# Protein Folding with Deep Reinforcement Learning

**Steven Laverty**
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180
`lavers@rpi.edu`

**Mohammed Zaki, PhD**
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180
`zakim@rpi.edu`

## Abstract

We implemented a novel lattice-free reinforcement learning task called fragmented protein reconstruction which simplifies the protein structure prediction task to a discrete set of 288 actions for every 8-residue protein fragment. We trained a 12-block transformer-style neural network [40] with Deep Q Learning [23] to predict Q-values for an optimal folding policy. After 40 hours of training with a single short-chain protein PDB ID 2IGD chain A [41, 4] on a single RPI AiMOS [29] compute node, the deterministic policy was able to near-optimally predict the protein's structure with average FAPE loss [11] 3.25 angstrom. After 70 hours of training with all available proteins from CATH [32] superfamily 3.10.20.10 on a single RPI AiMOS compute node, the deterministic policy achieved an average FAPE loss of 7.73 angstrom.

## 1 Background

Proteins are vital molecular components of all living organisms. All proteins are comprised of the same set of 20–22 amino acids [13], yet individual proteins are incredibly diverse in both structure and function [19]. As a whole, proteins are capable of performing a vast array of specialized biological tasks. In general, a protein's unique structure determines its functionality [21, 19].

Every protein is comprised of at least one linked chain of amino acids called a polypeptide [24]. The type and ordering of these amino acids are determined by genetic instructions encoded into strands of RNA. Proteins are created during the biological transcription process, whereby a single strand of RNA dictates how a protein chain should be assembled [21]. Specifically, every group of 3 RNA bases (a.k.a. codons) indicate which type of amino acid should be appended to the growing protein chain [21]. As the protein chain grows, neighboring atoms will begin to interact, forming attractive hydrogen bonds [30]. Subsequently, the protein chain begins to twist and bend under the influence of these atomic interactions. The shape of the protein evolves rapidly during and immediately following transcription as it transitions through many intermediate metastable structures [24]. However, under proper conditions, all transcribed copies of a particular protein will invariably fold into the same final structure [24, 2]. The full protein structure prediction task is to determine the final folded structure of a protein using only its raw genetic instructions [5].

Machine learning techniques have been utilized extensively for protein structure prediction [26]. Trainable models are able to functionally map a raw sequence of amino acids (often formatted as a string with a 22-character alphabet) to a set of 3-dimensional predicted atomic coordinates representing the final folded structure of the protein [11]. In supervised learning approaches, predicted structures are compared against empirically-derived measured structures [11]. Gradually, the parameters of the model are adapted so as to minimize the difference between the predicted protein structures and their measured counterparts. In recent years, supervised machine learning approaches such as AlphaFold 2 have achieved impressive levels of accuracy for full-scale protein structure prediction by leveraging deep transformer-style artificial neural networks [40, 11].

Preprint. Under review.

Reinforcement learning approaches have also been applied in protein structure prediction–though typically in a highly simplified context [27, 14, 42]. Reinforcement learning involves an *agent* and an *environment*. The agent continually interacts with its environment in discrete action steps, receiving a single environmental reward after every step [12]. Reinforcement learning algorithms aim to train an agent to maximize the expected reward it will receive from its environment by following a particular set of rules, called a policy [12]. In the context of protein structure prediction, the environment and reward are typically based a simplified protein model. For example, the hydrophobic-polar protein model simplifies the protein folding task by reassigning each amino acid as either polar or hydrophobic [42]. The RL agent interacts with its environment by placing one amino acid at a time onto a simple cubic lattice, with the restriction that connected amino acids must be adjacent in the lattice. Rewards are provided for every adjacent pair of hydrophobic amino acids in the lattice, based on the biological intuition that proteins tend to form densely-packed hydrophobic cores [42, 38].

Multiple other lattice-based models reinforcement learning models exist for protein structure prediction in both 2 and 3 dimensions [27]. Our objective is to apply reinforcement learning techniques to the protein structure prediction with a 3-dimensional lattice-free model.

## 2 The protein folding task

Our reinforcement learning model is based on fragmented protein reconstruction. In this model, a protein is represented as a sequence of small fragments. Each fragment consists of 8 consecutive amino acids. The agent is tasked with assembling the fragments, one at a time, into a full predicted structure.

