MODEL-BASED REINFORCEMENT LEARNING FOR PROTEIN BACKBONE DESIGN

Frederic Renard [†§*] **Cyprien Courtot** [†] **Alfredo Reichlin** [‡] **Oliver Bent** [†§]

ABSTRACT

Designing protein nanomaterials of predefined shape and characteristics has the potential to dramatically impact the medical industry. Machine learning (ML) has proven successful in protein design, reducing the need for expensive wet lab experiment rounds. However, challenges persist in efficiently exploring the protein fitness landscapes to identify optimal protein designs. In response, we propose the use of AlphaZero to generate protein backbones, meeting shape and structural scoring requirements. We extend an existing Monte Carlo tree search (MCTS) framework by incorporating a novel threshold-based reward and secondary objectives to improve design precision. This innovation considerably outperforms existing approaches, leading to protein backbones that better respect structural scores. The application of AlphaZero is novel in the context of protein backbone design and demonstrates promising performance. AlphaZero consistently surpasses baseline MCTS by more than 100% in top-down protein design tasks. Additionally, our application of AlphaZero with secondary objectives uncovers further promising outcomes, indicating the potential of model-based reinforcement learning (RL) in navigating the intricate and nuanced aspects of protein design.

1 INTRODUCTION

The inverse design of proteins to optimise predetermined attributes is core to applications spanning from pharmaceutical drug development (Lagassé et al., 2017) to materials science (DiMarco & Heilshorn, 2012) or plastic recycling (Zhu et al., 2022). Machine learning (ML) has showcased its versatility in protein design, notably in the prediction of protein structures using AlphaFold (Jumper et al., 2021) and the design of protein sequences through ProteinMPNN (Dauparas et al., 2022), significantly enhancing the capabilities of *in silico* protein design. Beyond structure prediction, ML has proven to be highly effective in optimizing the complex and often irregular fitness functions associated with protein structures (Gront et al., 2012; Wu et al., 2019; Gao et al., 2020). Optimizing these fitness functions is particularly challenging due to the necessity of exploring the immense combinatorial space of amino acid sequences and structural configurations.

The success of reinforcement learning (RL) in complex combinatorial problems (Mazyavkina et al., 2021), such as the bin-packing (Laterre et al., 2018) and traveling salesman (Khalil et al., 2017; Grinsztajn et al., 2023) problems, underscores its potential in optimizing protein fitness functions. At its core, RL operates on a simple yet powerful paradigm: an agent learns to make decisions by taking actions that maximize future rewards (Sutton & Barto, 2018). Model-based RL (Arulkumaran et al., 2017) differentiates itself by using models to simulate future states, allowing for strategic planning and foresight. This subfield of RL was revolutionized by the introduction of the AlphaZero class of generalised game-playing RL algorithms (Silver et al., 2016; 2017; 2018), which achieved state-of-the-art performance in the games of chess, shogi and go. AlphaZero navigates the vast tree of potential states and actions using a Monte-Carlo tree search (MCTS) guided by a policy-value neural network. DyNa-PPO (Angermueller et al., 2019) pioneered the use of model-based RL applied to protein design by modeling the design of proteins as a Markov decision process (MDP)

[*] Work done during internship at InstaDeep Ltd

[†] InstaDeep Ltd, 40B Rue du Faubourg Poissonnière, Paris, 75010

[‡] Computer Science and Engineering, KTH Royal Institute of Technology,Stockholm, Sweden

[§] Corresponding authors: {f.renard,o.bent}@instadeep.com where amino-acid sequences are filled from left to right and the reward is chosen depending on the objective, such as optimizing the energy of protein contact Ising models (Marks et al., 2011) or transcription binding sites. EvoPlay (Wang et al., 2023) later investigated the use of the single-player version of AlphaZero to design protein sequences and new luciferase variants.

Focusing on the top-down design of protein nanomaterials of predefined shape, Lutz et al. (2023) developed a MCTS approach to successfully design protein backbones while optimizing structural protein scores. This approach iteratively assembles protein secondary structures, alpha-helices and loops, to construct protein backbones. Cryo–electron microscopies (Bonomi & Vendruscolo, 2019) of the structures designed with this approach were almost identical to the *in silico* designs. This novel use of MCTS paves the way to investigate the application of AlphaZero on such a task.

We showcase the efficacy of AlphaZero in designing protein backbones of icosahedral shape while optimizing protein structural scores. Our contributions are threefold:

- We benchmark AlphaZero against the MCTS approach of Lutz et al. (2023) to compare performance in effectively sampling the protein backbone space to optimize the reward function.
- We demonstrate how the design of the MDP, and more specifically of the reward function, can influence the learning and performance of AlphaZero.
- We propose a novel AlphaZero approach including side objectives to regularize the policy-value network throughout learning and benchmark it against the original AlphaZero algorithm.

2 Methods

2.1 Markov Decision Process

State and action spaces We formulate the protein backbone design problem as a MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{P}, r)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $\mathbb{P}$ and reward r. Both the state and action spaces follow the approach of Lutz et al. (2023). The transition probabilities are 1 if the agent takes a legal action given its state, 0 else. The space of possible states is the ensemble of the protein backbones that fit inside of an icosahedron of predefined radius. The protein backbones are composed of alanines. Hence, each state is represented by a matrix of shape $(5 \times \text{number of amino-acids}, 3)$ where every quintuple contains the cartesian coordinates of the three carbon atoms, the nitrogen atom and the oxygen atom of every alanine. The space of possible actions is the union of three distinct subsets of action spaces; the actions of adding an alpha-helix of 9 to 22 residues on either end of the protein backbone; the actions of adding a loop sampled from one of 316 different loop clusters on either end of the protein backbone; or the action to terminate the episode. As described by Lutz et al. (2023), beta strands are excluded from the design space to restrict the size of the design space. They could be added to generalize to a greater subset of proteins.

