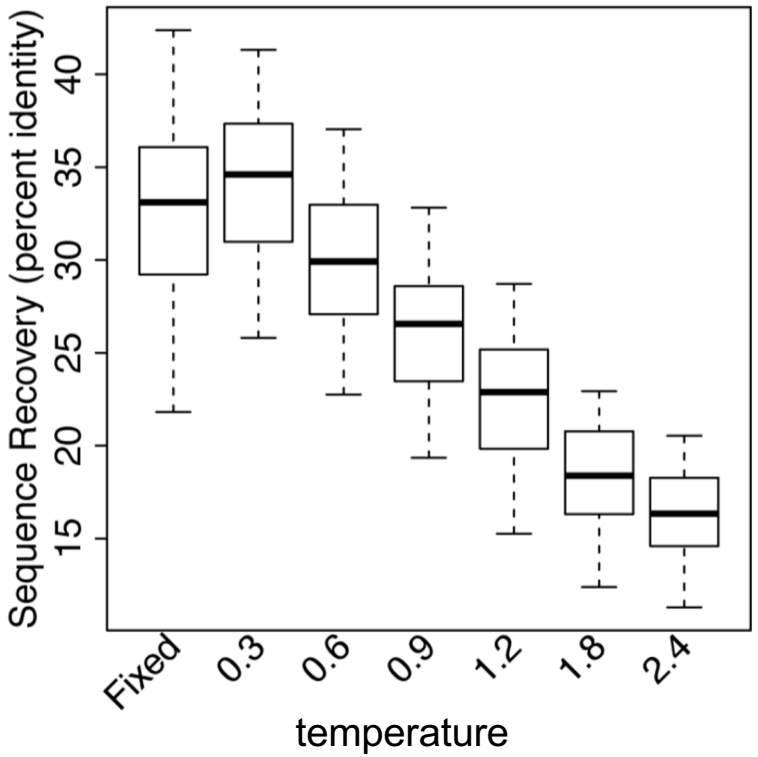




H2 - low resolution model

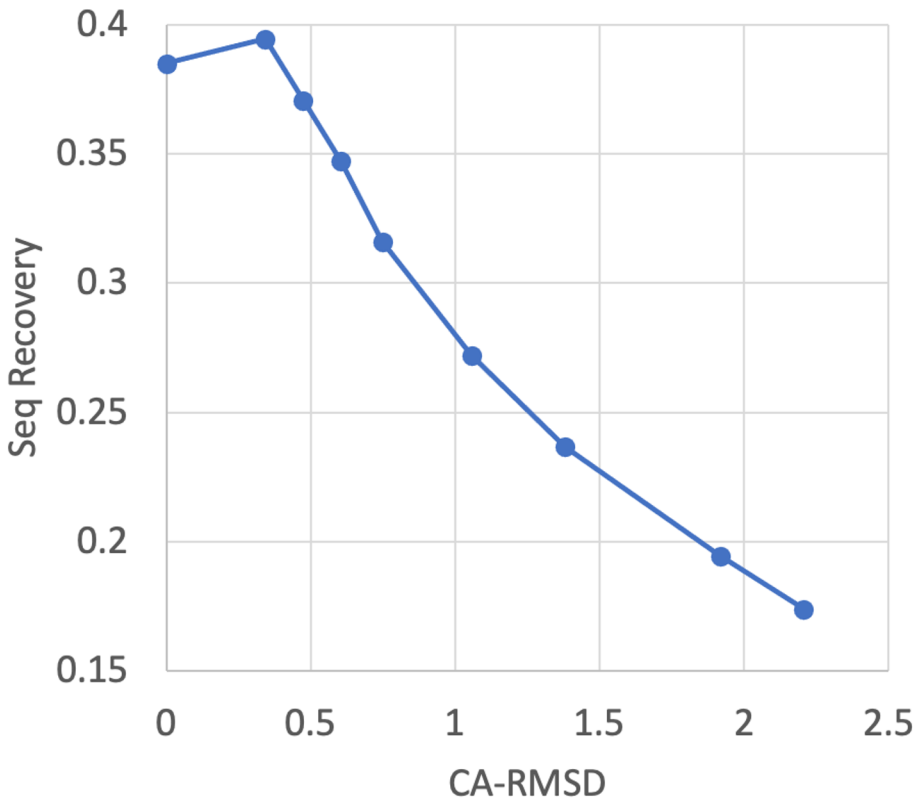
Use backrub to generate "low resolution" backbones and design new sequences with Rosetta

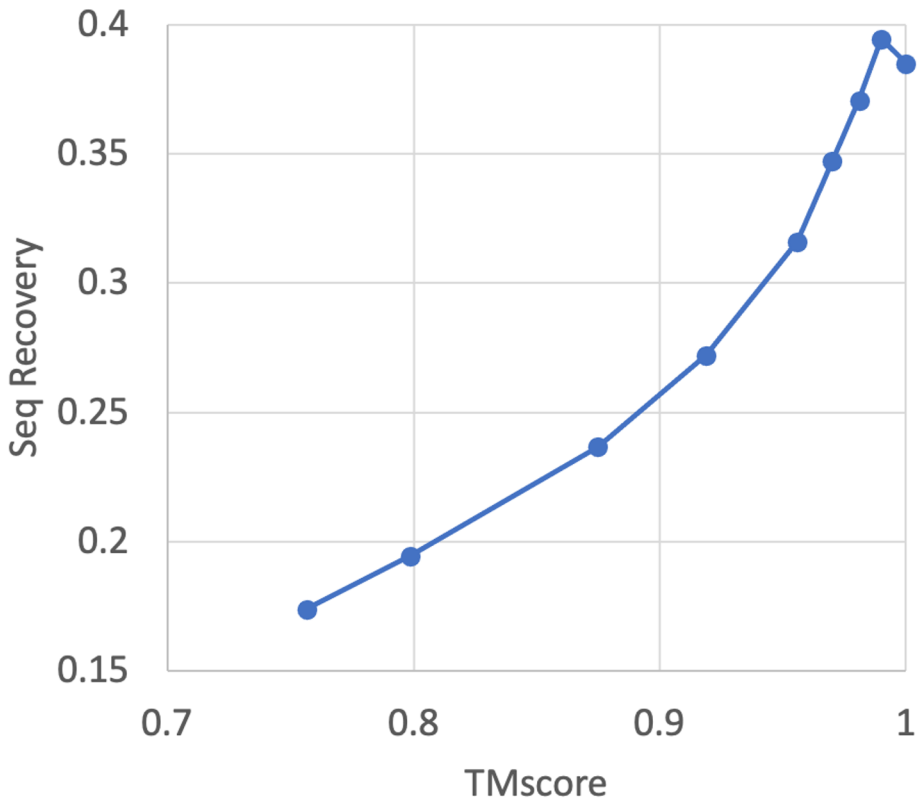




Ollikainen, Noah, and Tanja Kortemme. "Computational protein design quantifies structural constraints on amino acid covariation." *PLoS computational biology* 9, no. 11 (2013): e1003313.

Perturb backbone, redesign with Rosetta

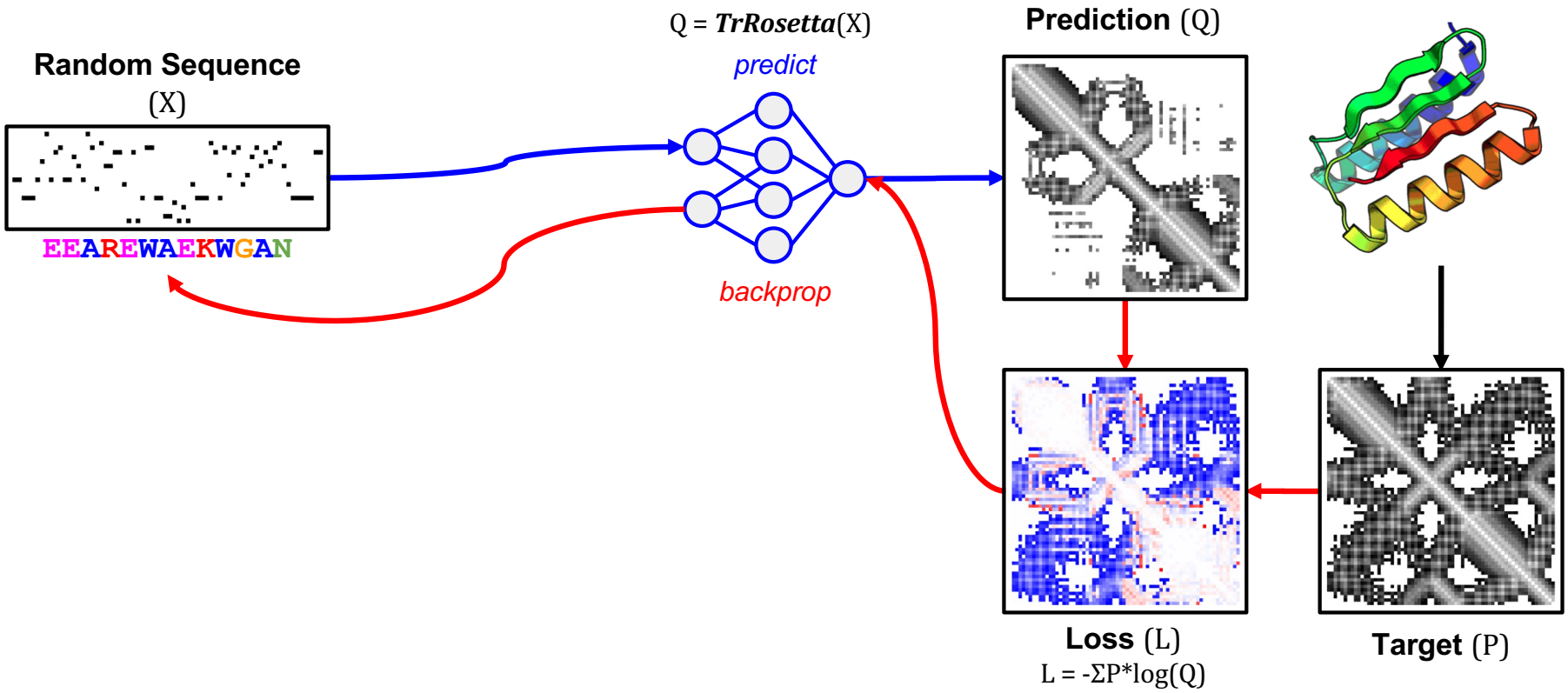




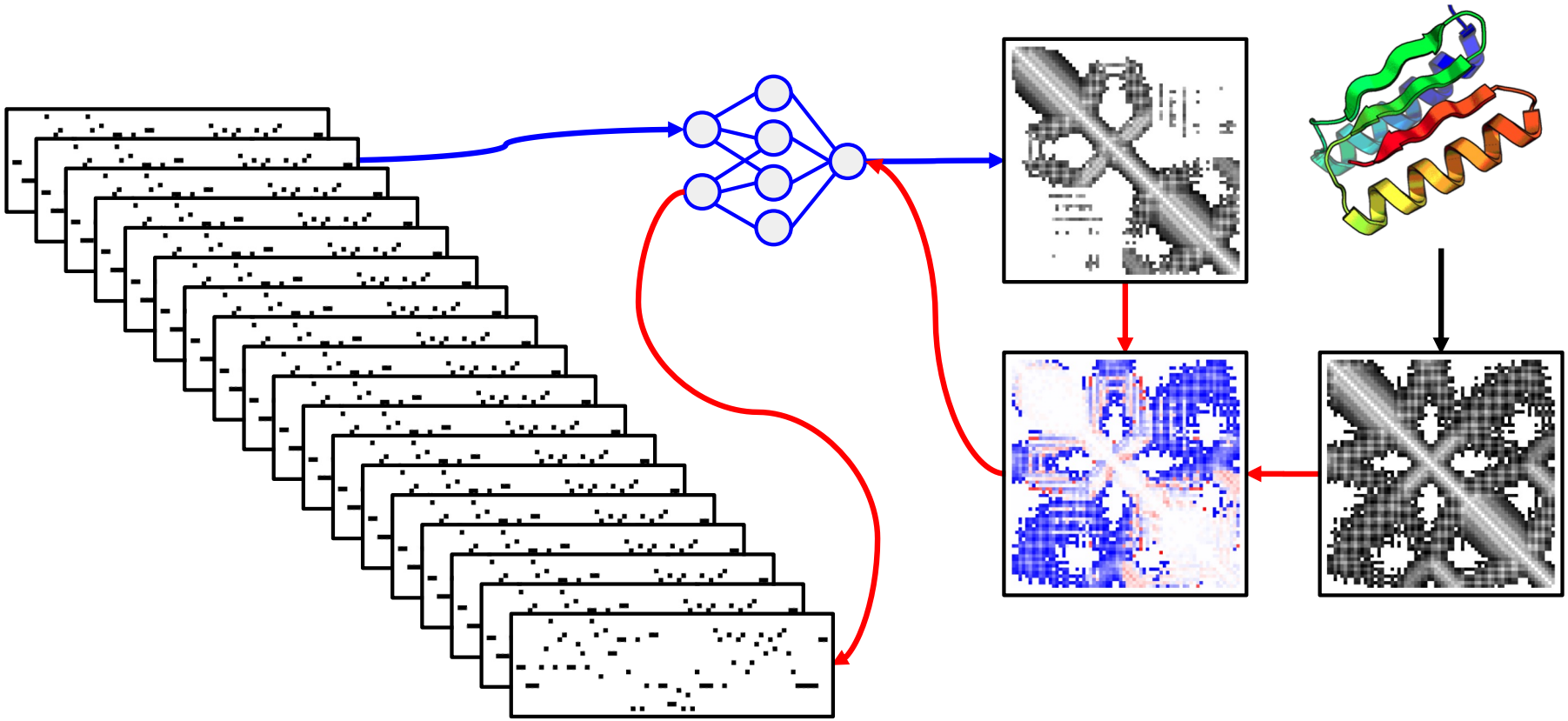
Given our structure accuracy for *denovo* design proteins is ~0.7 TMscore, ~0.15 sequence recovery is expected

H3 - only a subset of sequence or contacts being optimized

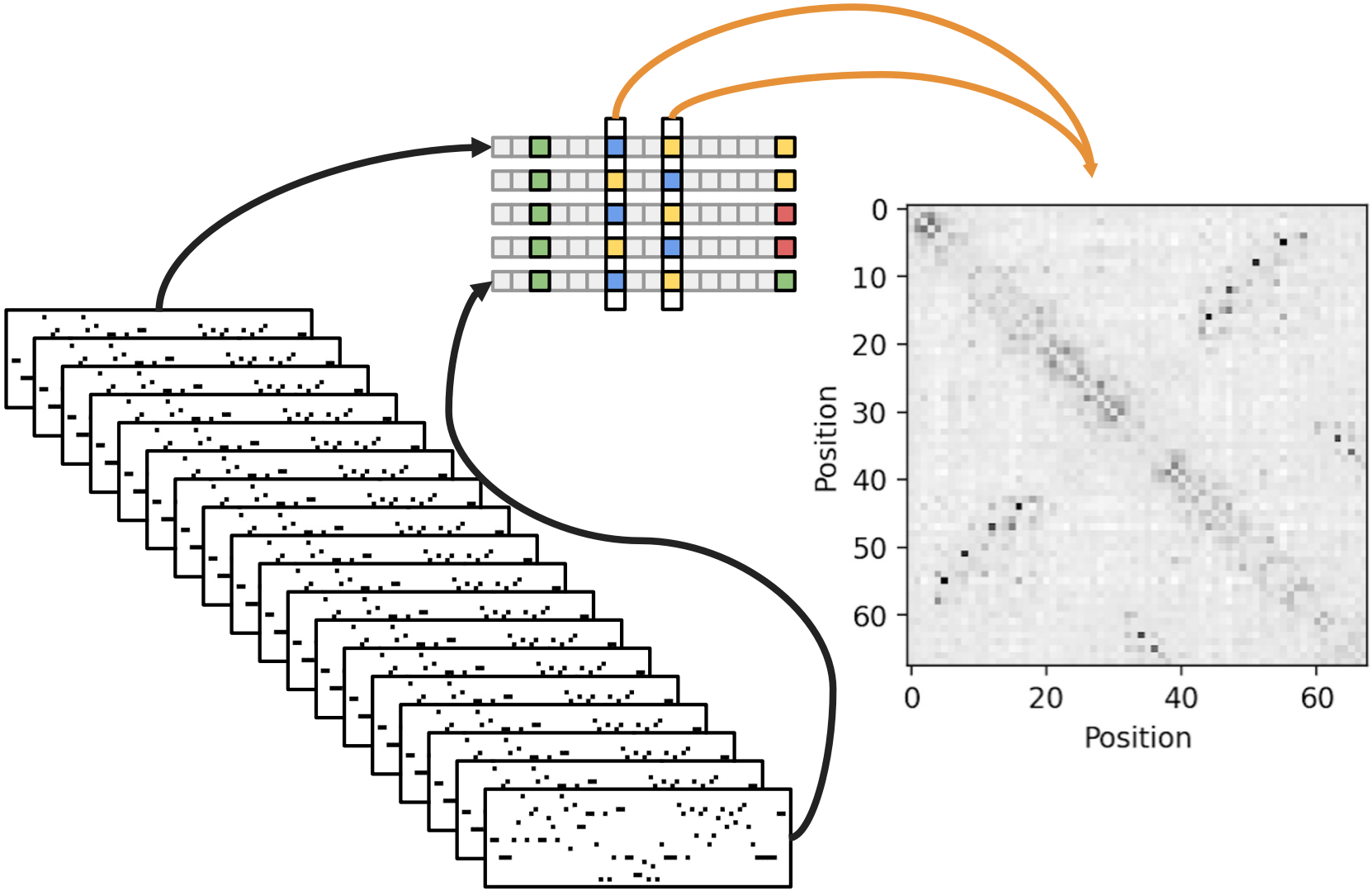
Run TrRosetta many time independently and generate many sequences



Run TrRosetta many time independently and generate many sequences



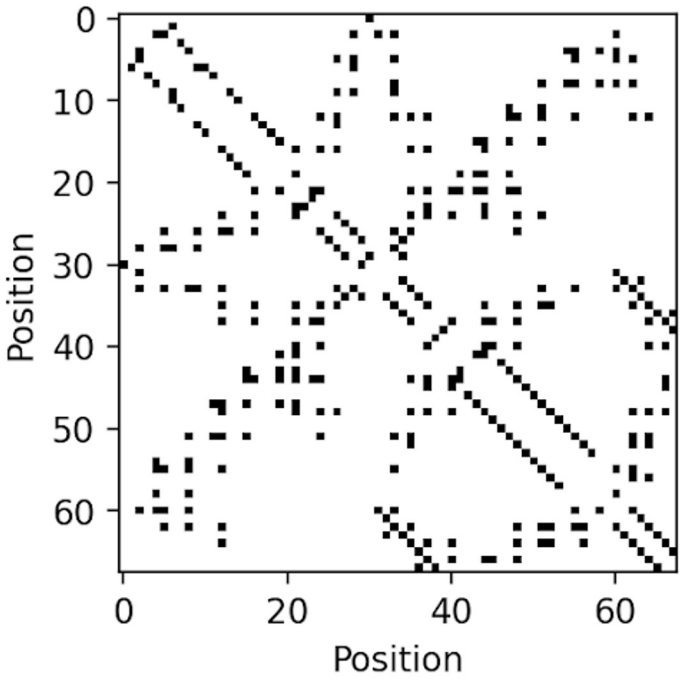
Pass sequences to **GREMLIN** (coevolution method) to find covariance between positions



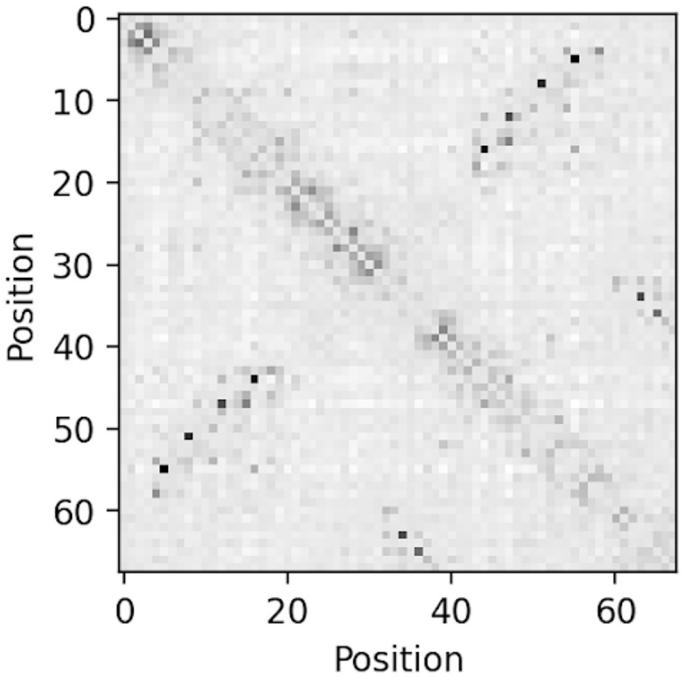
Generate 10K sequences and run coevolution analysis on these.

Only a subset of contacts coevolving in designed sequences!