The environment is first initialized with empirically measured structural information for a specified protein. The environment then breaks the protein apart into fragments with a randomized initial offset. The first and last fragments will contain between 1 and 8 amino acids, depending on this initial offset. Every fragment is converted to a vectorized format based on the displacement vector from the first backbone [24] atom in the fragment to the final backbone atom in the fragment. The environment's initial state contains only the first vectorized fragment, placed arbitrarily. The agent's first task is to choose a relative angle between the first and second vectorized fragments. The environment will then place the second vectorized fragment in accordance with this angle. The agent must then choose a relative angle between the second and third fragments, and then the third and fourth fragments, and so on (see figure 1).

After the final relative angle is chosen, the environment will produce the complete predicted structure. At this point, the agent is given a terminal reward based on the accuracy of the predicted structure as compared with the provided empirical structure. The specific metric used for comparison is negative frame-aligned point error (FAPE) loss [11]. To calculate FAPE loss, the relative displacements between each pair of fragments are compared with the fragmented empirical structure. A minimum FAPE loss of zero indicates perfect correspondence between predicted structure and empirical structure. Therefore, the maximum possible terminal reward is zero and the minimum reward is unbounded. For all earlier time steps, the agent receives a reward of zero.

### 2.1 Motivation

This model is biologically informed by secondary protein structure. Proteins fold in a hierarchical fashion, where secondary structures (i.e. local structures) are the result of interactions between near-neighboring amino acids in the polypeptide chain [30]. The full tertiary structure (i.e. global structure) of a protein is the result of interactions between amino acids in different secondary structures [24]. The most common secondary structures are alpha helices and beta sheets. In alpha helices, a complete cycle occurs every 3.6 amino acids [36]. In beta sheets, linear strands of 3 to 10 amino acids are aligned in parallel to form a 2-dimensional surface [36]. Thus, in our model, fragments contain 8 amino acids each with reasoning that a secondary structure should be treated as a single unit, allowing the agent to primarily focus on tertiary folding.

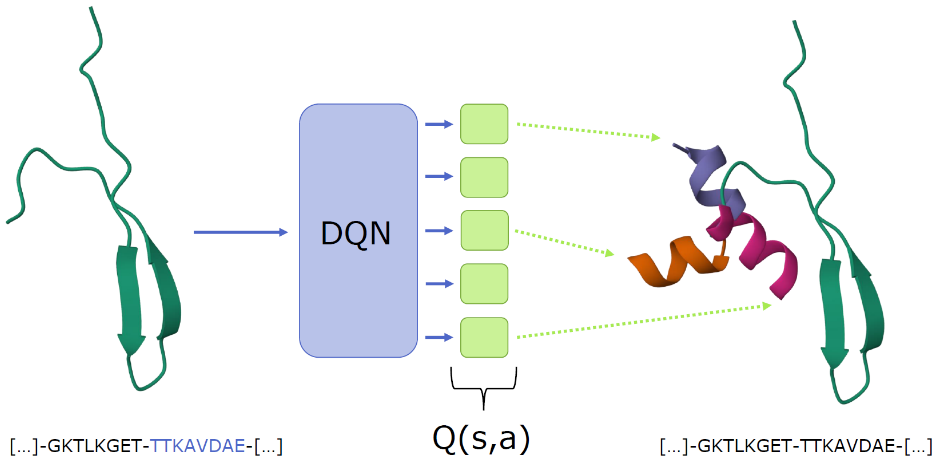

Figure 1: Deep Q learning in the fragmented protein reconstruction environment. The agent chooses how to extend the growing protein structure by appending the next fragment, which is a portion of an alpha helix. Protein visualization generated using Mol* [31].

## 3 Deep Q Learning

In Deep Q Learning, policy is derived from Q-value estimates [22]. The Q-value for a given state-action pair is equal to the expected future discounted reward the agent would receive by choosing said action (see equation 1) [37]. Accordingly, an optimal policy for the agent would be to choose the action with the highest Q-value at every time step [22].

$$Q_\pi(s,a) = R(s) + \gamma V_\pi(s') = R(s) + \gamma\left(\max_{a'} Q_\pi(s',a')\right) \tag{1}$$

Q-values are estimated by a deep neural network [22]. The input to the network is the current state. For every possible action, the network outputs a Q-value estimate. In order to train the network, estimated Q values are compared against actual received rewards as in equation 2 [22].