Reward Adhering to the approach established by Lutz et al. (2023), the reward is the combination of five different structural protein scores; the core score $C(s)$ quantifies the formation of an hydrophobic core; the helix score $H(s)$ quantifies if an alpha-helix is detaching from the rest of the protein backbone; the porosity score $P(s)$ assesses how porous the structure is; the monomer designability score $M(s)$ and the interface designability score $I(s)$ quantify how favorable the geometric interactions are, i.e. between core amino-acids and between the protein backbone's images by the icosahedral symmetries. The goal of the RL agents is to design protein backbones that meet score thresholds introduced by Lutz et al. (2023), specifically: $C_0 = 0.2,\ H_0 = 2.0,\ P_0 = 0.45,\ M_0 = 0.9,\ I_0 = 17$. Combining those scores in a reward is a key step to ensure correct learning of the RL agents.

Two different formulations of this reward are studied: Lutz et al. (2023) proposed a sigmoid reward formulation and we propose a novel thresholds reward formulation. The sigmoid reward is:

$$r(s) = \frac{m}{1 + \exp(-a(\tilde{r}(s) - b))} \quad \text{with } \tilde{r}(s) = 0.05 \prod_{S \in \{C,H,P,M,I\}} \sigma_S(S(s)) \tag{1}$$

With $m = 1.0, a = 0.03, b = 200.0$, and where each score is normalized by the sigmoids σ_S. When trained with the sigmoid reward, algorithms will be noted `Algorithm` (sigmoid).

The thresholds reward is :

$$r(s) = \frac{1}{5}\sum_{S} \max\left(\tau_0, \frac{S(s)}{S_0}\right) \text{ with } S \in \{C, H, P, M, I\} \tag{2}$$

Where τ_0 is a hyperparameter. When trained with the thresholds reward, algorithms will be noted `Algorithm` (thresholds). In contrast to the sigmoid reward which pushes extremely small rewards to the sigmoid curve's tail, the thresholds reward reduces reward scarcity, setting τ_0 as the target reward.

Episodes Our methodology adopts the sequential design workflow for protein backbones introduced by Lutz et al. (2023) and is described in Figure 7. First, an alpha-helix of five residues is initialized inside the icosahedron, uniformly sampling its position from a buffer of 5.000 initialization positions. Then, this alpha-helix is extended to a size between 9 to 22 residues on either the C-terminal or the N-terminal end. Next, an end is chosen and a loop is added on this end. Those last two steps are repeated until the terminal action is chosen, while enforcing that episodes can only terminate if the last secondary structure added was an alpha-helix. When the episode is finished, the reward is computed.
Between each step, geometric checks are performed such that the secondary structure to be added cannot clash with the protein backbone or with the image of the protein backbone by one of its icosahedral symmetries. If the secondary structure does not meet these geometric conditions, the action is deemed illegal and cannot be taken by the agent.

2.2 AlphaZero for protein backbone design

AlphaZero algorithm The original AlphaZero algorithm (Silver et al., 2018) alternates between phases of self-play and phases of learning. In the self-play phases, episodes are completed by selecting at each step actions through a neural-network guided MCTS search. At the end of each episode, the reward r_T, the state s_T and the tree policy $\boldsymbol{\pi}$ are stored in a buffer. Then, during the learning phase, tuples $(\boldsymbol{\pi}, r_T, s_T)$ are sampled from the buffer and the policy and value estimates of the state s_T, $(\boldsymbol{p}, v)$, are computed by the neural network. The policy $\boldsymbol{p}$ is a vector of the probability over the actions given s_T and the value v is a prediction of the future reward of s_T. The parameters θ of the neural network are updated to minimize the loss :

$$L_0 = (r_T - v)^2 - \boldsymbol{\pi}^T \log \boldsymbol{p} + c||\theta||^2 \tag{3}$$

Figure 1 presents how the AlphaZero algorithm can be used to iteratively design protein backbones.

The most crucial hyperparameters for the AlphaZero algorithm are the exploration hyperparameters of the MCTS search, typically noted c_{puct} for the upper confidence bound applied to trees (UCT) coefficient, τ which is a temperature parameter applied to the tree policy $\boldsymbol{\pi}$ to smoothen it and the number of MCTS simulations performed at each step. In particular, the number of MCTS simulations performed at each step impacts the diversity of the designs as a high number of MCTS simulations allows the agent to explore more of the protein backbone space.

AlphaZero algorithm with side-objectives In this paper, we propose to store for each terminal state, in addition to the previous elements, the value of the five scores for the protein backbone s_T : $C(s_T), H(s_T), P(s_T), M(s_T), I(s_T)$. The policy-value network is modified by adding five different heads which compute, given a protein backbone s_T, estimates for the five different scores: $\hat{C}(s_T), \hat{H}(s_T), \hat{P}(s_T), \hat{M}(s_T), \hat{I}(s_T)$.