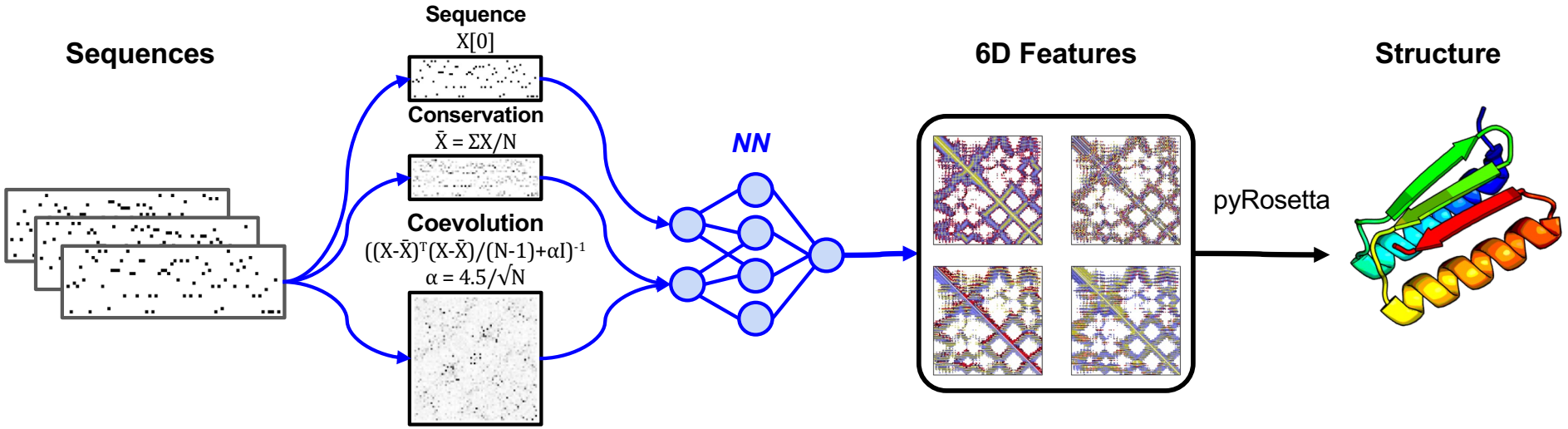
Residues in contact



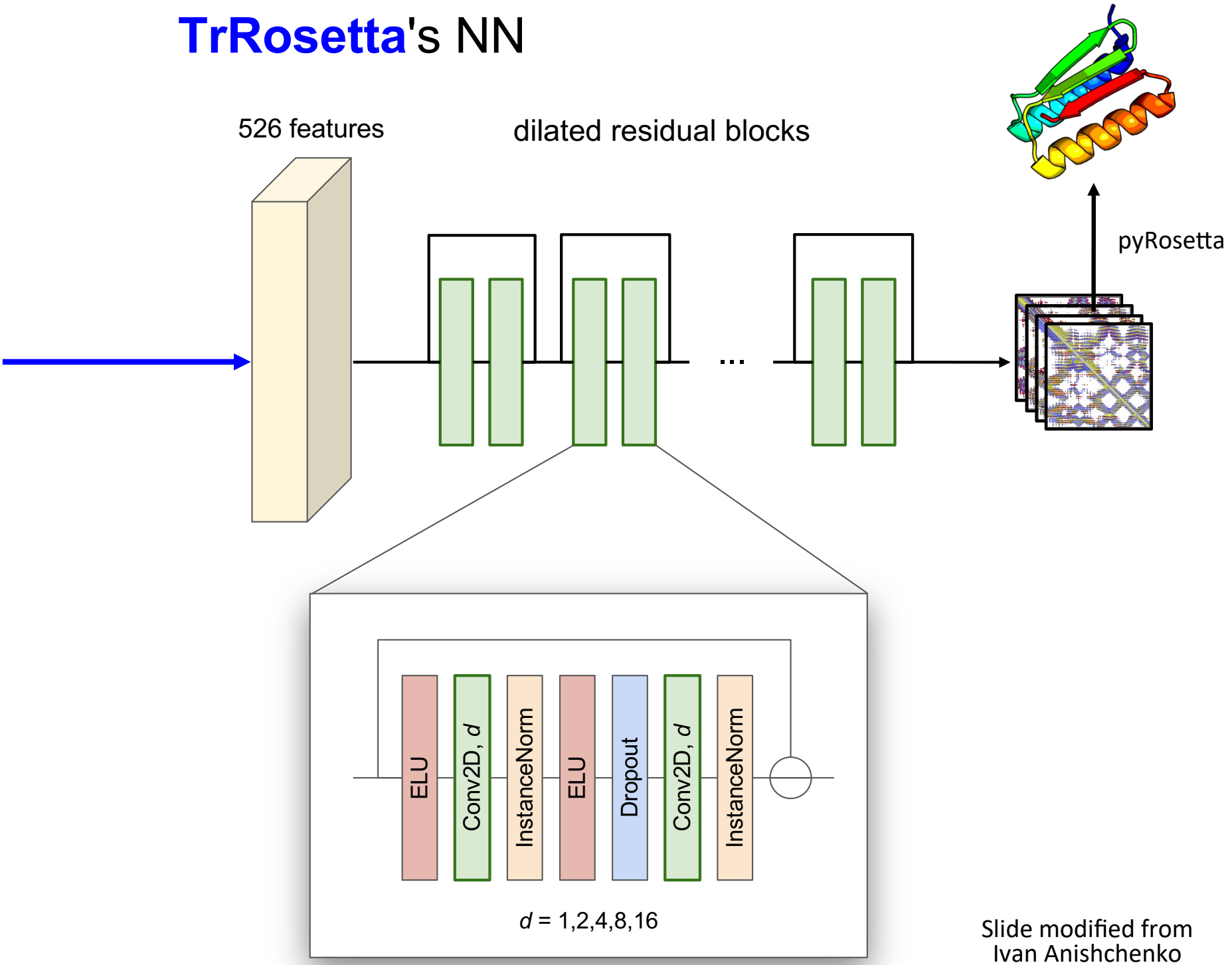
Residue pairs coevolving



TrRosetta - Takes sequences and returns structure

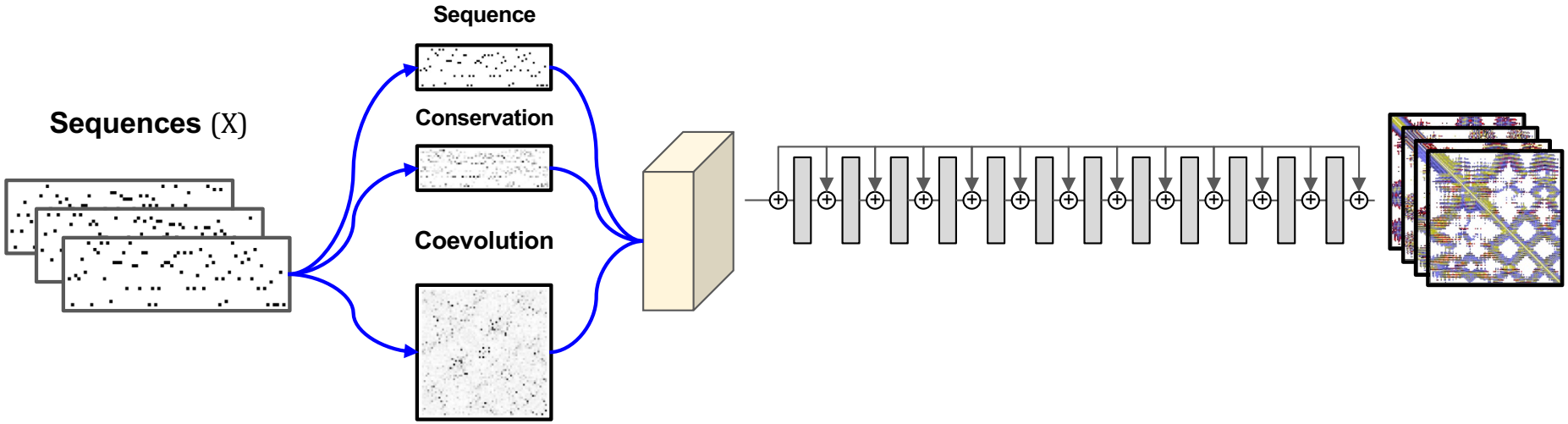


TrRosetta's NN

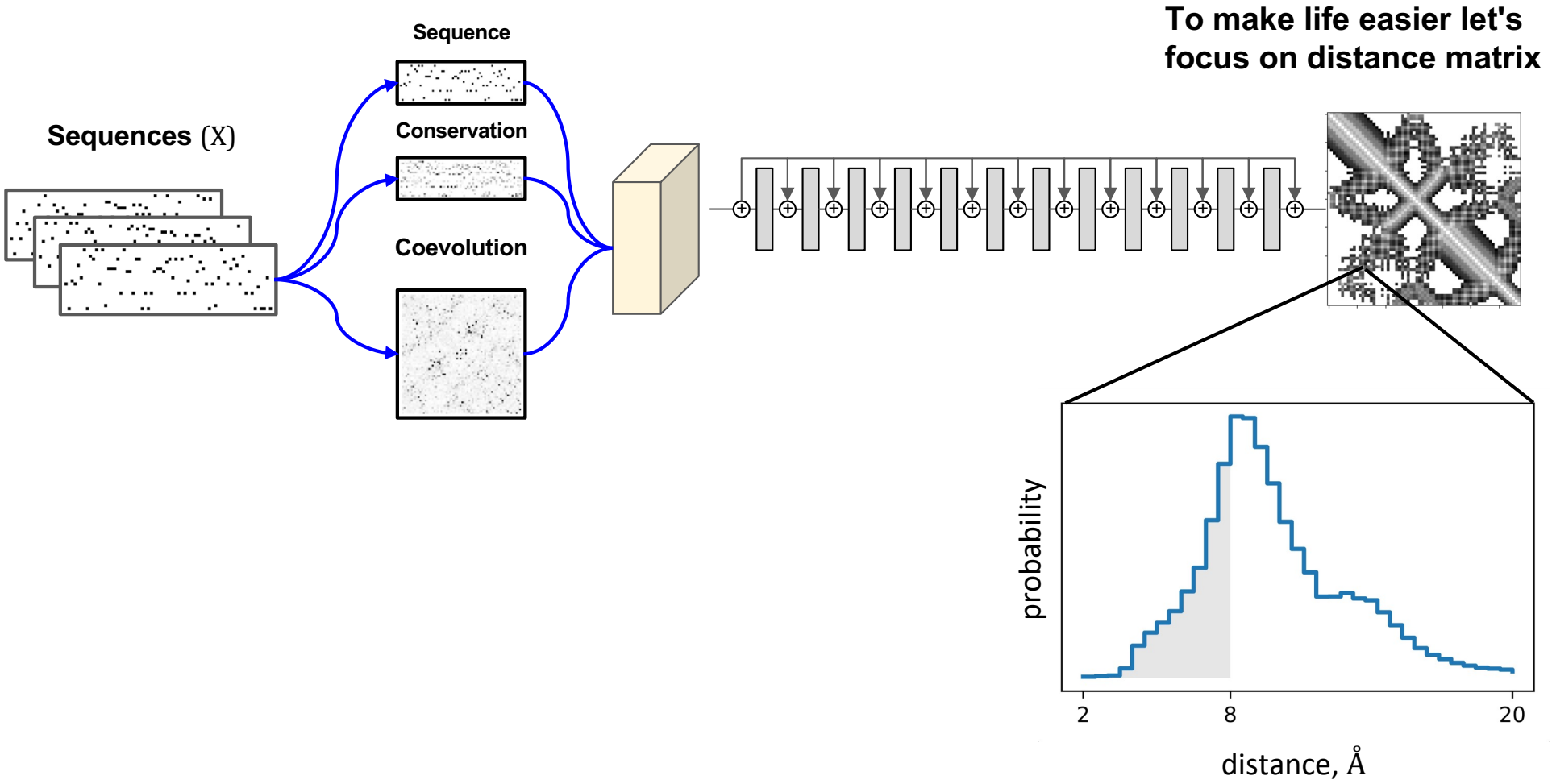


Slide modified from
Ivan Anishchenko

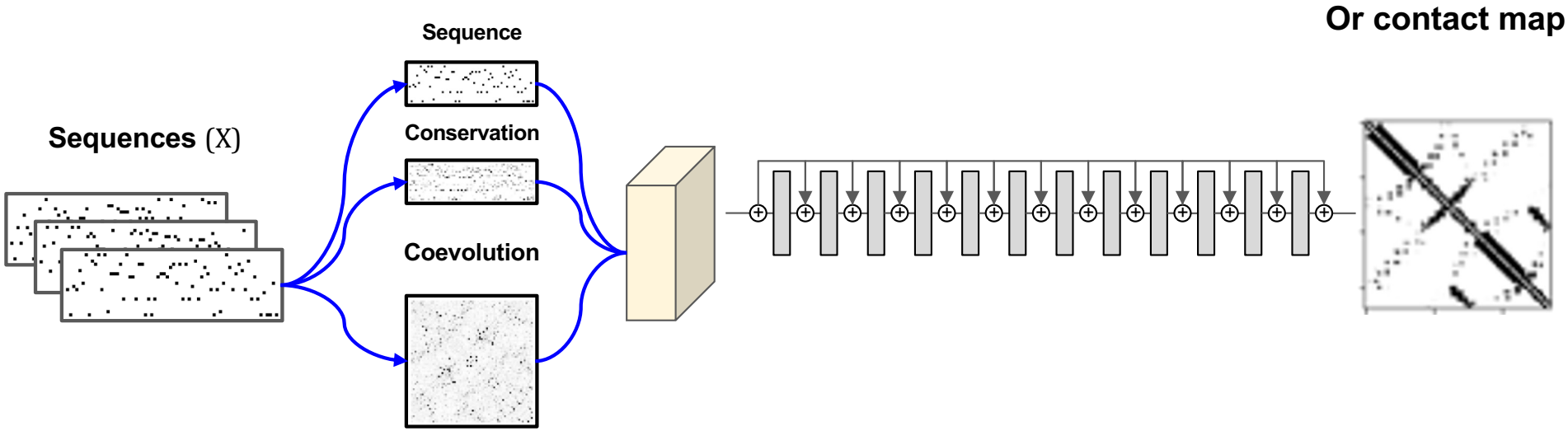
What is each layer doing?



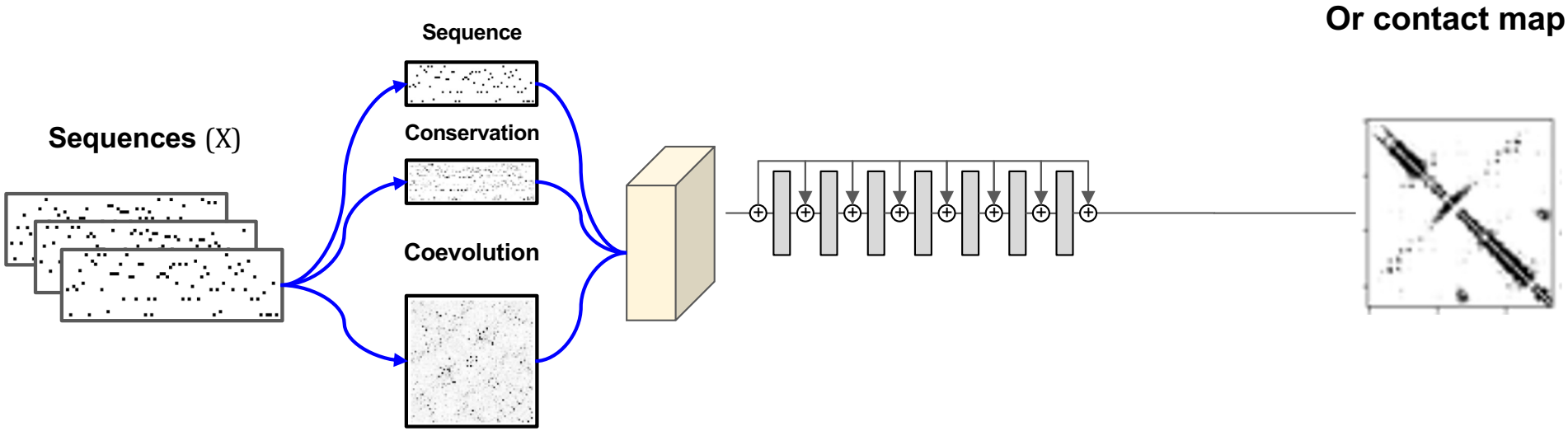
What is each layer doing?



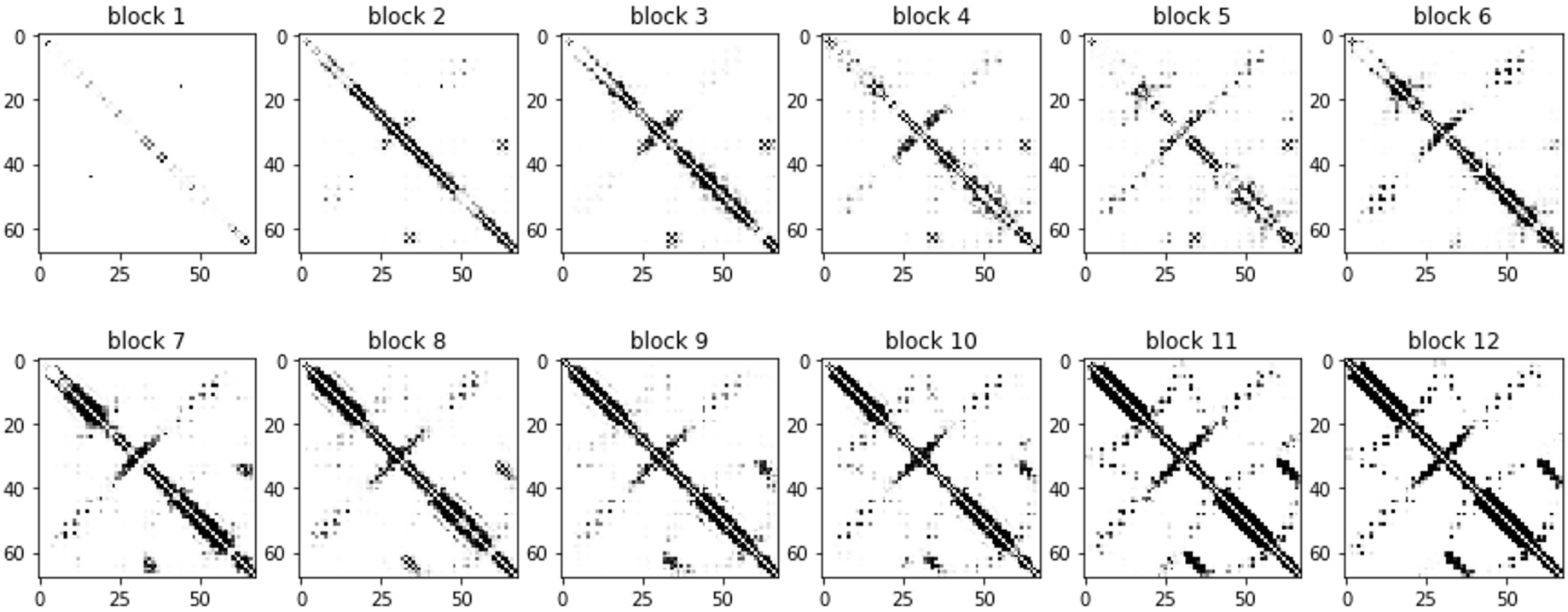
What is each layer doing?

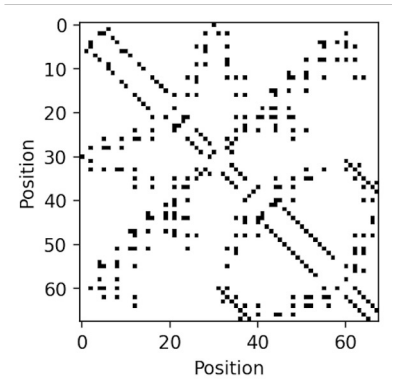


What is each layer doing?

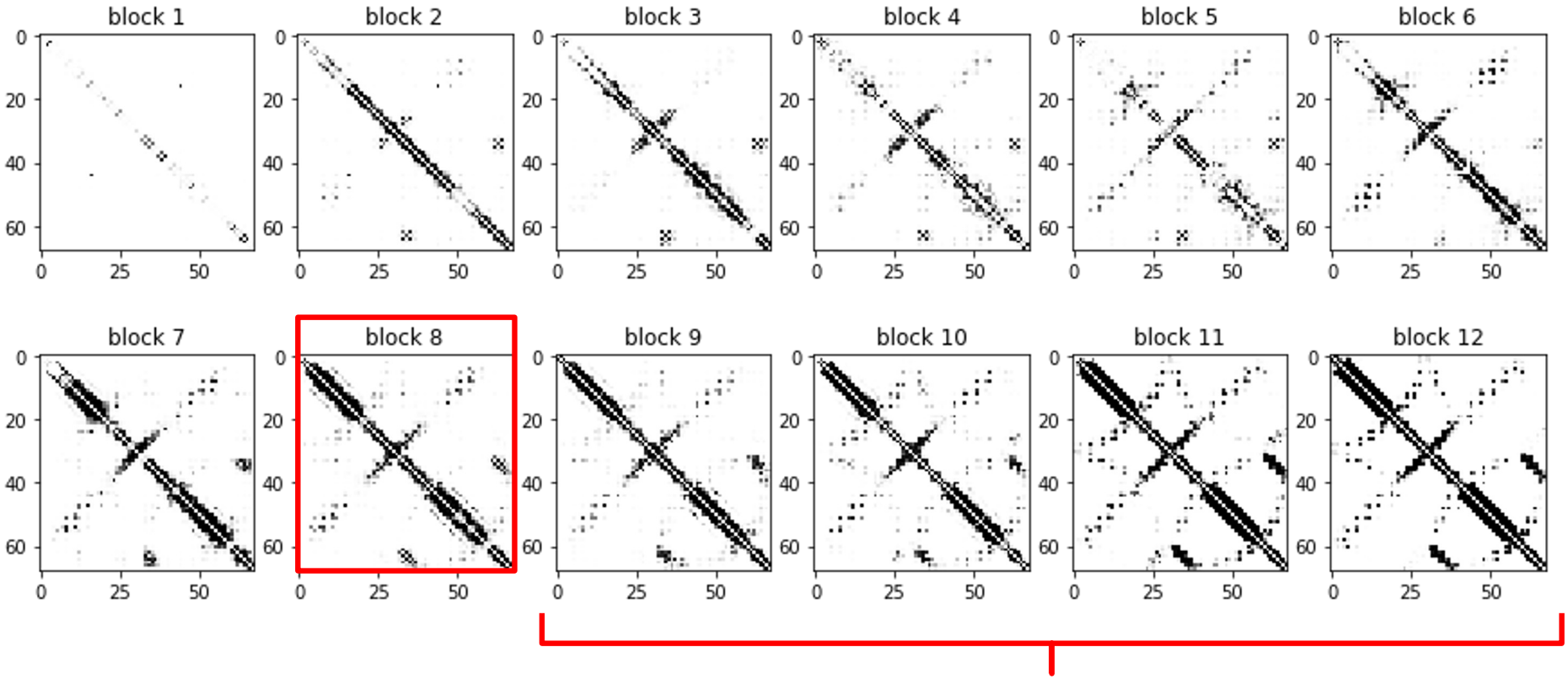


Investigating each block in resnet

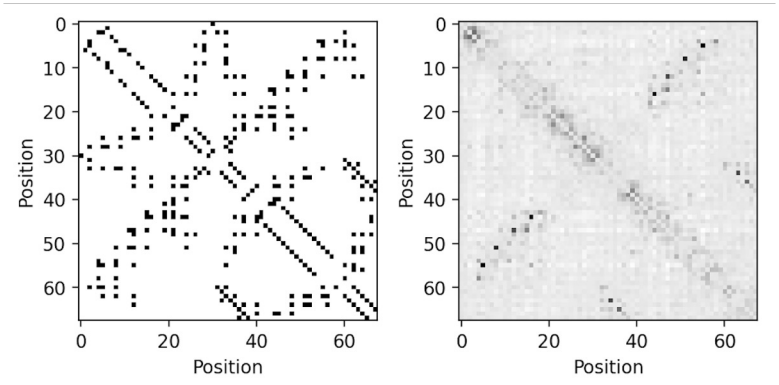




Covarying residues match those in block 8



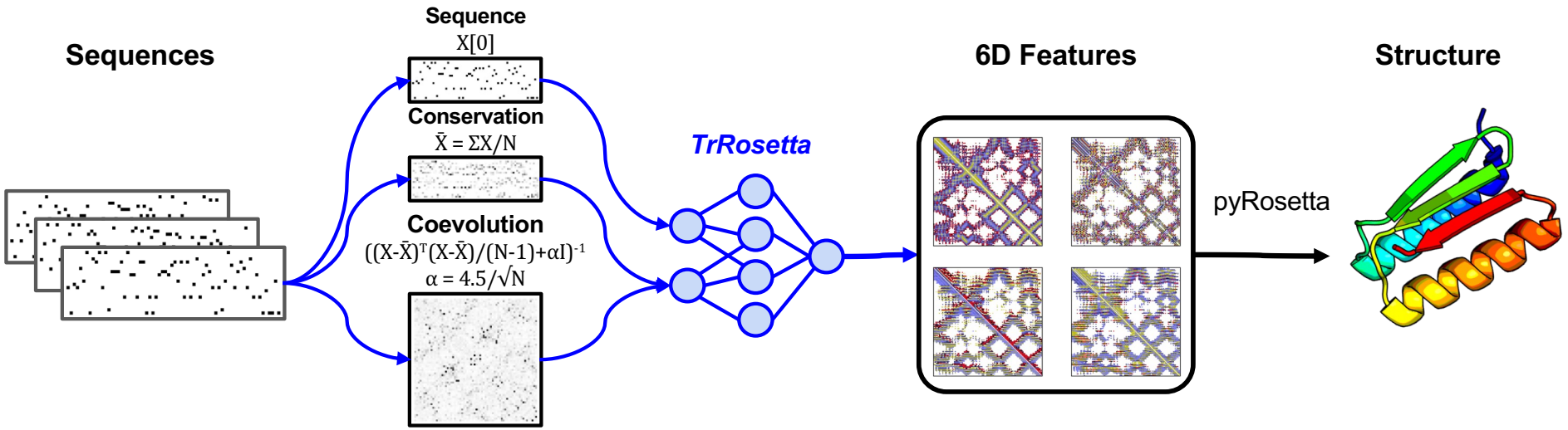
Are these layers just "connecting the dots"?



It appears only subset of sequence or contacts are being optimized.

Can we push the model to optimize all contacts?

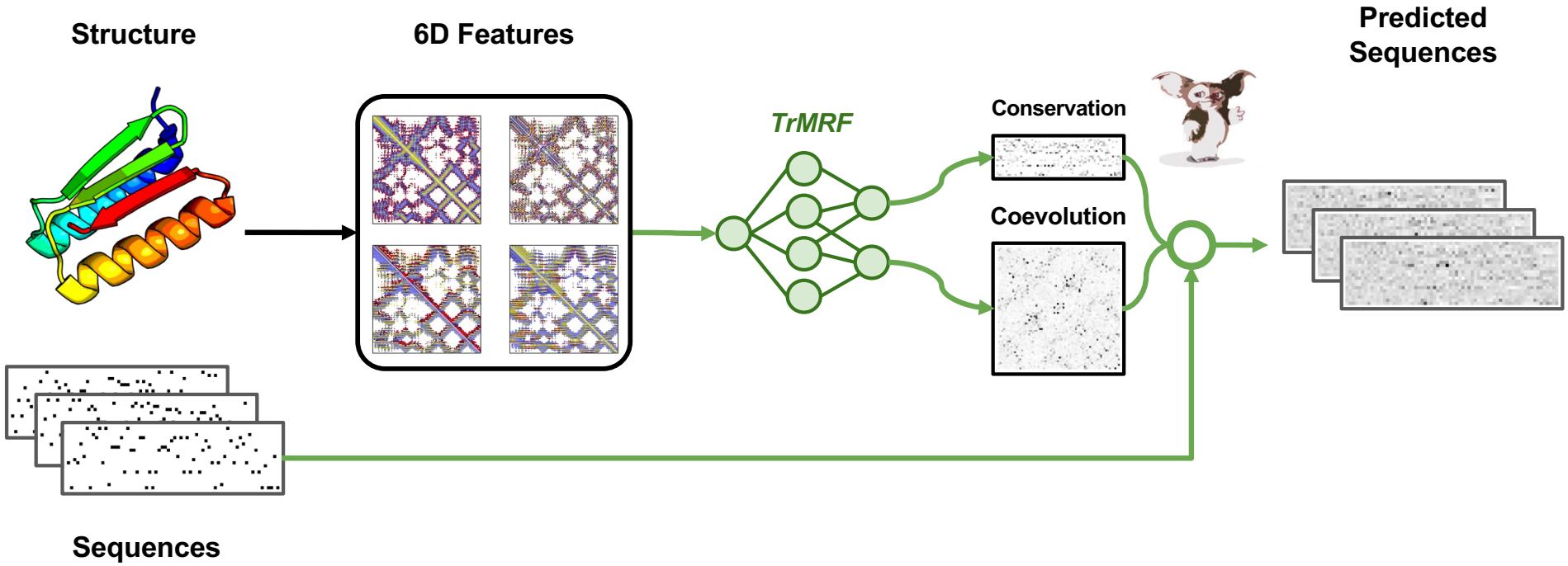
TrRosetta - Takes sequences and returns structure



pred_feats = **TrRosetta**(sequences)

loss = CCE(true_feats, pred_feats)

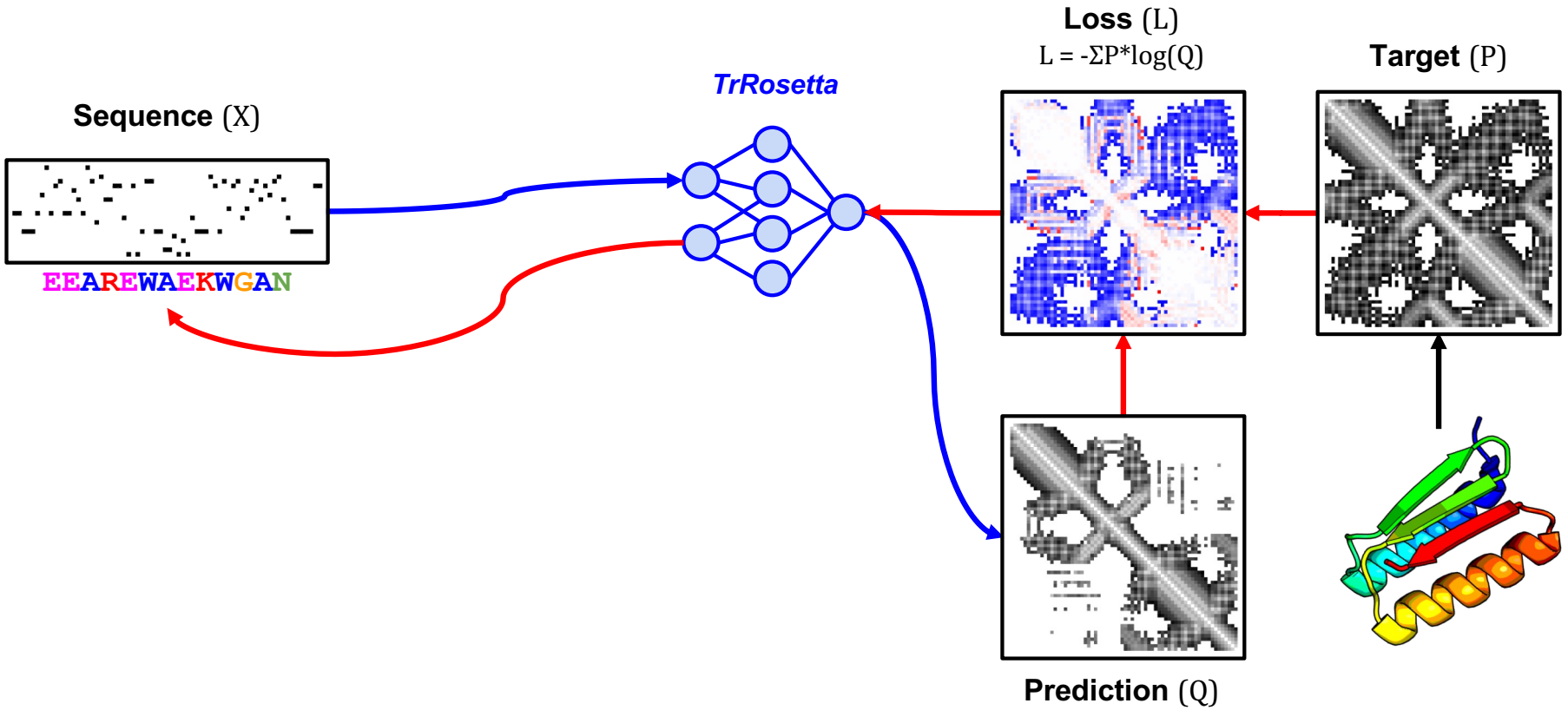
TrMRF - Takes structure and returns sequences