$$Q_\pi(s,a) \approx (1-\alpha)Q_\pi(s,a) + \alpha\left(R + \gamma\left(\max_{a'} Q_\pi(s',a')\right)\right) \tag{2}$$

During training, there is a tradeoff between exploration and exploitation [12]. If the agent always chooses the apparent best action, it may miss out on better rewards from choosing a different action with an underestimated Q-value. On the other hand, if the agent always chooses a random action, it will tend to explore all action possibilities, yet will not accurately estimate Q-values. A typical compromise is to use an $\epsilon$-greedy training policy [22], whereby the agent chooses a random action with probability equal to $\epsilon$, and otherwise chooses the apparent best action. $\epsilon$ is typically set to a high initial value to encourage exploration and then decreased over time to encourage exploitation.

Training stability is a persistent challenge in Deep Q learning. A common mitigation strategy is to train the Q-value network using random batched samples from a buffer of recent experiences [18]. Another useful strategy is to train two Q-value networks in parallel [23]. The first network's estimates are used to determine the policy, whereas the second network's estimates are used exclusively during training. The second network, called the target network, is responsible for estimating the maximum future action value in the right-hand side of equation 2 [23]. The policy network is trained directly, whereas the target network's parameters are gradually shifted towards the policy network according to the update rule in equation 3 [16].

$$W_{\text{target}} := (1-\tau)W_{\text{target}} + \tau W_{\text{policy}} \tag{3}$$

Deep Q learning models often struggle to learn optimal policies in environments with sparse rewards [3]. A mitigation strategy called reward backfill retroactively assigns non-zero rewards to previous zero-rewarded time steps at an exponentially decaying rate [3]. The backfilled rewards effectively provide more immediate feedback about the future value of an action.

## 4 Training

The fragmented protein reconstruction model is implemented as a Gymnasium [6] environment in Python. Model observations consist of a sequence of fragment encodings. Every fragment encoding is a tuple of the following information:

- A binary indicator of whether the fragment has already been placed.
- The absolute index of the fragment in the sequence, divided by the total number of fragments.
- The relative index of the fragment to the most-recently placed fragment, divided by the total number of fragments.
- A one-hot categorical encoding of the amino acid sequence within the fragment. There are 22 possible categories for every amino acid based on the standard 20-amino acid alphabet plus one category for unknown and one category for padding (first and last fragments only).
- A one-hot categorical encoding of the fragment's vector representation length. There are 15 evenly-spaced categorical bins from 0 to 30 angstrom, plus 1 category for lengths of more than 30 angstrom.
- A unit-scaled vector aligned with the direction of the placed fragment, relative to the local basis of the most-recently placed fragment. If this fragment has not yet been placed, then this vector is zero.
- A unit-scaled vector aligned with the position of the placed fragment, relative to the local basis of the most-recently placed fragment. If this fragment has not yet been placed, then this vector is zero.
- A one-hot categorical encoding of the distance between the placed fragment and the most-recently placed fragment. There are 20 evenly spaced categorical bins from 0 to 40 angstrom, plus 1 category for distances of more than 40 angstrom. If this fragment has not yet been placed, then all categories are assigned 0.

The policy and target Q-value networks are based on the same transformer-style [40] model, implemented in PyTorch [39]. The model consists of:

1. An embedding layer to map each fragment encoding to 256 dimensions.
2. A layer to prepend the embedded fragment sequence with a learnable 256-dimensional prediction token.
3. 12 multi-headed self-attention blocks [40], where each block consists of
   (a) An 8-headed self-attention layer with dropout probability 0.1 [35], along with a residual connection [9].
   (b) A projection layer into 1024 dimensions, followed by ReLU activation, followed by a projection back to 256 dimensions, along with a residual connection [9].
4. An 8-headed self-attention layer using only the prediction token to generate keys (no dropout).
5. A projection layer into 1024 dimensions, followed by ReLU activation, followed by a projection back to 256 dimensions.
6. A Q-value estimation layer with output dimension 288.

