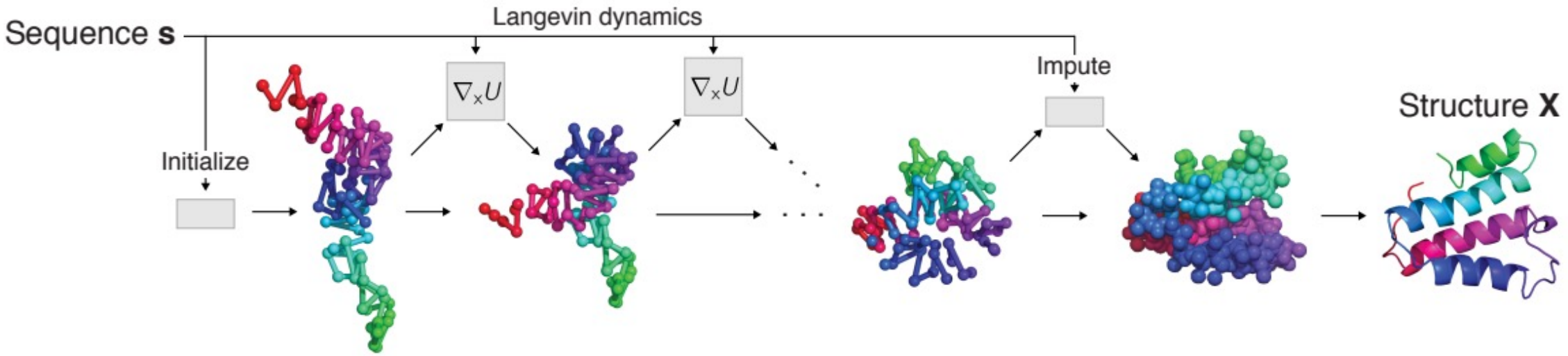


Learning Protein Structure with a Differentiable Simulator

John Ingraham, Adam Riesselman, Chris Sander, Debora Marks

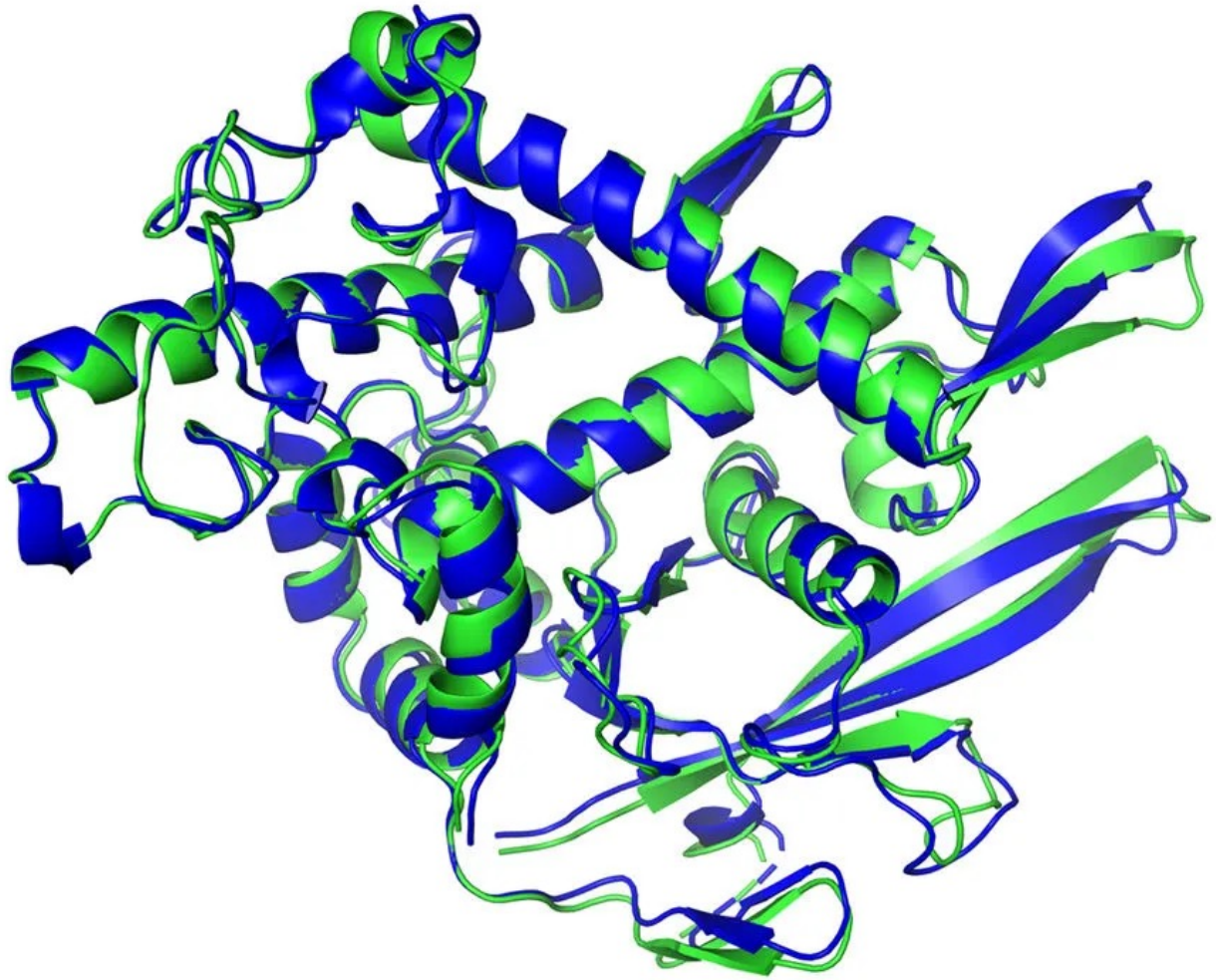
ICLR 2019

Protein Folding

Grand challenge in structural biology

Useful for understanding and interfacing with proteins

Famously “solved” by AlphaFold2!



https://www.science.org/content/article/game-has-changed-ai-triumphs-solving-protein-structures

Outline

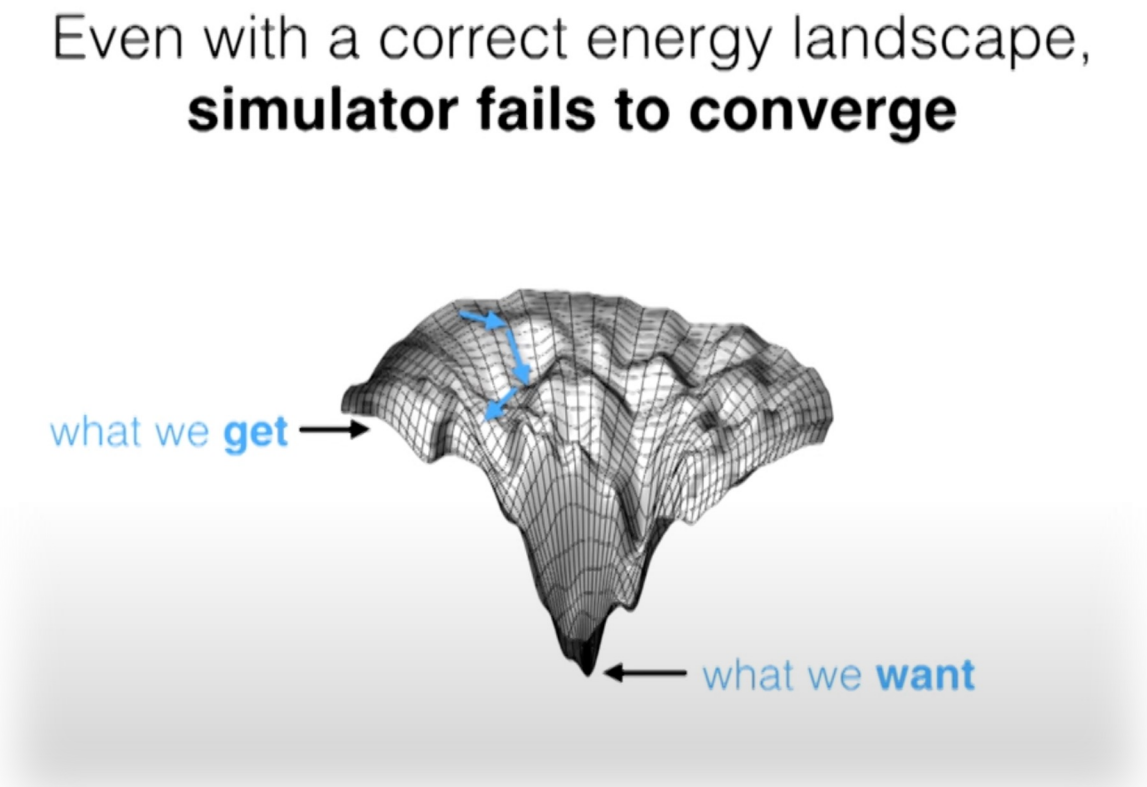
1. NEMO – Why we're excited
2. Background
3. Model Design
4. Engineering
5. Data & Hyperparameters
6. Results

How does a protein fold?