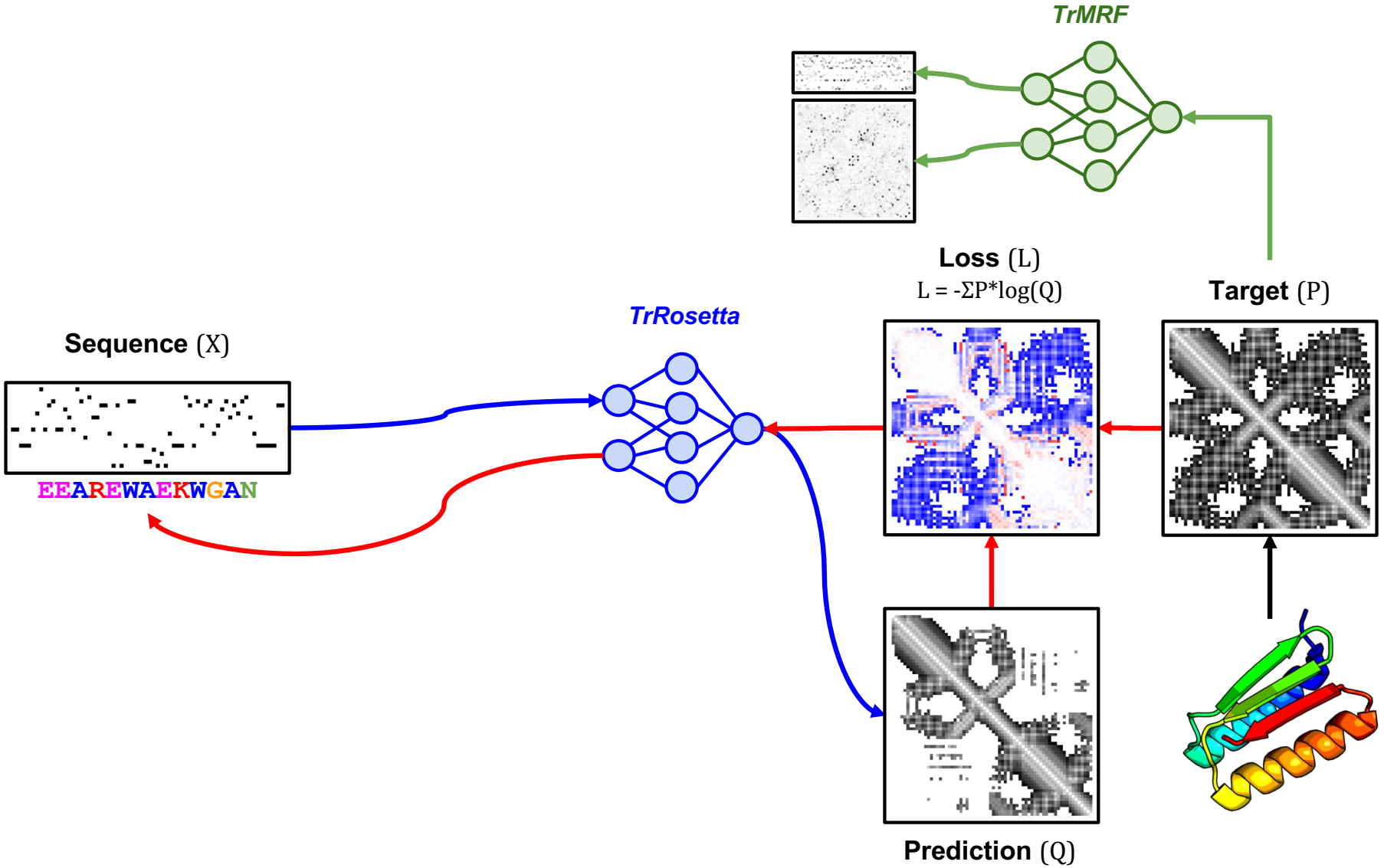


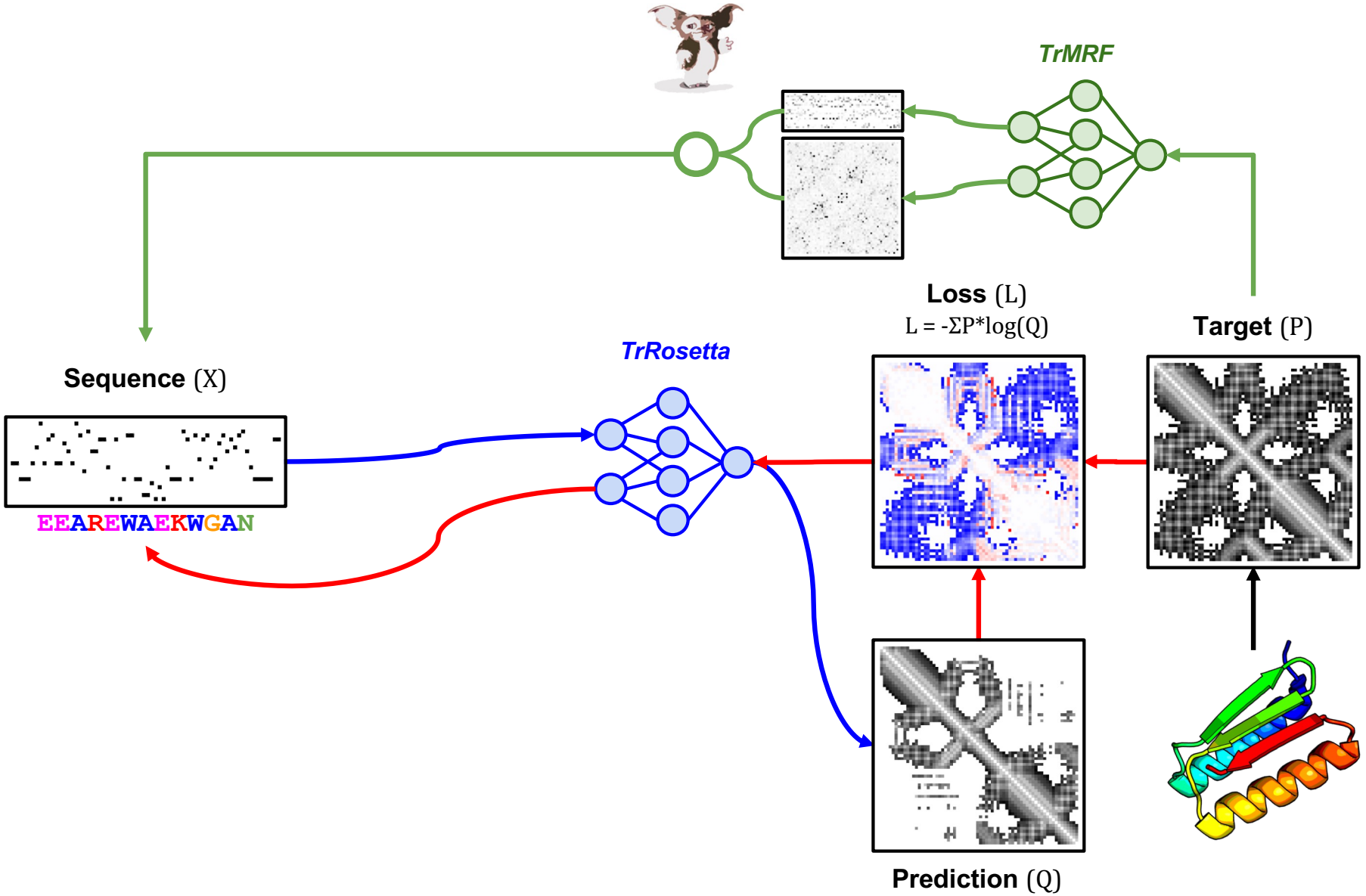
W,b = TrMRF(structure)

pred_sequences = softmax(sequences@W + b)

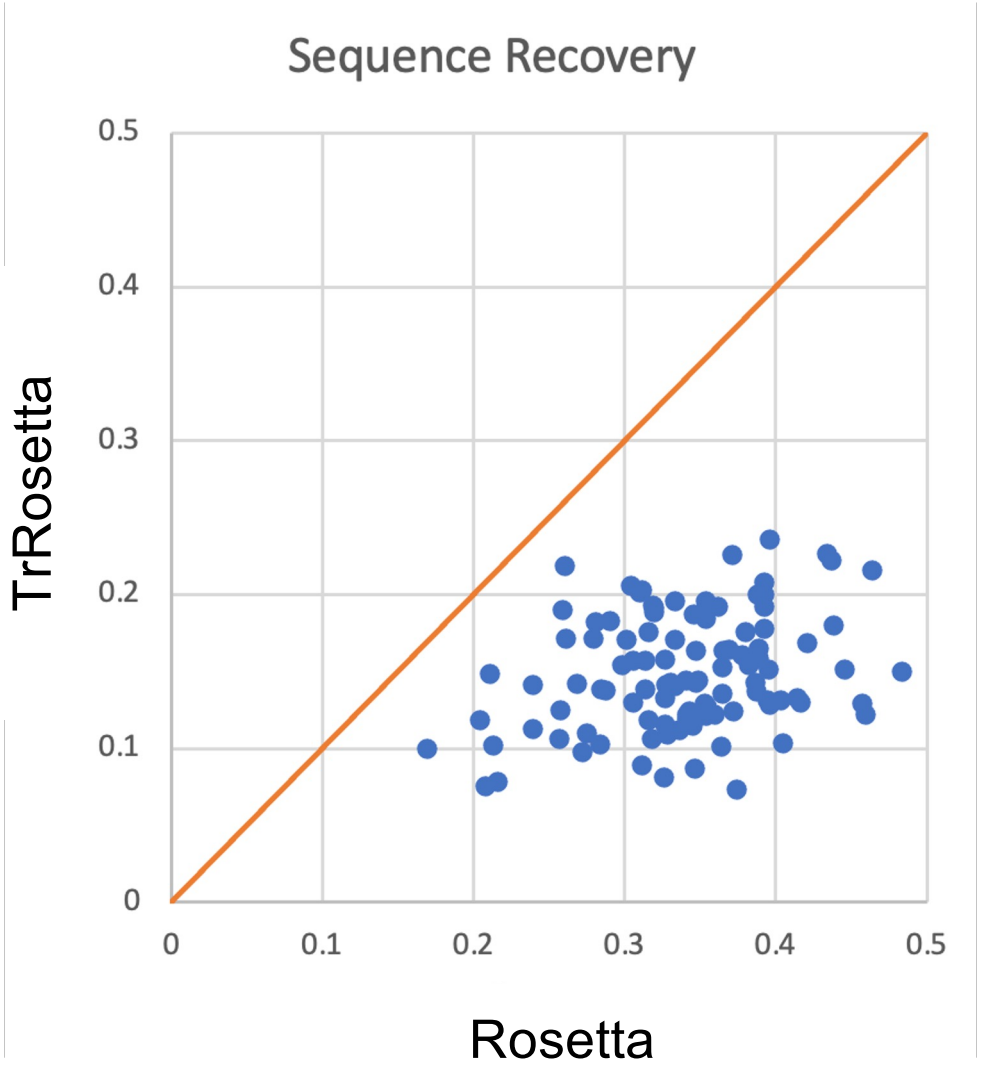
loss = CCE(sequences, pred_sequences)


Loss (L)
L = -ΣP*log(Q)
Target (P)
TrRosetta
Sequence (X)
EEAREWAEKWGAN
Prediction (Q)

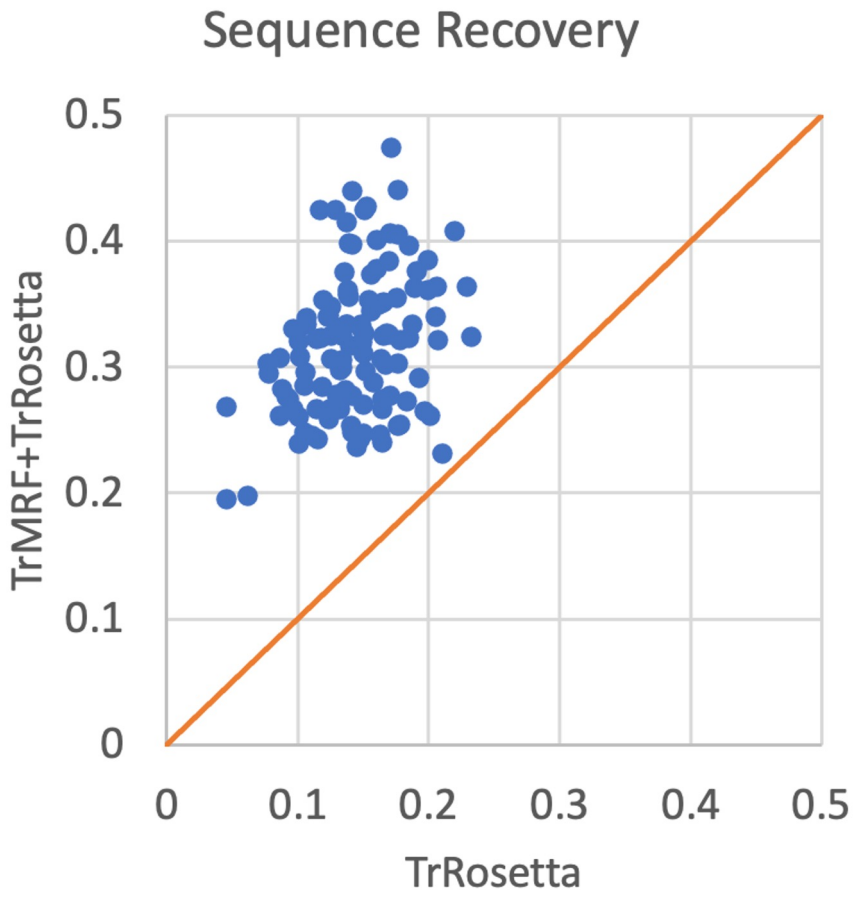

TrMRF
Loss (L)
L = -ΣP*log(Q)
Target (P)
TrRosetta
Sequence (X)
EEAREWAEKWGAN
Prediction (Q)


TrMRF
Loss (L)
L = -ΣP*log(Q)
Target (P)
TrRosetta
Sequence (X)
EEAREWAEKWGAN
Prediction (Q)

TrRosetta Sequence recovery is too low compared to Rosetta



Adding **TrMRF** loss significantly improved sequence recovery!



Experiment

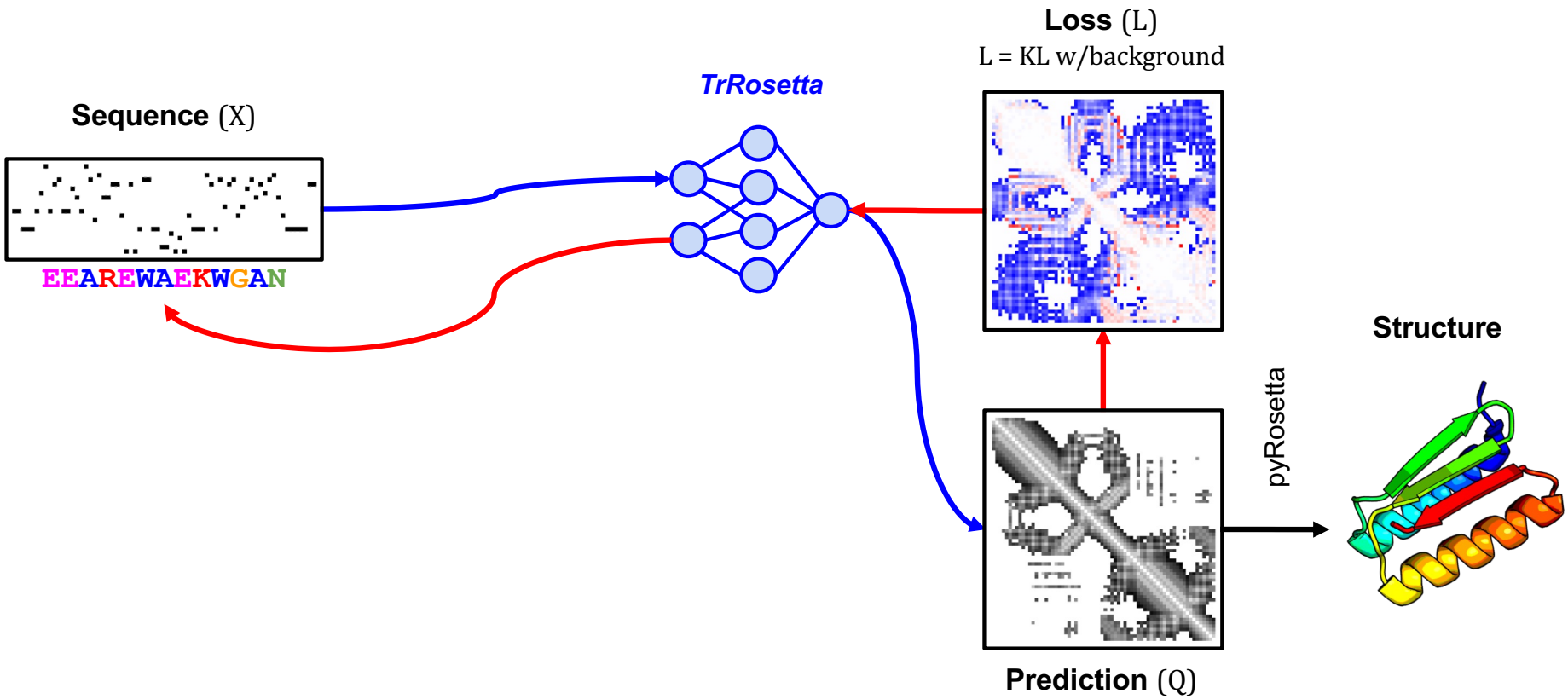
- TrROS - Hallucinate new proteins using TrRosetta.
- TrMRF - Redesign Proteins using TrMRF objective.
- TrROS+TrMRF - Redesign Proteins using TrRosetta+TrMRF objective.
- Experimentally test designs for protease resistance (proxy for stability).
- Compare initial hallucinated proteins to redesigned proteins.