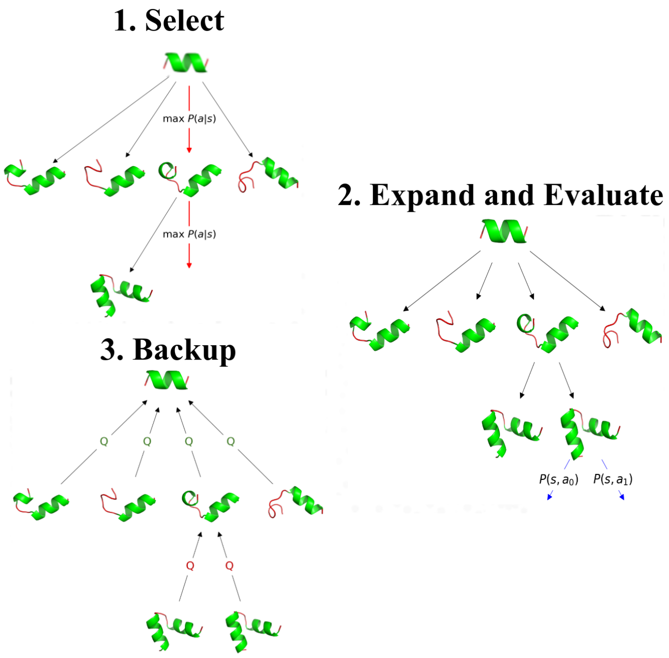

Figure 1: Diagram of the AlphaZero algorithm action selection process. Starting from the root node, the tree of states and actions is expanded by the repetition of the select, expand and evaluate, and backup phases. First, a new child node is selected by maximizing $P(a|s) = Q(s,a) + c_{puct}P(s,a)\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$ with $P(s,a)$ the policy network output, $Q(s,a)$ the mean action value of (s,a) and $N(s,a)$ the number of visits of (s,a). In the second phase, this new child node is evaluated by the neural network $f_\theta(s) = (P(s,a), V(s))$ with $V(s)$ the value network output. In the third phase, the value estimate $V(s)$ is used to update the Q values for the parent nodes. After a number of MCTS simulations, an action is selected according to $\boldsymbol{\pi}(a|s) = \frac{N(s,a)^{1/\tau}}{\sum_b N(s,b)^{1/\tau}}$. Once a terminal state is reached, $(\boldsymbol{\pi}, r_T, s_T)$ are stored in a buffer.

Then, during the learning phase, tuples $(\boldsymbol{\pi}, r_T, s_T, C(s_T), H(s_T), P(s_T), M(s_T), I(s_T))$ are sampled from the buffer, the output by the neural network $(\boldsymbol{p}, v, \hat{C}(s_T), \hat{H}(s_T), \hat{P}(s_T), \hat{M}(s_T), \hat{I}(s_T))$ of the state s_T is computed and the parameters θ of the neural network are updated to minimize the loss :

$$L = L_0 + \sum_{S \in \{C,H,P,M,I\}} \lambda_S (S(s_T) - \hat{S}(s_T))^2 \tag{4}$$

Where the λ_S coefficients were chosen to scale the sum of the score losses to the order of magnitude of $\frac{L_0}{10}$: $\lambda_C = 1000, \lambda_H = 1, \lambda_P = 10, \lambda_I = 0.1, \lambda_M = 1$. When comparing this AlphaZero approach to the original AlphaZero, it will be noted AlphaZero (side-objectives) as opposed to AlphaZero (original).

2.3 Implementation

The neural network architectures used for both AlphaZero algorithms are detailed in Appendix A.2. Appendix A.3 shows the details of the architecture search that was performed to select expert networks for the protein structure scores.

3 Results

3.1 Benchmark of MCTS against AlphaZero

Motivation and design The benchmark is designed to compare the score distributions of AlphaZero and MCTS with initializations inside the icosahedron that were not used to train AlphaZero. 100 alpha-helix initializations are generated inside an icosahedron of radius 75 Angstroms. 10.000 protein backbones are generated for each one of those initializations along with their scores with MCTS using the code of Lutz et al. (2023). Then, 300 protein backbones are generated with AlphaZero (sigmoid) and AlphaZero (thresholds) on each initialization with 300 MCTS simulations at each step. The hyperparameters used for training are referenced in Appendix A.2, Table 2. The metrics of interest are the score distributions of the protein backbones generated by the three methods. The mean and 95% bootstrap confidence intervals for each score are summarized in Figure 2.

Results First, we observe a clear superiority of both AlphaZero algorithms with respect to the MCTS approach developed by Lutz et al. (2023), with a mean improvement of factor 5 for the core score, 1.8 for the interface designability score, 7 for the helix score, 5 for the porosity score and 5 for the monomer designability score. The score distributions and p-values for the statistical tests are detailed in Appendix A.1, Table 1.
Second, the p-values obtained when performing a Wilcoxon Rank-Sum test comparing AlphaZero (thresholds) to AlphaZero (sigmoid), reported in Appendix A.2, indicate that AlphaZero (thresholds) consistently reaches better performance compared to AlphaZero (sigmoid).

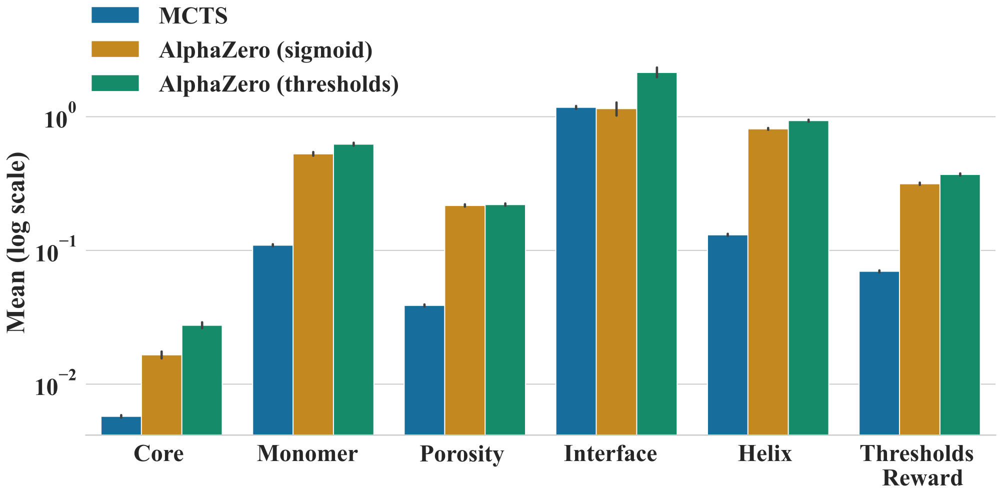

Figure 2: Protein score distributions means with 95% bootstrap confidence intervals. AlphaZero, and more specifically AlphaZero (thresholds) systematically outperforms MCTS on all scores.

