DeepFoldit
A Deep Reinforcement Neural Net Folding Proteins

Dimitra Panou[1] **& Martin Reczko**[1,2]

1. University of Athens, Department of Informatics and Telecomunications
2. Biomedical Sciences Research Center "Alexander Fleming"

reczko@fleming.gr — +30 210 9656310 (ext. 144)

ECCB 2018

Abstract

Despite considerable progress, ab initio protein structure prediction remains unsolved. A crowdsourcing approach is the online puzzle video game Foldit[1], that provided several useful results that matched or outperformed algorithmically computed solutions[2]. Using Foldit, the WeFold[3] crowd had several successful participations in the Critical Assessment of Techniques for Protein Structure Prediction. Based on the recent Foldit standalone version[4], we trained a deep reinforcement neural net called DeepFoldit to improve the score given an unfolded protein, using the Q-learning method[5] with experience replay. Initial results show that given a set of small unfolded training proteins, DeepFoldit learns action sequences that improve the score both on the training-set and on novel test proteins. Our approach combines the intuitive user interface of Foldit with the power of deep reinforcement learning.

Introduction

Protein structure prediction (PSP) is a developing and very important subfield of bioinformatics, as it targets to infer the proteins folding, which determines to a great extent its biological function. If structural homologs are not available, *ab initio* modeling suggests 3D structures from scratch. The current accuracy of *ab initio* modeling still needs improvemnet and is limited to small protein structures, as state-of-the-art *ab initio* methods have huge computational demands. With the rise of cost-effective parallel computation with graphical processing units (GPUs), deep learning methods have been used successfully in many applications. We introduce deep reinforcement learning for PSP.

Data

Our dataset consists of 40 proteins from the PDB database obtained by the X-ray Diffraction method, containing less than 100 residues and having less than 30% sequence homology. The set was split into 20 proteins for training and and 20 for testing. The structures in each set were modified as follows: For deepFoldit-Soloist, linear extended conformations were created by processing the extracting amino acid sequences with the LEaP program of Amber[6]. For deepFoldit-Evolver, the native structures were first optimized until convergence by the energy minimization function of FolditStandalone and then 100 random moves were performed from our action set creating a dataset of denatured proteins.

Materials and Methods

Using the Q learning method and the interactive interface of the Rosetta molecular modeling package provided by the standalone version of the FoldIt game (Figure 1), we trained two types of deep reinforcement neural nets (DRNN) to: a) improve the score given an unfolded protein (corresponding the to the 'Soloist' setting in FoldIt) and b) improve the score of a slightly denatured protein (the 'Evolver' setting in FoldIt).

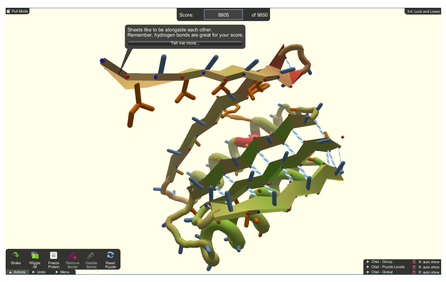

Figure 1: The user interface of the FoldIt game

log-Polar Transformation

The image of the protein has to be presented to the DRNN using an informative representation. The screen area containing the protein is resampled to 160x160 pixels. To obtain invariance with respect to rotations and scalings performed at the center of the image, a log-polar transformation is applied. Rotated or scaled images will result in translations in polar coordinates that can be invariantly detected by the convolutional net. Examples for the input images and their transformation are shown in Figure 2.

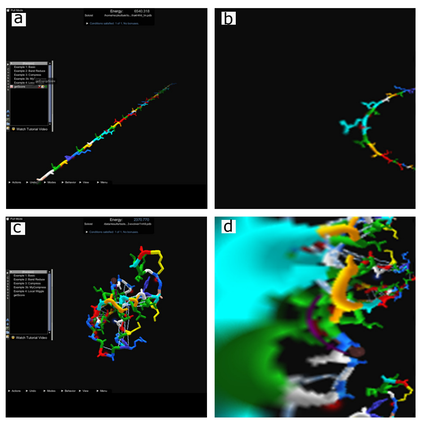

Figure 2: **a.** Linear input, **b.** log-polar transformation of a, **c.** Evolver input, **d.** log-polar transformation of c

Architecture

The input to the neural network is an 160x160x3 RGB image sampled from Foldit-standalone,followed by 3 rectifier hidden layers convolving 1 1x1 filter with stride 1 (for adaptive RGB to gray conversion), 40 20x20 filters with stride 2 and 5 40x40 filters with stride 8. The final hidden layer with of either 128 or 256 fully-connected rectifier units is connected to the output layer, representing 15 actions, shown in Figure 3. In some training runs, the adaptive RGB to grayscale layer marked with (*) in Figure 3, was omitted to assess the relevance of the color-coding of the amino acids.