Justas Dauparas

Tsuboyama Kotaro

Gabriel J Rocklin

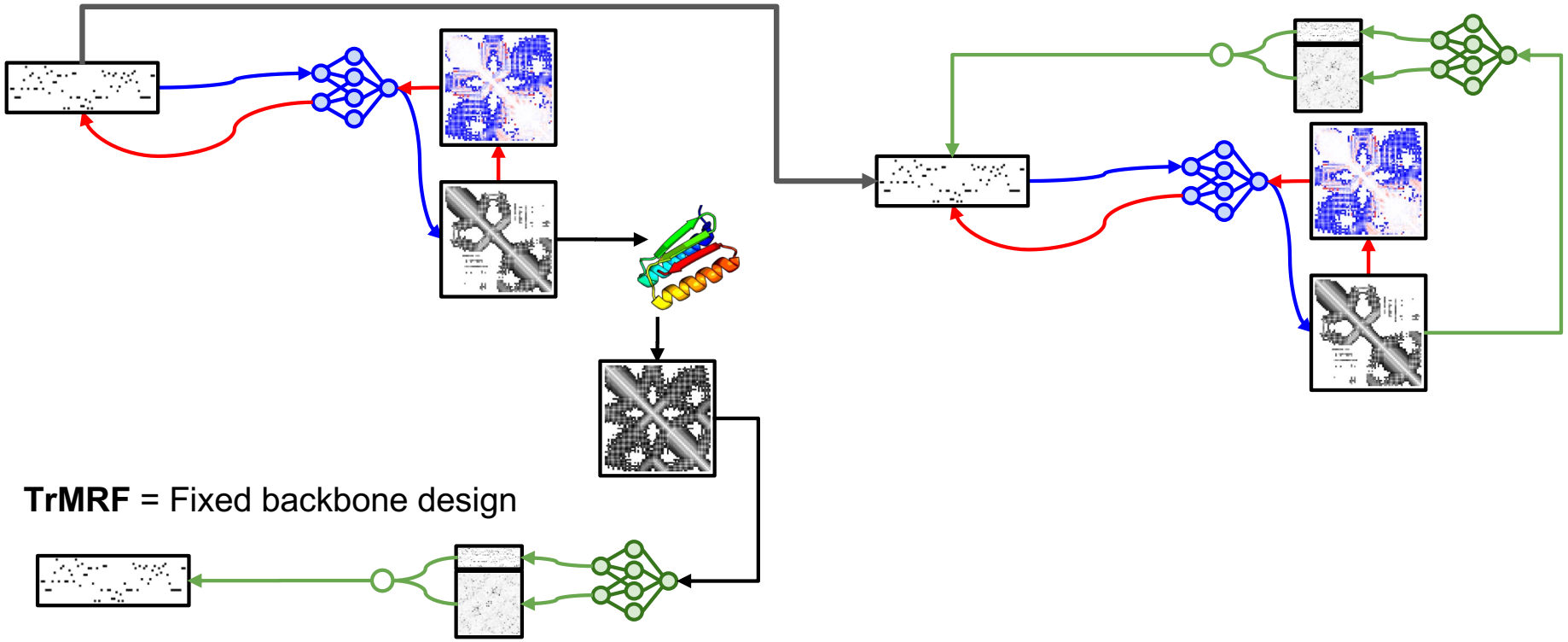
Hallucinate thousands of new proteins



Redesign proteins

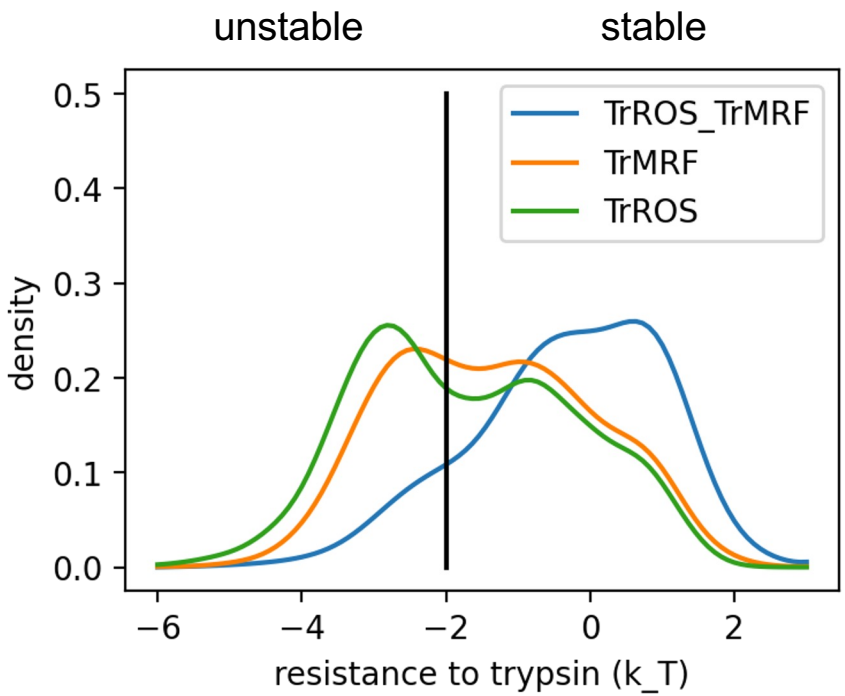
TrROS = Hallucination

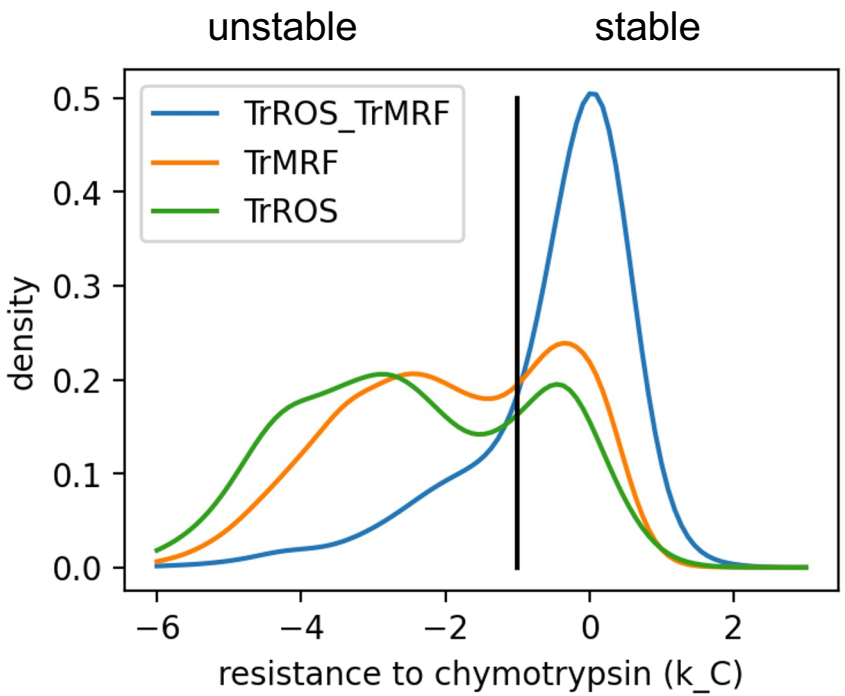
TrROS+TrMRF = Cont. Hallucination



TrMRF = Fixed backbone design

Results - Protease Resistance (Stability) Assay

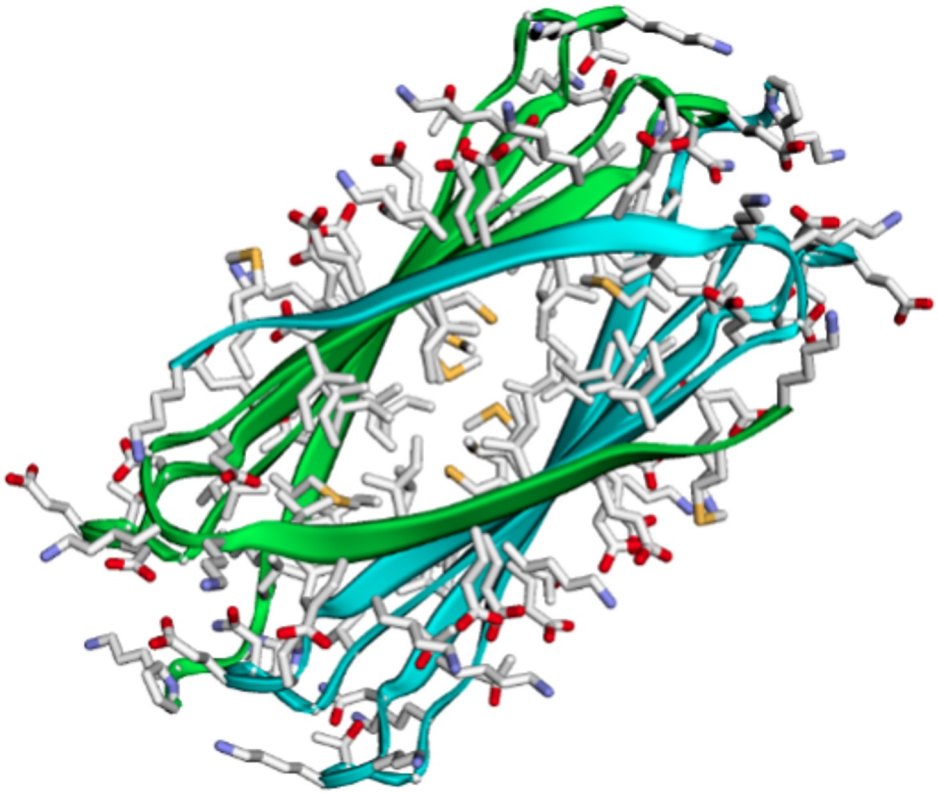




Tsuboyama
Kotaro

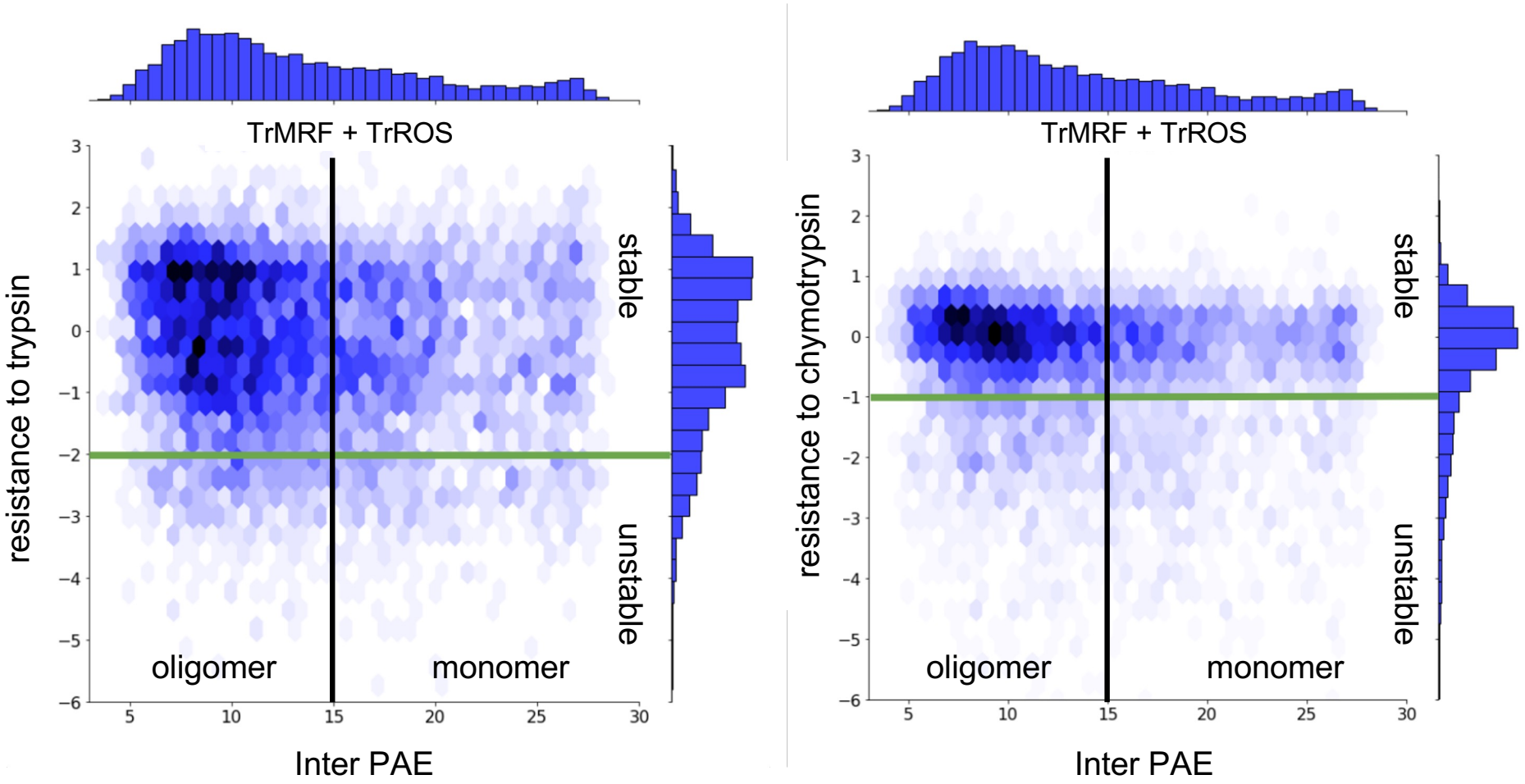
Results look TOO good

We noticed some of the most stable designs had hydrophobic patches, which AlphaFold predicts as homo-oligomers



Justas
Dauparas

Lets compute inter-PAE values for our designs



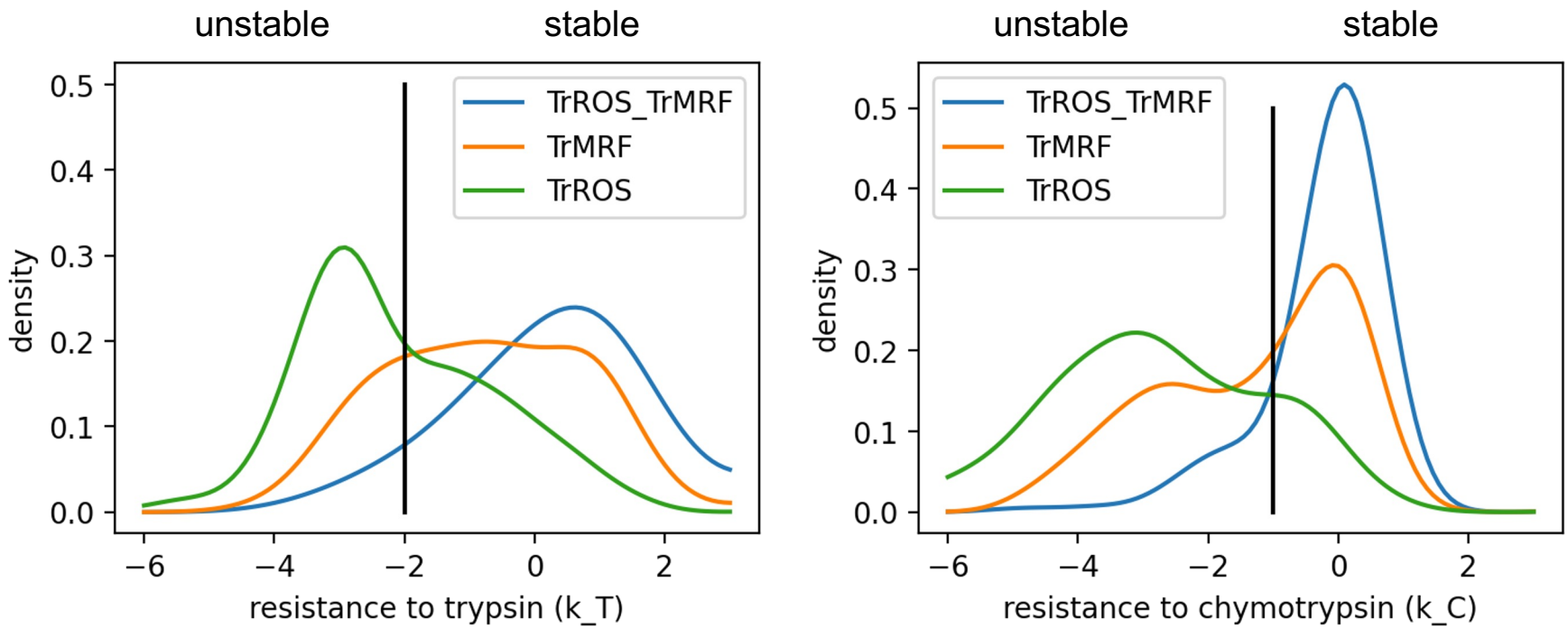
Though most designs are predicted as oligomers, the monomers are still "stable"

AlphaFold filtering

interPAE > 15
pLDDT > 70

Results - Protease Resistance (Stability) Assay

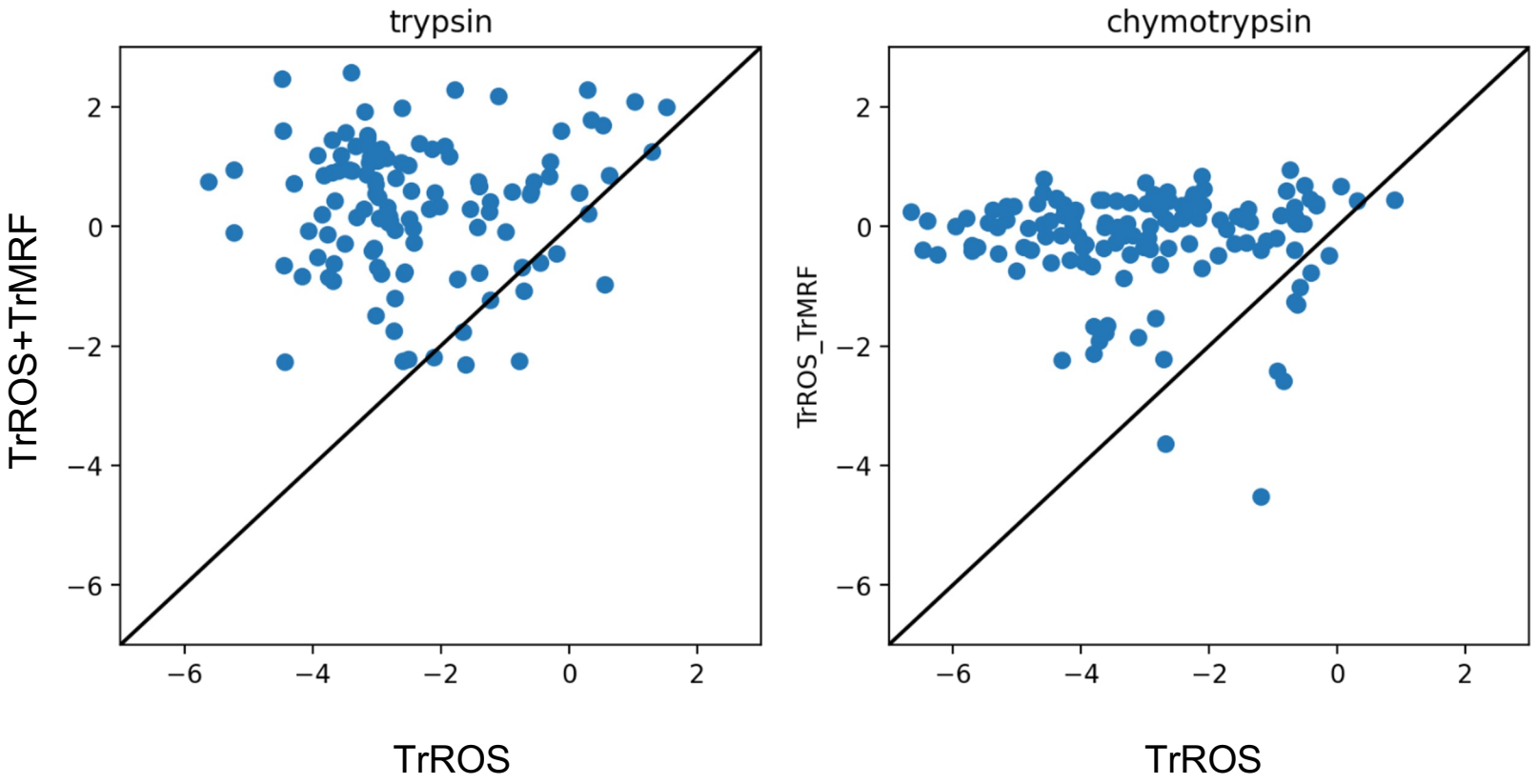
(removing proteins predicted as oligomers by AlphaFold)



interPAE > 15
pLDDT > 70

Results appear to hold even after filtering for oligomers

Combined loss significantly better!



Conclusions

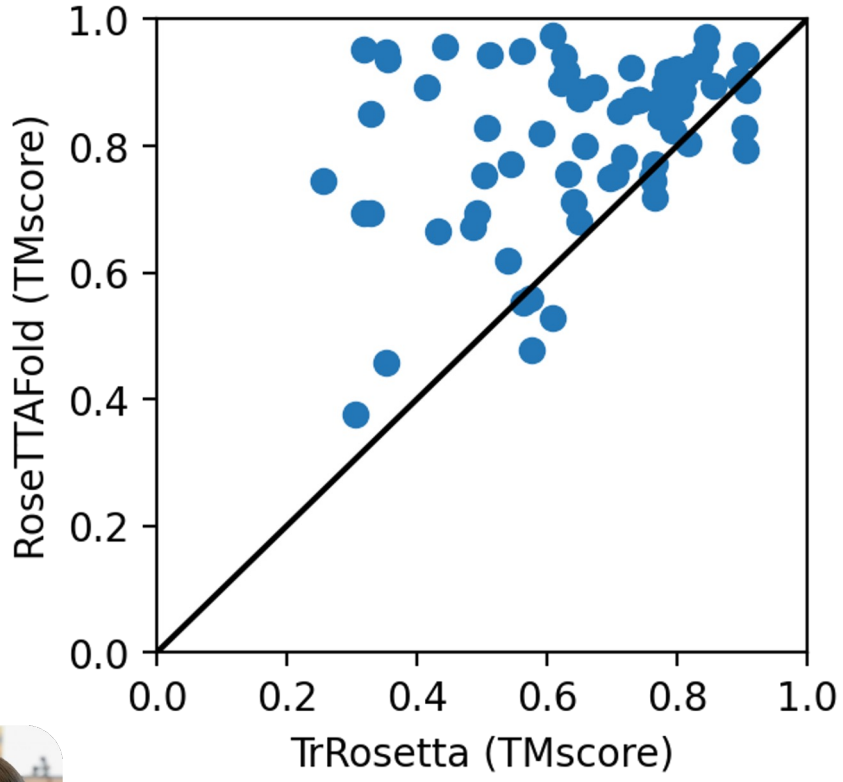
- Combing inverted NN that sees the entire conformational landscape with NN that optimizes energy given structure appears to results in more "stable" designs.

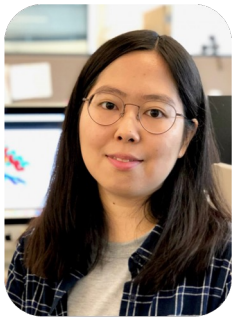
P(structure | sequence) ~ AbRelax (FF) ~ TrRosetta ~ **AlphaFold** ~ **RoseTTAFold**
P(sequence) = 1.0

P(sequence | structure) ~ Rosetta ~ TrMRF
P(structure) = 1.0

P(structure, sequence) ~ P(structure | sequence) * P(sequence | structure)

Modern methods: RoseTTAFold / AlphaFold





Minkyung Baek

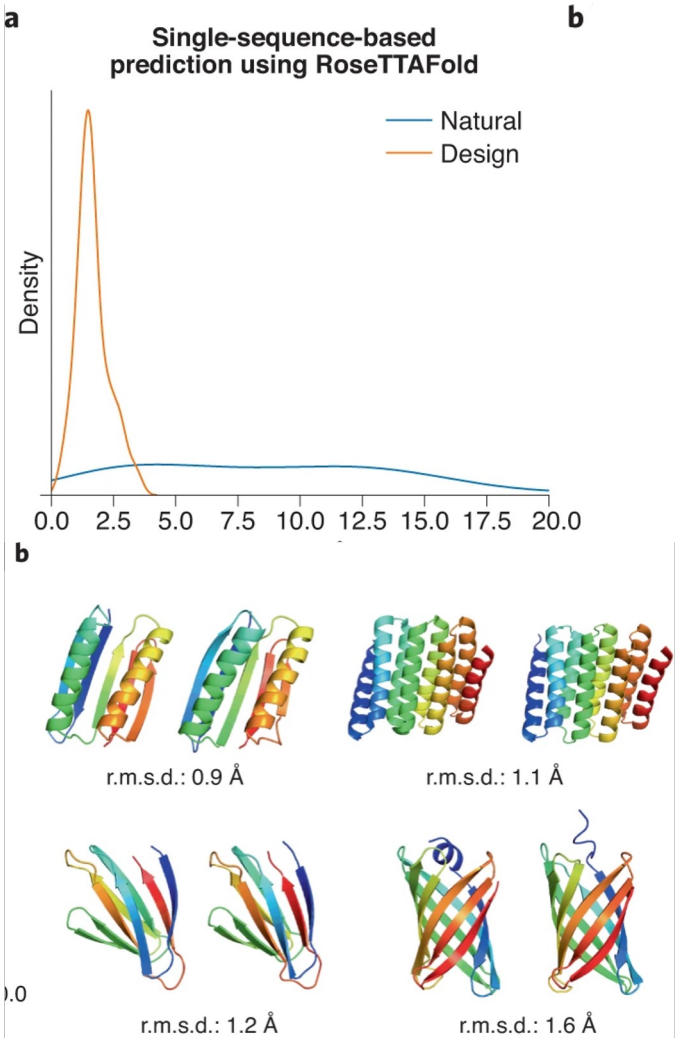


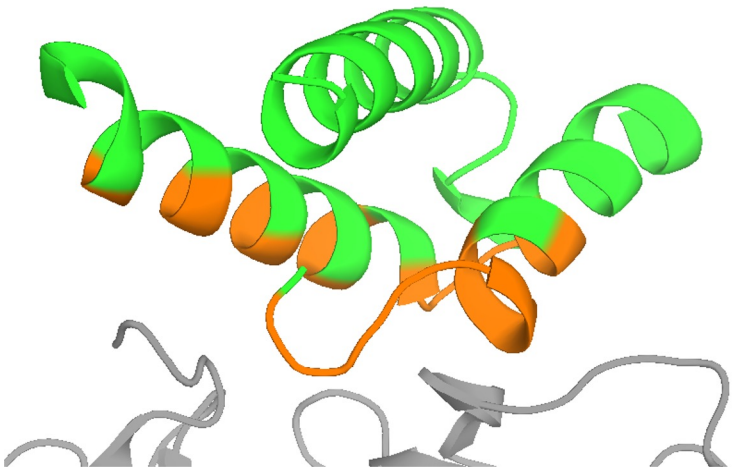
Fig. 1: RoseTTAFold accurately predicts structures of de-novo-designed proteins from their amino acid sequences.

Jue Wang

Partial Hallucination

Can we use DL models to design functional proteins?
Scaffolding functional motifs with hallucination

Goal: small, stable protein
with desired motif



Doug Tischer | Sidney Lisanza | David Juergens

New Results | Follow this preprint

Deep learning methods for designing proteins scaffolding functional sites

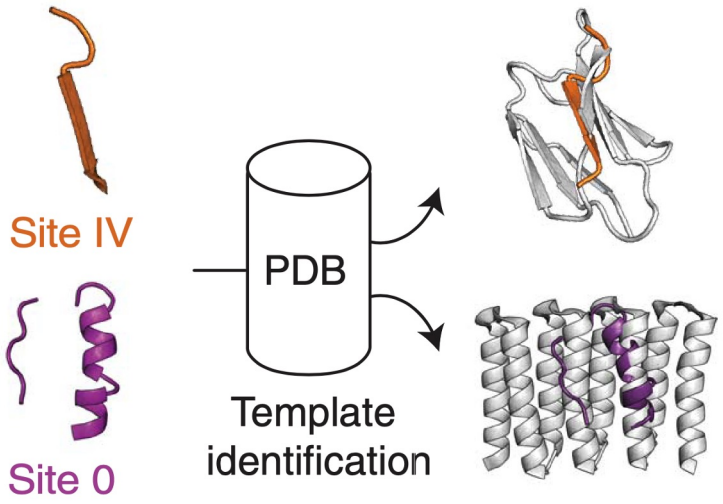
Jue Wang, Sidney Lisanza, David Juergens, Doug Tischer, Ivan Anishchenko, Minkyung Baek, Joseph L. Watson, Jung Ho Chun, Lukas F. Milles, Justas Dauparas, Marc Expòsit, Wei Yang, Amijai Saragovi, Sergey Ovchinnikov, David Baker

doi: https://doi.org/10.1101/2021.11.10.468128

https://github.com/RosettaCommons/RFDesign

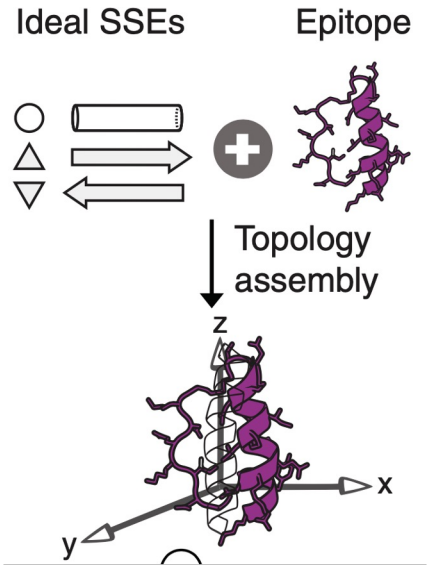
Current state-of-the-art

Graft motif into existing protein



Limited by databases

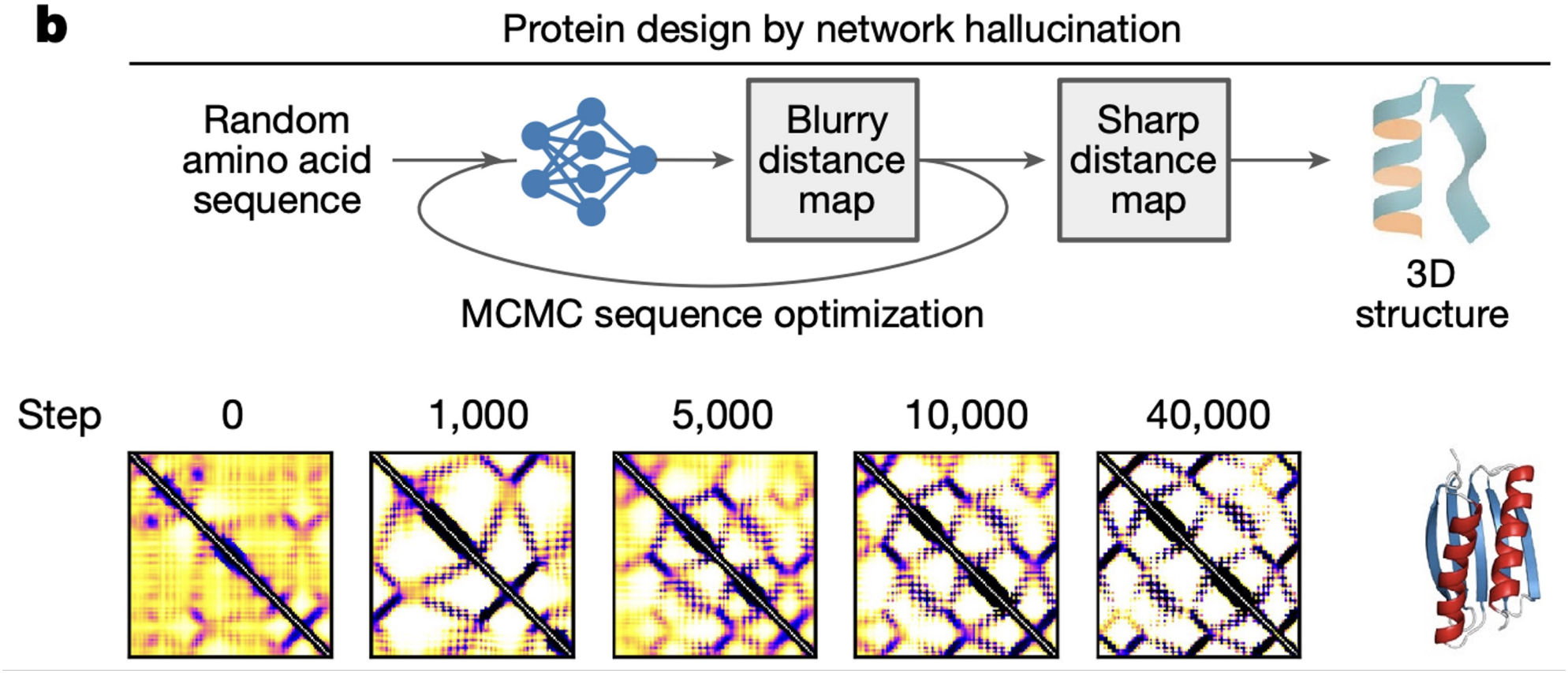
Build up scaffold from fragments

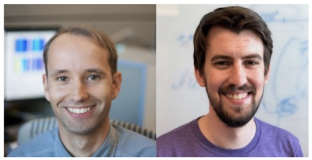


Limited to simple motifs; Need to choose fold beforehand

Sesterhehn et al. 2020

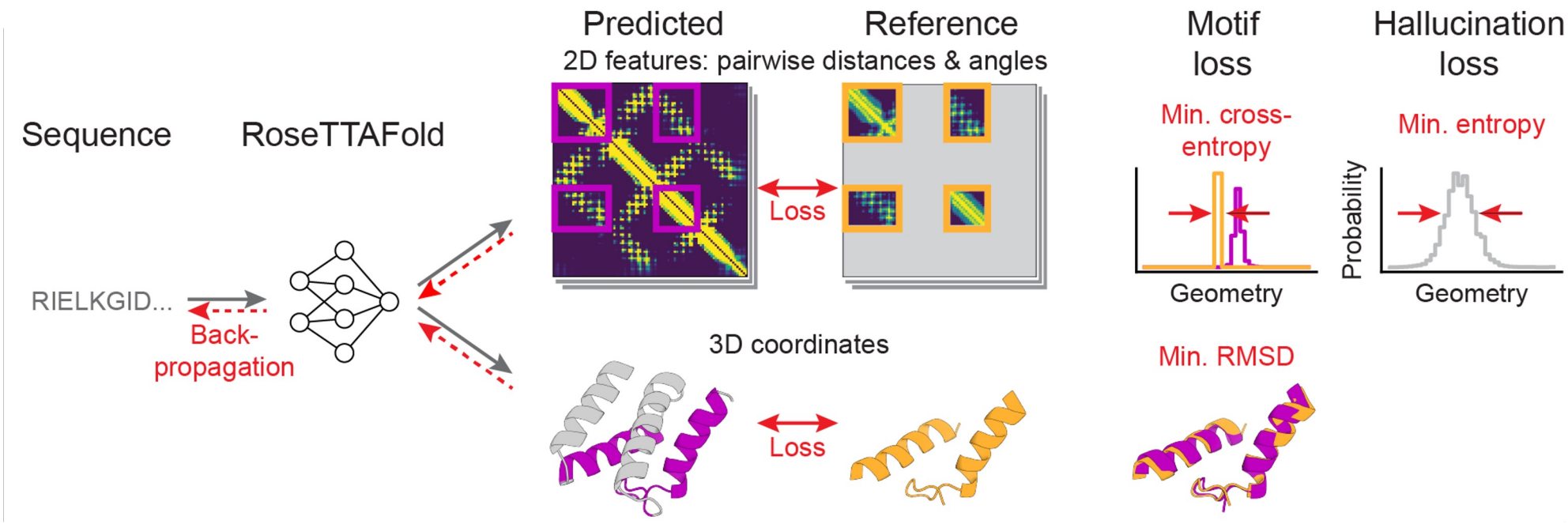
“Hallucination” can create new structures without referencing existing backbone