3.2 Benchmark of AlphaZero with and without side-objectives

Motivation and design To compare both algorithms, the reward distributions at each step of training are collected for 50 epochs. Training hyperparameters are referenced in Appendix A.2, Table 2. Both algorithms are compared with the thresholds reward formulation. The parameter τ_0 of Equation 2 is set to 1. The metrics of interest are the cumulative distribution functions (CDFs) of the rewards at epoch 1 and epoch 40, presented in Figure 3. The CDFs are computed on batches of 55 to 65 episodes that are self-played by the AlphaZero agent before each learning phase as describedin Section 2.2.

Results The CDF of AlphaZero (side-objectives) is consistently to the right compared to the CDF of the original AlphaZero, achieving higher rewards. The maximum reward per batch of episodes of AlphaZero (side-objectives) reaches 1.0 for 28% of the epochs, which indicates that all five objectives are simultaneously achieved. The average reward of AlphaZero (side-objectives) is consistently above to the mean reward of the AlphaZero (original). The mean episode reward and maximum episode reward of both algorithms throughout the training are reported in Appendix A.2 in Figure 4.

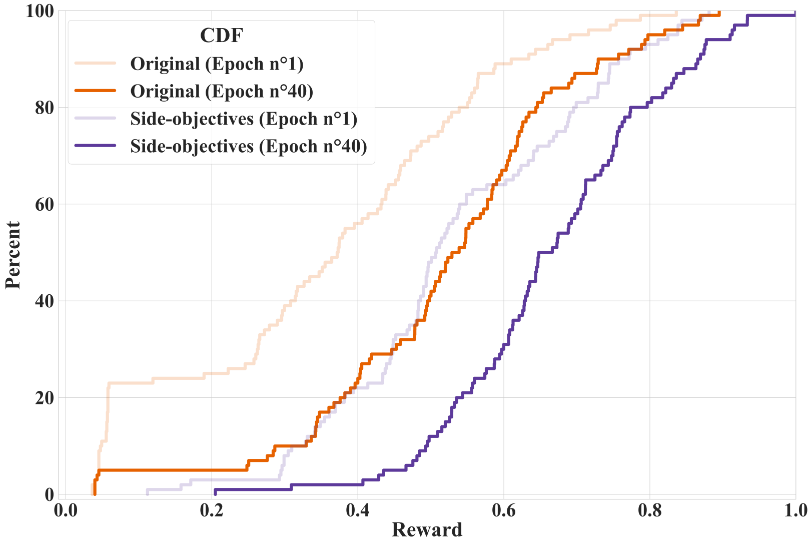

Figure 3: CDF of the reward of both algorithms at the first epoch and at epoch 40 of training. AlphaZero (side-objectives) consistently achieves higher rewards compared to the AlphaZero (original).

4 Discussion

In this work, we have demonstrated the pertinence of model-based deep RL, and more specifically of AlphaZero, for protein backbone design. Both the impact of the reward formulation and the addition of side-objectives emerge as crucial elements for achieving all objectives with AlphaZero. In comparison with the designability, diversity and novelty metrics of diffusion models (Yim et al., 2023), the designability is evaluated by the different scores and the diversity and novelties are constrained by the different protein secondary structures that can be chosen, as described in Appendix A.5.

Several improvements can be made to this work. First, considering the superiority of the AlphaZero agent trained with the threshold reward compared to the sigmoid reward, shaping the reward function is key to achieve better performance. Scheduling the increase of the parameter τ_0 throughout training could allow the agent to learn how to outperform the score thresholds through a curriculum learning approach (Narvekar et al., 2020). However, if such an approach is taken, the performance of the agent in the five different scores should be monitored to ensure the agent does not overspecialize in one score, hurting its performance with the other scores. Another possible improvement would be to search for better possible expert networks architectures for the AlphaZero algorithm with side-objectives. Those side-objectives regularize the policy-value network, improving its performance in designing protein backbones. Future work could also investigate the use of AlphaZero to generate protein backbones of other shapes such as nanopores as in the work of (Lutz et al., 2023) and the transfer learning capabilities of AlphaZero agents. The approach could be enriched by designing protein sequences from the protein backbones with the methodology of Lutz et al. (2023) and employing AlphaFold for a structure prediction test of the designs obtained.

This work paves the way for the use of RL in multi-objective optimization of protein structures. The application of AlphaZero to generate protein backbones alleviates some of the common drawbacks of machine learning, as every design and choices that were made are traceable. It unlocks new methods to design protein nanomaterials of specific shapes to target therapeutic sites of interest or innovative materials.

REFERENCES

Christof Angermueller, David Dohan, David Belanger, Ramya Deshpande, Kevin Murphy, and Lucy Colwell. Model-based reinforcement learning for biological sequence design. In *International conference on learning representations*, 2019.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

Massimiliano Bonomi and Michele Vendruscolo. Determination of protein structural ensembles using cryo-electron microscopy. *Current Opinion in Structural Biology*, 56:37–45, 2019.

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

Rebecca L DiMarco and Sarah C Heilshorn. Multifunctional materials through modular protein engineering. *Advanced Materials*, 24(29):3923–3940, 2012.

Wenhao Gao, Sai Pooja Mahajan, Jeremias Sulam, and Jeffrey J Gray. Deep learning in protein structural modeling and design. *Patterns*, 1(9), 2020.

Nathan Grinsztajn, Daniel Furelos-Blanco, Shikha Surana, Clément Bonnet, and Thomas D Barrett. Winner takes it all: Training performant rl populations for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Dominik Gront, Sebastian Kmiecik, Maciej Blaszczyk, Dariusz Ekonomiuk, and Andrzej Koliński. Optimization of protein models. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(3):479–493, 2012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

HA Daniel Lagassé, Aikaterini Alexaki, Vijaya L Simhadri, Nobuko H Katagiri, Wojciech Jankowski, Zuben E Sauna, and Chava Kimchi-Sarfaty. Recent advances in (therapeutic protein) drug development. *F1000Research*, 6, 2017.