During training, rewards are scaled according to equation 4 in order to normalize reward ranges for proteins of different sizes.

$$R_{\text{scaled}} = R \times 1.2/\{\text{number of fragments}\} \tag{4}$$

The experience replay buffer [18] contains all state transitions from all simulations run using the most recent 20 policy network iterations. For every policy network iteration, approximately 2500 simulations are run with proteins from a provided data set. The two datasets used during training were (a.) the single protein sequence PDB ID 2IGD chain A [41, 4] (b.) all available proteins from CATH [32] superfamily 3.10.20.10. During training, the $\epsilon$-greedy policy is determined by the current policy network iteration $t$ according to equation 5.

$$\epsilon = 0.01 + 0.99 \times 0.99^{t} \tag{5}$$

Rewards are retroactively backfilled [3] at the end of each simulation such that the earliest time step receives 0.1 times the final reward (see equation 6)

$$R_{\text{backfill}} = R_{\text{final}} \times 0.1^{\left(1-\frac{\{\text{fragment index}\}}{\{\text{total number of fragments}\}}\right)} \tag{6}$$

After all simulations have been run, the policy network is trained on 10 randomly-sampled batches of 1024 state transitions from the experience replay buffer. The model is trained to minimize Huber loss [10] between the predicted Q-value and the estimated target Q-value from equation 2, plus L2 weight regularization for all non-bias model parameters with penalty 0.01. The Q-learning rate $\alpha$ is set to 0.75 and the discount factor $\gamma$ is set to 0.9. The network is trained by an AdamW [20] AMSGrad-variant [28] optimizer with learning rate 0.0001 and beta-parameters 0.9 and 0.999 for computing the average gradient and its square, respectively. After the policy network is trained on 10 batches, the target network receives a soft parameter update [17] according to equation 3 with $\tau$ set to 0.01.

After every training iteration, the new policy network is evaluated. During evaluation, approximately 1250 simulations are run with proteins from a provided data set. $\epsilon$ is set to 0 so that the policy deterministically chooses the action with the highest Q-value estimate. Rewards are not scaled during evaluation.

The models were trained on Rensselaer Polytechnic Institute's AiMOS supercomputer [29] using a single compute node with 2 20-core IBM Power9 processors and 4 NVIDIA Tesla V100 GPUs. The training process cycles between 3 phases: training, evaluation, and data generation. During the training phase, batches are distributed across 4 processes to utilize all 4 GPUs. Likewise, during the evaluation and data generation phases, many simulations are run in parallel and served by four GPU-bound processes to maximize hardware utilization.

## 5 Results

After training for 40 hours, or around 1000 iterations, the policy network for the single-protein 2IGD dataset converged to near-optimality, with average FAPE loss 3.25 angstrom from the empirical structure. The discrete action space imposes a lower bound on FAPE loss. The full policy network evaluation history is shown in figure 2.

After training for 70 hours, or around 600 iterations, the policy network for the CATH superfamily dataset did not fully converge, nevertheless achieving an average FAPE loss of 7.73 angstrom. Once again, the discrete action space imposes a lower bound on FAPE loss. The full policy network evaluation history is shown in figure 2.

## 6 Discussion

The deep Q learning agent for the fragmented protein reconstruction task is effectively trained in a supervised setting. However, by reformulating protein structure prediction as a reinforcement learning problem, the agent utilizes an incremental approach to building protein structure, with self-attention allowing for selective focus of both neighboring and distant fragments at every time step [40]. In contrast, supervised protein structure prediction models such as AlphaFold 2 [11] predict atomic coordinates for the entire protein chain at once, necessitating a much more complex internal state.

The 12-self-attention-block deep Q learning model is able to learn an effective policy for minimizing FAPE loss, both for a single sequence and for a single superfamily of related proteins. During the first 250 iterations of training, when random actions are chosen with $\epsilon \geq 0.09$, the quality of the policy is highly variable, ranging from 13-25 angstrom averaged FAPE loss. As training continues,