A good heuristic (from experimental research):

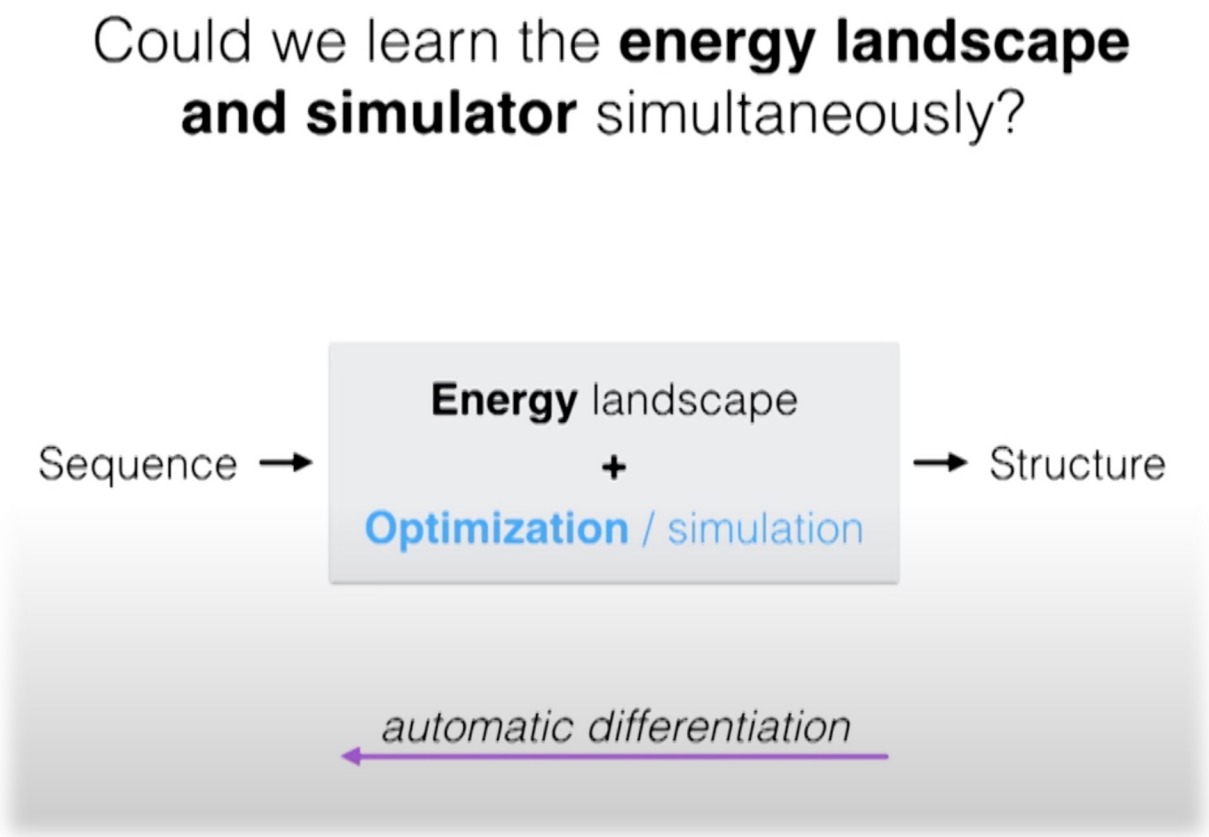
the **energy landscape model** assumes that proteins naturally achieve folds that minimize free energy.

Just fold the protein!

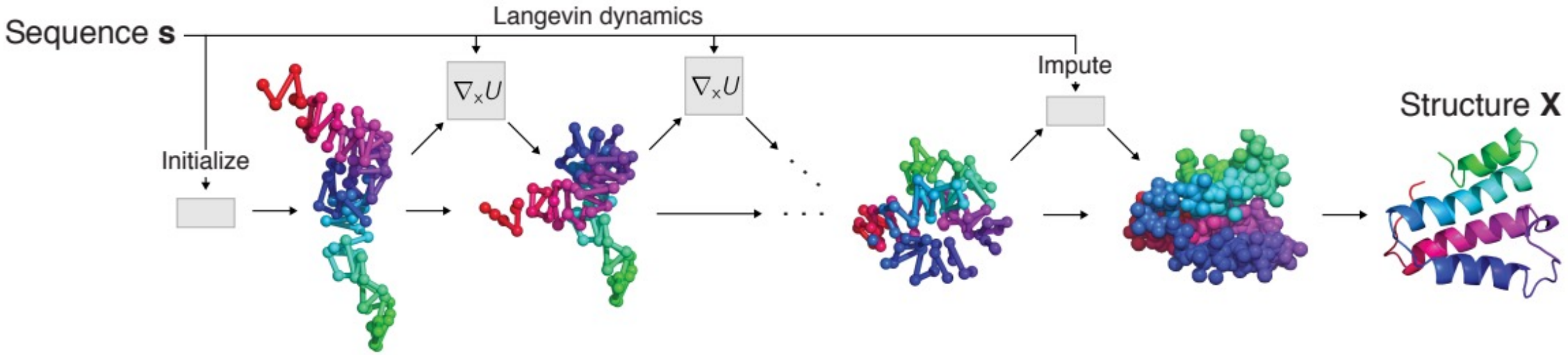


https://www.youtube.com/watch?v=R20_s8XPw8U

Just learn to simulate the universe!



https://www.youtube.com/watch?v=R20_s8XPw8U



NEMO

An end-to-end differentiable protein folding simulator.

Related Work

- Learning distributions and sampling (SPEN, diffusion)
- Protein modeling (like AlphaFold)
- Differentiable computing (neural implicit representations, etc.)
- RNNs and exploding gradients, RL and discount factor

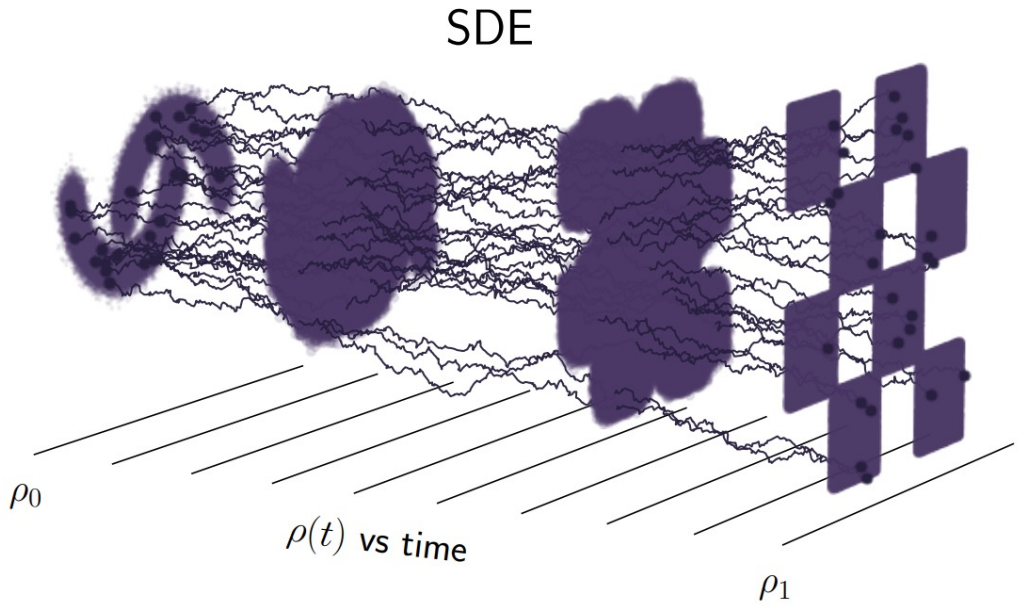


Image courtesy of Albergo et al. (2023)