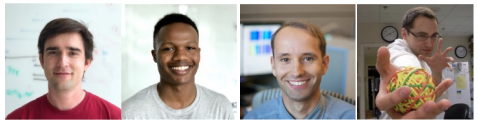


Anishchenko, Pellock et al. 2021

Partially constrained hallucination: embedding pre-specified motif

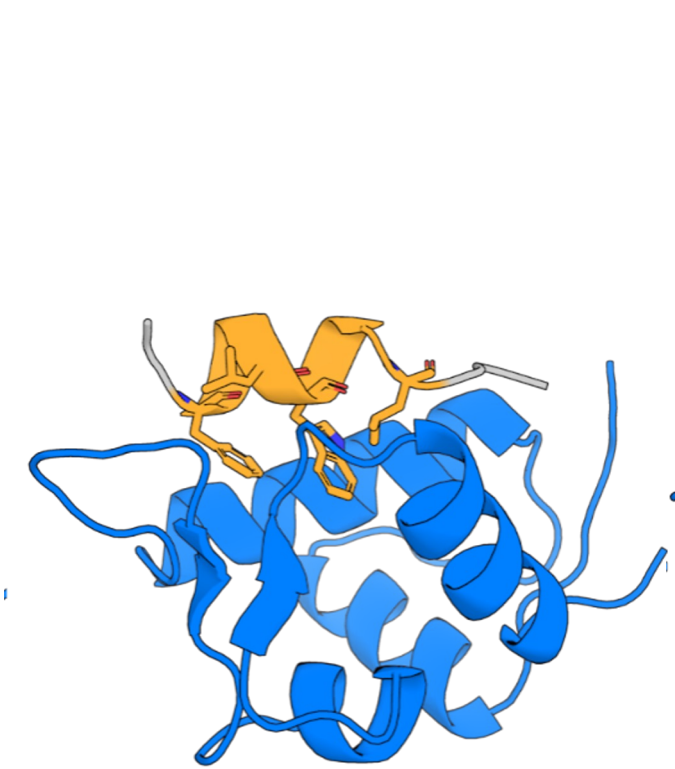


- Can sample all of protein structure space…?
- No assumptions about topology or secondary structure of scaffold

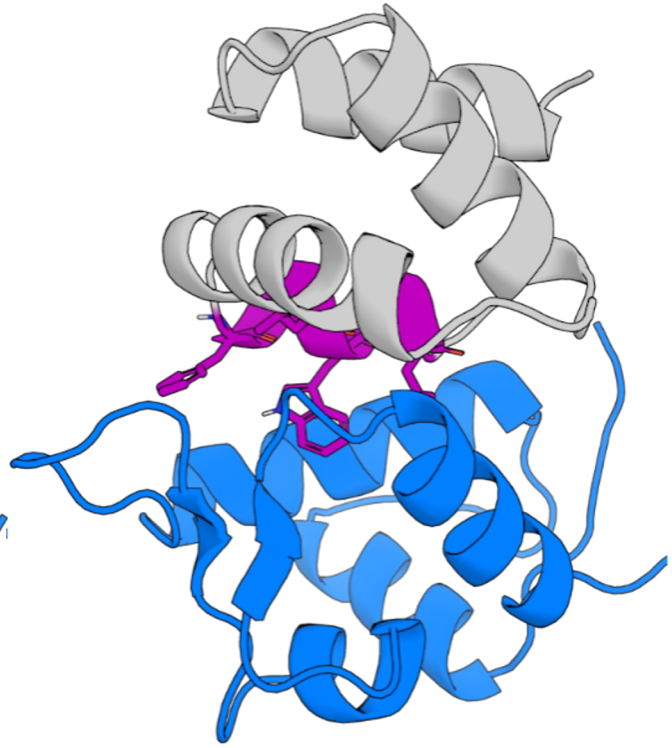


Doug Sidney Ivan Sergey

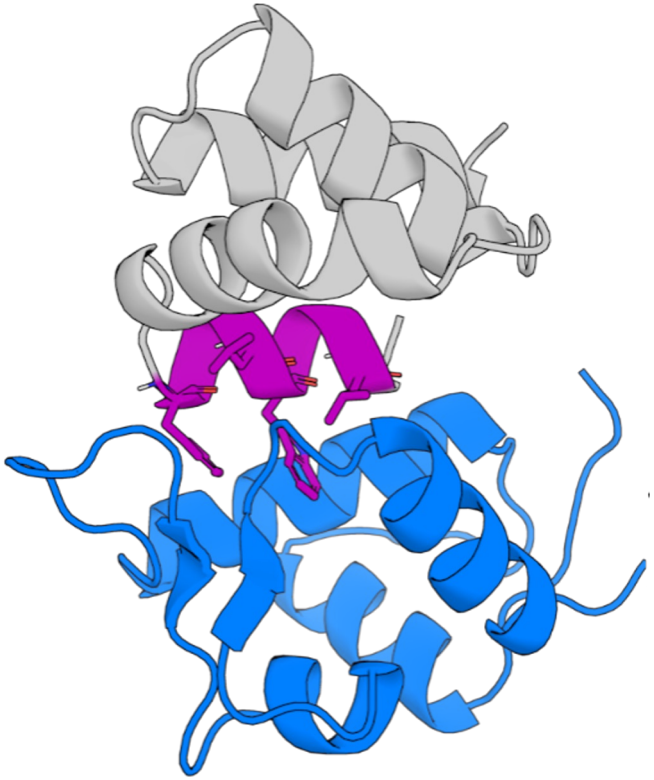
Scaffolding p53 helix binding to mdm2



Native

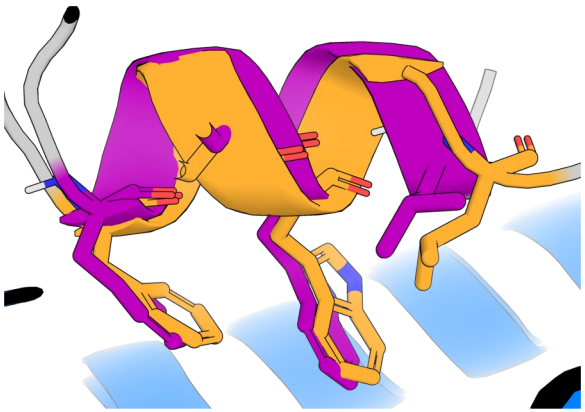


RF hallucination

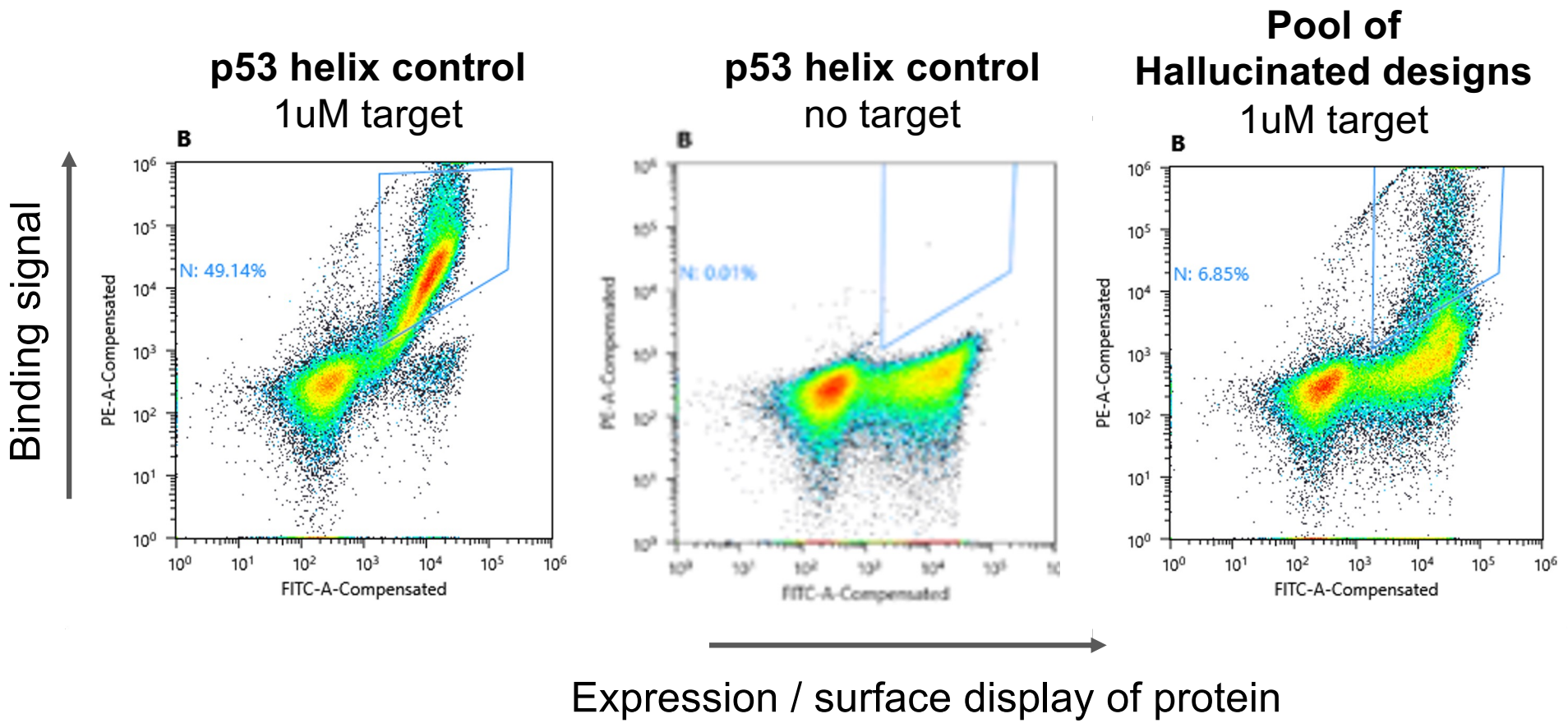


AF2 prediction

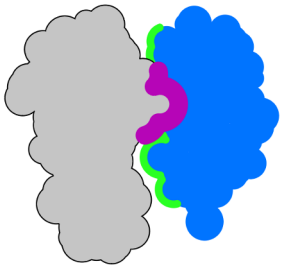
Native motif (p53 helix)
Design motif
Binding partner (mdm2)



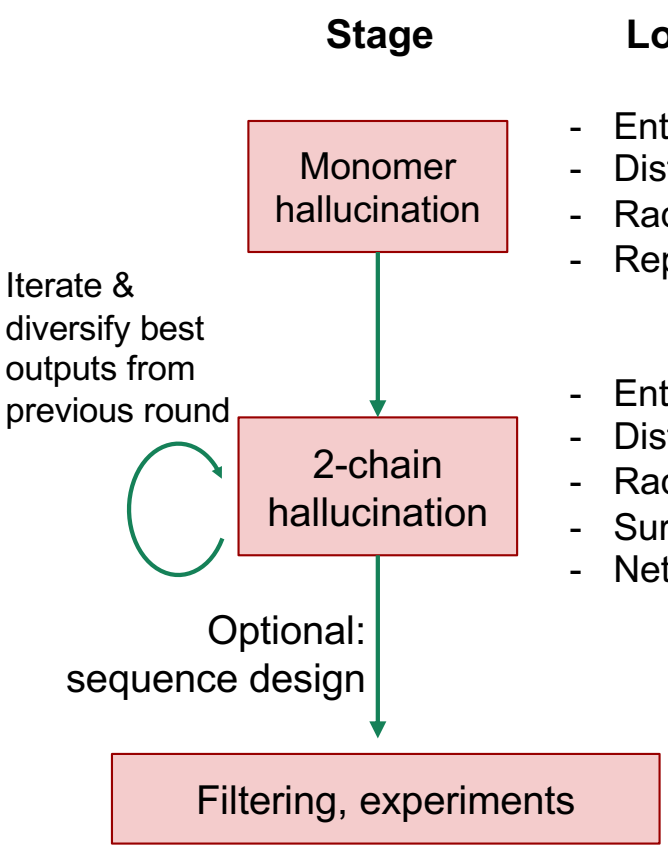
Designed p53 mimetics bind to mdm2 in yeast display

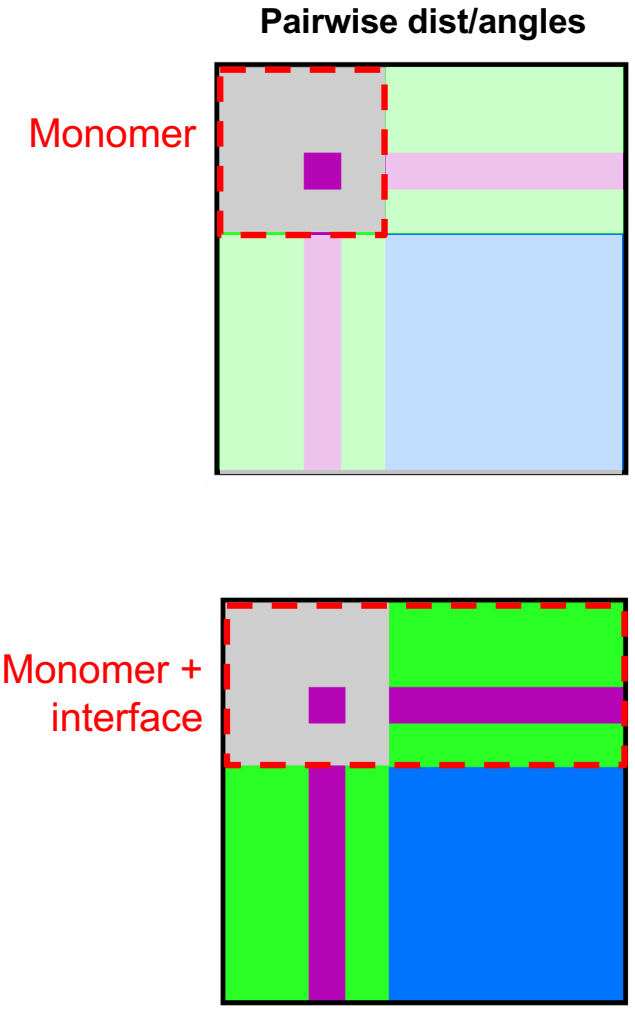


Binder hallucination pipeline



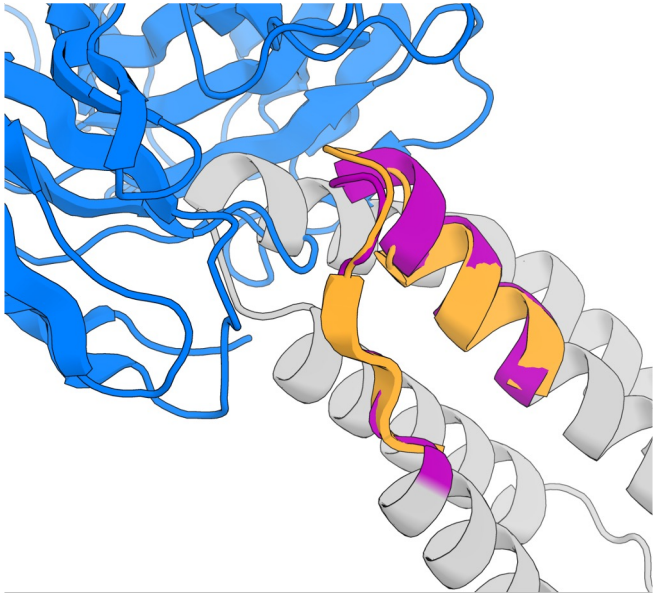
Hallucinated binder
Hallucinated interface
Stub (small interface motif)
Binding target (receptor)





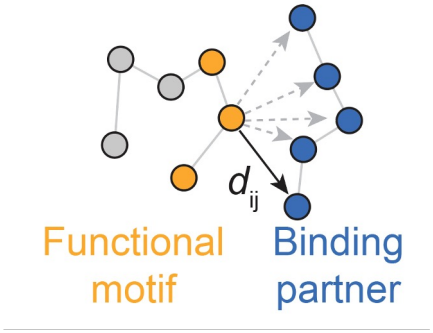
Problem-specific loss functions

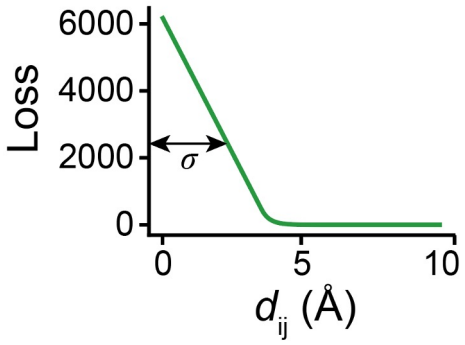
Problem: Hallucinations clash with binding partner (RSVF site V)



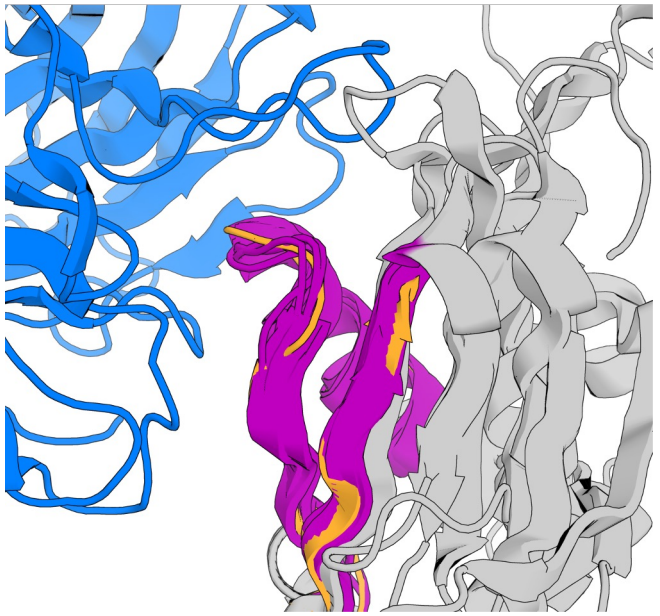
Native motif
Design motif
Binding partner

Solution: Include repulsive loss





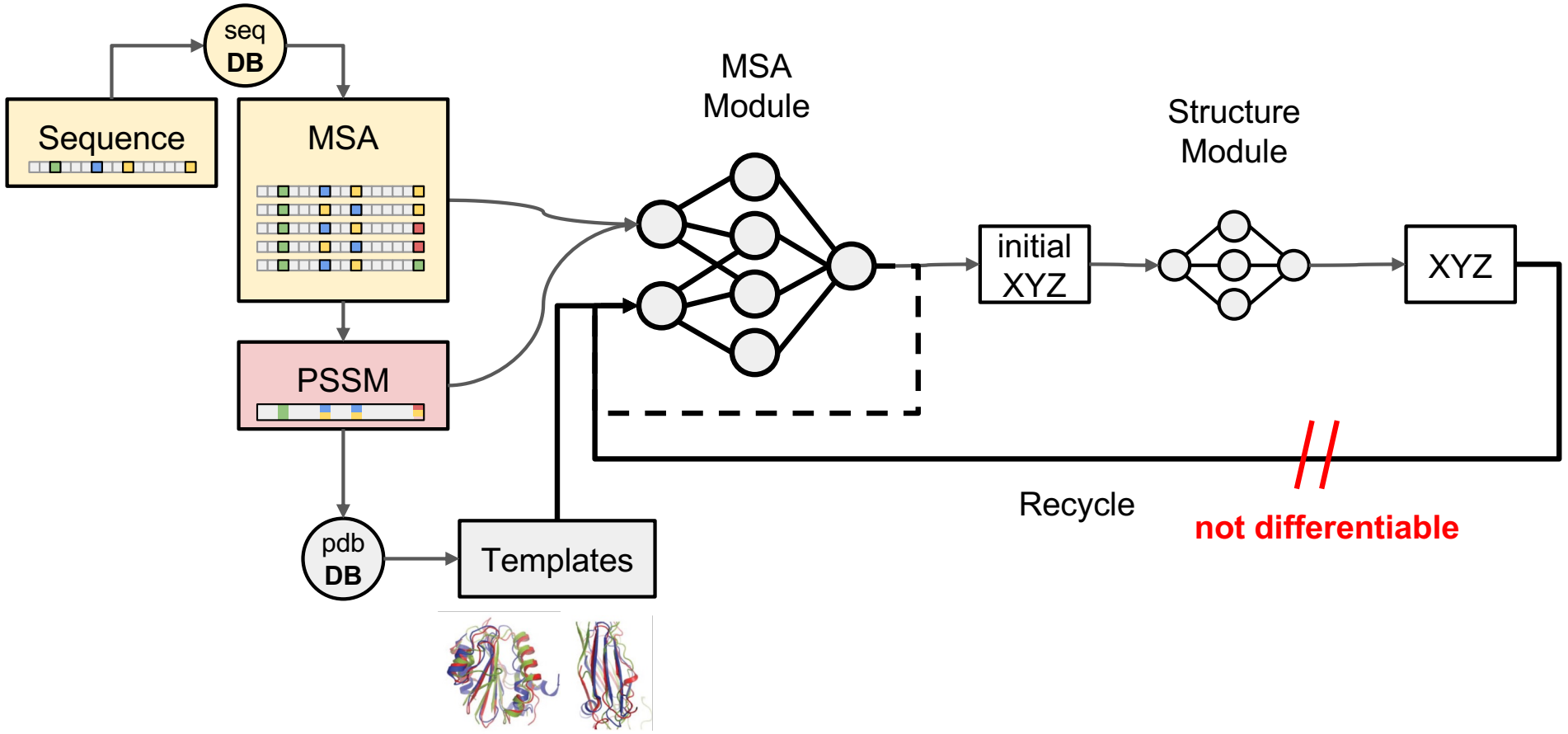
Result: (Almost) no clashes



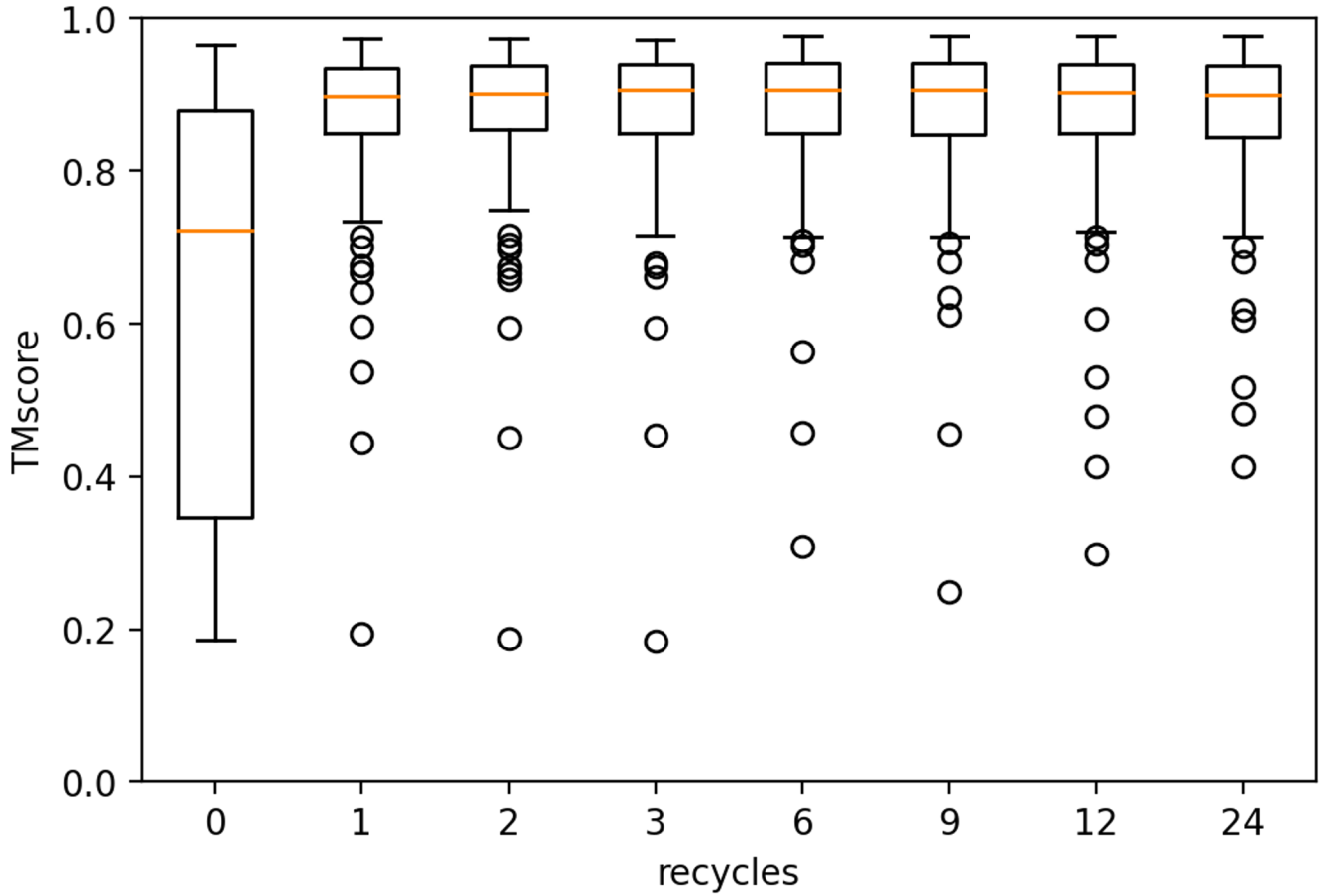
Back to Sergey

AfDesign

Alphafold2


seq
DB
Sequence
MSA
PSSM
pdb
DB
Templates
MSA
Module
Structure
Module
initial
XYZ
XYZ
Recycle
not differentiable

AlphaFold: Number of recycles needed to predict denovo designed proteins



Why is AlphaFold sooooo bad at num_recycle=0?