Alexandre Laterre, Yunguan Fu, Mohamed Khalil Jabri, Alain-Sam Cohen, David Kas, Karl Hajjar, Torbjorn S Dahl, Amine Kerkeni, and Karim Beguir. Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization. *arXiv preprint arXiv:1807.01672*, 2018.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Isaac D Lutz, Shunzhi Wang, Christoffer Norn, Alexis Courbet, Andrew J Borst, Yan Ting Zhao, Annie Dosey, Longxing Cao, Jinwei Xu, Elizabeth M Leaf, et al. Top-down design of protein architectures with reinforcement learning. *Science*, 380(6642):266–273, 2023.

Debora S Marks, Lucy J Colwell, Robert Sheridan, Thomas A Hopf, Andrea Pagnani, Riccardo Zecchina, and Chris Sander. Protein 3d structure computed from evolutionary sequence variation. *PloS one*, 6(12):e28766, 2011.

Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *The Journal of Machine Learning Research*, 21(1):7382–7431, 2020.

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.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

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

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Yi Wang, Hui Tang, Lichao Huang, Lulu Pan, Lixiang Yang, Huanming Yang, Feng Mu, and Meng Yang. Self-play reinforcement learning guides protein engineering. *Nature Machine Intelligence*, 5(8):845–860, 2023.

Zachary Wu, SB Jennifer Kan, Russell D Lewis, Bruce J Wittmann, and Frances H Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18):8852–8858, 2019.

Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023.

Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

Baotong Zhu, Dong Wang, and Na Wei. Enzyme discovery and engineering for sustainable plastic recycling. *Trends in biotechnology*, 40(1):22–37, 2022.

A Appendix

A.1 Benchmark of MCTS against AlphaZero details

Table 1 presents the mean and standard deviation of the distributions for all five scores for the protein backbones generated by the algorithms, and for each score the p-value of the Wilcoxon Rank-Sum test with the null hypothesis being : the mean of the scores collected by the AlphaZero agent is greater than the mean of the scores collected by the MCTS agent.
In this table, means are noted with μ, standard deviations with σ and the p-values of the Wilcoxon Rank-Sum test with p. P-values below 1×10^{-308} are rounded to zero.

Table 1: Benchmark of MCTS against AlphaZero agents.

	MCTS baseline	**Original AlphaZero with sigmoid reward**	**Original AlphaZero with thresholds reward**
$\boldsymbol{\mu_C}$	0.0056	0.017	0.028
$\boldsymbol{\sigma_C}$	0.02	0.027	0.038
$\boldsymbol{p_C}$	-	2.2×10^{-117}	22.1×10^{-274}
$\boldsymbol{\mu_H}$	0.13	0.81	0.93
$\boldsymbol{\sigma_H}$	0.38	0.36	0.43
$\boldsymbol{p_H}$	-	0	0
$\boldsymbol{\mu_P}$	0.0390	0.216	0.220
$\boldsymbol{\sigma_P}$	0.121	0.117	0.123
$\boldsymbol{p_P}$	-	0	0
$\boldsymbol{\mu_I}$	1.17	1.15	2.14
$\boldsymbol{\sigma_I}$	4.64	3.81	5.21
$\boldsymbol{p_I}$	-	7.0×10^{-4}	6.14×10^{-25}
$\boldsymbol{\mu_M}$	0.11	0.53	0.62
$\boldsymbol{\sigma_M}$	0.30	0.47	0.46
$\boldsymbol{p_M}$	-	0	0

A.2 AlphaZero algorithm

The neural network architecture used for AlphaZero (original) is a multi-layer perceptron (MLP) with shared parameters for the value and policy networks. The input of this network is the flattened protein backbone array. Then, two hidden layers with 512 and 256 neurons were added before two different two-layers heads: the policy head with output size of 663 and the value head with output size 1. LeakyRelu (He et al., 2015) activation with a negative slope of 0.01 were used between each layers.

The mixture of experts architecture used for AlphaZero (side-objectives) is shown in Figure 5. The convolutional neural network (CNN) and MLP used are identical to the ones defined in section A.3. The hyperparameters used for the training of AlphaZero are presented in Table 2. RL experiments were performed with 19 workers collecting rollouts and one worker performing the learning steps. Each worker used an AMD EPYC 7452 CPU with 5GB of RAM.

When performing a Wilcoxon Rank-Sum test with the null hypothesis being : the mean of the scores collected by AlphaZero (thresholds) is greater than the mean of the scores collected by AlphaZero (sigmoid), the p-values obtained are : 6.43×10^{-24} for the core score, 4.90×10^{-24} for the helix score, 1.31×10^{-16} for the monomer designability score, 8.89×10^{-8} for the interface designability score and 0.045 for the porosity score.

The mean episode reward and maximum episode reward of both agents is reported in Figure 4. Those reward statistics are computed on batches of 55 to 65 episodes that are self-played by the AlphaZero agent before each learning phase as described in Section 2.2. Both agents quickly converge and AlphaZero (side-objective) consistently reaches higher reward statistics. Training lasted 5h for the AlphaZero (original) and 7h for AlphaZero (side-objectives).

Table 2: Hyperparameters for training the AlphaZero agents.