Background

Boltzmann Distribution & Energy Landscapes

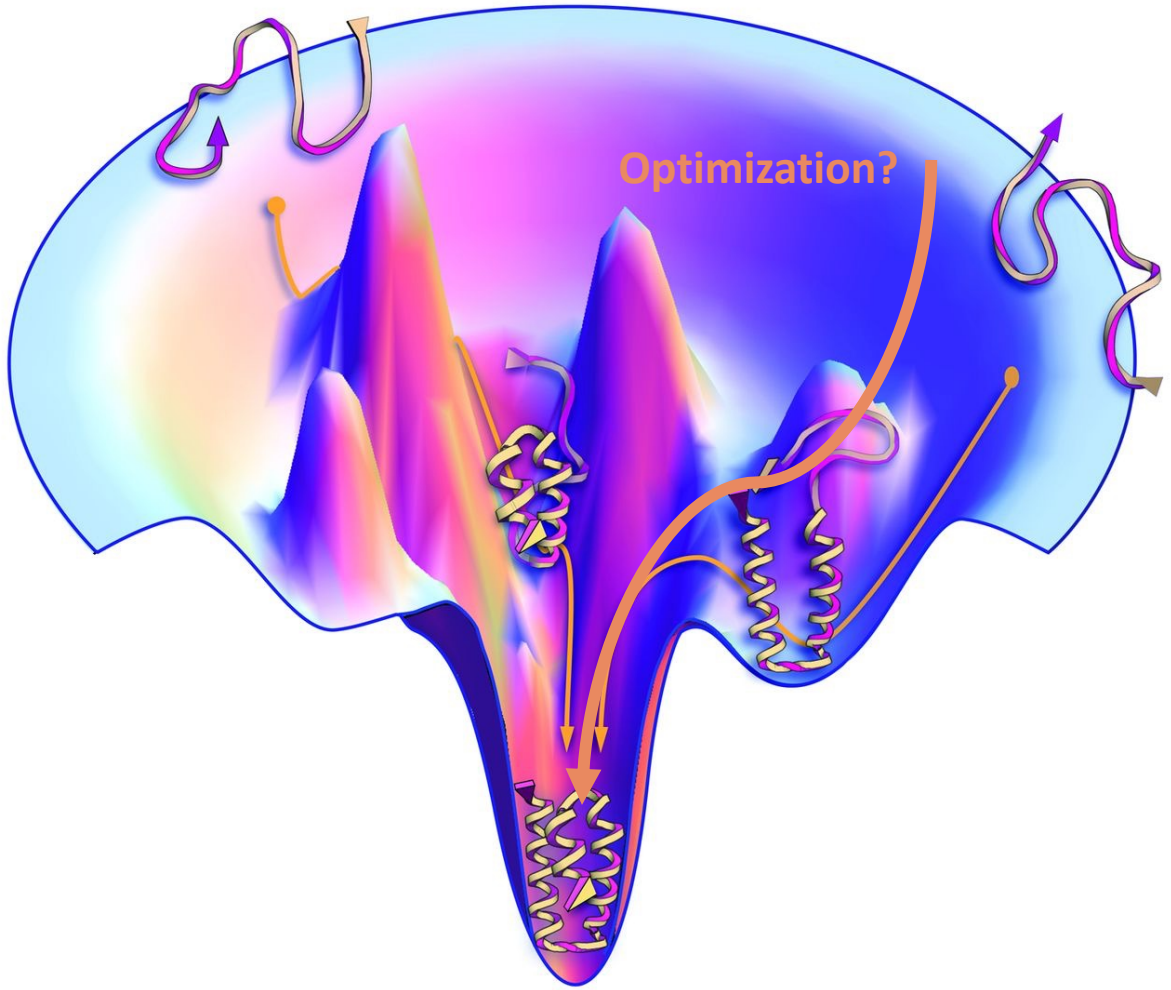


Image courtesy of: Dill & MacCallum (2012)

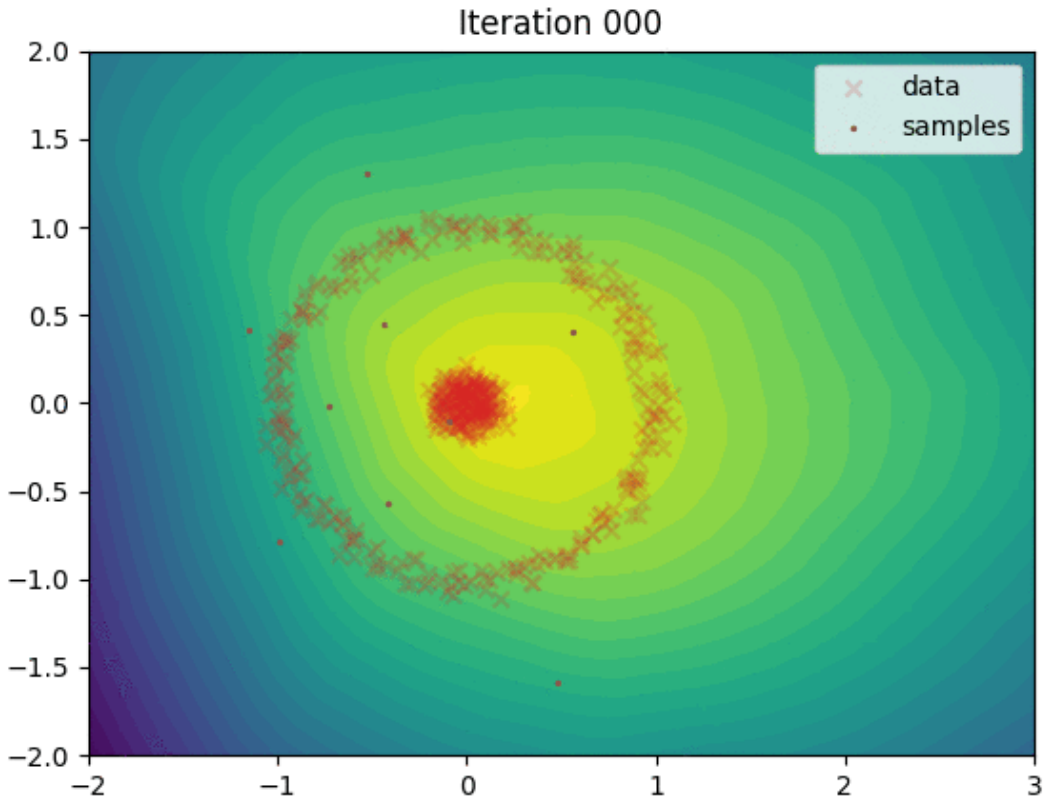
Sampling low energy states

$$x^{(t+\varepsilon)} \leftarrow x^{(t)} - \frac{\varepsilon}{2}\nabla_x U^{(t)} + \sqrt{\varepsilon}p$$

(Approximate) Langevin Dynamics

We update each atom's position, by stepping in the direction of the force, and perturbing with Gaussian noise.