James Roney

Why is num_recycle=0 so bad?

```
if self.config.num_recycle:
  emb_config = self.config.embeddings_and_evoformer
  prev = {
      'prev_pos': jnp.zeros(
          [num_residues, residue_constants.atom_type_num, 3]),
      'prev_msa_first_row': jnp.zeros(
          [num_residues, emb_config.msa_channel]),
      'prev_pair': jnp.zeros(
          [num_residues, num_residues, emb_config.pair_channel]),
  }

  .....

else:
  prev = {}
  num_iter = 0
```

```
if c.recycle_pos and 'prev_pos' in batch:
  prev_pseudo_beta = pseudo_beta_fn(
      batch['aatype'], batch['prev_pos'], None)
  dgram = dgram_from_positions(prev_pseudo_beta, **self.config.prev_pos)
  pair_activations += common_modules.Linear(
      c.pair_channel, name='prev_pos_linear')(
          dgram)

if c.recycle_features:
  if 'prev_msa_first_row' in batch:
    prev_msa_first_row = hk.LayerNorm([-1],
                                      True,
                                      True,
                                      name='prev_msa_first_row_norm')(
                                          batch['prev_msa_first_row'])
    msa_activations = msa_activations.at[0].add(prev_msa_first_row)

  if 'prev_pair' in batch:
    pair_activations += hk.LayerNorm([-1],
                                     True,
                                     True,
                                     name='prev_pair_norm')(
                                         batch['prev_pair'])
```

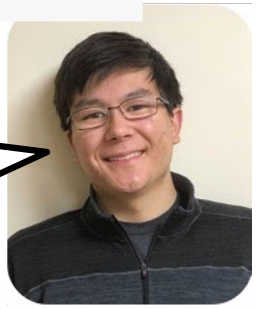
If num_recycle=0, this disables a series of layers

bugfix

```
if self.config.num_recycle:
  emb_config = self.config.embeddings_and_evoformer
  prev = {
      'prev_pos': jnp.zeros(
          [num_residues, residue_constants.atom_type_num, 3]),
      'prev_msa_first_row': jnp.zeros(
          [num_residues, emb_config.msa_channel]),
      'prev_pair': jnp.zeros(
          [num_residues, num_residues, emb_config.pair_channel]),
  }

  .....

else:
  prev = {}
  num_iter = 0
```

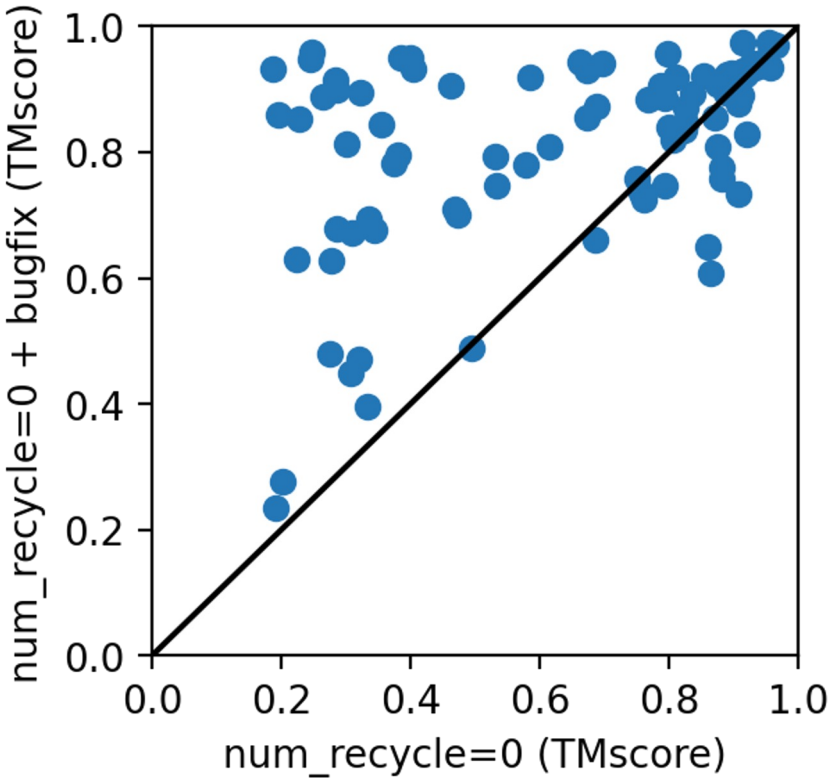
Alternative (without making source code changes)

```
model_config.model.num_recycle = 1
model_config.data.common.num_recycle = 1
processed_feature_dict["num_iter_recycling"] = np.array([0])
```

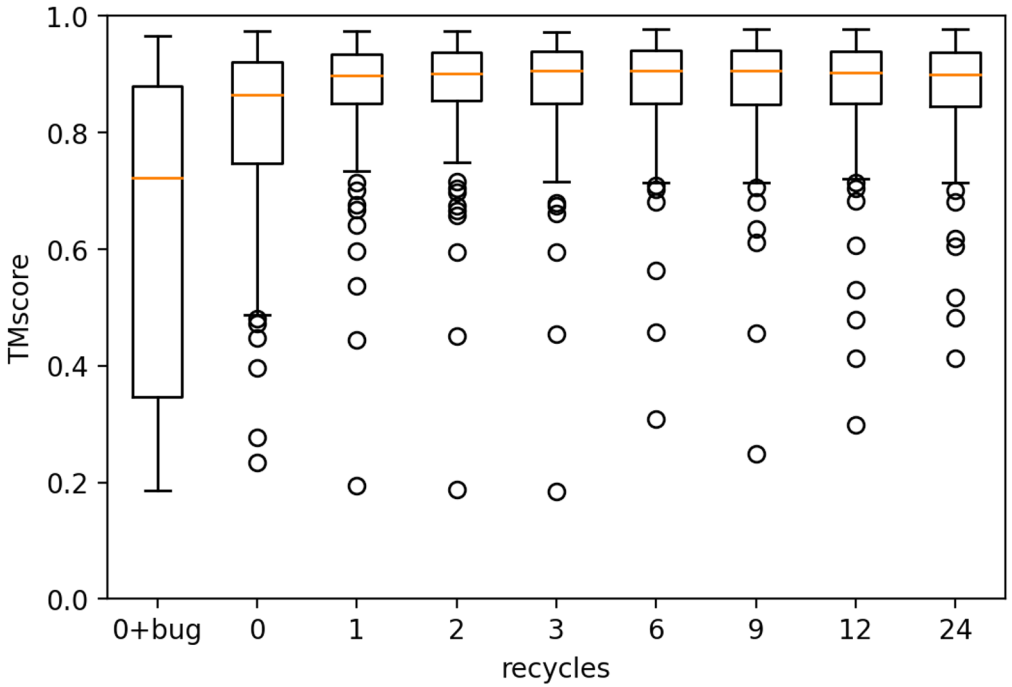


James Roney

Bugfix significantly improves the accuracy of denovo designed proteins at num_recycle=0



Number of recycles needed to predict denovo designed proteins

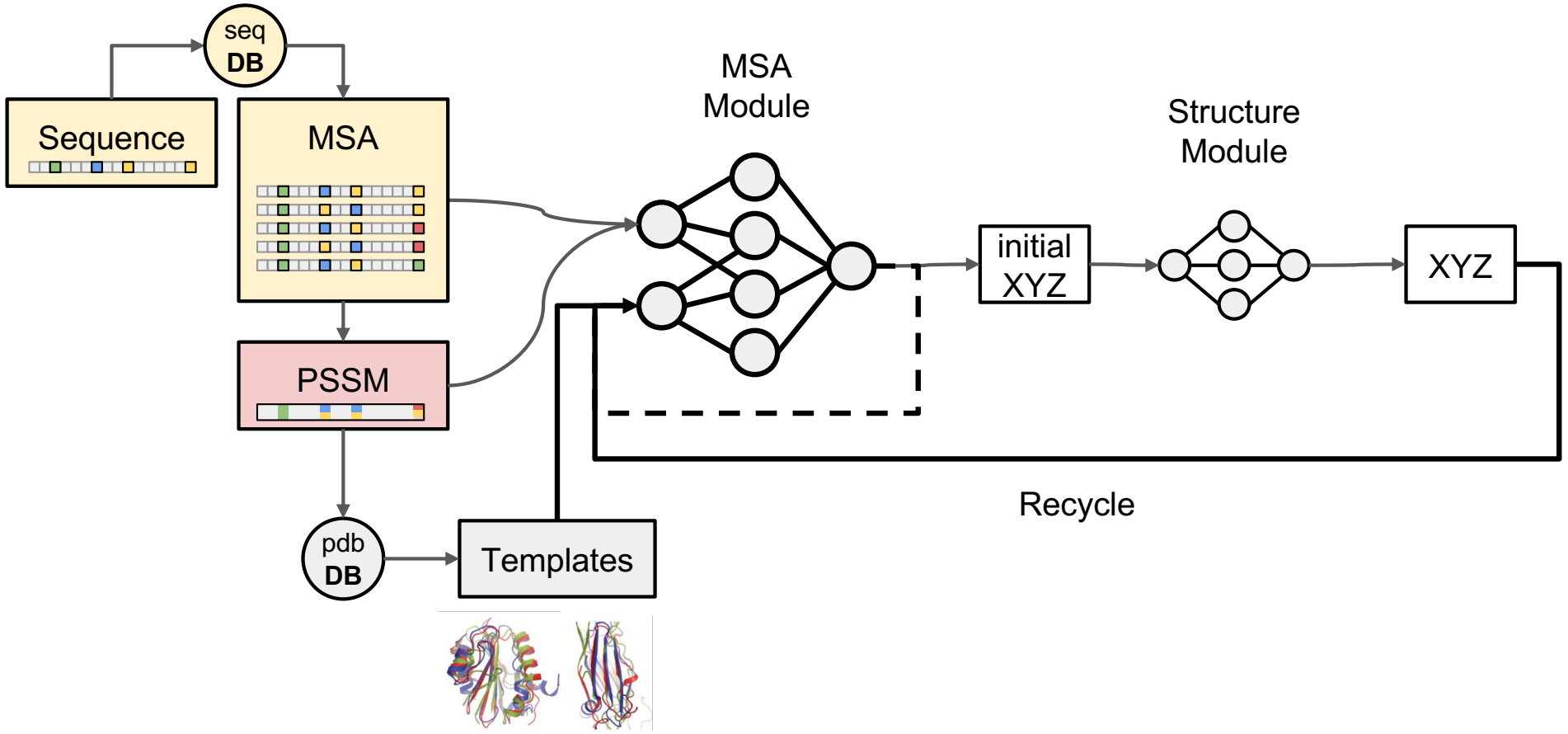


By changing a few configs we can reduce runtime! (using Google Colab Free = Tesla K80)

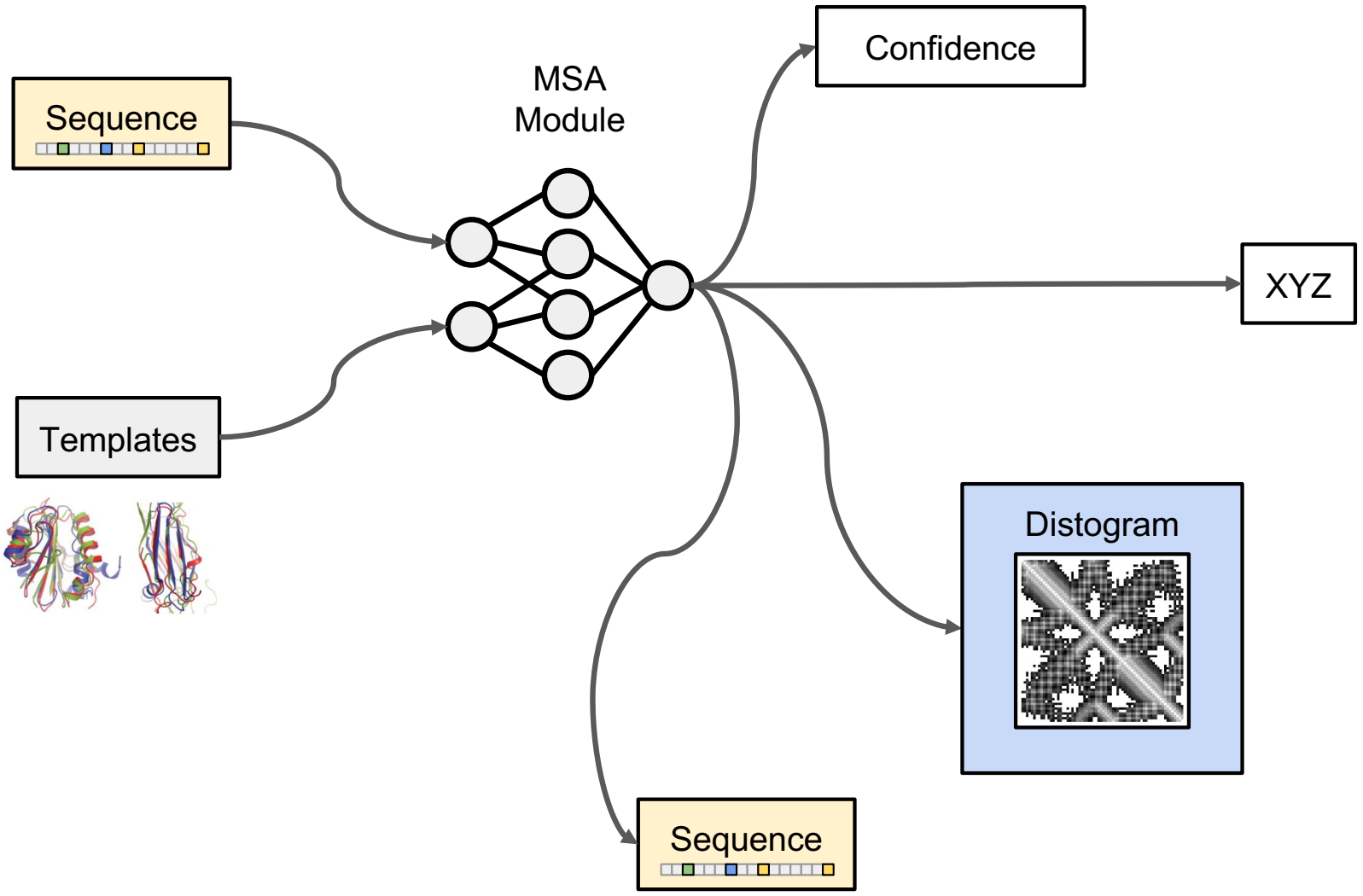
```
length=100 length=50
     26.7s     13.9s (msa=512, subbatch=4,    recycles=3, backprop=False)
      2.5s      0.8s (msa=1,   subbatch=4,    recycles=3, backprop=False)
      1.9s      0.5s (msa=1,   subbatch=None, recycles=3, backprop=False)
      0.5s      0.1s (msa=1,   subbatch=None, recycles=0, backprop=False)
      1.8s      0.4s (msa=1,   subbatch=None, recycles=0, backprop=True )
      3.2s      1.1s (msa=1,   subbatch=4,    recycles=0, backprop=True )
```

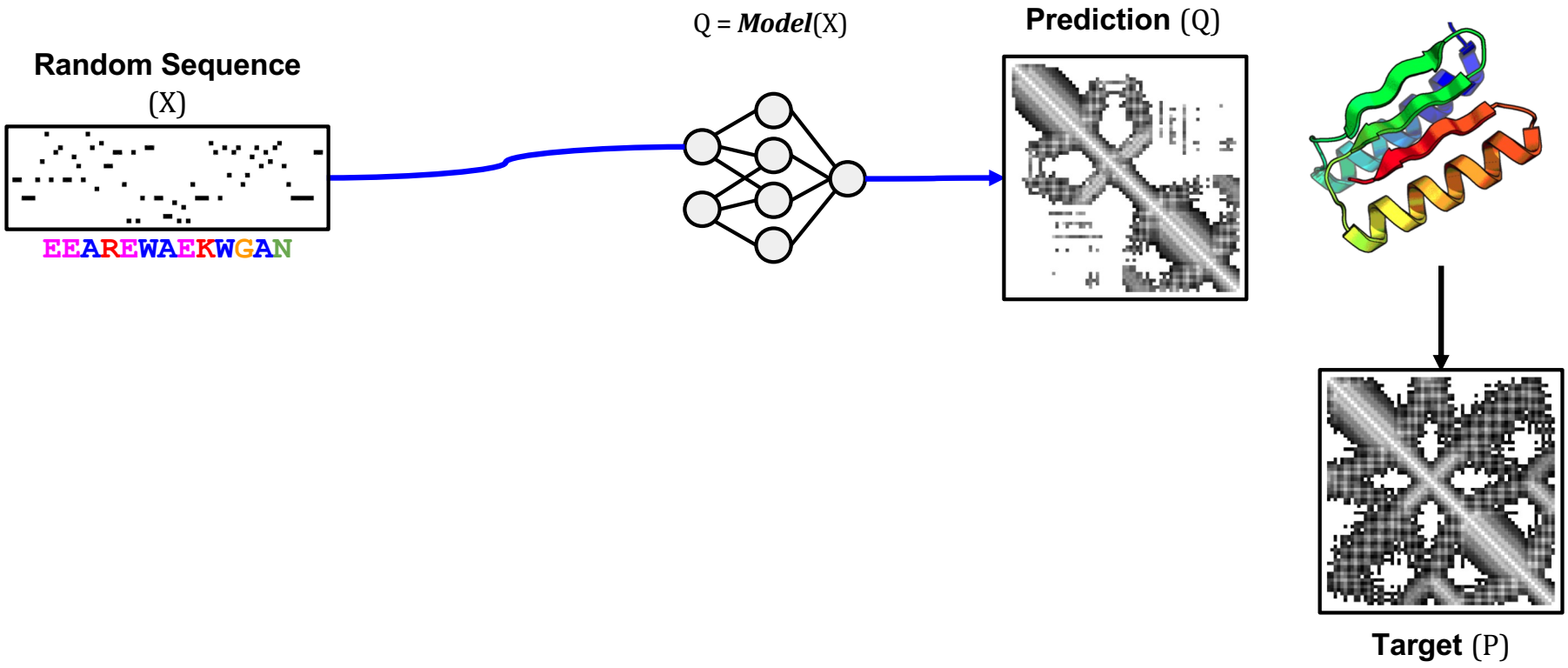
```python
# enable checkpointing (otherwise not enough memory to backprop)
cfg.model.global_config.use_remat = True

# disable subbatching
cfg.model.global_config.subbatch_size = None

# disable recycles
cfg.model.num_recycle = 1
cfg.data.common.num_recycle = 1
processed_feature_dict["num_iter_recycling"] = np.array([0])

# set max number of sequences
cfg.data.eval.max_msa_clusters = 1
cfg.data.common.max_extra_msa = 1
```

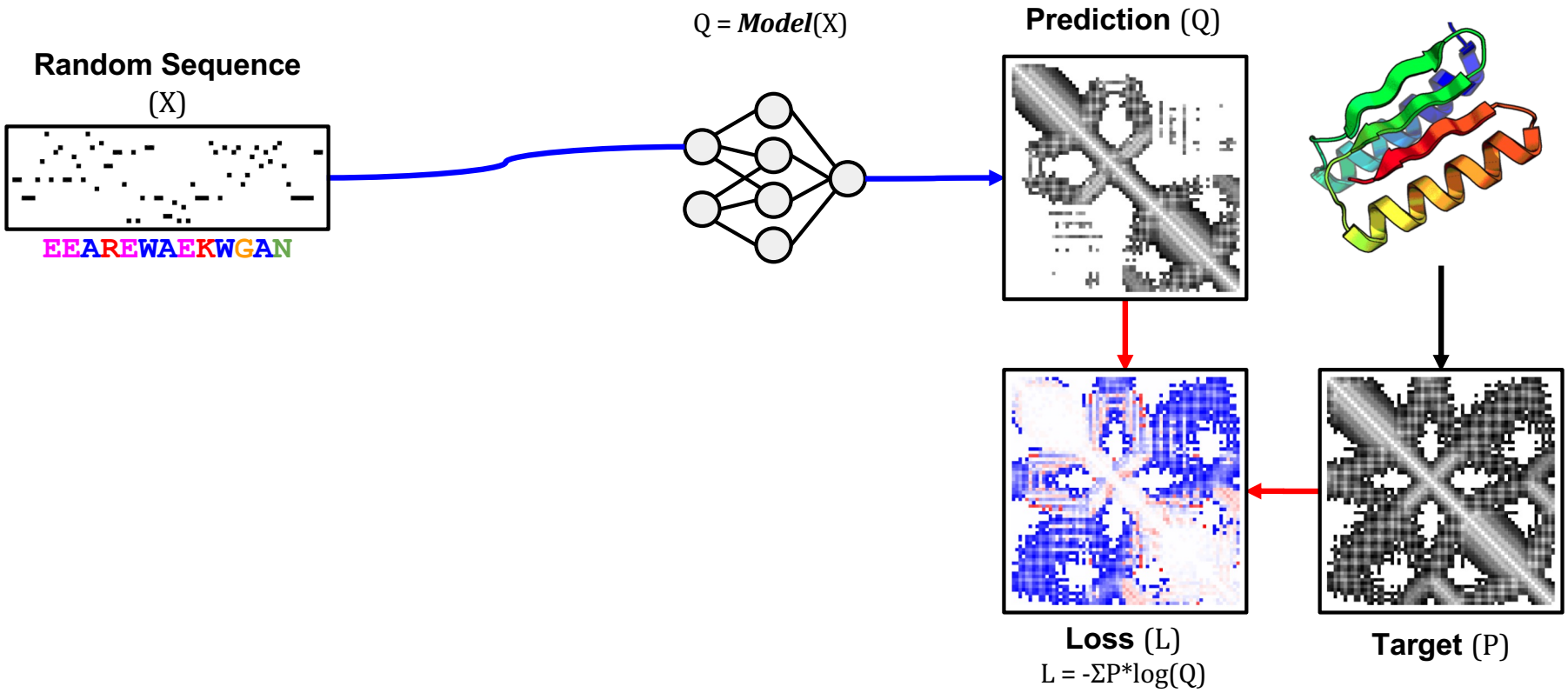
Alphafold2


seq
DB
Sequence
MSA
PSSM
pdb
DB
Templates
MSA
Module
initial
XYZ
Structure
Module
XYZ
Recycle

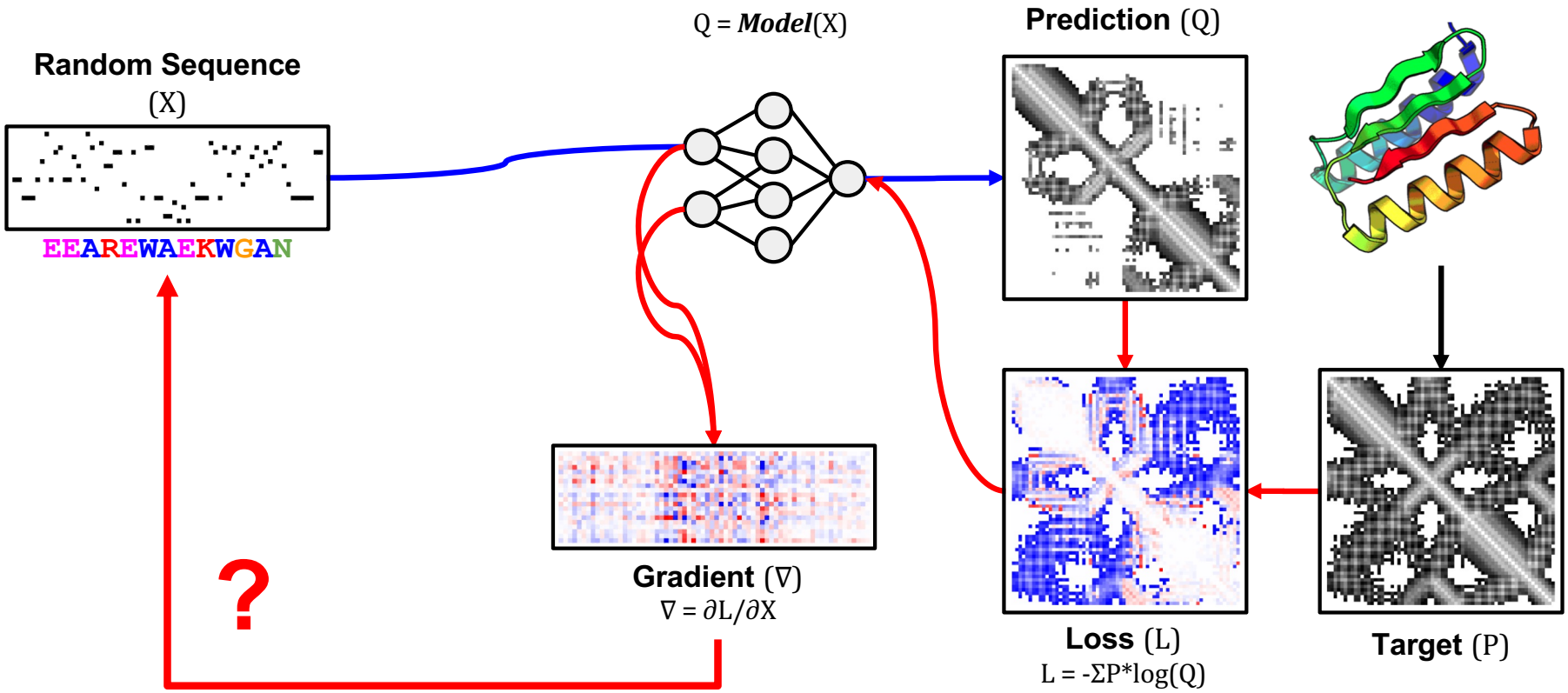
AlphaFold2: inputs/outputs




Random Sequence
(X)
EEAREWAEKWGAN
Q = *Model*(X)
Prediction (Q)
Target (P)


Random Sequence
(X)
EEAREWAEKWGAN
Q = *Model*(X)
Prediction (Q)
Loss (L)
L = -ΣP*log(Q)
Target (P)

How to backprop through discrete input?