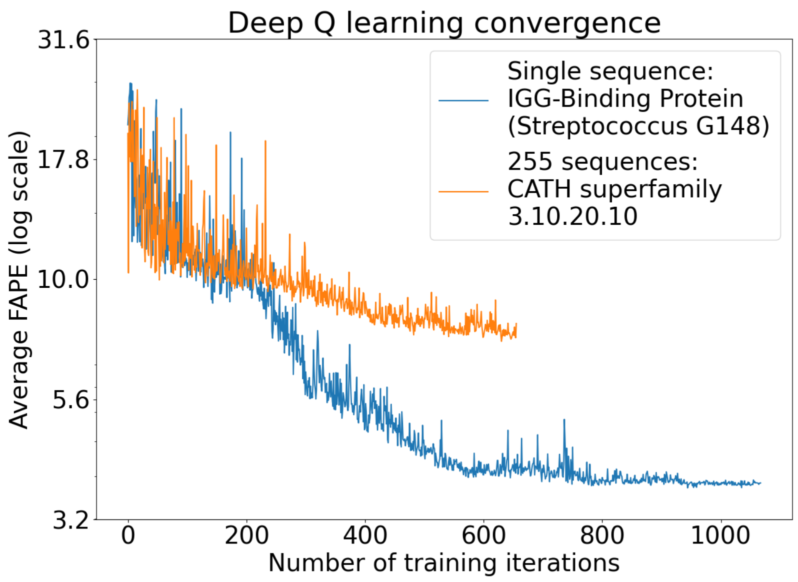

Figure 2: Policy network evaluation every 10 training batches. Note the vertical log scale.

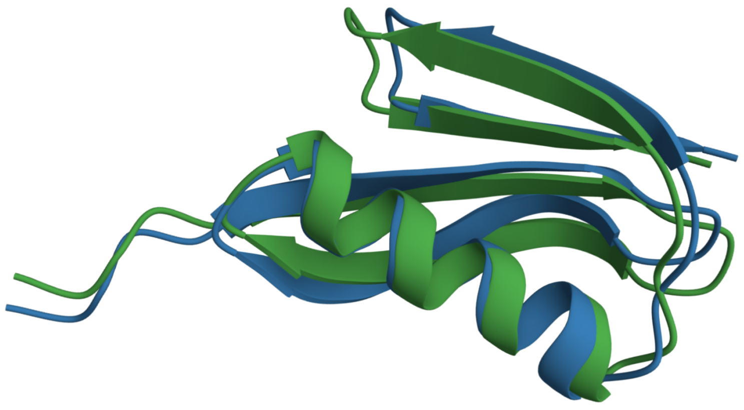

Figure 3: Pictured: PDB protein 2IGD chain A [41, 4]. The predicted structure is shown in blue. The empirical structure is shown in green. Protein visualization generated using Mol* [31].

the algorithm shifts its focus from exploration to exploitation. A paradigm shift occurs as broad policy updates transition into smaller fine-tuning updates. The training process is observed to full convergence for the single sequence dataset in figure 2.

Reward backfill [3] successfully mitigates the effect of sparse rewards in the fragmented protein reconstruction task. By retroactively providing immediate feedback to the learning agent, this technique permits the model to understand long-term consequences of immediate actions. Furthermore, batched training via use of an experience replay buffer [18] containing 2500 simulations from each of the past 20 policy network iterations stabilizes training significantly. After 1000 iterations of training, only slight fluctuations are observed in quality of the learned policy, on the order of $1 \times 10^{-2}$ angstrom.

The inference-time computational complexity of using a transformer-style Q-value model at every time step of the fragmented protein reconstruction task is $O(n^3)$. Although this cost can be partially mitigated by use of distributed hardware, this approach quickly becomes infeasible for very long protein chains.

### 6.1 Future work

The fragmented protein reconstruction task relies on preliminary knowledge of fragment structures. Due to this limitation, a model for this task cannot be used for standalone end-to-end protein structure prediction. However, if a model for the fragmented protein reconstruction task is combined with another model that is able to accurately predict local (secondary) structures within an 8-amino acid window, the stacked model could to generate a full tertiary structure with no preliminary structural knowledge.

This paper does not explore the performance of Deep Q Learning for fragmented protein reconstruction on diverse protein datasets. It stands to reason that a deeper, more expressive model than the one used in this paper (for instance, a 40-block transformer-style network, as in AlphaFold 2 [11]) could generalize well across a comprehensive set of protein structures. A generalized Q-value model would prove useful for inferential structure prediction given preliminary knowledge of fragment structures.

As mentioned in section 4, the discrete action space imposes a lower bound on reconstruction error. An alternative formulation of the fragmented protein reconstruction environment uses a continuous action space via direct prediction of coordinate transformations in unit quaternion or angle-axis format. In this proposed setting, Deep Q learning would no longer be viable. Instead, a policy-gradient method such as DDPG [15], TD3 [7] or SAC [8] could learn an optimal policy using the actor-critic approach. Without a lower bound on reconstruction error, the resulting policy could potentially converge to a near-perfect reconstruction.