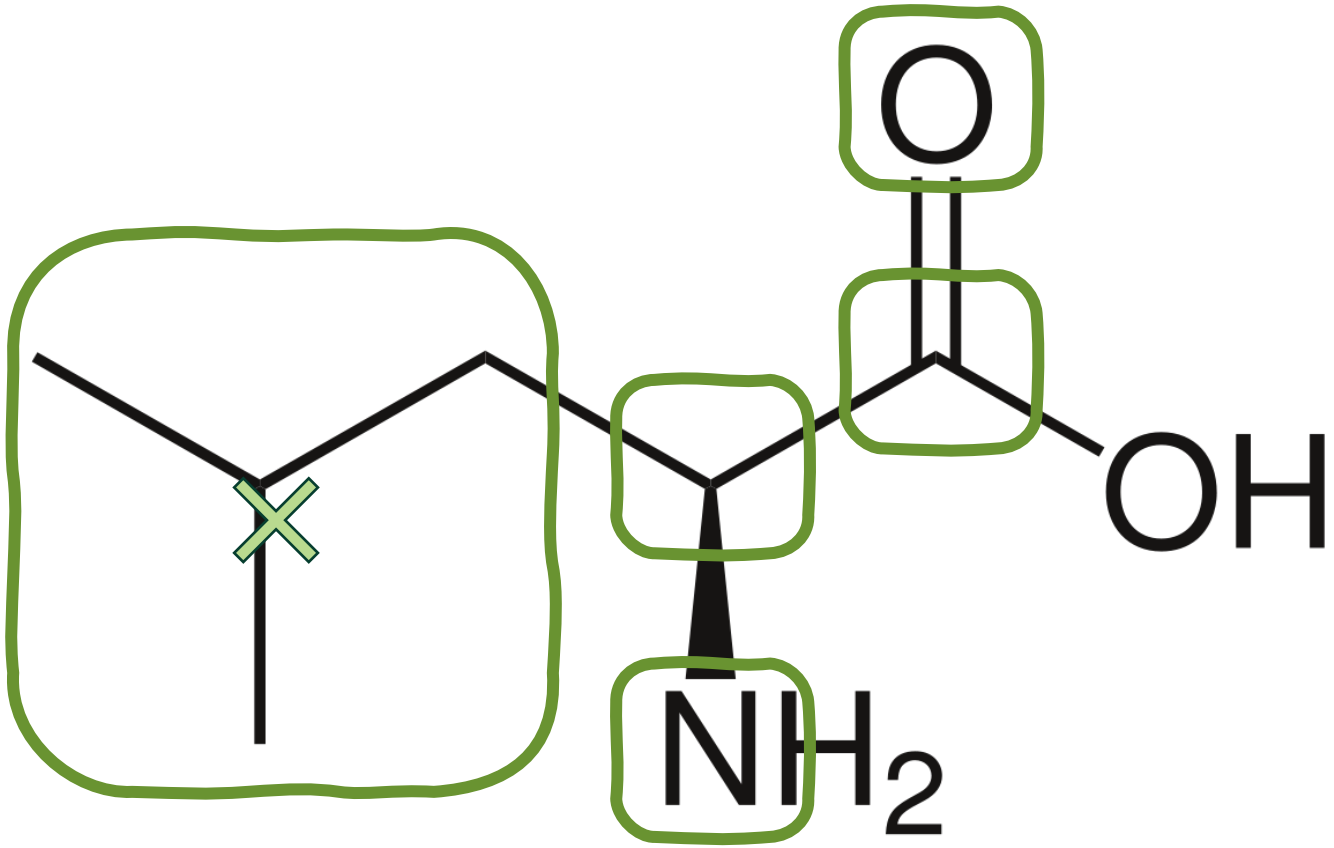
Langevin Sampling in Action



https://gist.github.com/vene/5fb0b44166001c8e8a8cc575a8949754

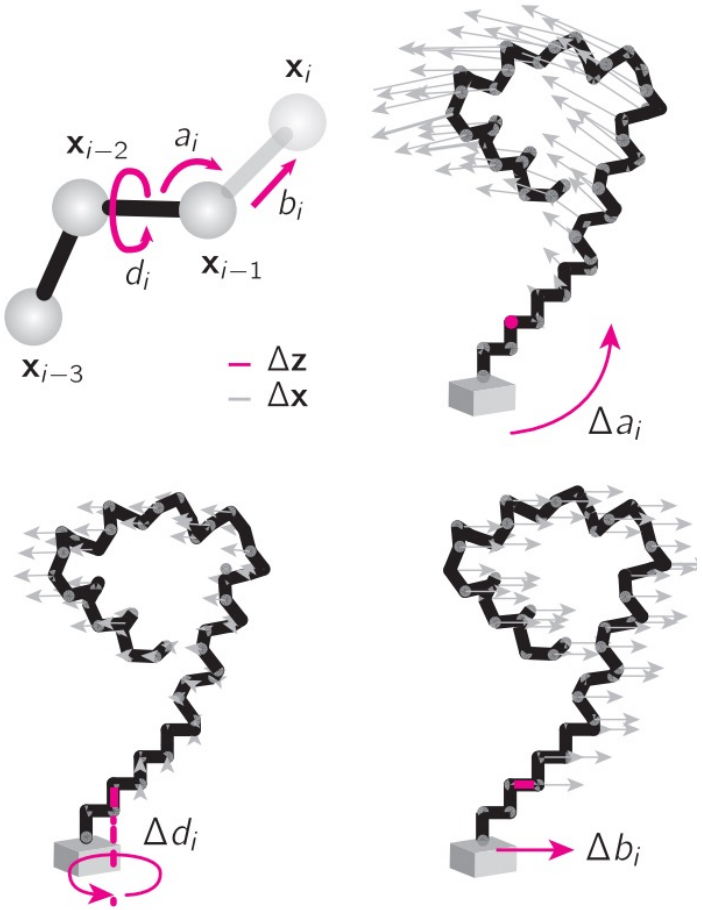
A Differentiable Simulator

Building Blocks



https://www.creative-biostructure.com/amino-acid-structure-chart.htm

Coordinate Choices



Internal coordinates favor coherent movement into new conformations.

Cartesian coordinates favor local structure rearrangements.

NEMO uses **both** coordinates, alternating between time steps.

Sampling Atomic Position

Algorithm 1: Direct integrator

Input : State $\boldsymbol{z}^{(0)}$, energy $U(\boldsymbol{x})$, step ϵ, time T, scale $\mathbf{C}$

Output : Trajectory $\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(T)}$

Initialize $\boldsymbol{x}^{(0)} \leftarrow \mathcal{F}(\boldsymbol{z}^{(0)})$;

while $t < T$ **do**

  Compute forces $\boldsymbol{f}_{\boldsymbol{z}} = -\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}^{T} \nabla_{\boldsymbol{x}} U$;

  Sample $\Delta \boldsymbol{z} \sim \mathcal{N}\left(\frac{1}{2}\epsilon \mathbf{C} \boldsymbol{f}_{\boldsymbol{z}}, \epsilon \mathbf{C}\right)$;

  $\boldsymbol{z}^{(t+\epsilon)} \leftarrow \boldsymbol{z}^{(t)} + \Delta \boldsymbol{z}$;

  $\boldsymbol{x}^{(t+\epsilon)} \leftarrow \mathcal{F}(\boldsymbol{z}^{(t+\epsilon)})$;

  $t \leftarrow t + \epsilon$;

end

Algorithm 2: Transform integrator

Input : State $\boldsymbol{z}^{(0)}$, energy $U(\boldsymbol{x})$, step ϵ, time T, scale $\mathbf{C}$

Output : Trajectory $\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(T)}$

Initialize $\boldsymbol{x}^{(0)} \leftarrow \mathcal{F}(\boldsymbol{z}^{(0)})$;

while $t < T$ **do**

  Compute forces $\boldsymbol{f}_{\boldsymbol{z}} = -\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}^{T} \nabla_{\boldsymbol{x}} U$;

  Sample $\Delta \boldsymbol{z} \sim \mathcal{N}\left(\frac{1}{2}\epsilon \mathbf{C} \boldsymbol{f}_{\boldsymbol{z}}, \epsilon \mathbf{C}\right)$;

  $\tilde{\mathbf{x}} \leftarrow \boldsymbol{x}^{(t)} + \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}^{(t)} \Delta \boldsymbol{z}^{(t)}$;

  $\boldsymbol{x}^{(t+\epsilon)} \leftarrow \boldsymbol{x}^{(t)} + \frac{1}{2}\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{z}}^{(t)} + \frac{\partial \tilde{\mathbf{x}}}{\partial \boldsymbol{z}}\right) \Delta \boldsymbol{z}^{(t)}$;

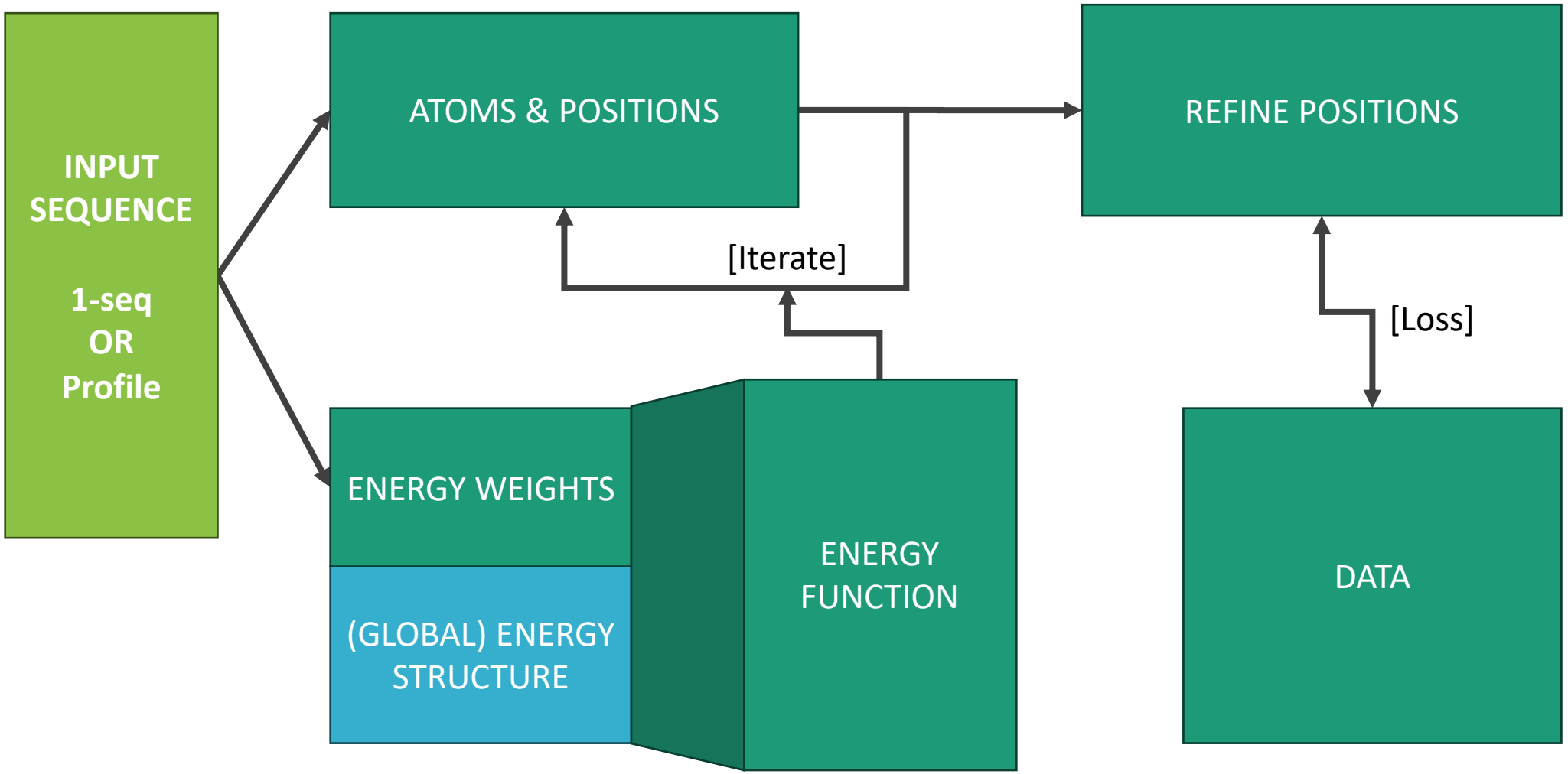
  $t \leftarrow t + \epsilon$;