Hyperparameter	**Value**
train batch size	1024
learning rate	5×10^{-5}
L2 regularization coefficient	1×10^{-5}
c_{puct}	1.5
τ	1.0
Number of MCTS simulations	5000
Buffer size	100.000
Number of training iterations	200

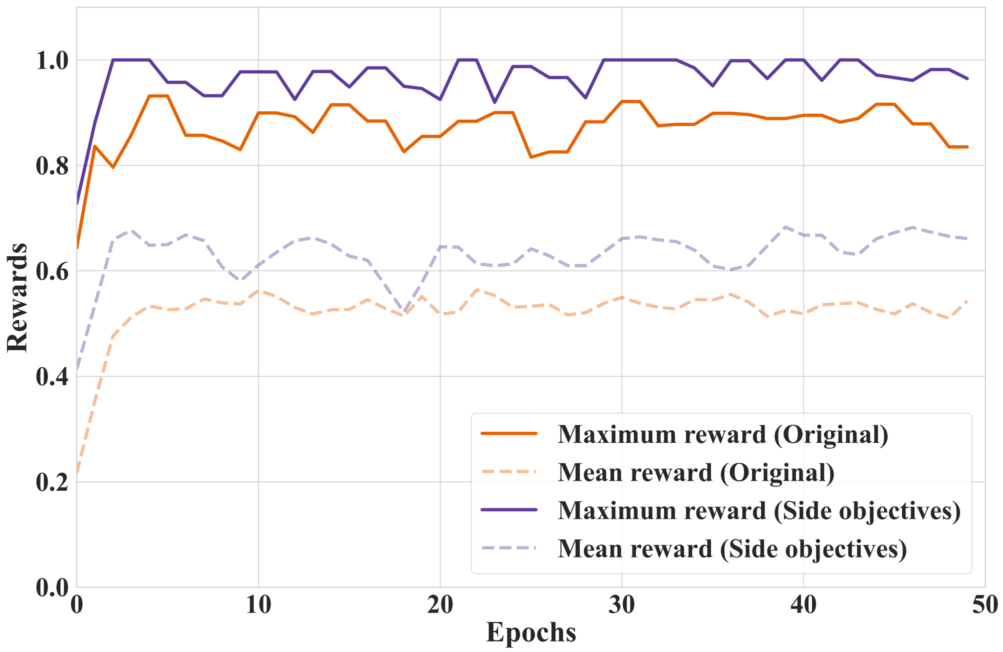

Figure 4: Rewards of the AlphaZero agents throughout training. Both AlphaZero (side-objectives) maximum and mean rewards are consistently higher than those of AlphaZero (original).

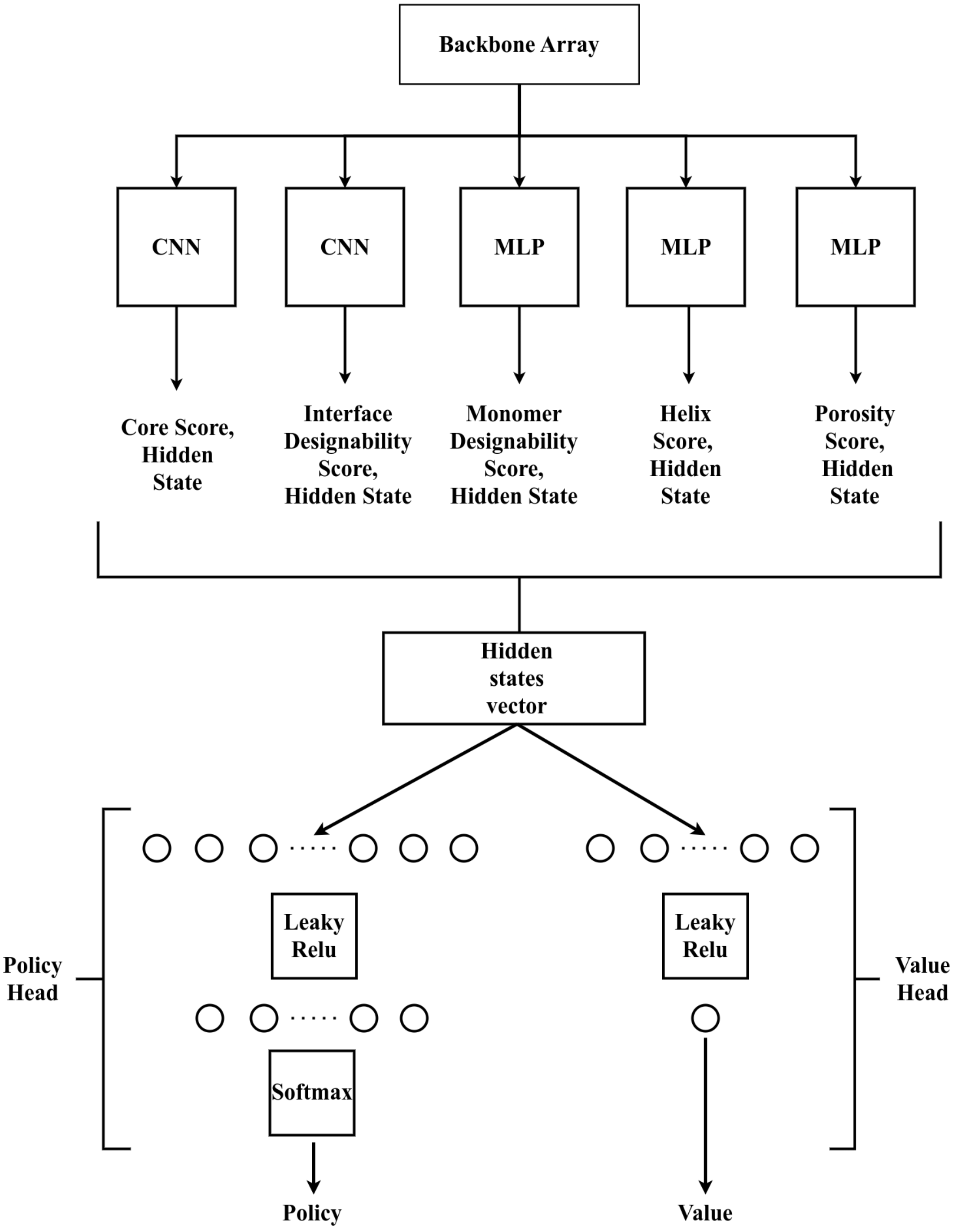
Backbone Array
CNN
CNN
MLP
MLP
MLP
Core Score, Hidden State
Interface Designability Score, Hidden State
Monomer Designability Score, Hidden State
Helix Score, Hidden State
Porosity Score, Hidden State
Hidden states vector
Leaky Relu
Leaky Relu
Softmax
Policy Head
Value Head
Policy
Value

Figure 5: Mixture of experts architecture for AlphaZero. Circles represent linear layers. CNNs are used to predict the core and interface designability score and MLPs for the other scores. The hidden states used to compute the scores are concatenated and used by two different heads : a policy and a value head. The neural network output the policy, the value and the five different protein structure scores.