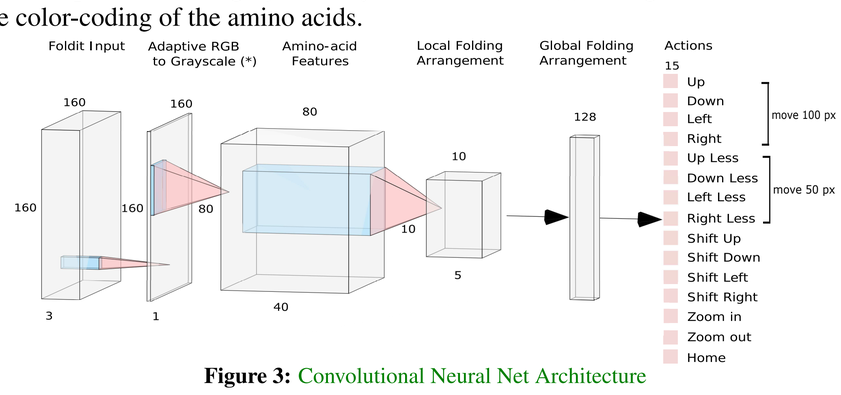

Figure 3: Convolutional Neural Net Architecture

Q-learning

The Q-function represents the maximum discounted future reward when an action a is performed in state s. Q(s, a) gives an estimation of the quality of action a in state s. In deep Q-learning this function is approximated by a deep neural net.

Q-learning: Learn DNN Q

Require:
Initialize replay memories D, D_2 to capacity N
Initialize DNN Q with random weights θ
for episode = 1, M **do**
  Initialize arbitrary first sequence of frames for initial state
  for t = 1, T **do**
    With probability ϵ select a random action a_t, otherwise select $a_t = max_a Q(s_t, a)$
    Execute action a_t and observe reward r_t and state s_{t+1}
    if $r_t <> 0$ **then**
      Store state transition $(s_t, a_t, r_t, s_{t+1}, term)$ in D_2
    else
      Store state transition $(s_t, a_t, r_t, s_{t+1}, term)$ in D
    Sample random mini batch $(s_i, a_i, r_i, s_{i+1}, term)$ from D and D_2
    Set $y_i = Q(s_i)$
    if $term = true$ **then**
      Set $y_{i,a_i} = r_i$
    else
      Set $y_{i,a_i} = r_i + \gamma max_a Q(s_{i+1}, a)$
    Train Q on (s_i, y_i)

All rewards are $r_t = tanh(0.1 \cdot \Delta S)$, where ΔS is the FoldIt score difference before and after an action. The usual Q-learning algorithm with experience replay uses replay memory that contains all state-action pairs, irrespective of the reward received. As rewards (or penalties) occur only very sparsely, we split the replay memory into two queues that contain the more informative states with non-zero reward (queue D_2) and the non-reward states (queue D). Training is accelerated by randomly selecting the mini-batch to contain equal shares of each queue. Each protein is manipulated in $T = 200$ actions.

Results

In Figure 4, the relative score improvements are shown both for deepFoldIt-Soloist on the training and test sets after 220000 training steps. The mean score improvement is 5.8% on the training set and 4.5% on the test set, indicating that the trained action sequences can generalize to novel proteins with unrelated sequences.