end

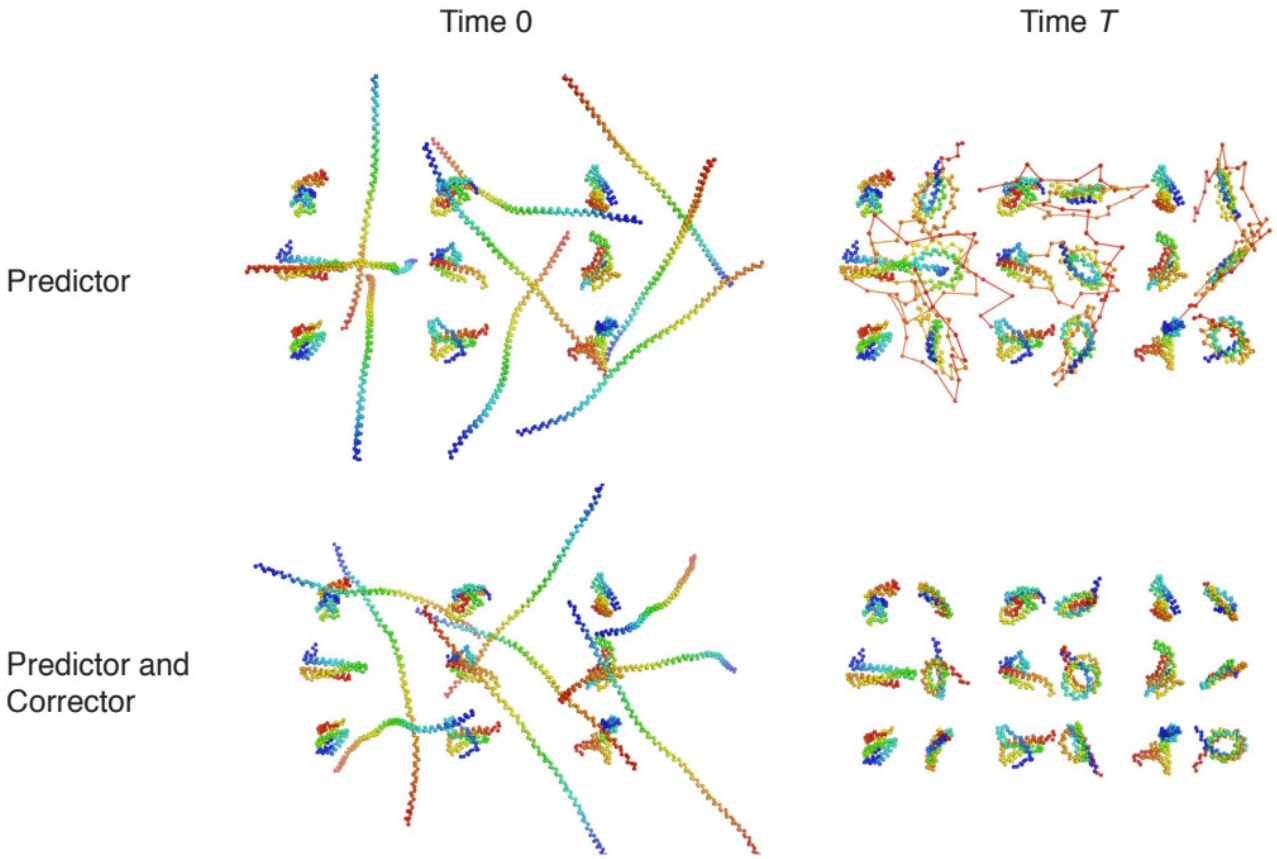
Parallelizable!

Heun's Method

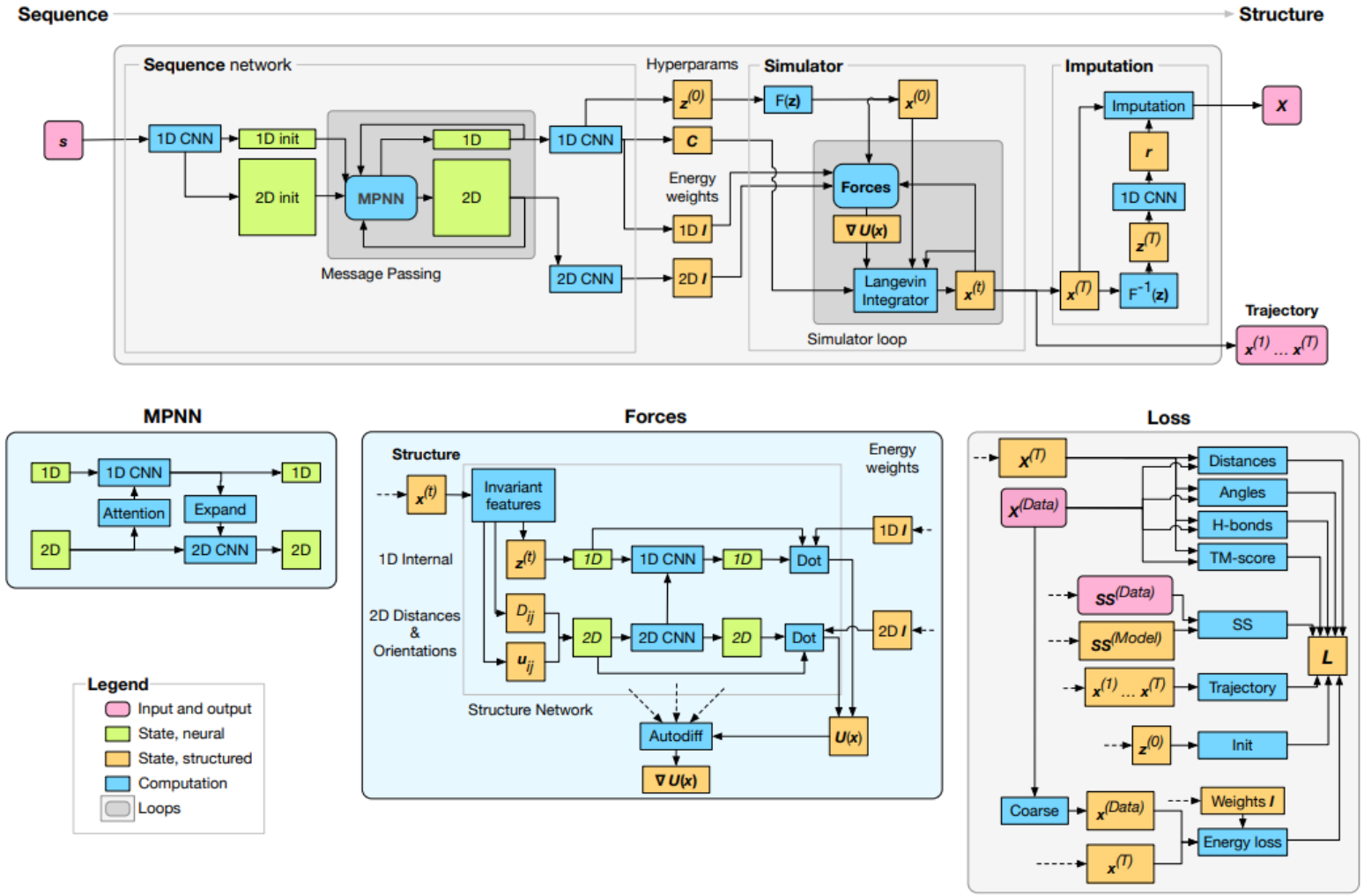
Learning to simulate



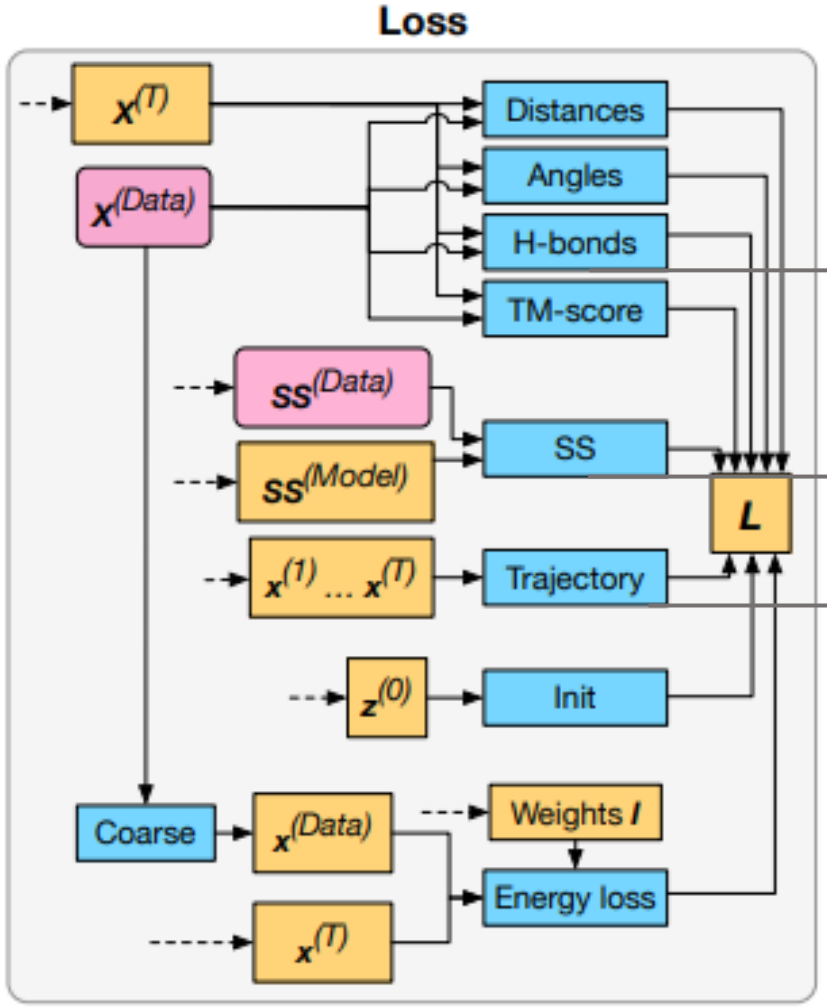
Corrector & Imputation



Full model architecture



Focusing on Loss



$$\mathcal{L}_{\mathrm{ML}} = U_\theta(\perp(\boldsymbol{x}^{(\mathrm{D})}); \boldsymbol{s}) - U_\theta(\perp(\boldsymbol{x}^{(\mathrm{M})}); \boldsymbol{s})$$