A.3 Neural networks architecture search

In order to determine what neural network architecture to use for the AlphaZero algorithm, 80.000 protein backbones, represented by matrices of shape $(400, 3)$ were generated through an MCTS search with their corresponding scores. Then, several architectures were tested on the supervised learning task of predicting the scores from the protein backbones, including a MLP, a long-short term memory network (LSTM) (Yu et al., 2019) and a mixture of experts composed of two one-dimensional CNNs for the core and interface designability score and three MLPs for the other scores. This dataset is augmented by computing the image for each backbone by 8 icosahedral symmetries, yielding a dataset of 640.000 protein backbones. It is split between a train dataset of size 512.000, a validation dataset of size 80.000 and a test dataset of size 80.000. Datasets are standardized using the train dataset mean and standard deviation. Neural networks are trained to minimize the Mean Squared Error with the training hyperparameters shown in Table 4.
The MLP architecture takes as input the flattened backbone array of shape 1200 and outputs a vector of shape 5. It is a three layers network with respectively 512, 256 and 5 neurons in each layer and LeakyReLu activation functions with a negative slope of 0.01 after each layer, including the output layer.
The LSTM architecture has the same inputs and outputs as the MLP. It is composed of two stacked LSTM layers with a hidden size of 128, followed by a linear layer of size 64, LeakyRelu activation with a negative slope of 0.01 and a final linear layer of size 5.
The CNN architecture accepts as an input backbone arrays of shape (400, 3) and outputs a vector of size 1. LeakyReLu activation functions were used after each layer with a negative slope of 0.01, including the output layer. Its hyperparameters are presented in Table 3.

Table 3: CNN hyperparameters.

Hyperparameter	**Value**
Number of channels first convolutional layer	3
Kernel size first convolutional layer	5
Stride first convolutional layer	3
Number of channels second convolutional layer	16
Kernel size second convolutional layer	2
Stride second convolutional layer	2
Size of first fully connected layer	528
Size of second fully connected layer	64
Size of third fully connected layer	32

All networks are trained with the Adam algorithm (Kingma & Ba, 2014) and a Cosine Annealing learning rate schedule (Loshchilov & Hutter, 2016) to minimize the Mean Squared Error loss between the scores and the network's predictions. For the mixture of experts, the networks are trained with different learning rates. The parameters used for training are shown in Table 4.
Supervised learning experiments were performed on a 12th Gen Intel® Core™ i7-1265U × 12 central processing unit (CPU) with 32 Gio of RAM.

Figure 6 presents the validation losses for all three tested architectures. The mixture of experts proves to be the best architecture both in terms of speed of convergence and validation loss.

Table 4: Training hyperparameters for the architecture search.

Hyperparameter	**Value**
MLP Learning Rate	5×10^{-3}
LSTM Learning Rate	5×10^{-3}
Core score CNN Learning Rate	5×10^{-4}
Interface Designability score CNN Learning Rate	1×10^{-3}
Monomer Designability score MLP Learning Rate	1×10^{-3}
Helix score MLP Learning Rate	5×10^{-3}
Porosity score MLP Learning Rate	1×10^{-3}
Minimum learning rate of cosine annealing	5×10^{-6}
Minimum learning rate reached at epoch	100
Number of training epochs	100
Adam weight decay	1×10^{-5}
Batch size	1024

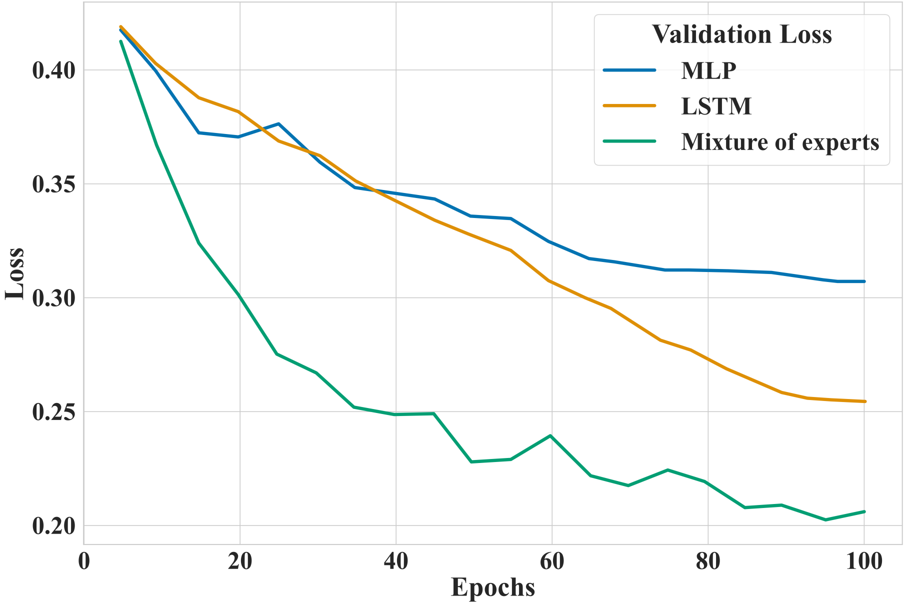

Figure 6: Validation losses for the prediction of protein structure scores from protein backbones. Results show a clear superiority of the mixture of experts network both in terms of speed of convergence and validation loss.