## 7 Acknowledgements

We re-use code from the OpenFold project [1] to read and write PDB files in the `.mmCIF` and `.pdb` file formats. Our software implementation (available on GitHub) also makes use of an alternative reinforcement learning algorithm defined by AlphaZero [33]. A PyTorch tutorial entitled “Reinforcement Learning (DQN) Tutorial” [25] was used as an implementation reference for Deep Q Learning in PyTorch. A blog post entitled “From-scratch implementation of AlphaZero for Connect4” [34] was used as an implementation reference for AlphaZero’s Monte-Carlo Tree Search algorithm.

## References

[1] Gustaf Ahdritz, Nazim Bouatta, Sachin Kadyan, Qinghui Xia, William Gerecke, Timothy J O’Donnell, Daniel Berenberg, Ian Fisk, Niccolò Zanichelli, Bo Zhang, Arkadiusz Nowaczynski, Bei Wang, Marta M Stepniewska-Dziubinska, Shang Zhang, Adegoke Ojewole, Murat Efe Guney, Stella Biderman, Andrew M Watkins, Stephen Ra, Pablo Ribalta Lorenzo, Lucas Nivon, Brian Weitzner, Yih-En Andrew Ban, Peter K Sorger, Emad Mostaque, Zhao Zhang, Richard Bonneau, and Mohammed AlQuraishi. Openfold: Retraining alphafold2 yields new insights into its learning mechanisms and capacity for generalization. *bioRxiv*, 2022. doi: 10.1101/2022.11.20.517210. URL `https://www.biorxiv.org/content/10.1101/2022.11.20.517210`.

[2] J.M. Berg, J.L. Tymoczko, and L. Stryer. *Biochemistry, Fifth Edition: International Version*. W. H. Freeman, 2002. ISBN 9780716746843. URL `https://books.google.com/books?id=zSHitAEACAAJ`.

[3] Mohammad Reza Bonyadi, Rui Wang, and Maryam Ziaei. Self-punishment and reward backfill for deep q-learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–8, 2022. doi: 10.1109/tnnls.2021.3140042. URL `https://doi.org/10.1109%2Ftnnls.2021.3140042`.

[4] S. Butterworth, V.L. Lamzin, D.B. Wigley, J.P. Derrick, and K.S. Wilson. ANISOTROPIC STRUCTURE OF PROTEIN G IGG-BINDING DOMAIN III AT 1.1 ANGSTROM RESOLUTION, July 1998. URL `https://doi.org/10.2210/pdb2igd/pdb`.

[5] Márcio Dorn, Mariel Barbachan e Silva, Luciana S. Buriol, and Luis C. Lamb. Three-dimensional protein structure prediction: Methods and computational strategies. *Computational Biology and Chemistry*, 53: 251–276, Dec 2014. doi: 10.1016/j.compbiolchem.2014.10.001.

[6] Farama Foundation. Gymnasium, 2023. URL `https://gymnasium.farama.org/`.

[7] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018.

[8] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[10] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964. doi: 10.1214/aoms/1177703732. URL `https://doi.org/10.1214/aoms/1177703732`.

[11] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, and et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

[12] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey, 1996.

[13] A.L. Lehninger, D.L. Nelson, and M.M. Cox. *Lehninger Principles of Biochemistry*. Lehninger Principles of Biochemistry. W. H. Freeman, 2005. ISBN 9780716743392. URL `https://books.google.com/books?id=7chAN0UY0LYC`.

[14] Yanjun Li, Hengtong Kang, Ketian Ye, Shuyu Yin, and Xiaolin Li. Foldingzero: Protein folding from scratch in hydrophobic-polar model, 2018.

[15] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[16] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[17] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019.

[18] Long-Ji Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon University, USA, 1992. UMI Order No. GAX93-22750.

[19] H. Lodish. *Molecular Cell Biology*. W. H. Freeman, 2008. ISBN 9780716776017. URL `https://books.google.com/books?id=K3JbjG1JiUMC`.

[20] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[21] C.K. Mathews and K.E. Van Holde. *Biochemistry*. Benjamin/Cummings series in the life sciences and chemistry. Benjamin/Cummings Publishing Company, Incorporated, 1996. ISBN 9780805339314. URL `https://books.google.com/books?id=U9_7PgAACAAJ`.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, and et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236.