Probabilistic model of H-bonds + cross-entropy loss with data

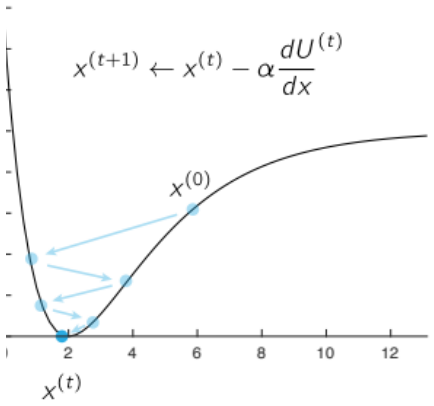
Predicting secondary structure (8 classes)

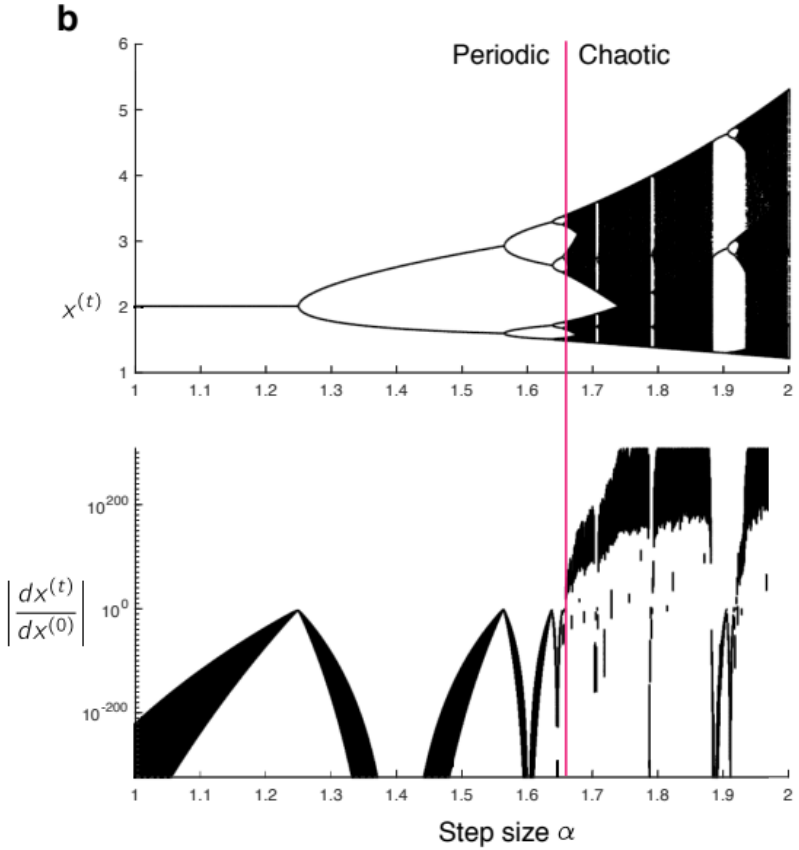
Required because of discount factor

Training is *very* unstable!

Some problems:

- Gradients accumulate **exponentially** over time
- High forces explode simulations!
- Sensitivity to initial conditions → instability through time steps





Damped Gradients

Gradient clipping? Doesn't work...

...so multiply each backwards iteration by γ,
like in reinforcement learning!

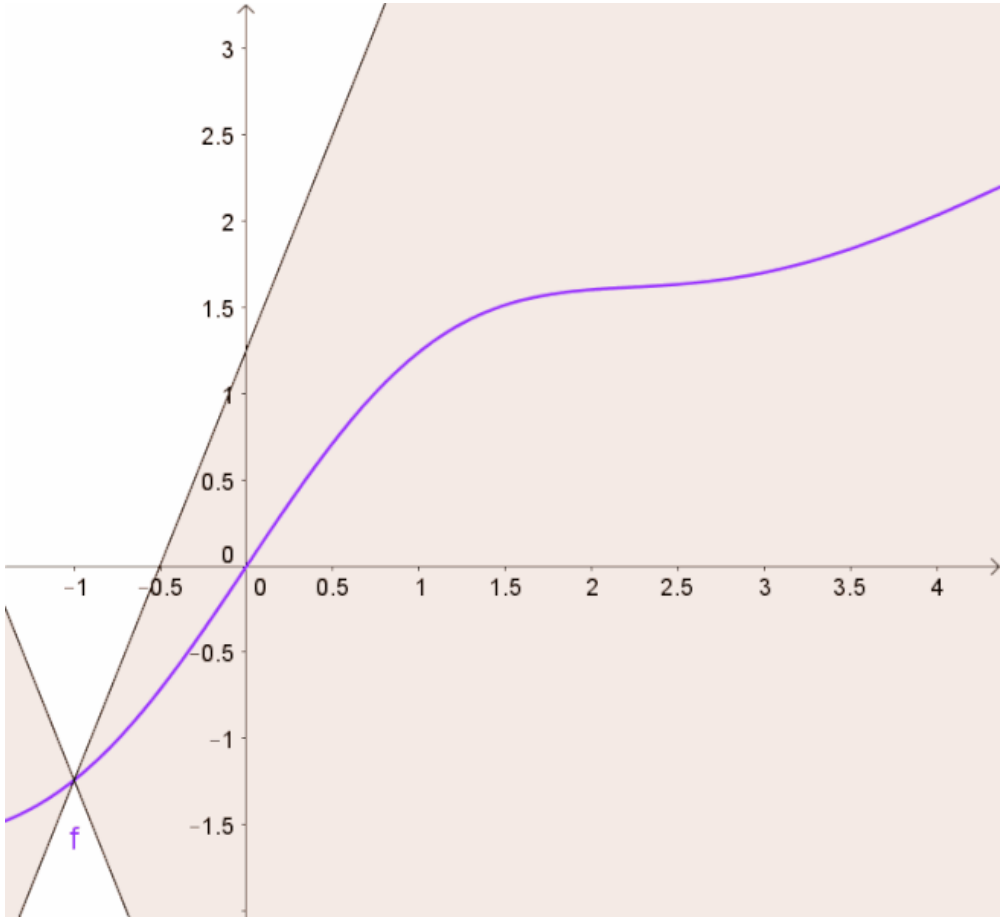
$$\partial \frac{\hat{\boldsymbol{x}}^{(t)}}{\partial \boldsymbol{x}^{(t-k)}} = \left(\left(\left(\gamma \frac{\partial \boldsymbol{x}^{(t)}}{\partial \boldsymbol{x}^{(t-1)}} \right) \gamma \frac{\partial \boldsymbol{x}^{(t-1)}}{\partial \boldsymbol{x}^{(t-2)}} \right) \cdots \gamma \frac{\partial \boldsymbol{x}^{(t-k+1)}}{\partial \boldsymbol{x}^{(t-k)}} \right) = \gamma^k \frac{\partial \boldsymbol{x}^{(t)}}{\partial \boldsymbol{x}^{(t-k)}}.$$

Speed Clipping

Algorithm 3: Mixed Integrator

Input : Initial state $\boldsymbol{z}^{(0)}$, energy $U(\boldsymbol{x})$, time steps $\epsilon_{\boldsymbol{x}}, \epsilon_{\boldsymbol{z}}$, total time T, preconditioners $\mathbf{C}_{\boldsymbol{x}}, \mathbf{C}_{\boldsymbol{z}}$,
Output : Trajectory $\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(T)}$
Initialize $\boldsymbol{x}^{(0)} \leftarrow \mathcal{F}(\boldsymbol{z}^{(0)})$;
while $t < T$ **do**
    $\boldsymbol{f}_{\boldsymbol{x}} \leftarrow \nabla_{\boldsymbol{x}} U$;
    $\Delta \boldsymbol{x}^{(Cart)} \leftarrow \texttt{CartesianStep}(\boldsymbol{x}^{(t)}, \boldsymbol{f}_{\boldsymbol{x}}, \epsilon_{\boldsymbol{x}}, \mathbf{C}_{\boldsymbol{x}})$;
    $\Delta \boldsymbol{x}^{(Int)} \leftarrow \texttt{ClippedInternalStep}(\boldsymbol{x} + \Delta \boldsymbol{x}^{(Cart)}, \boldsymbol{f}_{\boldsymbol{x}}, \epsilon_{\boldsymbol{z}}, \mathbf{C}_{\boldsymbol{z}})$;
    $\boldsymbol{x} \leftarrow \boldsymbol{x} + \texttt{Detrend}(\Delta \boldsymbol{x}^{(Cart)} + \Delta \boldsymbol{x}^{(Int)})$;
    $t \leftarrow t + \epsilon$;
end