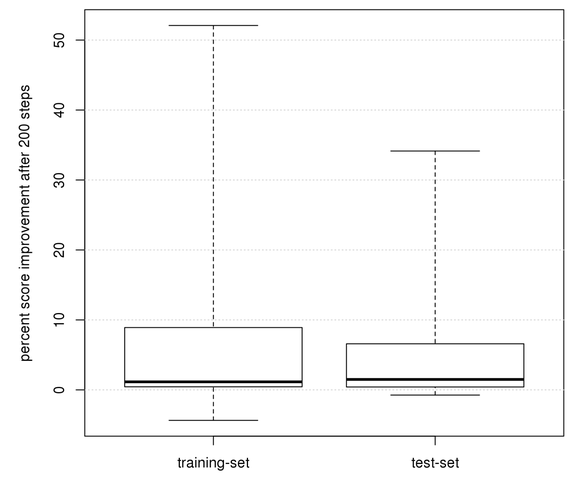

Figure 4: Relative score improvements

In Figure 5 and 7, the trained weights of the amino-acid feature layer are visualized for deepFoldIt-Evolver and Soloist, respectively. As the features of the Evolver net are directly connected to the RGB image, is is evident that the net has developed feature detectors specific to groups of amino acids that exploit the amino acid color code used by Foldit (shown in Figure 6).

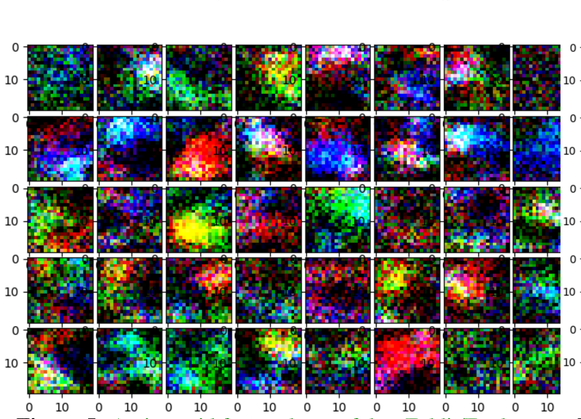

Figure 5: Amino acid feature layer of deepFoldit/Evolver

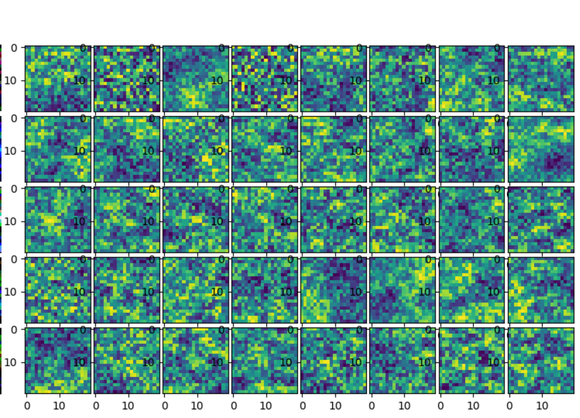

Figure 7: Amino acid feature layer of deepFoldit/Soloist

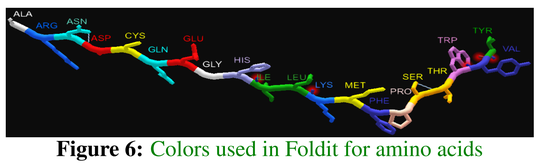

Figure 6: Colors used in Foldit for amino acids

Forthcoming Research

- Parallelize training: Using multiple virtual X framebuffers and interact with several FolditStandalone sessions to optimize multiple proteins in parallel.
- Run our implementation on the GPU nodes of the ARIS national high performance computing system.
- Final results will be compared to reference protein structures using RMS and other structural similarity metrices.
- The algorithmic complexity of protein folding can be assessed using different numbers of optimization steps for each protein: $\mathcal{O}(1)$ vs $\mathcal{O}(n)$ vs $\mathcal{O}(n \log n)$ ... , n being the length of the protein sequence.
- Provide a deepFoldit optimization module to the Foldit players community.

References

1. Cooper, S. *et al.* Predicting protein structures with a multiplayer online game. *Nature* **466,** 756–760 (Aug. 2010).
2. Horowitz, S. *et al.* Determining crystal structures through crowdsourcing and coursework. *Nat Commun* **7,** 12549 (Sept. 2016).
3. Khoury, G. A. *et al.* WeFold: a coopetition for protein structure prediction. *Proteins* **82,** 1850–1868 (Sept. 2014).
4. Kleffner, R. *et al.* Foldit Standalone: a video game-derived protein structure manipulation interface using Rosetta. *Bioinformatics* **33,** 2765–2767 (Sept. 2017).
5. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529,** 484–489 (Jan. 2016).
6. Salomon-Ferrer, R., Huang, A., Case, D. A. & Walker, R. An overview of the Amber biomolecular simulation package. *Molecular Science* **3,** 198–210 (Jan. 2013).