[24] R.K. Murray. *Harper's Illustrated Biochemistry*. Appleton & Lange, 2006. ISBN 9780071461979. URL `https://books.google.com/books?id=lbu7tAEACAAJ`.

[25] Adam Paszke and Mark Towers. Reinforcement learning (dqn) tutorial, 2023. URL `https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html`.

[26] Zhenling Peng, Wenkai Wang, Renmin Han, Fa Zhang, and Jianyi Yang. Protein structure prediction in the deep learning era. *Current Opinion in Structural Biology*, 77:102495, Dec 2022. doi: 10.1016/j.sbi.2022.102495.

[27] Farzad Peyravi, Alimohammad Latif, and Seyed Mohammad Moshtaghioun. Protein tertiary structure prediction using hidden markov model based on lattice. *Journal of Bioinformatics and Computational Biology*, 17(02):1950007, Apr 2019. doi: 10.1142/s0219720019500070.

[28] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=ryQu7f-RZ`.

[29] Rensselaer Polytechnic Institute. Artificial intelligence multiprocessing optimized system (aimos), 2023. URL `https://cci.rpi.edu/aimos`.

[30] George D Rose, Patrick J Fleming, Jayanth R Banavar, and Amos Maritan. A backbone-based theory of protein folding. *Proc. Natl. Acad. Sci. U. S. A.*, 103(45):16623–16633, November 2006.

[31] David Sehnal, Sebastian Bittrich, Mandar Deshpande, Radka Svobodová, Karel Berka, Václav Bazgier, Sameer Velankar, Stephen K Burley, Jaroslav Koča, and Alexander S Rose. Mol* Viewer: modern web app for 3D visualization and analysis of large biomolecular structures. *Nucleic Acids Research*, 49(W1): W431–W437, 05 2021. ISSN 0305-1048. doi: 10.1093/nar/gkab314. URL `https://doi.org/10.1093/nar/gkab314`.

[32] Ian Sillitoe, Nicola Bordin, Natalie Dawson, Vaishali P Waman, Paul Ashford, Harry M Scholes, Camilla S M Pang, Laurel Woodridge, Clemens Rauer, Neeladri Sen, Mahnaz Abbasian, Sean Le Cornu, Su Datt Lam, Karel Berka, Ivana Hutařová Varekova, Radka Svobodova, Jon Lees, and Christine A Orengo. CATH: increased structural coverage of functional space. *Nucleic Acids Research*, 49(D1):D266–D273, 11 2020. ISSN 0305-1048. doi: 10.1093/nar/gkaa1079. URL `https://doi.org/10.1093/nar/gkaa1079`.

[33] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, and et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. doi: 10.1038/nature24270.

[34] Wee Tee Soh. From-scratch implementation of alphazero for connect4, 2019. URL `https://towardsdatascience.com/from-scratch-implementation-of-alphazero-for-connect4-f73d4554002a`.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, jan 2014. ISSN 1532-4435.

[36] Peter D Sun, Christine E Foster, and Jeffrey C Boyington. Overview of protein structural and functional folds. *Curr. Protoc. Protein Sci.*, Chapter 17(1):Unit 17.1, May 2004.

[37] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press, 1998. ISBN 9780262303842. URL `https://books.google.com/books?id=U57uDwAAQBAJ`.

[38] C Tanford. The hydrophobic effect and the organization of living matter. *Science*, 200(4345):1012–1018, June 1978.

[39] The Linux Foundation. Pytorch, 2019. URL `https://pytorch.org/`.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[41] wwPDB consortium. Protein Data Bank: the single global archive for 3D macromolecular structure data. *Nucleic Acids Research*, 47(D1):D520–D528, 10 2018. ISSN 0305-1048. doi: 10.1093/nar/gky949. URL `https://doi.org/10.1093/nar/gky949`.

[42] Kaiyuan Yang, Houjing Huang, Olafs Vandans, Adithya Murali, Fujia Tian, Roland H.C. Yap, and Liang Dai. Applying deep reinforcement learning to the HP model for protein structure prediction. *Physica A: Statistical Mechanics and its Applications*, 609:128395, jan 2023. doi: 10.1016/j.physa.2022.128395. URL `https://doi.org/10.1016%2Fj.physa.2022.128395`.