Limiting Sensitivity to Initial Conditions



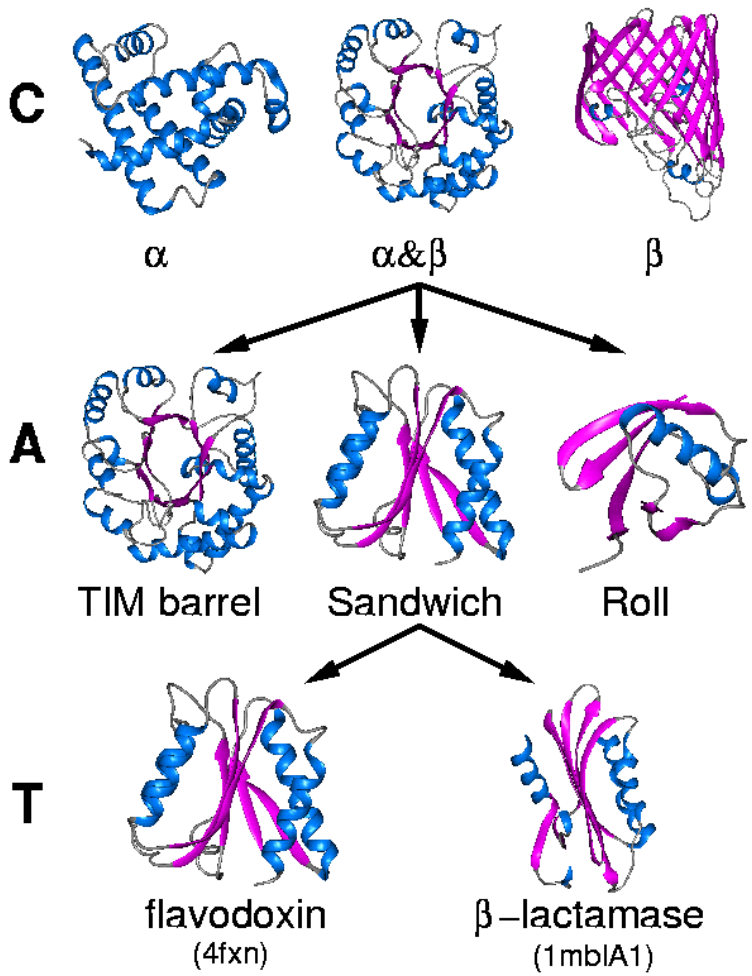
$$|F^{(t)}(\boldsymbol{x}, \theta) - F^{(t)}(\boldsymbol{x} + \Delta\boldsymbol{x}, \theta)| \leq |\Delta\boldsymbol{x}|$$

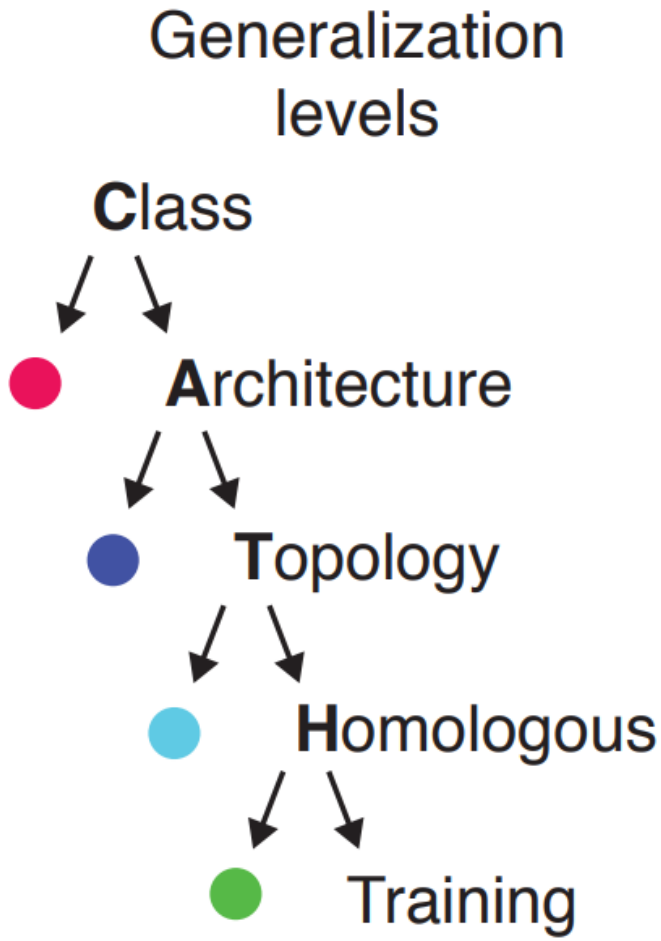
$$\mathcal{L}_{Lyapunov} = \max\left(0, \log \frac{|F(\boldsymbol{x}^{(t)}) - F(\boldsymbol{x}^{(t)} + \boldsymbol{\delta})|}{|\boldsymbol{\delta}|}\right)$$

https://en.wikipedia.org/wiki/File:Lipschitz_Visualisierung.gif

Data & Metadata

Dataset: CATH -- hierarchy of protein structure





Data Split

- Train: 35k folds
- Validation: 21k folds
- Test: 10k folds

- Testing generalization:
  - **H** validation: superfamilies excluded from train
  - **T** validation: topologies excluded from train
  - **A** validation: secondary structures excluded from train

Guesswork

figure Figure 12. We note that the validation set was not explicitly used to tune hyperparameters due to the large cost of training (2 months on 2 M40 GPUs), but we did keep track of validation statistics during training.

!!

So... how did we do?