Test Case - 1TEN (Fibronectin type-III domain 3)

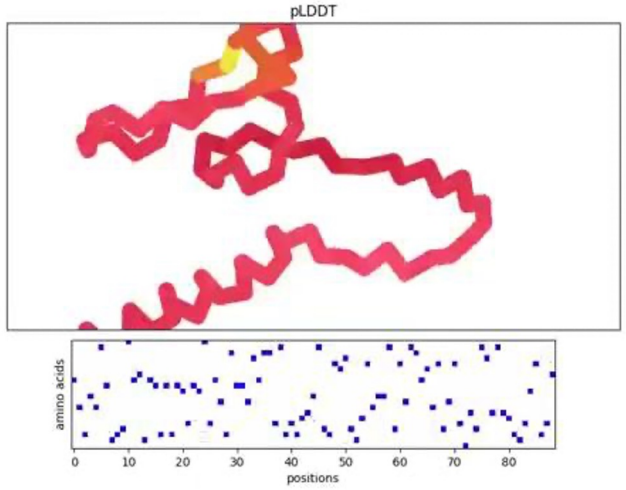
Starting from random sequence, the MCMC protocol failed[1] for this example, **RMSD = 4.36**

Let's try backprop!

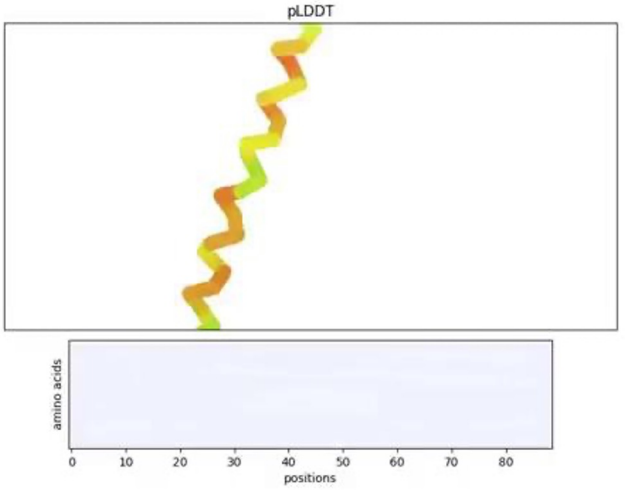
1. **Using AlphaFold for Rapid and Accurate Fixed Backbone Protein Design**
   Lewis Moffat, Joe G. Greener, David T. Jones
   https://doi.org/10.1101/2021.08.24.457549

1TEN - Inputs

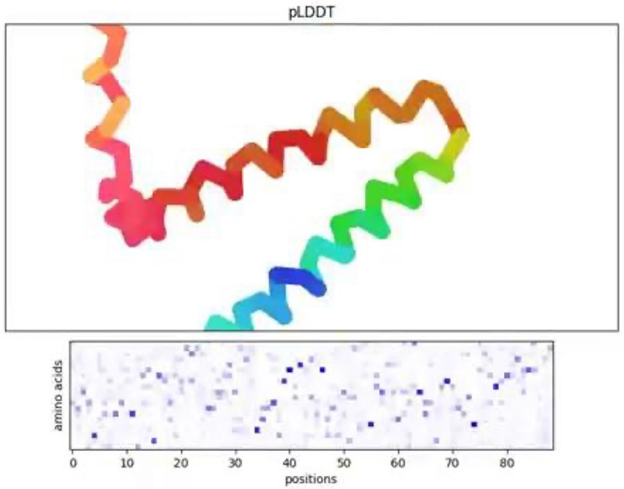
stop_gradient(argmax - softmax(logits)) + softmax(logits)



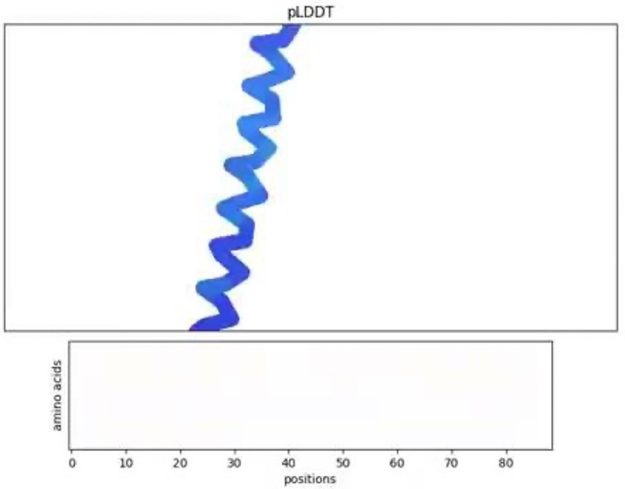
softmax(logits)



softmax(logits + gumbel_noise)

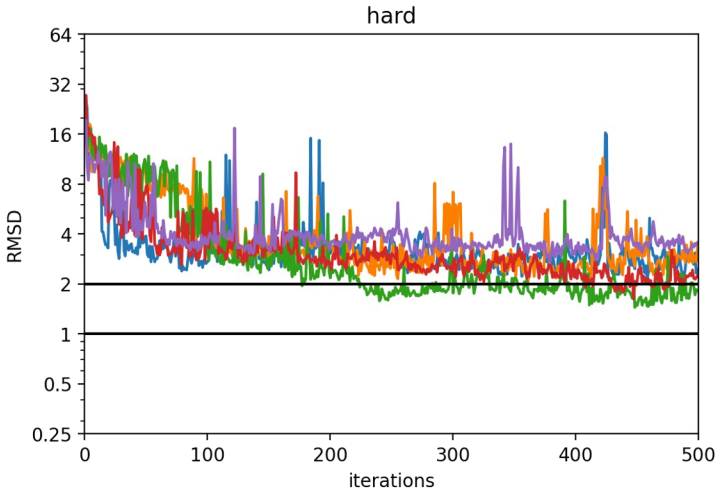


logits

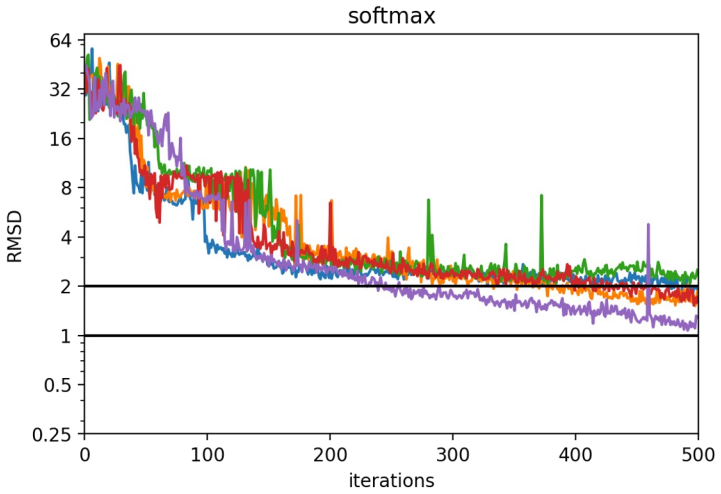


1TEN - Trajectories - RMSD

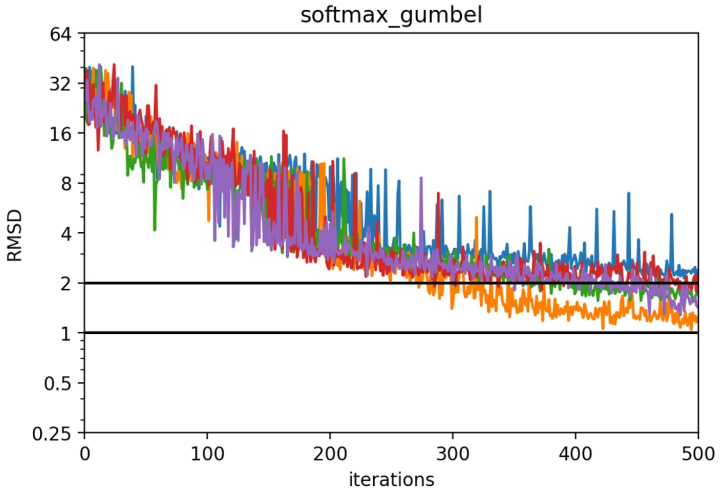
stop_gradient(argmax - softmax(logits)) + softmax(logits)



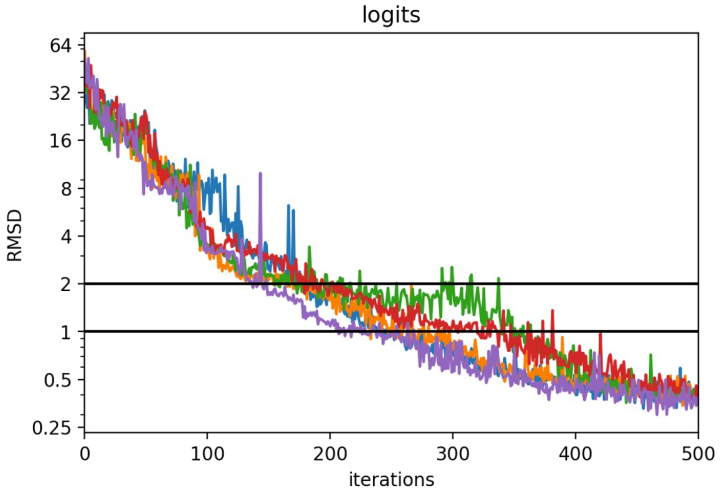
softmax(logits)



softmax(logits + gumbel_noise)

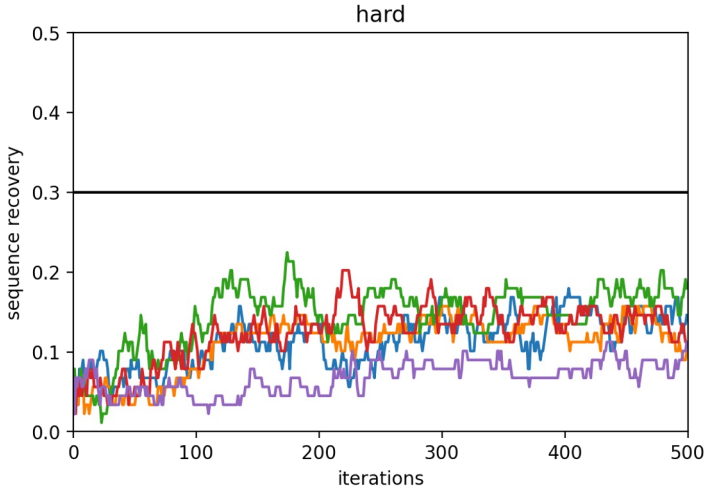


logits

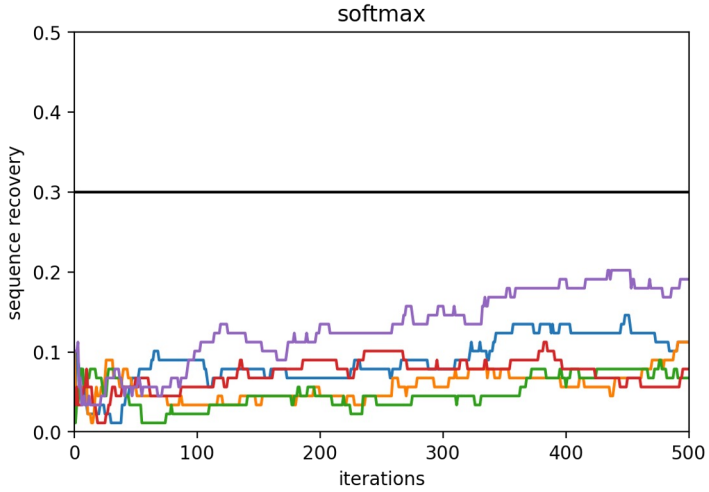


1TEN - Trajectories - Sequence Recovery

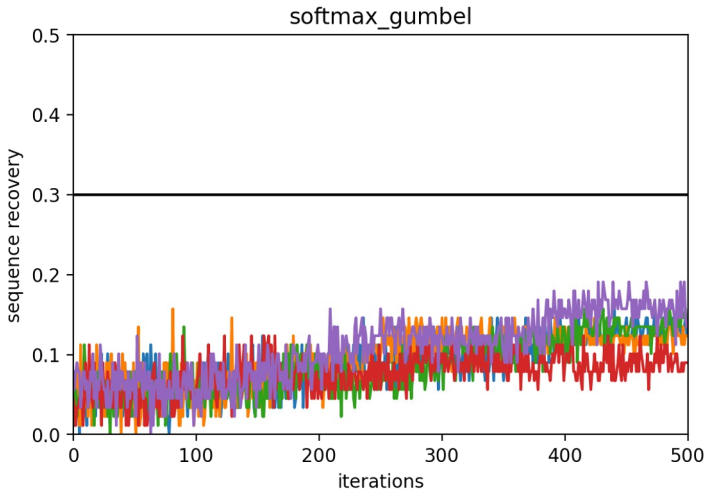
stop_gradient(argmax - softmax(logits)) + softmax(logits)



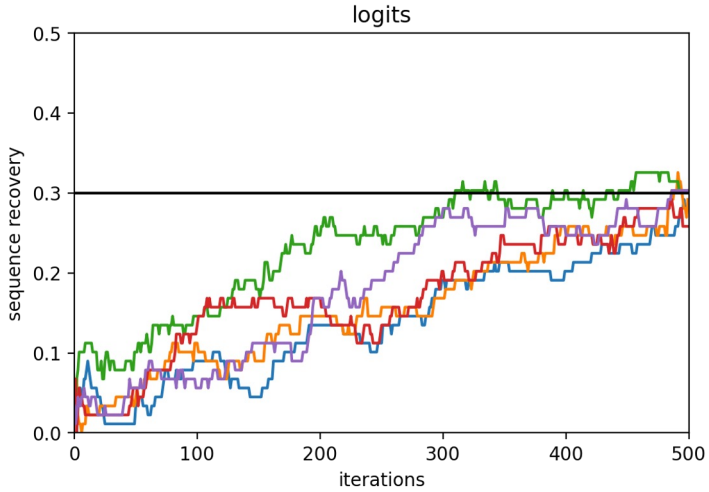
softmax(logits)



softmax(logits + gumbel_noise)



logits



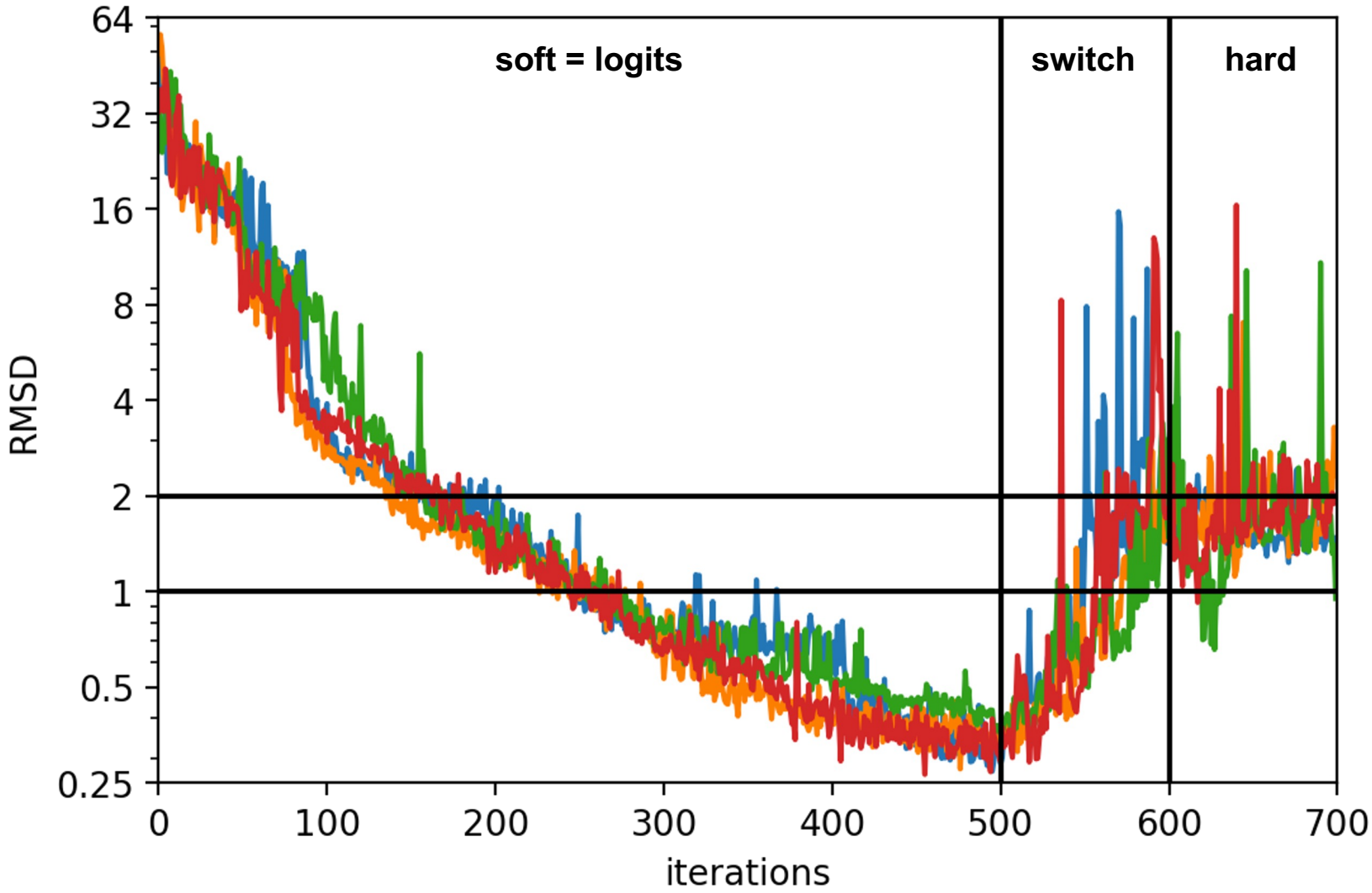
Observations

1. Decrete optimization for this complex topology is stuck at ~2-4 RMSD, conversely the sequence recovery is bad (< 0.15%)
2. If we "cheat" and use "logits" (continious values) as inputs we can find a solution with RMSD < 0.5!
3. Even though logits make "no sense" and technically "adverserial", surprisingly if you take max category at each position, the sequence recovery >30%!

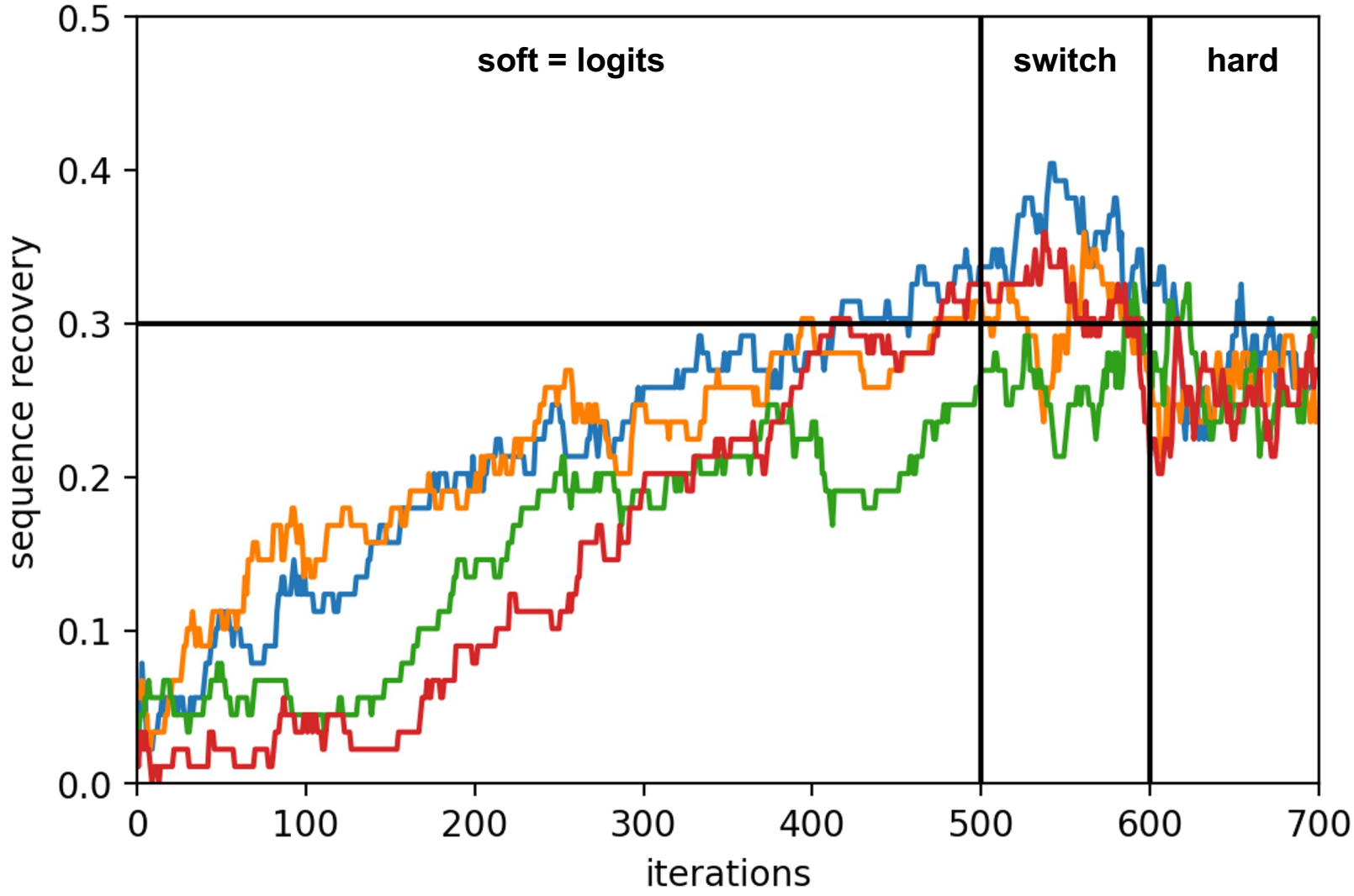
Which one should we use?

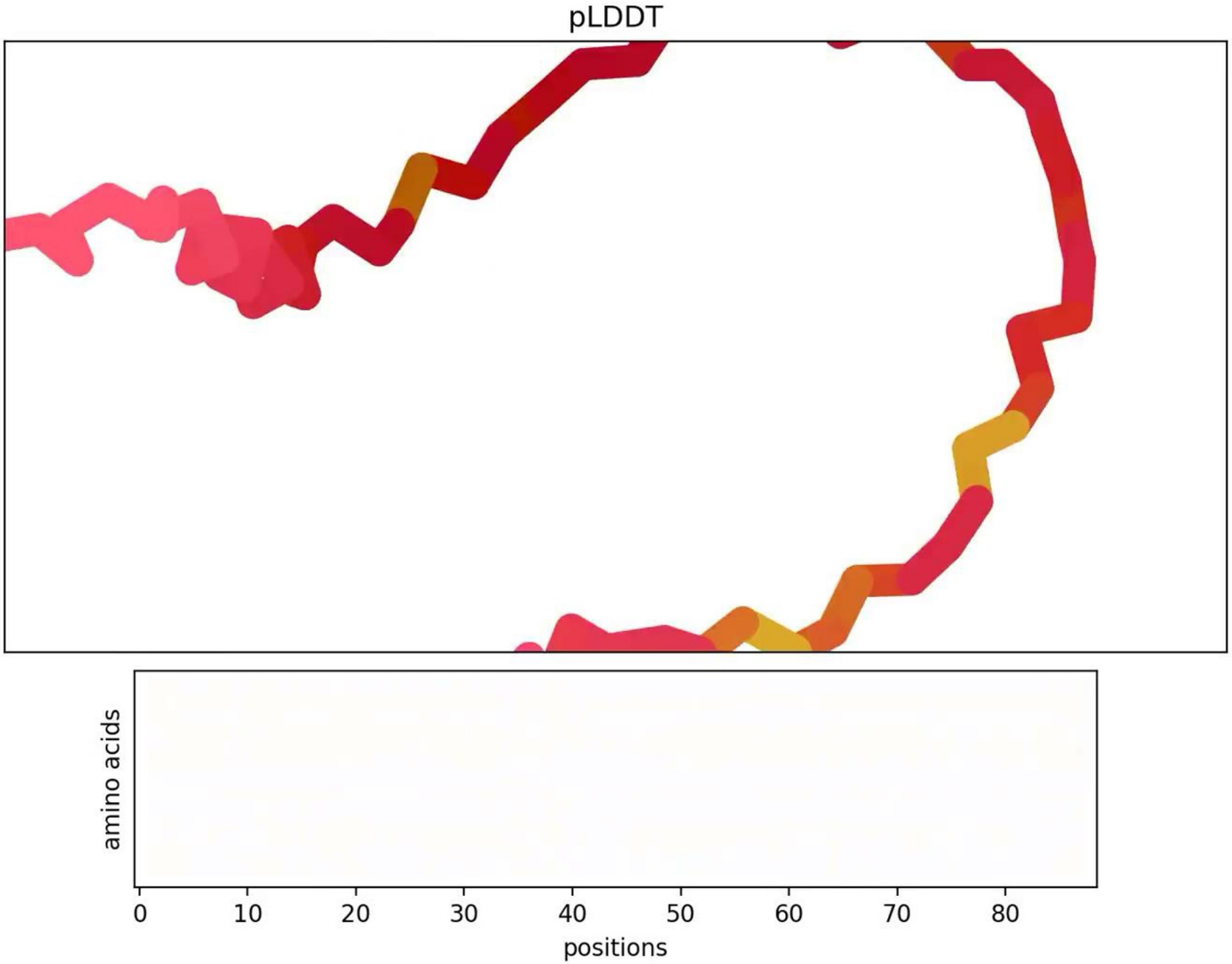

Both. Both is good.

Wait till convergences switch to hard!