A.4 Protein Backbone Design Episode

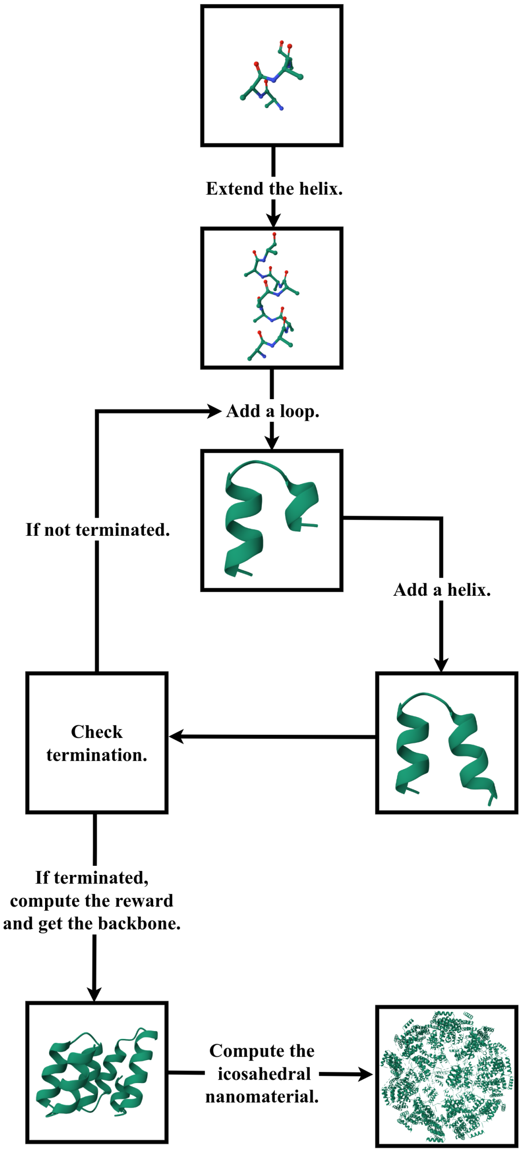
Extend the helix.
Add a loop.
If not terminated.
Add a helix.
Check termination.
If terminated, compute the reward and get the backbone.
Compute the icosahedral nanomaterial.

Figure 7: Flowchart of an episode. During an episode, alpha-helix of between 9 to 22 residues and loops sampled from 316 different loop clusters are iteratively added to the protein backbone. Between each step, geometric checks are performed to avoid clashes between the structure to be added and the current protein backbone or one of its icosahedral symmetries. Geometric checks also prevent the protein backbone from exiting the icosahedral shape. If an action passes the geometric checks, it is legal. If no legal actions can be found, the episode is ended. If the episode ends because no legal actions can be found and the last action was to add a loop, this loop is removed. An episode is terminal if more than 7 alpha-helices were added, if the number of amino-acids is superior to 80 or if the terminal action was chosen. The terminal action can be chosen if the backbone has more than 3 alpha-helices and is always chosen if no legal actions can be found. At each helix-addition step, all legal helix action additions can be chosen. At each loop-addition step, a subset of 50 loop clusters is randomly chosen and one loop from each cluster is sampled. The loop actions the agent can take are the legal loop actions of this subset of loop actions.

A.5 Protein Secondary Structures

This section describes the different protein secondary structures used to construct protein backbones. Figure 8 presents an example of alpha-helix used to construct the protein backbones and Figure 9 presents two different protein loops, amongst more than 20.000, that can be chosen by the agent. The different orientations given to the alpha-helices by the protein loops increase the diversity of the generated backbones while ensuring designability by optimizing the different protein structure scores.

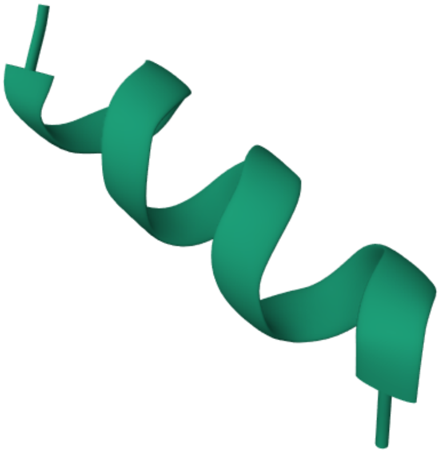

Figure 8: Example of alpha-helix used to construct protein backbones. Image generated with Mol* (Sehnal et al., 2021).

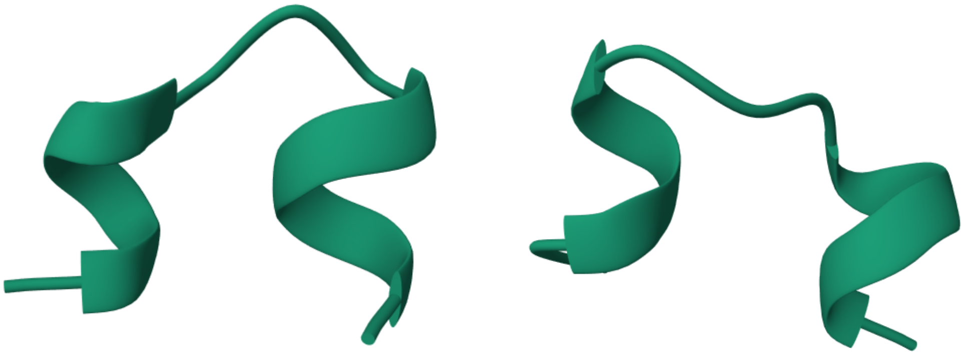

Figure 9: Examples of protein loops used to construct protein backbones. Images generated with Mol* (Sehnal et al., 2021).