A scary question

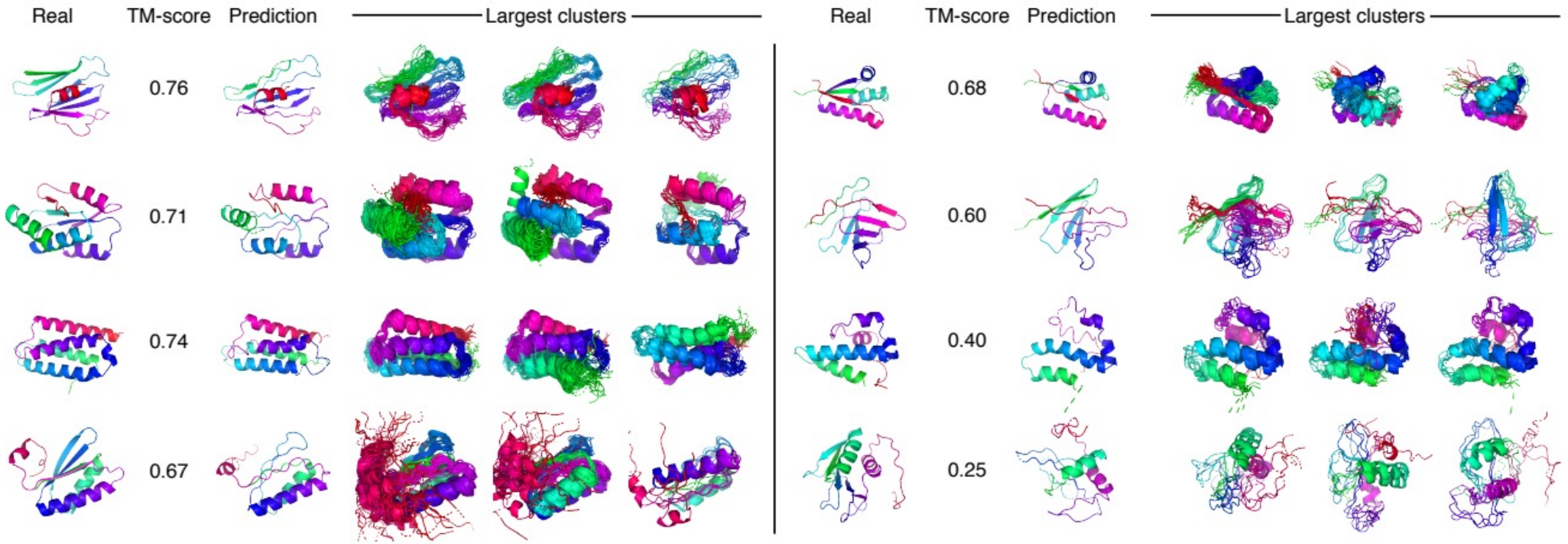
One answer

Table 1: Test set performance across different levels of generalization

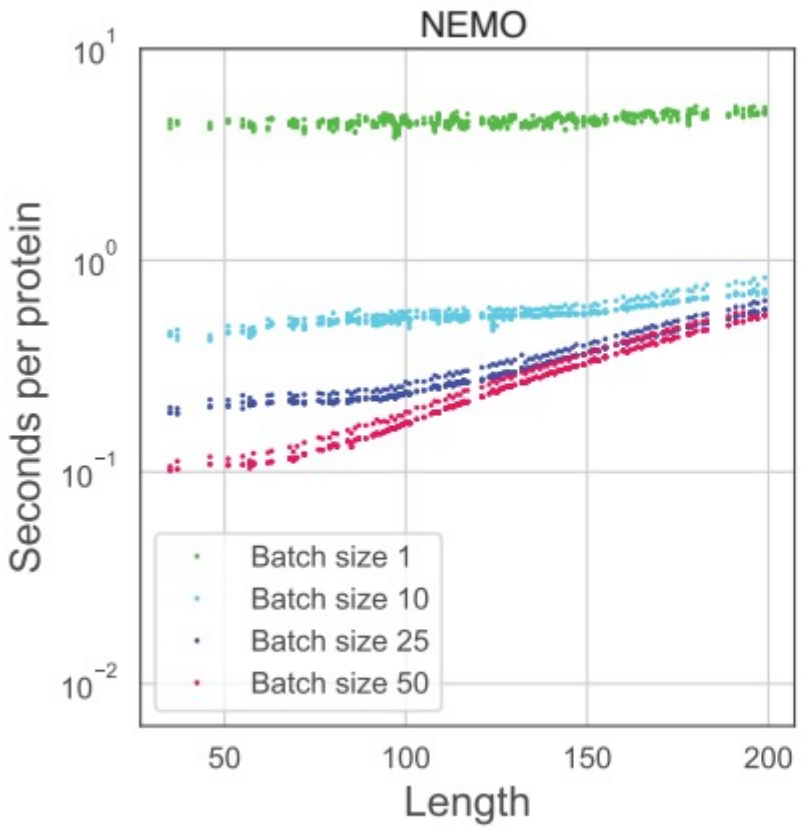
Model	# params	Total	C	A	T	H
NEMO (ours, profile)	21.3m	**0.366**	**0.274**	**0.361**	**0.331**	0.431
NEMO (ours, sequence-only)	19.1m	0.248	0.198	0.245	0.254	0.263
RNN baseline model (profile)						
2x100	5.9m	0.293	0.213	0.230	0.247	0.388
2x300 (avg. of 3)	8.8m	0.335	0.229	0.282	0.278	0.446
2x500	13.7m	0.347	0.222	0.272	0.286	**0.477**
2x700	21.4m	0.309	0.223	0.259	0.261	0.403
Number of structures		10381	1537	1705	3198	3941

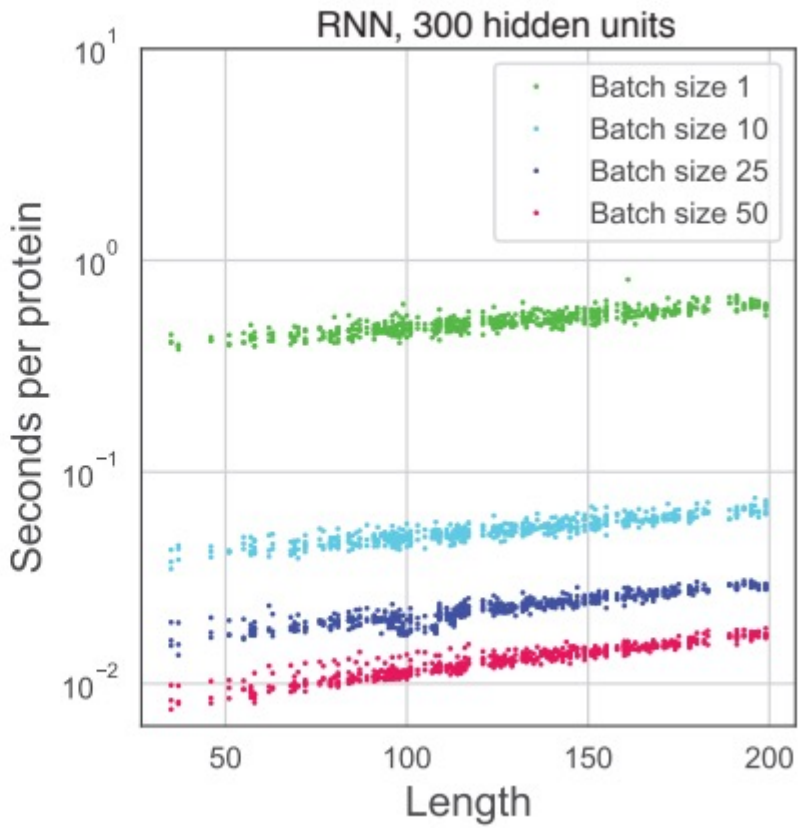
Confidence

Uncertainty measured by “clusters” – similar structures sampled in distribution



Time


NEMO
Seconds per protein
10^{1}
10^{0}
10^{-1}
10^{-2}
Batch size 1
Batch size 10
Batch size 25
Batch size 50
50
100
150
200
Length

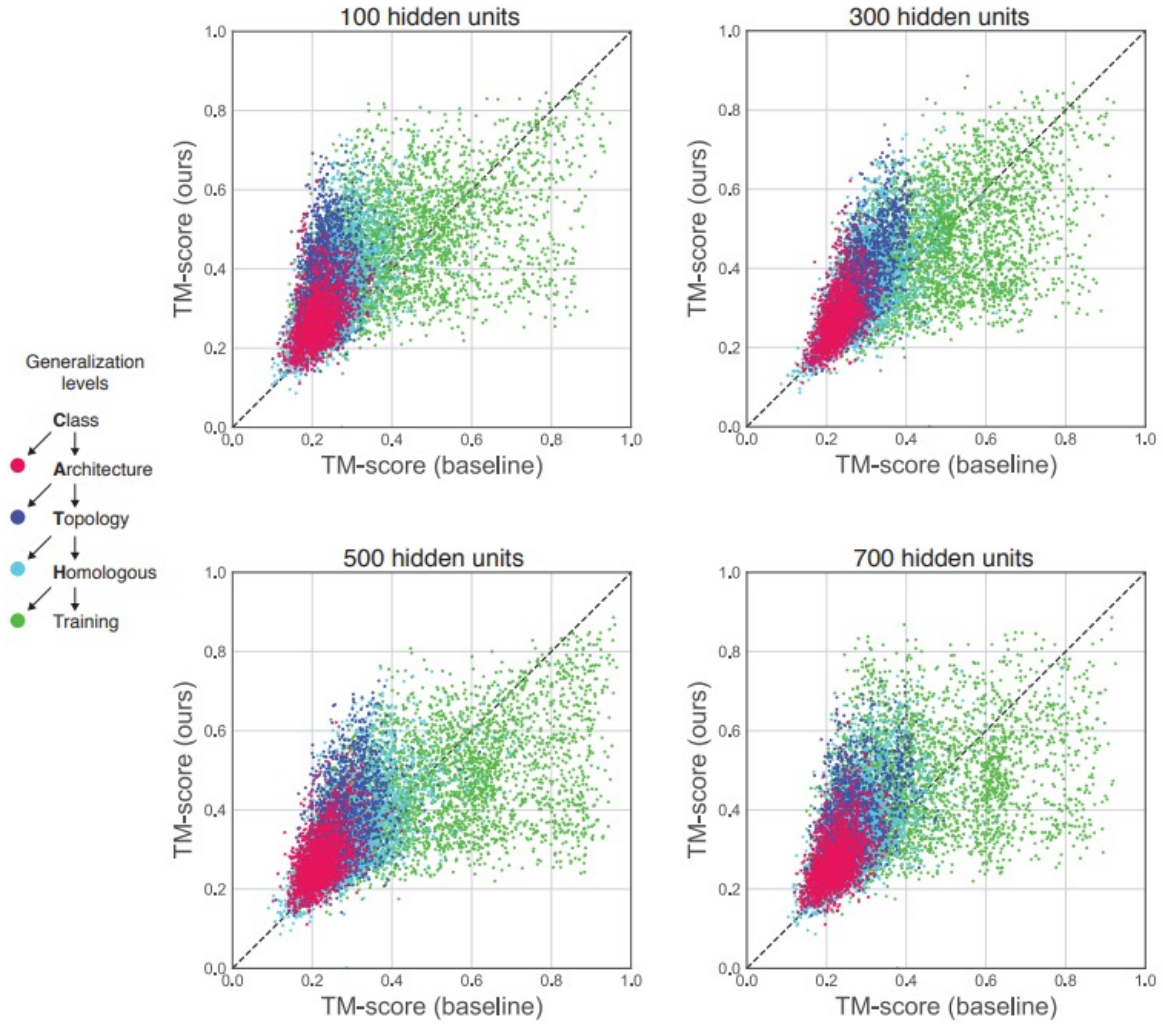

RNN, 300 hidden units
Seconds per protein
10^{1}
10^{0}
10^{-1}
10^{-2}
Batch size 1
Batch size 10
Batch size 25
Batch size 50
50
100
150
200
Length

One measurement metric

- TM-Score
  - Scored from 0 to 1
  - >0.5 is considered “good”
  - Best possible score across all possible positions:
    - Found by optimizing (with autograd throughout)

$$\sum_i \frac{1}{1+\left(\frac{D_i}{D_0}\right)^2},$$

Baselines


Generalization levels
Class
Architecture
Topology
Homologous
Training
100 hidden units
300 hidden units
500 hidden units
700 hidden units
TM-score (ours)
TM-score (baseline)
0.0
0.2
0.4
0.6
0.8
1.0

loss.backward()

Any questions?