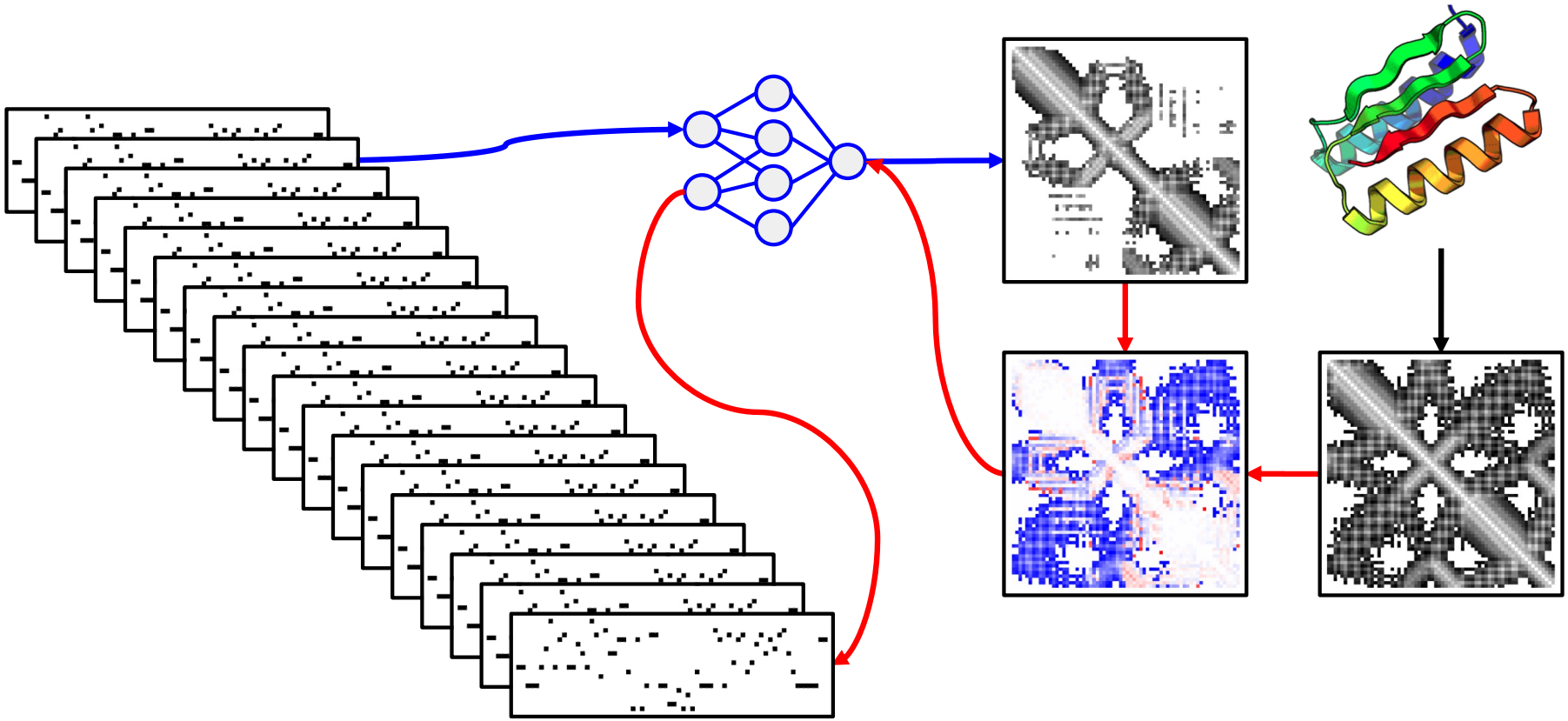
After switch, sequence recovery >25%



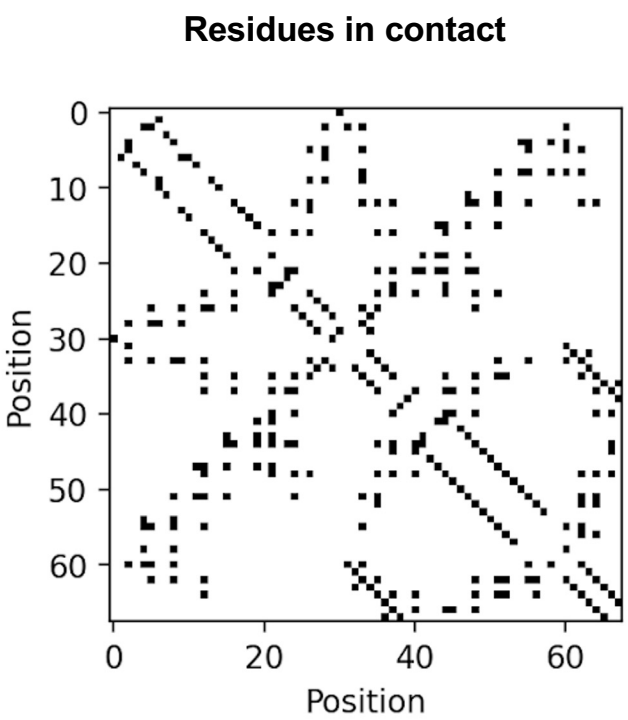

pLDDT
amino acids
0
10
20
30
40
50
60
70
80
positions

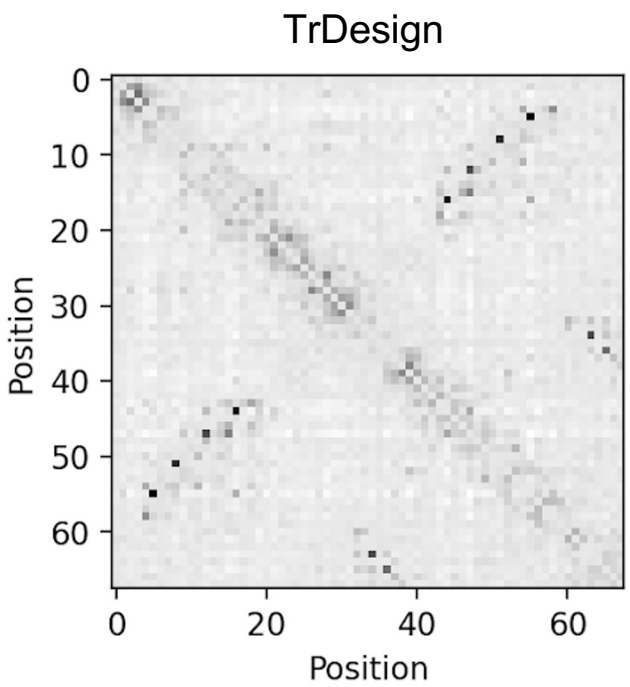
Are the sequences better than TrRosetta?

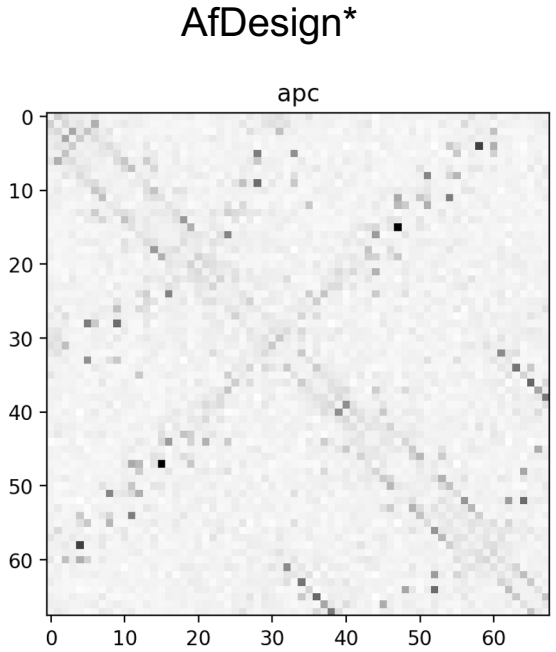
Run **AlphaFold** many time independently and generate many sequences



Coevolution test recovers more contacts!







*1.5K sequences

DEMO

Related work

Inverting prediction model for design

1. **Fast differentiable DNA and protein sequence optimization for molecular design**
   Johannes Linder, Georg Seelig
   https://arxiv.org/abs/2005.11275
2. **Using AlphaFold for Rapid and Accurate Fixed Backbone Protein Design**
   Lewis Moffat, Joe G. Greener, David T. Jones
   https://doi.org/10.1101/2021.08.24.457549
3. **AlphaDesign: A de novo protein design framework based on AlphaFold.**
   Michael Jendrusch, Jan O. Korbel, S. Kashif Sadiq
   https://www.biorxiv.org/content/10.1101/2021.10.11.463937v1

Design sequence given structure

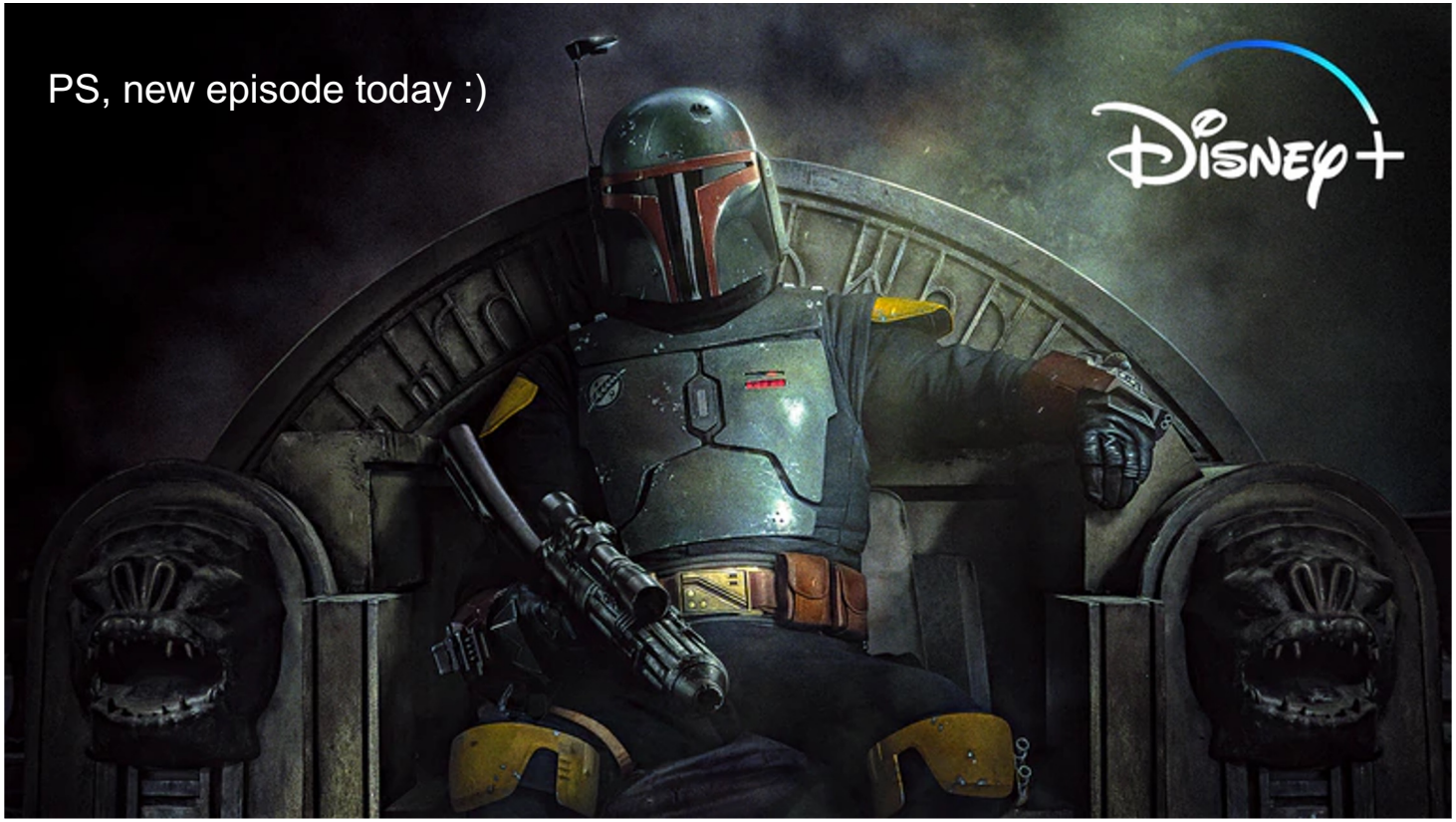
1. **Generative Models for Graph-Based Protein Design**
   John Ingraham, Vikas Garg, Regina Barzilay, Tommi Jaakkola
   http://papers.nips.cc/paper/9711-generative-models-for-graph-based-protein-design
2. **Protein Sequence Design with a Learned Potential**
   Namrata Anand, Raphael R. Eguchi, Alexander Derry, Russ B. Altman, Po-Ssu Huang
   https://www.biorxiv.org/content/10.1101/2020.01.06.895466v1
3. **Learning from Protein Structure with Geometric Vector Perceptrons**
   Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael J.L. Townshend, Ron Dror
   https://arxiv.org/abs/2009.01411
4. **TERMinator: A Neural Framework for Structure-Based Protein Design using Tertiary Repeating Motifs**
   AlexLi, Vikram Sundar, Gevorg Grigoryan, Amy Keating
   https://www.mlsb.io/papers_2021/MLSB2021_TERMinator:_A_Neural_Framework.pdf

Design sequence given general fold description

1. **Design of metalloproteins and novel protein folds using variational autoencoders**
   Joe G. Greener, Lewis Moffat & David T Jones
   https://www.nature.com/articles/s41598-018-34533-1

GORDON AND BETTY MOORE FOUNDATION
EARLY INDEPENDENCE AWARD
SIMONS FOUNDATION
NSF
Rosetta Commons

Thank you!



CODE: github.com/sokrypton

Collaborate? Join? **solab.org**

EXTRA

How to backprop?

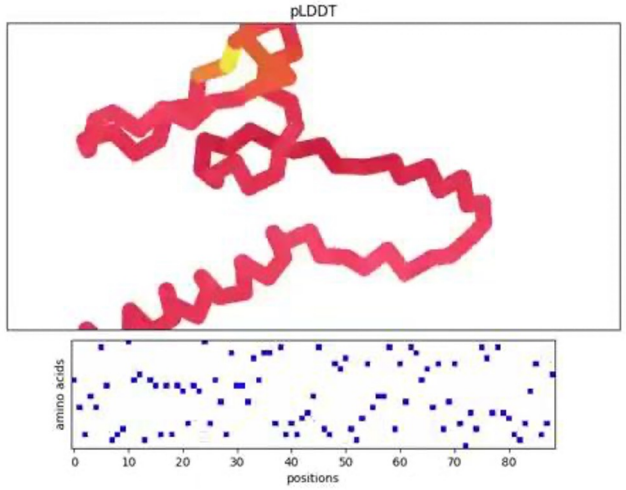
model.py

```
57    def _forward_fn(batch):
58      model = modules.AlphaFold(self.config.model)
59      return model(
60          batch,
61          is_training=False,
62          compute_loss=False,
63          ensemble_representations=False)
64
```

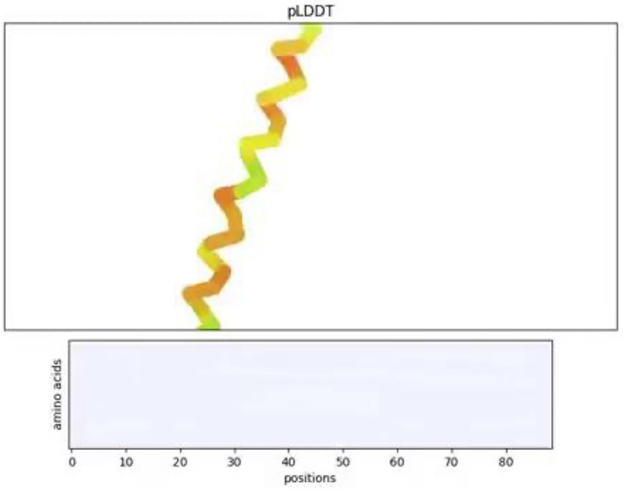
```python
def fn(seq, inputs):

  # update sequence
  inputs["msa_feat"] = jnp.zeros_like(inputs["msa_feat"]).at[...,0:20].set(seq).at[...,25:45].set(seq)
  inputs["target_feat"] = jnp.zeros_like(inputs["target_feat"]).at[...,1:21].set(seq)

  # get outputs
  key = jax.random.PRNGKey(0)
  outputs = model_runner.apply(model_params, key, inputs)

  # def random loss fn
  ca_coords = outputs["structure_module"]["final_atom_positions"][:,1,:]
  loss = jnp.square(ca_coords).sum()
  return loss

grad_fn = jax.jit(jax.grad(fn))

# initialize sequence
seq = jnp.zeros((50,20))
# get gradient
grad = grad_fn(seq, processed_feature_dict)
```

1TEN - Inputs

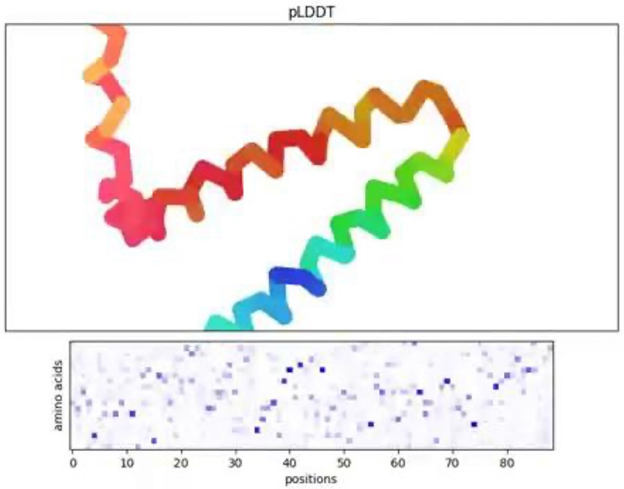
stop_gradient(argmax - softmax(logits)) + softmax(logits)



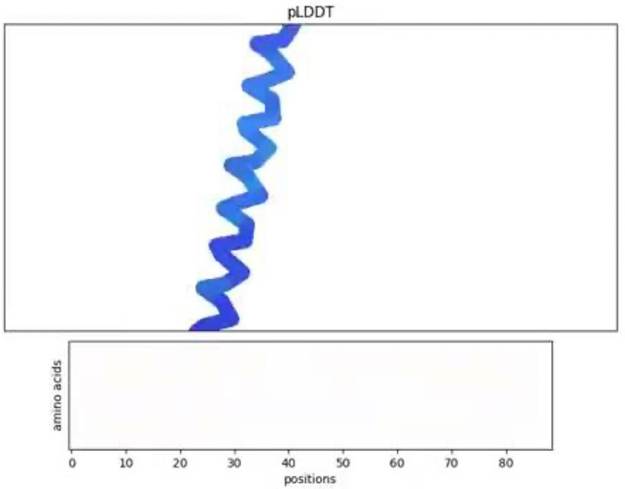
softmax(logits)

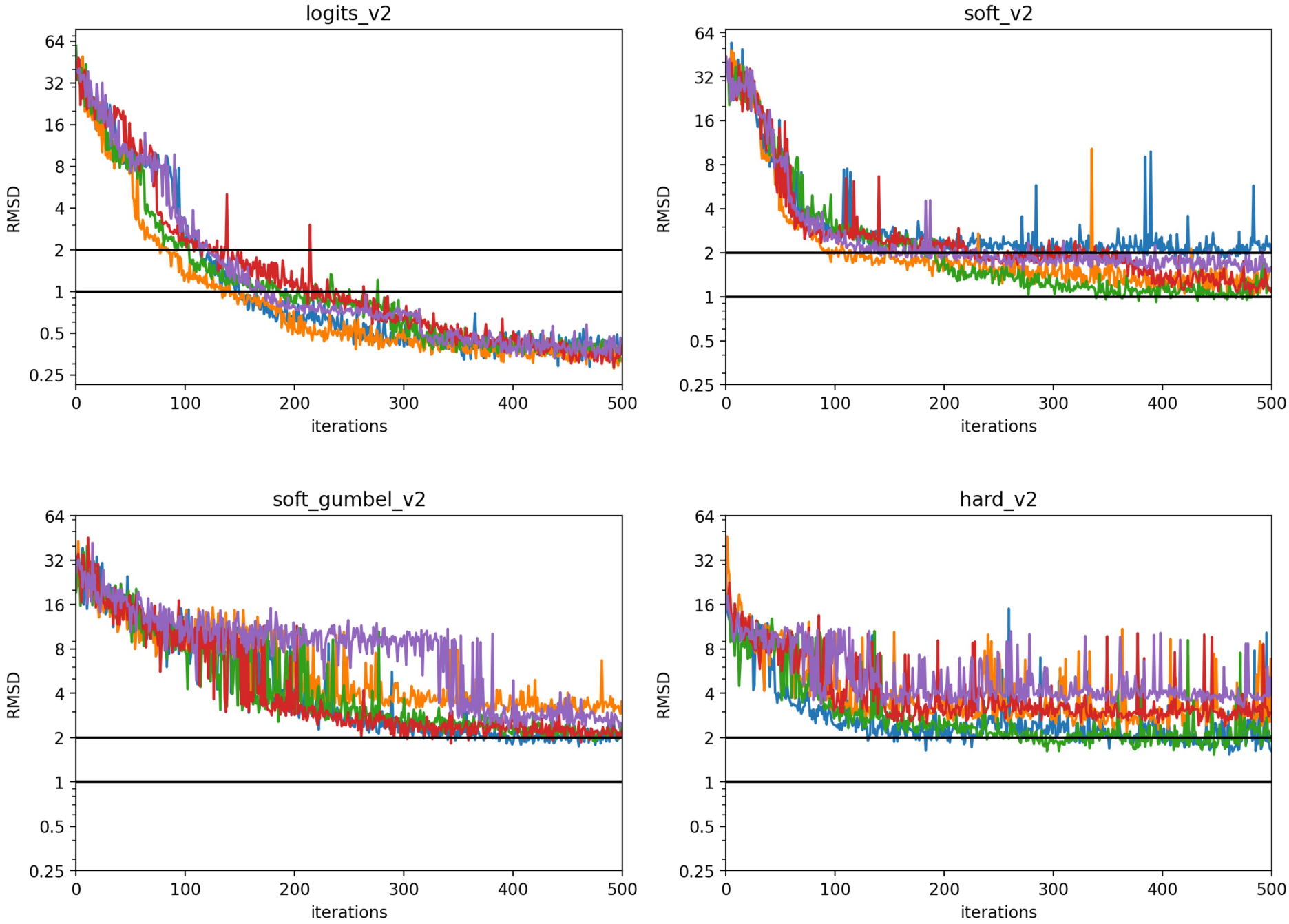


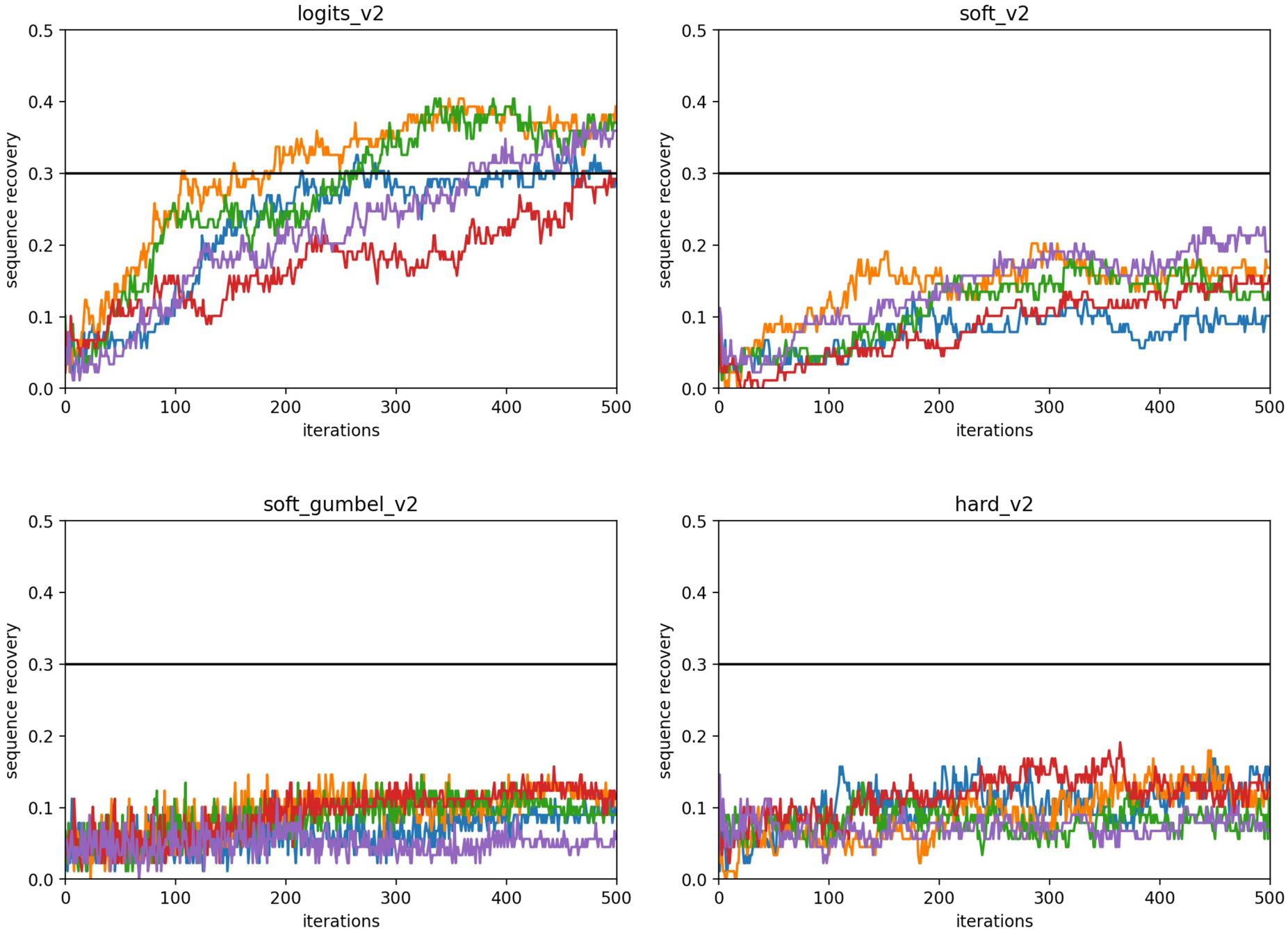
softmax(logits + gumbel_noise)



logits




logits_v2
soft_v2
soft_gumbel_v2
hard_v2
RMSD
iterations
64
32
16
8
4
2
1
0.5
0.25
0
100
200
300
400
500


logits_v2
soft_v2
soft_gumbel_v2
hard_v2
sequence recovery
iterations
0.0
0.1
0.2
0.3
0.4
0.5
0
100
200
300
400
500